Linguistically Informed Question Answering

Gerhard Fliedner

21st March 2007

Zusammenfassung

Question Answering hat in den letzten Jahren lebhaftes Interesse von Forschern in den Informationswissenschaften erfahren. Insbesondere die Einführung von Wettbewerben zur Evaluation hat große Aufmerksamkeit erregt (Voorhees and Dang, 2006; Magnini et al., 2006).

Ein Question-Answering-System (QA-System) erwartet Fragen in natürlicher Sprache als Eingabe, sucht in großen Dokumentsammlungen nach passenden Antworten und präsentiert dem Benutzer umfassende Antworten (Maybury, 2004b).

Question Answering wird oft als Sonderfall des Information Retrieval angesehen. Viele QA-Systeme basieren auf Methoden aus dem Information Retrieval und verwenden nur wenig linguistische Information (Hirschman and Gaizauskas, 2001).

Solche Systeme haben daher oft Schwierigkeiten, Antworten genau zu identifizieren. Dies führt typischerweise zu niedriger Antwortpräzision (d. h., ein großer Teil der Antworten des Systems sind falsch) und einer Abhängigkeit von Antwortredundanz (d. h., das System muss eine Antwort mehrfach in verschiedenen Formulierungen vorfinden).

Wenn man Question Answering als linguistisches, insbesondere als computerlinguistisches Problem auffast, eröffnet dies meiner Meinung nach interessante neue Perspektiven für praktische QA-Systeme und bietet Lösungen für die genannten Probleme.

Ich werde daher Arbeiten zu Fragen und Antworten in der linguistischen Literatur als Ausgangspunkt nehmen und Phänomene im Zusammenhang mit Fragen und Antworten in menschlicher Kommunikation erörtern. Ich komme zu dem Schluss, dass sowohl syntaktische als auch semantische und pragmatische Gesichtspunkte eine Rolle bei der Beschreibung von Fragebeantwortung spielen (*anwerhood*, d. h., die Beziehung zwischen einer Frage und ihrer Antwort). Ich stelle fest, dass linguistische Arbeiten zu Fragen und Antworten an der Beschreibung und Erklärung der Beziehung der Fragebeantwortung zwischen einer gegebenen Frage und einer gegebenen Antwort interessiert sind. Um Antworten auf Fragen in einer Textsammlung zu finden, ist diese Herangehensweise nicht geeignet, da es häufig nötig ist, Antworten von Texten durch zusätzliche Inferenzschritte abzuleiten, insbesondere wenn Unterschiede in der Formulierung vorliegen. Vom linguistischen Konzept der indirekten Fragebeantwortung (*indirect answerhood*, Higginbotham and May, 1981) leite ich daher indirekte Fragebeantwortung als Kernkonzept für Question Answering mit linguistischen Informationen (*Linguistically Informed Question Answering*) ab: Dies ist eine Relation, die genau dann zwischen einem Text und einer Frage besteht, wenn eine Antwort auf die Frage aus dem Text inferiert werden kann.

Ein nahe liegender Ansatz für automatisches Question Answering bestünde darin, für alle Texte in der Dokumentsammlung und für alle Fragen von Benutzern semantische Repräsentationen abzuleiten und Methoden des automatischen Schließens anzuwenden, um so Antworten aus den in der Dokumentsammlung enthaltenen Informationen abzuleiten. Dieser Ansatz ist allerdings aus verschiedenen Gründen nicht als Grundlage einer praktischen Implementation geeignet. Die wichtigsten Probleme sind die folgenden: Zum einen ist es zurzeit nicht möglich, volle semantische Repräsentationen aus allgemeinen Texten abzuleiten. Zum anderen ist die Abdeckung von Wissensbasen, die als Quelle für Inferenzen benötigt würden, nicht ausreichend.

Ich gebe daher eine einfachere Näherung der indirekten Fragebeantwortung an. Sie basiert darauf, syntaktische Dependenzbäume, die mit lexikalisch-semantischen Informationen angereichert sind, zu matchen: Wenn ein solches Match zwischen einer Frage- und einer Antwortrepräsentation gefunden werden kann, so kann man davon ausgehen, dass zwischen ihnen die Beziehung der Fragebeantwortung gegeben ist. Wir ergänzen dieses Matching durch die Anwendung von Inferenzregeln, die als Umetikettierung (*Relabelling*) der linguistischen Strukturen dargestellt werden.

Diese Spezifikation der Inferenzregeln erlaubt es, Informationen aus beliebigen Quellen modular zu integrieren. Zurzeit verwende ich vor allem zwei linguistische Ressourcen als Quelle für Inferenzregeln, nämlich GermaNet (die deutsche Version von WordNet) und FrameNet.

Um aus diesem Konzept des Matchings von Bäumen einen praktisch anwendbaren Suchalgorithmus abzuleiten, entwickele ich einen Algorithmus, der auf einem bekannten, effizienten Algorithmus für ungeordnete Pfadeinbettung basiert (Kilpeläinen, 1992, $O(n^{5/2})$). Ich modifiziere den Algorithmus so, dass er Suche und Inferenzen zu einem kombinierten Prozess integriert.

Ich passe den Algorithmus dann weiter an, so dass er eine relationale Datenbank verwendet, um die Speicherung großer Dokumentsammlungen zu ermöglichen und das Retrieval zu beschleunigen.

Um die Umsetzbarkeit meines Ansatzes zu zeigen, habe ich SQUIGGLI als Prototyp-QA-System für das Deutsche implementiert. Kern des Systems bildet eine Toolchain von natürlich-sprachlichen Modulen, die aus deutschen Texten syntaktische Dependenzstrukturen ableiten, diese mit lexikalisch-semantischen Informationen anreichern und anaphorische Bezüge auflösen. Mithilfe dieser Toolchain werden Textrepräsentationen aus Dokumentsammlungen abgeleitet und in der Datenbank gespeichert. Das eigentliche QA-System verwendet die Toolchain, um Fragen in Repräsentationen im gleichen Format zu übersetzen, sucht dann nach Antworten in der Datenbank und präsentiert dem Benutzer Antworten, die aus den gefundenen Repräsentationen mithilfe eines Generierungsmoduls erzeugt werden. Zusätzliche Funktionen, wie gezielte Anwortbegründungen und Anaphernauflösung in Fragen, bieten eine Grundlage für interaktives Question Answering.

Die Ergebnisse einer End-to-End-Evaluation zeigen, dass das System eine interessante kombinierte Performanz erreicht: Es beantwortet über 33 % aller Fragen korrekt, und zwar mit einer Antwortpräzision von über 66 %.

Der Suchalgorithmus ist effizient: Der Großteil der Antworten wird in einigen Sekunden gefunden. Dabei ist zu beachten, dass die Suche auf Grundlage linguistischer Strukturen über den Repräsentationen der gesamten Dokumentsammlung vorgenommen wird.

Wir hoffen mit dieser Arbeit zu zeigen, dass Question Answering mit linguistischer Information einen interessanten neuen Ansatz für das Question Answering darstellt und dass es als Grundlage für die Entwicklung benutzerfreundlicher interaktiver QA-Systeme herangezogen werden kann.

Beiträge zum Forschungsstand

Die hauptsächlichen Beiträge dieser Arbeit zum Forschungsstand sind die folgenden:

Question Answering mit linguistischer Information. Ich entwickle Question Answering mit linguistischer Information (*Linguistically informed Question Answering*), unter Berücksichtigung von in der linguistischen Literatur beschriebenen Phänomenen im Zusammenhang mit Fragen und Antworten. Ich stelle einen Ansatz zum Question Answering vor, der auf der Nutzung reicher linguistischer Informationen anstelle wissensarmer Methoden aus dem Information Retrieval basiert.

Modellierung Indirekter Fragebeantwortung als Matching von Baumstrukturen. Als Kern des Question Answering mit linguistischer Information modelliere ich indirekte Fragebeantwortung (*indirect answerhood*) als lokale

Inferenzen über syntaktischen Dependenzstrukturen, die mit lexikalisch-semantischer Information angereichert sind, und Matching von Frage- und Antwortstrukturen.

Nutzung lexikalischer Datenbasen als Quelle für Inferenzen. Inferenzen und Matching in der indirekten Fragebeantwortung werden durch eine linguistische Datenbasis gesteuert. Ich zeige, wie Information, insbesondere aus lexikalischen Ressourcen (nämlich GermaNet und FrameNet) in diese Datenbasis überführt werden kann.

Ein effizienter, auf indirekter Fragebeantwortung basierender Suchalgorithmus. Ausgehend von meiner Definition der indirekten Fragebeantwortung definiere ich einen effizienten Suchalgorithmus, der auf einem bekannten Algorithmus für ungeordnete Pfadeinbettung basiert, diesen allerdings dahingehend erweitert, dass Inferenzschritte direkt in die Suche integriert werden.

Eine Korpusstudie von Fragen und Antworten. In einer Korpusstudie von Fragen und Antworten früherer QA-Wettbewerbe ermittle ich wichtige Quellen für Inferenzen, die benötigt werden, um Antworten gezielt zu finden, so zum Beispiel WordNet and FrameNet.

Interaktive Funktionen. Ich beschreibe mehrere Funktionen, die verwendet werden können, um QA-Systeme interaktiv und benutzerfreundlicher zu gestalten. Dieses sind insbesondere Antwortbegründung und die Nutzung von Anaphern- und Ellipsenauflösung in Fragen.

Evaluation. In der Evaluation gebe ich einen detaillierten Überblick über die Systemperformanz. Ein generell interessanter Punkt für das Design von QA-Systemen ist die Evaluation der Beiträge einzelner Module und Ressourcen, die zeigt, dass Question Answering nur durch ein Zusammenspiel vieler verschiedener Komponenten erfolgreich implementiert werden kann. Unter diesen sind die Nutzung linguistischer Strukturen, die richtige Identifikation der Information in Adverbialen, die korrekte Auflösung von Anaphern in Texten und die Nutzung lexikalisch-semantischer Relationen als Quelle von Inferenzen die wichtigsten.

Gliederung

In Kapitel 2 fasse ich den aktuellen Forschungsstand im Bereich Informationszugriff zusammen, mit einem Schwerpunkt auf QA-Systemen. Ich beschreibe außerdem die benachbarten Bereiche Information Retrieval, natürlich-sprachliche Schnittstellen zu Datenbanken und Information Extraction. Die meisten QA-Systeme verwenden Information-Retrieval-Module und nur eine begrenzte Menge linguistischer Information. Ich zeige, dass dies zu Recall- und Präzisionsproblemen führen kann.

In Kapitel 3 stelle ich die konzeptuellen Grundlagen für mein Projekt dar. Ich gebe einen Überblick über linguistische Forschungen zu Fragen und Antworten und führe dabei syntaktische, semantische und pragmatische Aspekte auf. Ich ermittle indirekte Fragebeantwortung (*indirect Answerhood*, „Fragebeantwortung plus Inferenz“) als eine geeignete konzeptuelle Grundlage für die Suche nach Antworten in Dokumentsammlungen. Ich beschreibe Ansätze zu Question Answering auf Basis strukturierter semantischer Repräsentationen und Methoden des automatischen Schließens. Ich zeige auf, warum diese Ansätze zurzeit keine geeignete Grundlage für Question Answering darstellen. Ich stelle dann die Ergebnisse einer Korpusstudie von Fragen und Antworten aus früheren QA-Wettbewerben vor.

In Kapitel 4 untersuche ich mehrere mögliche Quellen für lokale Inferenzregeln. Ich stelle fest, dass sowohl GermaNet – die deutsche Version von WordNet – als auch FrameNet geeignete Information in Form von semantischen Relationen zwischen Konzepten bieten.

In Kapitel 5 spezifiziere ich eine Näherung indirekter Fragebeantwortung, die syntaktische Dependenzstrukturen verwendet, die mit lexikalisch-semantischer Information angereichert werden. Diese Näherung modelliert Inferenzen als lokale Umettiketierungsoperationen (*Relabelling*) auf Bäumen und Frage-Antwort Matching als Matching von Bäumen. Von dieser Grundlage leite ich einen effizienten Suchalgorithmus ab, der auf einem bekannten Algorithmus für das ungeordnete Pfadeinbettungsproblem basiert. Ich erweitere diesen Algorithmus, so dass er zusätzlich lokale Inferenzen *während* der Suche durchführt. Ich zeige dann, wie dieser Algorithmus auf Basis einer relationalen Datenbank für die Speicherung und das Retrieval linguistischer Strukturen implementiert werden kann.

In Kapitel 6 beschreibe ich Design und Implementierung von SQUIGGLI, einem deutschen QA-System, das auf dem Ansatz des Question Answering mit linguistischer Information basiert. Der Kern des Systems ist eine Toolchain zur Verarbeitung natürlicher Sprache für das Deutsche, die eine kaskadierte Parsing-Architektur verwendet und einen topologischen Parser, einen Named-Entity-Erkenner, einen NP/PP-Chunker, ein GermaNet- und ein FrameNet-Annotationsmodul sowie ein Anaphernauflösungsmodul umfasst. Ich beschreibe darüber hinaus die Benutzerschnittstelle des Systems.

In Kapitel 7 werden die Resultate einer Systemevaluation zusammengefasst. Ich habe eine End-to-End-Performanzevaluation des Systems durchgeführt und dazu ein Korpus von Fragen und Antworten zu Texten in der deutschen Wikipedia herangezogen. Die Performanzevaluation wird ergänzt durch eine diagnostische Evaluation, die sowohl die Beiträge der einzelnen Module und Ressourcen für die erfolgreiche Beantwortung von Fragen als auch die Gründe für Fehler analysiert. Die Performanz des Parsers und des Anaphernauflösungsmoduls wurden in einer eigenen Komponentenevaluation ermittelt.

In Kapitel 8 fasse ich die Ergebnisse der Arbeit kurz zusammen und stelle Ziele für künftige Arbeiten vor.

Acknowledgements

Work that forms the basis of this thesis was carried out by the author at Saarland University, Department of Computational Linguistics, Saarbrücken and at DFKI GmbH, Saarbrücken, for the most part within the projects Collate and Collate 2 (*Computational Linguistics and Language Technology for Real Life Applications*), funded by the German Ministry for Education and Research, Grant numbers 01 IN A01 B and 01 IN C02.

The following people have provided me with data that was used in system development and evaluation. I would like to heartily thank them again.

- Christian Braun has developed large parts of the processing tool-chain that derives PREDS from German texts in his time at our department.
- Julia Besser and Niko Felger have annotated the corpus used in parser evaluation at our department.
- Michael Strube of EML GmbH, Heidelberg, made the Heidelberg Text Corpus with its coreference annotation available, which was used in the evaluation of my anaphora resolution module.
- Manfred Stede, at Institut für Linguistik (Department of Linguistics),University of Potsdam, has made the Potsdam Coreference Corpus available to me. Unfortunately, I found that I could not employ it for evaluation purposes, as the annotation format differed to much from mine.
- Sabine Schulte im Walde (then at our department, now at Computational Linguistics Department at Stuttgart University) made results from her Ph. D. thesis (rules and lexicon entries with probabilities from a statistical parser for German) available to me, which were used to integrate optimising heuristics into some rules in our PREDS parser.

- Katrin Erk and Sebastian Padó at our group gave me a ‘sneak preview’ of the annotated Salsa corpus and also provided tools for the automatic extraction of information from the data.

- Irene Cramer, Jochen Leidner and Dietrich Klakow at Signal Processing Department, Saarland University provided me with access to the Smart-Web AnswerGatherer corpus, used for the end-to-end system evaluation.

Besides, a lot of people have in some way or other contributed to this thesis through advice, ideas, discussion etc.

First and foremost, I would like to thank my supervisors, Manfred Pinkal and Anette Frank.

Then, thanks go to people at our department, with whom I have discussed different issues concerned with the thesis: Aljoscha Burchardt (FrameNet, detour), Christian Braun (parsing, annotation, evaluation), Katrin Erk (FrameNet, grammar induction), Anette Frank (evaluation), Kerstin Hadelich (data collection, evaluation), Alexander Koller (algorithms and complexity), Andrea Kowalski (FrameNet, Multi-Word expressions), Marco Kuhlmann (Dependency Grammar), Sebastian Padó (FrameNet, grammar induction), Sabine Schulte im Walde (probabilistic parsing, heuristics), Stefan Thater (all of the above and more: Stefan, you have been extremely helpful throughout, especially for just running ideas past you and discussing them, a special thank you!), Olga Uryupina (anaphora resolution), Stephan Walter (parsing and storing) and Magda Wolska (who has provided hints for literature search and sweets, another special thank you!).

Peter Baumgartner and Fabian M. Suchanek (both at Max Planck Institute, Saarbrücken at that time) have discussed issues of knowledge representation and inferencing with me, thanks!

With Martin Theobald and Gerd Weikum (also Max Planck Institute), I have talked about databases, XML and structured access.

Christoph Koch (Database Theory, Saarland University) has provided valuable discussion on databases, especially databases using graph structures. Thank you!

The Berkeley FrameNet people, especially Collin Baker, Michael Elsworth, Chuck Fillmore and Miriam Petruck have, at different workshops, conferences etc. been good enough to discuss my diverse questions concerning FrameNet, frame annotation and frame semantics. Also a big thank you!

Other people from different places have at different times also discussed my ideas with me and encouraged me, thank you very much to all of you: Sanda Harabagiu (QA), Ian Horrocks (Description Logic), Maghi King (User-Oriented Evaluation), Marc Light (QA and its challenges), Bernardo Magnini (evaluation, data collection), Michael Minock (database storage and access), Oliviero Stock (system design, user interfaces), Jörg Tiedemann (linguistically informed IR and linguistically informed QA), Roman Yangarber (evaluation)

Then, there were these people who make sure that things run smoothly, such as systems administration (especially Christoph Clodo and Rainer Martin) and the secretary people (especially Helga Riedel and Angelika Maurus).

People at CLT Sprachtechnologie GmbH (I've been managing director there in my 'spare' time) have also been very supportive and helpful; they have especially kindly adjusted all schedules according to the needs of my writing. Thanks to Daniel Bobbert, Kerstin Otte, Andrea Kowalski, Diana Steffen, Hajo Keffer and all the others.

Lots of friends and family have also been extremely helpful and listened patiently to my moanings and groanings throughout. Dear all, I hope I can make up for that now!

And, last but certainly not least, very big and special thanks go to Claudia, who has given me dearly needed support throughout and who had a knack of asking the right questions at the right time, such as *'When do you start writing up?'* and *'When do you finally* finish *writing up?'* I don't know how I would have managed without you.

Contents

Zusammenfassung **iii**

Acknowledgements **ix**

Contents **xiii**

List of Figures **xxi**

List of Tables **xxv**

List of Algorithms **xxvii**

1 Introduction **1**

1.1 Contributions of the Thesis 3

1.2 Outline 4

2 Question Answering and Other Information Access Systems **7**

2.1 Information Access Systems 7

2.1.1 Information Retrieval 8

2.1.1.1 Indexing 9

2.1.1.2 Retrieval Models 9

2.1.1.3 Adding Linguistic Information 11

2.1.1.4 Document Relevance 12

2.1.1.5 Evaluation 12

2.1.2 Natural Language Interfaces to Databases 13

2.1.2.1 Mapping between Natural Language and Database 13

2.1.2.2 Advantages 14

2.1.2.3 Disadvantages 15

2.1.3 Information Extraction 16

2.1.3.1 Knowledge Engineering Approach 16
2.1.3.2 Automatic Training Approach 17
2.1.3.3 Bootstrapping Approach 17
2.1.3.4 Discussion 18
2.2 Question Answering 18
2.2.1 General Overview 19
2.2.1.1 Early Systems 19
2.2.1.2 Current Systems 20
2.2.1.3 Current Research and Future Directions . . 20
2.2.2 Technical Overview 21
2.2.2.1 Document Collection Preprocessing 23
2.2.2.2 Question Analysis 24
2.2.2.3 Candidate Document Selection 24
2.2.2.4 Candidate Document Analysis 25
2.2.2.5 Answer Extraction 26
2.2.2.6 Response Generation 27
2.2.3 Discussion 27
2.3 Research Questions 28
2.3.1 Improving Answer Precision 28
2.3.2 Maintaining High Answer Recall 31
2.3.3 Answering Times 31
2.3.4 Efficient Structured Searching 31
2.4 Conclusions 32

3 Linguistically Informed Question Answering 33
3.1 Developing a Framework for Question Answering 34
3.1.1 Motivation 34
3.1.2 Setting the Stage 36
3.2 Questions and Answers 39
3.2.1 Syntactic Aspects 40
3.2.1.1 Interrogatives and Declaratives 40
3.2.1.2 Question Types 41
3.2.1.3 Embedded Interrogatives 44
3.2.1.4 German as an Example 44
3.2.1.5 Discussion 47
3.2.2 Semantic Aspects 48
3.2.2.1 Belnap's Erotetic Logic 49
3.2.2.2 Categorial Approach 53
3.2.2.3 Propositional Approach 61
3.2.2.4 Partition Approach 63
3.2.2.5 Additional Issues 70

3.2.3 Pragmatic Aspects 74
3.2.3.1 Questions and Speech Acts 75
3.2.3.2 The Gricean Conversational Maxims 78
3.2.3.3 Responses 79
3.2.3.4 Resolvedness Conditions of Questions and Answers 80
3.2.3.5 Discussion 84
3.2.4 Updating the Question Answering Framework 85
3.2.4.1 Question Analysis 85
3.2.4.2 Answer Finding 88
3.2.4.3 Answer Presentation 89
3.3 Question Answering Based on Structured Semantic Information 96
3.3.1 Question Answering Based on Componential Semantics 97
3.3.2 Question Answering Using Knowledge Representation and Reasoning 99
3.3.2.1 Knowledge Representation 100
3.3.2.2 Reasoning 101
3.3.2.3 A Large-Scale Experiment: The HALO Project 103
3.3.2.4 Discussion 105
3.3.3 Question Answering Based on Full Semantic Representations for Natural Language: Some Issues 105
3.3.3.1 Representation Formalism: Issues of Expressivity and Complexity 106
3.3.3.2 Automatically Deriving Structured Semantic Information from Natural Language 109
3.3.3.3 Axiomatising Knowledge 112
3.3.4 Discussion 113
3.4 Recognising Textual Entailment 113
3.4.1 The Textual Entailment Task and its Relevance for Question Answering 114
3.4.2 Circumscribing the Task 115
3.4.3 System Descriptions 118
3.5 Indirect Answers 119
3.5.1 Specification 120
3.5.1.1 Textual Inference and Direct Answerhood . 121
3.5.1.2 Answers with Inverted Polarity 122
3.5.1.3 Uncertain Answers 123
3.5.2 Relevant Phenomena 124
3.5.2.1 Corpus Study 125
3.5.2.2 Syntax and Morphology 126

3.5.2.3 Lexical Semantics 128
3.5.2.4 Anaphora 132
3.5.2.5 Direct Answerhood Phenomena 135
3.5.2.6 World Knowledge 139
3.5.3 Discussion 143
3.6 Conclusions 144

4 Resources **147**
4.1 Dependency Grammar and Dependency Structures 148
4.1.1 Dependency Grammar 148
4.1.2 Partially Resolved Dependency Structures 150
4.2 WordNet and GermaNet 153
4.2.1 WordNet 153
4.2.1.1 Princeton WordNet 153
4.2.1.2 Extended WordNet 157
4.2.2 GermaNet 158
4.3 FrameNet 161
4.3.1 English FrameNet 161
4.3.1.1 Frame Semantics 162
4.3.1.2 FrameNet Annotation 164
4.3.1.3 Frame-to-Frame Relations 164
4.3.1.4 Using FrameNet for Linguistically Informed Question Answering 166
4.3.2 German FrameNet: The SALSA Project 167
4.3.2.1 Frame Annotation in the Salsa Project . . . 169
4.3.3 Frame Resources Used in SQUIGGLI 170
4.4 Ontologies 172
4.4.1 Cyc 173
4.4.2 Suggested Upper Merged Ontology: SUMO 174
4.5 Conclusions 175

5 Matching Structured Representations **177**
5.1 Defining Indirect Answerhood 178
5.1.1 Relevance 179
5.1.2 Text Representations 180
5.1.3 Textual Inferences 183
5.1.4 Direct Answerhood 187
5.1.5 Answer Checking 190
5.2 The Linguistic Knowledge-Base 193
5.2.1 Related Work: Matching Graph-Based Text Representations 193

5.2.1.1 Matching Semantic Networks Using Query Expansion for Question Answering 194
5.2.1.2 Approaches Based on Approximate Tree Matching 195
5.2.1.3 Concept Graph Subsumption for Modelling Inferences 196
5.2.1.4 Graph Matching for Modelling Inferences . 197
5.2.1.5 Partial Matching of Graphs with LFG and FrameNet Information 197
5.2.1.6 Conceptual Graph Matching 198
5.2.1.7 Conclusions 199
5.2.2 Linguistic Knowledge and Inference 201
5.2.2.1 Structural Uniformity and Robustness . . . 202
5.2.2.2 Instantiating Relevance Values 203
5.2.2.3 Syntactic Dependency Information 206
5.2.2.4 Named Entities 208
5.2.2.5 GermaNet Information 208
5.2.2.6 Frame Information 212
5.2.2.7 Additional Role Relations 215
5.2.2.8 Coreference and Predication 215
5.2.2.9 Merging the Levels 216
5.2.3 Conclusion 218
5.3 Searching for Indirect Answers 218
5.3.1 Linguistic Graph Matching 219
5.3.1.1 Searching Answers as Graph Matching . . . 219
5.3.1.2 Graph Matching through Subgraph Isomorphism 220
5.3.1.3 Tree Matching by Path Inclusion 222
5.3.2 Answer Search 224
5.3.2.1 Generating Answers from Questions 225
5.3.2.2 The Search Algorithm 226
5.3.2.3 Practical Speed-Up 234
5.3.2.4 Conclusions 236
5.4 Storing and Retrieving Structured Representations 237
5.4.1 Using a Database System for Storage and Retrieval . . 237
5.4.1.1 A Note on Complexity of Database Queries 239
5.4.2 Database Scheme 240
5.4.3 Retrieval 241
5.5 Conclusions 244

6 SQUIGGLI: A System for Question Answering in German Grounded on Linguistic Information 245
6.1 Overall System Design 246
6.1.1 Off-line Processing 247
6.2 Deriving Representations from Text 248
6.2.1 Overall Parsing Architecture 250
6.2.1.1 Requirements 250
6.2.1.2 Cascaded Parsing 254
6.2.1.3 Alternative 1: Statistical Parsing 256
6.2.1.4 Alternative 2: The Hybrid Approach 257
6.2.2 Implementation Issues 258
6.2.3 Morphology 258
6.2.3.1 Reformed German Orthography 259
6.2.4 Topological Parser 259
6.2.5 Named Entity Recognition 260
6.2.5.1 Grammar 263
6.2.5.2 Learning 264
6.2.5.3 Other Approaches. 264
6.2.6 Noun Phrase Chunking 265
6.2.7 Constructing Dependency Structures 267
6.2.7.1 Representing Underspecification 267
6.2.7.2 Construction Process 271
6.2.8 Adding GermaNet Information 273
6.2.9 Anaphora Resolution 274
6.2.9.1 Other Approaches 274
6.2.9.2 Algorithm 276
6.2.9.3 Limitations 279
6.2.10 Constructing Frame Structures 280
6.2.10.1 Systems for Frame Assignment 280
6.2.10.2 Our Approach 282
6.2.11 Sample Output 285
6.3 Storing and Retrieving Text Representations 288
6.3.1 MySQL as Relational Database 288
6.3.2 Storing Linguistic Knowledge 289
6.3.3 Storing Text Representations 289
6.3.4 From Question to Database Query 290
6.3.5 Query Optimisation 291
6.4 User Interface 292
6.4.1 Basics 292
6.4.2 Walk-Through Example 293

6.4.3 Answer Generation 296
6.4.3.1 Motivation 297
6.4.3.2 Generating Answers from PREDS 297
6.4.4 Towards Interactive Question Answering 300
6.4.4.1 Anaphora and Coreference in Questions to Qustion Answering Systems 301
6.4.4.2 Anaphora Resolution 301
6.4.4.3 Ellipses 302
6.4.5 Making Use of Answer Relevance 303
6.4.6 Answer Support and Answer Justification 304
6.4.6.1 Answer Support: Displaying the Original Text 304
6.4.6.2 Answer Justification: Describing Textual Inference 305
6.4.7 From Answers to Responses: Providing Additional Information 306
6.5 Conclusions 306

7 Evaluation 309
7.1 Evaluating Question Answering Systems 310
7.1.1 Evaluating Natural Language Processing Systems . . . 310
7.1.2 Question Answering Competitions 311
7.1.3 Discussion 312
7.2 End-to-End Evaluation 313
7.2.1 Performance Evaluation 314
7.2.1.1 Evaluation Design 314
7.2.1.2 Test Data 317
7.2.1.3 Test Procedure 323
7.2.1.4 Results 324
7.2.1.5 Conclusions and Comparison 334
7.2.2 Diagnostic Evaluation 335
7.2.2.1 Contribution of Different Components and Resources 335
7.2.2.2 Error Analysis: Recall Errors 338
7.2.2.3 Error Analysis: Precision Errors 343
7.2.2.4 Results for Different Question-Answer-Types. 346
7.2.2.5 Conclusions 348
7.3 Component Evaluation 349
7.3.1 Evaluation of the Dependency Parser 350
7.3.1.1 Corpus Annotation 351
7.3.1.2 Evaluation Results 354
7.3.1.3 Error Analysis 357

7.3.1.4 Discussion 358
7.3.2 Anaphora Resolution 359
7.3.2.1 Measures 359
7.3.2.2 Evaluation and Results 362
7.3.2.3 Discussion 363
7.3.3 Parsing Speed 366
7.4 Conclusions 368

8 Conclusions 369
8.1 Summary 370
8.2 Results 371
8.3 Directions for Future Work 372
8.3.1 Additional Knowledge Sources 372
8.3.1.1 Extending Linguistic Resources 373
8.3.1.2 Integrating Additional Resources 374
8.3.2 Extension of the Indirect Answerhood Relation 375
8.3.2.1 Relaxed Answer Matching: Tree Inclusion . 375
8.3.2.2 Leaving Out Modifiers 375
8.3.2.3 Balancing Recall, Precision and Search Time 376
8.3.3 Evaluation of the User Interface 376
8.3.4 Answering Open Questions Through Summarisation . 376

Appendices 379

Appendix A Sample XML-Output of the PReDS-Parser 381

Index 387

Name Index 397

Bibliography 409

List of Figures

2.1 Overview of a Generic QA System, Adapted from Hirschman and Gaizauskas (2001, 286) 22
2.2 Answers Returned by Participating Systems for the TREC 2004 QA Track Question 1.2: *'What does the name [Crips] mean or come from?'* 30

3.1 Partitions for two questions (adapted from Groenendijk and Stokhof, 1984, 146) 64

4.1 PREDS for 'Lee Harvey Oswald has murdered John F. Kennedy in Dallas.' (Simplified) 149
4.2 Basic Constituent Structures for Active and Passive Sentence: '*Lee Harvey Oswald murdered John F. Kennedy.*' vs. '*John F. Kennedy was murdered by Lee Harvey Oswald.*' 152
4.3 PREDS for Active and Passive Sentence: *'Lee Harvey Oswald murdered John F. Kennedy.'* vs. *'John F. Kennedy was murdered by Lee Harvey Oswald.'* 152
4.4 Extended WordNet: Example for a Logical Form Derived Automatically from a WordNet Gloss for the Synset *Tennis, Lawn Tennis* (Adapted from Harabagiu et al., 1999, their Table 2) . 158
4.5 Frame Annotation Example: *'Vodafone bought Mannesmann.'* and '*Mannesmann was bought by Vodafone.*' 163
4.6 Frame Annotation Example: *'Vodafone bought Mannesmann.'* and '*Mannesmann acquired Vodafone.*' 163
4.7 Dependency Structure Extended with Frame Annotation for *'Vodafone Bought Mannesmann.'* 170

5.1 Text Representation for *'Lee Harvey Oswald has murdered John F. Kennedy in Dallas.'* 181

5.2 Local Inference as Relabelling of Elementary Trees 183
5.3 Inference: From Passive to Active Sentence 185
5.4 FrameNet Information as Source of Inference 186
5.5 Text representation for the question *'Which US-president was killed in Dallas?'* 187
5.6 Matching the representations for *'Which US-president was murdered in Dallas?'* and *'John F. Kennedy was murdered in Dallas.'* 190
5.7 Example for a Match Between Dependency Structures: PP *'Manufacturer of Cars'* matches Compound Noun *'Car Manufacturer'* 206
5.8 One Example for Matching Named Entities Using Relevance: Proper Names 209
5.9 Relabelling Using GermaNet Information: *'Vodafone took over Mannesmann'* allows inferring *'Vodafone bought Mannesmann'*, overall relevance 0.9 211
5.10 GermaNet and Grammatical Function Relabelling: *'Prime Minister of Japan'* is equivalent with *'Japanese Premier'*, overall relevance 0.81 211
5.11 Example for Relabelling Using Frame-to-Frame Relations: *'Mannesmann was sold to Vodafone'* is equivalent with *'Vodafone bought Mannesmann'* 214
5.12 Example for Matching Question and Answer: *'Who makes cars?'* is answered by *'SFT has been producing jeeps [for Mercedes for years].'* 217
5.13 Indirect Answerhood and Its Inverse Relation 225
5.14 'Multiplying out' Equivalences 229
5.15 Tree Matching: *'Who sold his House?'* Wrongly Matches *'Peter sold Tom's house.'* 231
5.16 Database Scheme for Flattened Text Representations 240

6.1 Design of the SQUIGGLI System 246
6.2 System Components in Parsing 249
6.3 Examples for Structures Recognised by the Topological Parser 261
6.4 Examples for Structures Recognised by the Topological Parser (continued) 262
6.5 Examples for Complex NPs/PPs 266
6.6 Underspecified PReDS Representation for *'Das Unternehmen hat 1997 die Zahl seiner Kunden auf über eine Million verdoppelt.'* 268
6.7 Underspecified PReDS Representation for *'Das Unternehmen hat 1997 die Zahl seiner Kunden auf über eine Million verdoppelt.'* 286

6.8 PReDS Representation for *‘E-Plus Mobilfunk GmbH, Düsseldorf: Das Unternehmen hat 1997 die Zahl seiner Kunden auf über eine Million verdoppelt.’* 287

7.1 F-Score with Weighting Factor β 326
7.2 Hierarchy of Grammatical Relations (from Briscoe et al., 2002, 5) 351
7.3 Example: Grammatical Relation Annotation 352
7.4 Model-Theoretic Scoring Scheme for Coreference Resolution: Formulæ for Recall and Precision according to Vilain et al. (1995) 360
7.5 Recall, Precision and F-Score for ‘Correct Chain’ and ‘Correct First Antecedent’ Scheme 361

List of Tables

4.1 Semantic Relations in WordNet 155
4.2 Some Database Statistics for WordNet 2.1 156
4.3 Relations in GermaNet (May 2003 Version) 160
4.4 Some GermaNet Statistics (May 2003 Version) 160
4.5 Some FrameNet Statistics (Release 1.2, December 2004, 609 frames) 168
4.6 Coverage of Frame Assignment Rules 171

6.1 Types of Named Entities Recognised by the Named Entity Recogniser 263
6.2 Initial Weights for the Salience Factors for Anaphora Resolution, adapted from Lappin and Leass (1994, 541) 278

7.1 Question-Answer Pairs in the SmartWeb AnswerGatherer Corpus 318
7.2 SmartWeb AnswerGatherer Corpus: Distribution of Question Words and Question Constructions 321
7.3 SmartWeb AnswerGatherer Corpus: Distribution of Question-Answer-Type 322
7.4 Results for All Questions, First Answer 324
7.5 Non-NIL Question Answer Recall, Precision and F-Score, First Answer 325
7.6 NIL Recall, Precision and F-Score 326
7.7 Evaluating Answer Ranking: Position of First Correct Answer 326
7.8 Mean Reciprocal Rank 327
7.9 Justification: Can Correct Answers be Recognised by Looking at Justification Sentence(s) Alone? 329
7.10 Justification: Can Wrong Answers be Disregarded when Looking at Justification Sentence(s) Alone? 329
7.11 Inter-sentential Anaphora in Questions 330

7.12 Uncertain Answers and Warnings: Was the Answer/Warning Correct? 332
7.13 Database Search Times for all Questions: Overview 333
7.14 Database Search Times: Percentage of Answers Found within *n* Seconds 334
7.15 Resources Used for Answering Questions: Results for 200 Correctly Answered Non-NIL Questions 336
7.16 Reasons of Failure: No Answer Found 339
7.17 Types of Errors: Spurious Answers for NIL Questions 343
7.18 Types of Errors: Wrong Answers for Non-NIL Questions . . . 343
7.19 Evaluation Results, Broken Down by Question-Answer-Type . 347
7.20 Details of the Manually Annotated Evaluation Corpus 353
7.21 Parser Evaluation Overview: Precision, Recall and F-Score for Sub-Corpus 'Business News' 355
7.22 Parser Evaluation Overview: Precision, Recall and F-Score for Sub-Corpus 'Legal' 356
7.23 Parser Evaluation Results, Classified according to Grammatical Relation, Business News (Lenient Scoring) 356
7.24 Parser Evaluation Results, Classified according to Grammatical Relation, Legal (Lenient Scoring) 356
7.25 Evaluation Results Using Dependency Structures for Amit Dubey's Statistical German Parser Sleepy (Dubey, 2004, 83–87) 359
7.26 Evaluation against the Heidelberg Text Corpus: Recall, Precision and F-Scores according to the Model-Theoretic Scoring Scheme (Vilain et al., 1995) 363
7.27 Evaluation against the Heidelberg Text Corpus: Detailed Analysis by Types of Anaphoric Expressions 363
7.28 Evaluation against the Heidelberg Text Corpus: Recall, Precision and F-scores for Different Types of Anaphoric Expressions 364
7.29 Detailed Evaluation Results for Strube et al.'s German Coreference Resolution System (Strube et al., 2002, their tab. 7) . . . 365
7.30 Evaluation of Parse Times: Minimum, Maximum and Average Parse Time per Sentence 366
7.31 Evaluation of Parse Times: Minimum, Maximum and Average Parse Time per Sentence 367

List of Algorithms

5.1 Unordered Path Inclusion, adapted from Kilpeläinen (1992, pp. 41–2, his alg. 4.14) 223
5.2 Algorithm for Constructing Indirect Answer Representations for a Given Question Representation 227
5.3 Algorithm for Searching Answer Representations for a Given Question Representation 233
5.4 Constructing SQL Queries from Question Representations . . 242

Chapter 1

Introduction

Question Answering has recently received lively interest from researchers in information sciences. Especially the establishment of shared competitions has attracted a lot of attention (Voorhees and Dang, 2006; Magnini et al., 2006).

A question answering (QA) system takes questions in natural language as its input, searches for suitable answers in large document collections and presents a comprehensive answer to the user (Maybury, 2004b).

Question answering is often seen as a special case of information retrieval and most QA systems are based upon methods from information retrieval and use only little linguistic information (Hirschman and Gaizauskas, 2001).

Question Answering systems therefore often have difficulties in reliably pinpointing answers. These difficulties typically lead to low answer precision (i. e., a high proportion of the system's answers are wrong) and a dependence on answer redundancy (i. e., the systems must hit upon an answer several times in different formulations).

We suggest that looking at QA as a problem of linguistics, and especially computational linguistics, opens up interesting new perspectives for usable QA systems and provides solutions for the mentioned problems.

We accordingly take work on questions and answers in linguistics as a starting point and explore phenomena associated with questions and answers in human interaction. We find that syntactic, semantic and pragmatic aspects play a rôle in describing answerhood (i. e., the relation that holds between a question and its answer). We note that work on questions and answers in linguistics is concerned with describing and explaining the relation of answerhood between a given question and a given answer. For finding answers in document collections, this view is not well-suited, as it is often necessary to derive the answer from the text through a number of additional inferencing steps, specifically when there are differences in wording between question and text. From the linguistic concept of indirect answerhood (Higginbotham and May, 1981), we derive indirect answerhood as a key concept for linguistically informed QA: Indirect answerhood is a relation between a text and a question that holds exactly if an answer to the question can be *inferred* from the text.

One obvious approach to automatic question answering would therefore be to derive semantic representations from all texts in the document collection and from the users' questions and employ automated reasoning methods to infer answers from the information in the document collection. However, this approach fails as a basis for practical applications for a number of reasons. The two most important issues are that it is currently not possible to derive full semantic representations from general texts and that knowledge bases that would be needed as a source of inferences do not provide adequate coverage (cf. 3.3).

We therefore specify a more shallow approximation of indirect answerhood. This approximation is based on the matching of syntactic dependency tree structures extended with lexical semantic information: If a match can be found between a question representation and an answer representation, answerhood is assumed to hold between the respective question and answer. We complement this matching with inference rules, expressed as relabellings of the linguistic structures.

This specification of the inference rules allows to easily integrate information from arbitrary sources in a modular fashion. We currently employ chiefly two linguistic resources, namely GermaNet (the German version of WordNet) and FrameNet, as sources from which inference rules are derived.

In order to arrive at a practically usable search algorithm from the notion of tree matching, we develop an algorithm that is based on an efficient algorithm for unordered path inclusion (Kilpeläinen, 1992, $O(n^{5/2})$). We modify the algorithm so that it combines search and inferences in one interleaved process.

We further adapt the algorithm so that it can be used with a standard relational database system to allow storage of large document collections and to speed up retrieval.

To show the practicability of our approach, we have implemented SQUIGGLI, a prototype QA system for German. The core of the system is a Natural Language Processing chain that derives syntactic dependency structures from German texts, enriches them with lexical semantic information and resolves anaphoric references. Using this processing chain, text representations are derived from a given document collection and stored in a relational database. The QA system itself uses the processing chain to translate questions into representations in the same format, searches for answers in the database and presents answers generated from the retrieved representations using a generation module to the user. Additional features, like focussed answer justification and anaphora resolution in questions provide a basis for interactive QA.

The results of an end-to-end evaluation proved that the system reaches an interesting level of combined performance: It correctly answered over 33 % of the questions, with an answer precision of over 66 %.

The search algorithm is quite efficient: The majority of the answers were found within a few seconds, even though searching is done using structured linguistic information over the representations of the full document collection.

We aim to show that linguistically informed QA forms an interesting new approach to QA and that it can advantageously be used as a basis of building user-friendly, interactive QA systems.

1.1 Contributions of the Thesis

The main contributions of this thesis are the following ones:

Linguistically Informed Question Answering. We derive linguistically informed question answering, taking phenomena in questions and answers described in the linguistic literature into account. We present an approach to QA that focusses on the use of rich linguistic information instead of knowledge-lean Information Retrieval methods.

Modelling Indirect Answerhood by Matching Tree Structures. As the core of linguistically informed QA, we model indirect answerhood as local inferences on syntactic dependency structures extended with lexical semantic information and matching question and answer structures.

Use of Lexical Databases as Source of Inferences. Inference steps and matching in indirect answerhood are controlled by a linguistic knowledge base. We show how information, especially from lexical resources (namely GermaNet and FrameNet) can be transferred into the knowledge base.

An Efficient Search Algorithm Based on Indirect Answerhood. Starting from our definition of indirect answerhood, we define an efficient search algorithm that is based on a known algorithm for unordered path inclusion, but extends the algorithm to also integrate inference steps into the search itself.

A Corpus Study of Questions and Answers. In a corpus study of questions and answers from past QA competitions, we show important sources of inferences needed to find answers in a directed fashion, such as WordNet and FrameNet.

Features for Interactive QA. We describe a number of features that can be used to make QA systems interactive and more user-friendly. This especially includes answer justification and the use of anaphora and ellipsis resolution on questions.

Evaluation. In the evaluation, we show a detailed picture of the system's performance. One point of general interest for the design of QA systems is the evaluation of the contributing modules and resources, which shows that QA can only be done successfully by an interplay of many different components. Among these, using linguistic structures, properly identifying information in adverbials, correctly resolving anaphora in texts and the use of lexical semantic relations as sources of inferences are the most important ones.

1.2 Outline

In chapter 2, we will summarise the current state of the art in Information Access systems, focussing on Question Answering systems. We also describe the related areas of Information Retrieval, Natural Language Interfaces to Databases and Information Extraction. Most current QA systems use information retrieval methods and only a limited amount of linguistic information. We point out that this may lead to recall and precision problems.

In chapter 3, we explore the conceptual background of our project. We give an overview of linguistic research on questions and answers, listing syntactic, semantic and pragmatic aspects. We identify indirect answerhood ('answerhood plus inferences') as a suitable conceptual basis for searching answers to questions in document collections. We describe approaches to QA based on structured semantic representations and reasoning methods. We show why these approaches currently do not offer a suitable basis for QA. We then report the results from a corpus study of questions and answers in past QA competitions.

In chapter 4, we investigate a number of possible sources of local inference rules. We find that both GermaNet – the German version of WordNet – and FrameNet provide suitable information in the form of lexical semantic relations between concepts.

In chapter 5, we specify an approximation of indirect answerhood, which uses syntactic dependency structures extended with lexical semantic information. This approximation models inferences as local relabelling operations on trees and question-answer matching as tree matching. From this basis, we derive an efficient search algorithm based on an existing algorithm for the unordered path inclusion problem. We extend this algorithm to additionally perform local inferences *during* search. We then show how this algorithm can be implemented using a standard relational database for the storage and retrieval of linguistic structures.

In chapter 6, we describe the design and implementation of SQUIGGLI, a German QA system based on linguistically informed question answering. The core of the system is a natural language tool-chain for German, which uses a cascaded parsing architecture and comprises a topological parser, a Named Entity Recogniser, an NP/PP chunker, a GermaNet and a FrameNet annotation module and an anaphora resolution module. We also describe the user interface of the system.

In chapter 7, the results of a system evaluation are reported. We conducted an end-to-end performance evaluation of the system, using a corpus of questions and answers pertaining to the German Wikipedia. The performance evaluation is complemented by a diagnostic evaluation, showing the contributions of the different modules and resources to successful answer search, as well as the sources for failures. The performance of the parser and the anaphora resolution module are assessed in a separate component evaluation.

In chapter 8, we sum up the thesis and propose directions for future work.

Chapter 2

Question Answering and Other Information Access Systems

In this chapter, we will give an overview of the state of the art in Information Access (IA) systems, that is, systems that manage information and provide users with means to access it. The overview will be centred around Question Answering (QA) system as the subject of this thesis. The chapter will not try to list all IA systems or all techniques that have been used in IA, but rather introduce central points that will be taken up in later chapters, when our approach is described more closely in chapter refsec:SystemDesign.

As a prerequisite for introducing QA, we will first define and describe several other research areas in Information Access (2.1), namely Information Retrieval, Natural Language Interfaces to Databases and Information Extraction. This is followed by a more detailed overview of QA systems themselves (2.2). We will then formulate the research questions of this thesis, outlined in 1.1, more precisely (2.3).

2.1 Information Access Systems

In this thesis we will only be concerned with systems that manage textual information. While there has been research on systems managing other media types (such as automatically recognising and searching for the subjects of pictures, e. g., Del Bimbo, 1999), managing textual data is still the most common direction in IA. The knowledge base underlying the IA system will in most cases be a collection of text documents, such as articles in a newspaper or magazine archive, word processor documents in an Intranet or web pages on the Internet.

We will focus on systems that search for information in such collections. We will not deal with other types of systems, such as document categorisation systems, which automatically put documents into predefined groups for easy retrieval, or text summarisation systems.

We will shortly introduce and describe the different types of systems, especially focussing on the distinctions between them that we will assume.

Information Retrieval Systems. Information Retrieval (IR) systems search text collections for documents that best match the users' query to the system (Baeza-Yates and Ribieiro-Neto, 1999). Queries are expressed as lists of keywords. A typical example are Internet Search engines.

Question Answering Systems. In contrast to IR systems, question answering (QA) systems search *answers* in text collections for *questions* that a user poses in natural language (Hirschman and Gaizauskas, 2001). The output of a QA system is a concise answer to the question rather than a document or a passage.

Natural Language Interfaces to Databases. Natural Language Interfaces to Databases (NLIDB systems) also answer questions, similar to QA systems (Copestake and Sparck Jones, 1990). However, they use structured databases instead of text collections as knowledge base.

Information Extraction Systems. Information Extraction (IE) systems fill (instantiate) predefined templates that describe certain chunks of information from text documents (Gaizauskas and Wilks, 1998). This can be, for example, named entity templates for names of persons or organisations, but also more complex event templates, which describe, say, the launch of a satellite.

Note that the types of systems are sometimes defined differently in the literature. For example, Question Answering systems are sometimes seen as a specialised subtype of IR systems and NLIDB systems are sometimes taken to be a subtype of QA systems, respectively. We will use the 'exclusive' specifications described here.

2.1.1 Information Retrieval

A textual information retrieval (IR) system searches a given document collection for pieces of texts that best match a user's query to the system. A query consists of one or more words, in IR jargon called the search terms. Information retrieval systems may be used to retrieve whole documents (document retrieval) or passages (typically paragraphs, passage retrieval systems). IR systems are widely used today. The best-known examples are Internet search engines, such as Google and Yahoo.

In our description of IR systems, we will follow Baeza-Yates and Ribieiro-Neto (1999), where additional details and references to the literature can be found.

2.1.1.1 Indexing

In order to allow efficient searching, IR systems use indices. They first need to process all documents in the document collection to build a suitable index structures. This step is generally called indexing.

In the simplest case, a so-called inverted file index is used (Baeza-Yates and Ribieiro-Neto, 1999, chapter 8). It has pointers, for every word in the document collection, to the documents in which this word is contained. Searching is done by first retrieving the search terms of the user's query in the index and identifying the documents best suited (most relevant) for the query by suitably combining the respective pointers to the documents. Depending on the used retrieval model (2.1.1.2), different index structures are employed.

2.1.1.2 Retrieval Models

Different retrieval models have been used in information retrieval. The retrieval model defines how to compute a relevance score for a document and a given query. The most commonly used retrieval models are the Boolean model and the vector model. We will shortly characterise them in the following. Other models, such as the probabilistic model or different forms of hybrid models are mostly used in research and will not further be discussed here (cf. Baeza-Yates and Ribieiro-Neto, 1999).

Boolean Model. In the Boolean model, users can combine their search terms using Boolean connectors (such as `AND`, `OR` or `NOT`) to define a filter: Only documents matching that filter are returned as the result of the query (i. e., documents that contain all terms connected by `AND`, at least one of the terms connected by `OR`, no term prefixed by `NOT` etc.). Relevance is reduced to a binary decision: Either a document matches a query or not. The Boolean model is easy to implement. However, casual users often have difficulties to correctly express their information need as a Boolean query. For many tasks the vector model is considered more suitable since it uses fine-grained relevance scores and can rank documents for relevance.

Vector Model. The vector model uses the concept of similarity between document vectors and query vectors: A document vector is built using all words contained in the document collection as its dimensions. In the basic case, only the word frequency within that document is used as the value for the word's dimension in the document vector; for words that are not present in the document, this value will be set to zero. Most systems use an additional weighting to compute the values in the document vector (see below).

From the user's query, a query vector is constructed that uses the same dimensions and weighting as the document vectors. Searching is then done by looking for the most similar document vector. Similarity is defined through a correlation measure between the vectors. Frequently, the cosine of the angle between the vectors is used. In contrast to the Boolean search, where a document either matches or does not match the query, this similarity search can rank documents according to their relevance for the query. The vector model is currently the most-used retrieval model in general IR systems.

One commonly used weighting scheme is called TF/IDF (term frequency/inverse document frequency, cf. also Baeza-Yates and Ribieiro-Neto, 1999, 29–30):

$$\begin{aligned} \mathrm{tfidf}_{i,j} &= \mathrm{freq}_{i,j} \times \mathrm{idf}_i \\ \mathrm{idf}_i &= \log \tfrac{N}{n_i} \end{aligned} \tag{2.1}$$

$\mathrm{tfidf}_{i,j}$ TF/IDF measure for term k_i and document d_j.

$\mathrm{freq}_{i,j}$ Frequency of term k_i om document d_j.

N Number of document in the document collection

n_i Number of documents containing term k_i.

idf_i Inverse document frequency of term k_i.

The TF/IDF measure normalises the frequency of the term in the document through dividing it by the (normalised) ratio of documents containing this term in the document collection. Thus, the TF/IDF measure better captures the relative importance of a term within a document: Very frequent terms with a low distinguishing power receive only small TF/IDF scores and are thus less important for the search than terms that are less common in the document collection.

2.1.1.3 Adding Linguistic Information

Most current IR systems use additional linguistic processing steps for indexing and searching. We will describe the most important such steps here. While the ‘core’ IR system is language independent, these additional processing steps make an IR system language dependent (and, to a certain extent, domain specific). That means that only modules implementing language dependent processing steps need to be replaced when adapting an IR system to a new language or domain.

Stop-Word Removal. Most IR systems only use content words, both in indexing and in retrieval. Function words, such as prepositions or conjunctions, the so-called stop-words, are first stripped using language-specific lists of stop-words. Instead of using a pre-defined stop-word list, some implementations simply consider all words as stop-words, whose relative frequency exceeds a certain threshold.

The assumption is that stop-words have a very low discriminatory power (as they appear in many documents) and distort the search results when used. See, however, Riloff (1995), who proposes more fine-grained criteria for selecting stop-words.

Stemming. Different inflected forms of a word are used in documents.[1] When users use a word as a search term, they typically expect it to match all inflected forms. Most systems therefore use a stemming component that reduces all word forms to a word stem before document indexing. This can be an heuristic component, which cuts off all possible inflectional endings from a word (such as the well-know Porter stemmer, Porter, 1997). It can also be a more sophisticated, lexicon-based lemmatiser module, up to a morphology that can handle different ways of word formation (such as inflection, derivation and compounding, e. g., Koskenniemi, 1983).

Query Expansion. Another linguistic addition that many IR systems employ is query expansion (also called synonym search): For a user’s query, the systems looks up possible synonyms for each term in a list of synonyms. These synonyms are added to the query as alternative terms. Quite often, lexical databases such as WordNet are used for that purpose (4.2.1, Fellbaum, 1998c). Word similarities can also be automatically induced from a document collection based on word co-occurrences (Qiu and Frei, 1993).

[1]Only, of course, in inflectional or, *mutatis mutandis*, in agglutinative languages.

Query expansion can help to improve recall (i. e., more relevant documents are found), as the user does not have to use exactly the right term to find a document. However, if too general terms are added to a query, many irrelevant documents may be returned.

Other Processing Steps. Information Retrieval systems may employ additional processing steps, such as Named Entity Recognition (see also 2.1.3). Languages other than English may need to employ additional linguistic preprocessing steps (such as word boundary detection for East Asian languages).

2.1.1.4 Document Relevance

We will shortly touch upon the possibility of defining a query-independent notion of importance of a document relative to a collection. If every document in the collection receives an importance score during the indexing phase, this can be combined with the (query-dependent) document relevance score so documents that are both relevant *and* important are ranked highest.

One of the most successful measures of importance is the PageRank measure (Brin and Page, 1998, named after one of its inventors, Larry Page), which can be used on document collections that contain hypertext links between documents. PageRank was first used in the Google Internet search engine and forms one of the bases for its success. The core idea is as follows: Documents that are often cited, i. e., that have many incoming links, are likely to be more important than others. Every document receives a value, its PageRank, that mirrors its importance. The bigger a document's rank, the more of an authority this document is considered to be. Now, every document spreads its rank through its outgoing links. Therefore, a document cited by an authority receives a higher rank in turn. This spreading activation is computed for the whole network of documents until some measure of stability is reached.

2.1.1.5 Evaluation

For the evaluation of Information Retrieval systems, a number of evaluation conferences have been set up. The first and still the by far most influential of these is TREC (Text Retrieval Conference, `http://trec.nist.gov/`), a yearly series of shared-task competitions-cum-conference funded by the US government. It is conducted by the National Institute for Science and Technology (NIST) and was first held in 1992. Other important conferences in similar vein are TREC's European equivalent CLEF (Cross Linguistic Evaluation Forum, `http://www.clef-campaign.org/`, with a bias towards multilingual and cross-lingual systems) and its Japanese counterpart, NTCIR (NII-

NACSIS Test Collection for IR Systems[2], `http://research.nii.ac.jp/ntcir/`).

The shared-task competitions all use the same general procedure: All participants taking part in the evaluation receive a document collection and a set of test queries that must be evaluated over the document collection. As all participants receive the same data, the results are directly comparable. Shared-task competitions for QA systems are described in greater detail in 7.1.

2.1.2 Natural Language Interfaces to Databases

Natural Language Interfaces to Databases (NLIDB, often also called Natural Language Database Front-ends, NLDB) provide an intuitive and easy-to-use method to access databases. The user poses questions in natural language to the system, and the system provides answers from its knowledge base. For NLIDBs, the knowledge base is a structured database that is organised under technical rather than under natural language considerations. The targeted main user group are users with no expertise in database systems. These non-expert users are to be enabled to find and use information from the database without knowing anything about its structure or its workings. A good and still reasonably up-to-date overview can be found in Copestake and Sparck Jones (1990).

In the general case, it is assumed that the information to be searched is already present in an existing database. Automatically adding information from textual sources to the database was not one of the goals of this area of research. It was generally assumed that this could in principle be done by suitable Information Extraction systems (but see 2.1.3 for a less optimistic view of this issue). Some NLIDBs have also experimented with providing users with possibilities of not only searching information but also adding new data to the database by natural language assertions (e. g., Thompson and Thompson, 1983, 1985).

One famous example, which lead to high expectations in this area of applications, is the LUNAR system. Its knowledge source is a database of rock samples brought back to earth by the lunar missions. It was successfully demonstrated at a conference of lunar scientists and it was reportedly able to answer over 90 % of questions of scientists who had had no introduction to the system (Woods, 1977, 1973).

2.1.2.1 Mapping between Natural Language and Database

Natural Language Interfaces to Databases have to be tailored to the specific applications, at least to a certain extent. This high domain specificity usually

[2]NII: National Institute of Informatics, Tōkyō, Japan; NACSIS: National Center for Science Information Systems.

leads to a good coverage of both the vocabulary and the used queries. As the design of the database is known, this knowledge can be directly used in building the interface. The interface designer has the possibility of anticipating many different ways of referring to this information, both lexically and structurally. Put differently: The natural language coverage needed to search in the database can, up to a certain point, be foreseen and taken into account.

An NLIDB must provide a mapping from natural language questions to database queries. In early systems, this mapping was essentially hard-coded. It was, therefore, very difficult to extend the system coverage. Porting the system to new domains meant essentially that one had to rebuild the system more or less from scratch. Later systems tried to address this problem by employing intermediate (often logical) representations of the users' questions that were then mapped onto actual database queries. Ideally, in such a system the natural language processing modules are highly domain independent and only the actual mappings from the logical representation to the database have to be defined by a knowledge engineer.

Some advanced systems provided a special interface, with which the knowledge engineer (or even 'normal' users) could extend the system coverage (especially vocabulary) by teaching the system the meaning of each new word. This was done by answering a number of questions about it and linking it to the database design (e. g., Martin et al., 1986).

Several recent systems have experimented with automatically finding mappings from natural language representations to database queries.

The system described in Popescu et al. (2003) looks for matches of words in the natural language question in the whole database. Then, reasoning techniques are used to construct an unambiguous data base query.

In Frank et al. (2005, 2007), a system is described that can answer questions using either text documents, semi-structured data in XML documents or databases as knowledge base. Questions are translated into generic proto-queries that are translated into queries for the different 'back-ends' using meta-information about the way data is stored in the back-end combined with ontological information.

These automated techniques seem to work well for small databases. It remains to be seen, however, how well they scale up to larger databases.

2.1.2.2 Advantages

We will now shortly summarise the most important advantages and disadvantages of typical NLIDBs. This list follows Androutsopoulos et al. (1995). Note that NLIDBs are here mostly compared to querying database systems using a database query language such as SQL (Structured Query Language).

The most important advantage of NLIDBs in the 1970s and 1980s was that users were able to use natural language to formulate their questions instead of having to learn a database query language *and* the structure of the underlying database to use it. This has become less important, as current database systems come with graphical user interfaces, which allow users to easily access the data.

Natural Language Interfaces to Databases often allow the use of features of the underlying database such as complex table joins and the use of aggregate functions. Users can thus ask complex questions, such as *'List all countries that border countries that border Israel.'* or *'Who are the employees in departments whose manager earns more than 50,000?'* or (implicitly) use aggregate functions such as *'Which country has most Nobel prize laureates in literature?'*. The NLIDB uses the database to collect this information. In contrast, current textual QA systems can be used to find factive information that is expressed somewhere in their document collection, but not to gather information in a comparable way.

2.1.2.3 Disadvantages

As potential disadvantages of NLIDBS, Androutsopoulos et al. (1995) lists the following ones:

One problem of NLIDBs is a general problem of all natural language interfaces: It is quite difficult to explain to users the coverage and the limitations of a system. For example, a system may not be able to handle a user's input, even though only a slight re-phrasal would make the input acceptable.

The major problem with NLIDBs, however, is their low scalability and portability that was already mentioned above. For systems with a hard-coded mapping from natural language to database queries, one can hardly talk about scalability and portability at all: Every extension to the system must in turn be hard-coded, in general by a team consisting of linguists, knowledge engineers and database specialists.

But even for NLIDBs that are designed for scalability and portability (even the ones with an integrated 'knowledge acquisition' mode, see above), setting up the system for a new database, let alone a new domain may be a complex task.

Recognition of this fact has lead to a strong decrease in NLIDB research since the early 1990s. Early commercial NLIDB extensions for database systems were not successful and have since dwindled from use.

The recent research in systems that automatically find mappings between natural language concepts and database contents on the one hand and the integration and combination with different knowledge sources (2.1.2.1) may change that, however.

2.1.3 Information Extraction

Information Extraction (IE) is concerned with automatically extracting domain-specific information from text documents. This overview is based on Appelt and Israel (1999); Gaizauskas and Wilks (1998); Cowie and Lehnert (1996).

Research in IE has especially been focussed by the Message Understanding Conferences (or Competitions), MUC, a series of (mostly) biennial conferences, funded by the US agency DARPA, starting in 1987 and ending with MUC-7 in 1997. The MUCs were shared-task competitions were users could test and compare their systems against a given, previously unknown reference corpus.

In the original MUC scenario, an IE system would be used to watch a stream of information (especially news-tickers or message systems, hence Message Understanding which has sometimes been used as a synonym for Information Extraction). Information within the domain of the IE system was to be spotted and stored in a structured form using so-called templates. This information was then to be used either directly by the user (scan the filled templates for interesting information, then call up the original document) or as a means to directly feed databases, possibly in conjunction with an NLIDB system (2.1.2).

Information Extraction is generally taken to subsume a number of different tasks, namely Named Entity Recognition (i. e., automatically spotting the names of, say, people or firms), coreference resolution (i. e., correctly identifying antecedents for anaphora, 6.2.9) and, as the most advanced task, filling scenario templates that describe domain-specific types of events.

Two examples of domains and domain templates, as defined in different MUCs are the following: Changes of key personnel in companies (MUC-6, 1995, ‘succession events’ with slots like organisation, post, reason of change, and ‘in and out’, i. e., the persons involved) and Satellite launches (MUC-7, 1997, with slots like carrier vehicle, payload, mission date etc.).

In general, three basic approaches have been used in IE: The knowledge engineering approach, the automatic training approach[3] and the bootstrapping approach.

2.1.3.1 Knowledge Engineering Approach

In the knowledge engineering approach, a knowledge engineer uses some suitable formalism to manually encode mappings from text to entities and templates. This means that for every type of entity (such as a person name) or template (such as the satellite launch event mentioned above), possible textual

[3]These names are proposed by Appelt and Israel (1999)

fillers must be defined. Different systems have employed different kinds of linguistic resources (up to syntactic parsing, e. g., Yangarber and Grishman, 1997).

Depending on the linguistic resources used, the actual mapping to the templates use different representation levels as starting points, such as patterns over surface words in the basic case and (partial) parse tree descriptions when a full parse is available. In general, systems using only limited, highly reliable components have fared best at the MUCs (cf. the discussion in Appelt et al., 1995).

2.1.3.2 Automatic Training Approach

Systems using the automatic training approach use machine learning approaches to automatically induce entity and template mappings from corpora annotated for the structures that the final system is expected to produce (e. g., Soderland, 1999; Miller et al., 1998). The systems induce patterns from the annotated examples automatically. These patterns are either defined in terms of surface text words or, in more sophisticated systems, through patterns on syntactic structures derived from the input by parsing. This approach has the main advantage that no knowledge engineering is required to implement the patterns for a given domain, as these are derived from corpus examples automatically. On the other hand, large amounts of consistently annotated data is required. Annotating such a corpus is expensive and time consuming.

2.1.3.3 Bootstrapping Approach

To avoid the necessity of annotating large corpora, bootstrapping has been used for named entity recognition and also for relation learning. Bootstrapping approaches employ large, unannotated corpora of text and a very small number of seed words (or seed relations) for each required category. Examples of systems can be found in Moschitti et al. (2003); Ghani and Jones (2002); Riloff and Jones (1999); Agichtein et al. (2000); Agichtein and Gravano (2000).

The bootstrapping algorithm proceeds by first identifying possible extraction patterns in the context of the seed words. If, for example, the algorithm is given *‘Washington’* as a location seed word, it may come up with a pattern like *‘headquartered in X’* as a candidate extraction pattern for locations if the phrase *‘headquartered in Washington’* is contained in the corpus. The candidate extraction patterns are then used to identify new candidates for the required named entity type. These are, in turn, used to identify new candidate patterns and so forth. This process is called mutual bootstrapping. In evaluations, this technique works well for relatively clear-cut, general categories such as location, but far less so for specialised types. In addition, these techniques currently handle only entity and (two-place) relation extraction but not more complex templates such as full event templates.

2.1.3.4 Discussion

The first two approaches share a common disadvantage: Scaling the system and porting it to other domains is relatively complicated. In the knowledge engineering approach, a large number of new rules must be written and tested. Only a small number of general rules will carry over, most others need to be newly implemented. Corpus-based systems need to be retrained for every change in the templates and/or domain. Of course, a suitable annotated corpus must be available that reflects the required changes. Bootstrapping approaches can partly solve this problem for some tasks but not yet for the full-fledged scenario template filling task.

In addition to this disadvantage, the scenario template filling task has proven to be far more difficult than first expected. Part of the difficulty may be due to the fact that it is very hard to rigorously define the task: In experiments for corpus annotation, inter-annotator agreement for human annotators for scenario templates has been as low as 60 %. The best systems in last MUC conferences have reached about 60 % of the human performance (Appelt and Israel, 1999).

This is certainly not sufficient for the scenario sketched above where the results from the IE system would be directly used to feed a domain database (cf. Appelt and Israel, 1999, 5). This conclusion has lead to a decrease of interest in stand-alone IE systems. Currently, IE systems are often regarded as one module to be used in larger natural language processing systems, such as QA or IR systems, to take care of, for example, Named Entity Recognition (see also 6.2.5).

2.2 Question Answering

Question Answering systems search for answers to users' questions in document collections and output concise answers. In this section, we will first present an overview of the development of QA systems from early systems over current systems to future directions (2.2.1). We will then describe the architecture of a generic QA system (2.2.2). Most current QA systems more or less use this general architecture. We will highlight on some interesting techniques that have been used.

2.2.1 General Overview

In this section, we will give an overview of the development of Question Answering as an area of research. We start with early systems, describe current systems, which are mostly research systems, and conclude with a discussion of directions for future research as described in different road-map papers. This overview is based on Hirschman and Gaizauskas (2001).

2.2.1.1 Early Systems

Asking questions and answering them is used in human communication as an important means of requesting and sharing information (cf. 3.2). Thus, being able to put questions to a computer has always been seen as a natural way of interaction. This lead researchers in computer sciences to build first computer systems for answering questions in natural language as early as 1960 (Hirschman and Gaizauskas, 2001).

The early systems did not extract information from document collections, but were in fact either NLIDB systems such as LUNAR (cf. 2.1.2) and LADDER (Hendrix et al., 1978) or used manually coded information as knowledge base (e. g. Lehnert, 1978, see also 3.3.1)

One of the first QA system that searches for answers in a document collection was MURAX (Kupiec, 1993). It used the electronic version of an encyclopedia to answer question from the quiz game Trivial Pursuit and was already designed along the lines of the generic QA system described in 2.2.2.

In 1999, TREC (cf. 2.1.1.5) for the first time ran a separate evaluation of QA systems, the QA track (Voorhees, 2000). This lead to a great deal of interest in the area of QA. Among the spurring influences, the shared-task competitions, and especially TREC, cannot be over-estimated. The general design of the shared evaluations has remained the same: Participants are given a document collection and a list of questions. They use their QA system to answer the questions and hand in the results for comparative scoring (cf. 7.1.2).

At first, QA was perceived as a subtask of IR: QA systems in the TREC QA track had to return snippets from documents containing an answer for a given question. These snippets were 250 bytes, then 50 bytes long. Only in 2001, the current form that requires exact answers was established. Now, system answers are marked as inexact (and thus not scored as correct) if they contain anything but the shortest possible answer.

The different competitions in IR mentioned above (2.1.1.5) now all run one or more subtasks in QA (often called ‘tracks’, such as the Question Answering Track at the TREC or the QA@CLEF track). These competitions have lead to a large number of systems being built, most of them research systems that are being developed at universities and research institutions.

2.2.1.2 Current Systems

Most current QA systems focus on a subtype of questions called ‘factoid questions’, that is, questions with fact-based, short answers. This is the main question type used in the competitions. Some examples are the following: *‘What is the capital of Madagascar?’*, *‘How did James Dean die?’* and *‘What is the Crips’ gang colour?’*. Recent experimental systems return more complex answers for definitional questions such as *‘What is the Hajj?’* (Blair-Goldensohn et al., 2004).

A standard architecture for QA systems comprising a number of components has emerged. Most systems can, more or less directly, be described in the terms of this standard architecture. This architecture is described below (2.2.2).

The best current systems now answer 60 % to 80 % of all questions in the TREC QA track correctly (Harabagiu et al., 2006; Sun et al., 2006; Clifton and Teahan, 2005; Cui et al., 2005), see also 2.2.3 and 7.1.2. However, this applies for the three best-performing systems; most participants reach 20 % to 35 % accuracy.

In spite of the good performance of the best systems in the competitions, everyday use of QA systems remains limited. Ulicny (2004) lists, for example, a number of QA systems, both for Internet and Intranet searches. These are, however, usually pilot installations. None of the systems is currently widely used.

This is certainly, at least partly, due to the fact that IR systems, especially the Internet search engines, today do a very reasonable job at document retrieval. Gregor Erbach has shown in a recent study that experienced users will be able to answer questions from the 2003 QA@CLEF competition using Google as search engine in a average of 70 to 80 seconds (Erbach, 2004). A recent user study even seems to suggest that users may be more interested in paragraph retrieval than in factoid QA (Lin et al., 2003).

2.2.1.3 Current Research and Future Directions

Directions for future research in QA have been set out in several road-map papers (Burger et al., 2001; Maybury, 2002, 2004b; Nyberg et al., 2004). We will list three important points, as they throw a light on current system performance. Most of these issues are currently being especially addressed in the context of the DARPA AQUAINT program (Advanced Question Answering for Intelligence, `http://www.nbc.gov/aquaint.cfm`).

The first research issue is that of handling more complex questions and answers. Current systems are usually focussed on factoid questions and short answers. They can typically not handle questions that require complex answers. Questions like *‘How is X done?’*, *‘Why did X happen?’* or *‘What would happen to X if Y happened?’* usually require the fusion of different information ‘nuggets’ and the generation of a coherent and complex answer. While this type of questions has been perceived as being of great interest to users, research has only begun (Narayanan and Harabagiu, 2004a,b; Voorhees, 2003; Burger et al., 2001). We will return to this type of questions below (3.2.2.5, 8.3.4).

The second research issue is that of improved user modelling, both during a QA session and across sessions. Using adaptive user models and reasoning about users’ goals could help to filter answers that do not fit the users’ expectations and goals (Harabagiu, 2006; Strzalkowski et al., 2006; Burger et al., 2001, see also 3.2.3).

A third research issue is that of interactive QA (Burger et al., 2001). An interactive QA system allows an information seeking dialogue, where the user can ask follow-up questions. In follow-up questions, anaphora and ellipsis are frequently used and must be resolved. Besides, a fully interactive system could ask clarification questions to resolve ambiguities in the user’s questions and suggest additional interesting topics (Strzalkowski et al., 2006; Harabagiu, 2006; Harabagiu et al., 2005).

The TREC 2004 conference has made a move towards a more dialogue-oriented system for the first time by using topics with four to eight questions rather than single questions, where only the first question actually mentions the topic while the other questions use anaphoric references to it. So far, the participating systems have mostly used simple techniques for resolving the anaphora such as simply replacing all pronouns with the original topic (Voorhees, 2005).

Our system also implements anaphora and ellipsis resolution for questions. This is described in 6.4.4.

There have also been first experiments with using speech input for interactive QA systems (op den Akker et al., 2005; Hori et al., 2003). This offers interesting new perspectives as it would allow very efficient interaction with the QA system. The experiments showed, however, that the error rate of Automatic Speech Recognition (ASR) systems is currently too high for efficient open-domain QA (Hori et al., 2003).

2.2.2 Technical Overview

In this section, we will describe the architecture and the components of a typical QA system (cf. Hirschman and Gaizauskas, 2001). Figure 2.1 shows the generic architecture of a QA system.

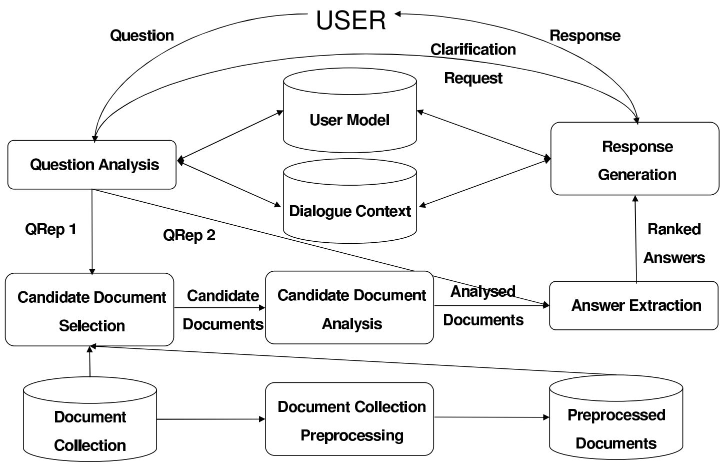


Figure 2.1: Overview of a Generic QA System, Adapted from Hirschman and Gaizauskas (2001, 286)

Virtually all current QA systems rely on IR techniques for searching. They use IR systems (2.1.1) to search their document collection for documents likely to contain answers to a users' question.

Questions are analysed to extract keywords (question analysis). These keywords are used as query terms for information retrieval (candidate document selection). The documents returned by the IR engine are then further processed to find the actual answer to the question (candidate document analysis, answer extraction). Different techniques are used for this answer extraction, from simple pattern matching over matching syntactic representations to semantic inferencing.

We will now describe the different processing stages in more detail. We will focus on typically used techniques, but also list some interesting additional ones.

2.2.2.1 Document Collection Preprocessing

As a first step, the document collection that is to be searched for answers is preprocessed. This often includes tasks like converting the documents into a standard input format. The normalised document collection is then indexed by the IR engine that is later employed to find candidate documents. Indexing often includes additional processing steps like stop-word removal and stemming (cf. 2.1.1).

Some recent systems run additional preprocessing steps to mark certain structures in the document collection before indexing. Especially Named Entities Recognisers are often employed to mark, for example, names of persons and organisations or dates (e. g., Bouma et al., 2006; Neumann and Xu, 2003). By adding these named entities to the IR module's index, finding relevant documents for questions containing them is facilitated.

Several current systems use not only an IR module for finding documents, but additionally use IE techniques to extract certain facts with high precision from the document collection and store them in a separate fact database (Prager et al., 2006; Bouma et al., 2006; Cui et al., 2005; Clifton and Teahan, 2005; Fleischman et al., 2003a). TREC has, for example, in the past years, quite often asked for the date or the manner of a person's death (such as the TREC 2001 question: *'How did James Dean die?'*). Information extraction systems are trained to find information that forms possible answers for such typical questions with high precision. When the QA system encounters a question that fits such a pattern, an answer is first searched in the fact database. If the answer is found, this answer is output, as it is likely to be correct. Only if the lookup fails, the system falls back the standard answer searching in all documents. Systems using such additional fact bases are among the currently best-performing QA systems in the competitions.

2.2.2.2 Question Analysis

The user's questions to the system are analysed linguistically. Most systems use a module for question type recognition based on the syntactic structure and on the semantic type of the expected answer, often utilising WordNet information. Question analysis is used for two purposes: On the one hand, keywords are extracted from the resulting structures, which are used as the query terms for retrieving candidate documents from the document collection. On the other hand, the analyses are used during answer extraction from the candidate documents (see below).

An elaborate example of such a question type analysis is described in Hovy et al. (2002a). It uses a question-type hierarchy based on the evaluation of some 17 000 natural language questions. This fine-grained hierarchy is especially important for answer extraction. It allows, for example, finding PERSON as the answer type of the question *'Who discovered America?'* and WHY-FAMOUS as that of *'Who was Christopher Columbus?'*, respectively. Thus, a named entity such as *'Christopher Columbus'* is acceptable as answer for the former, but not for the latter, which rather expects a definite NP such as *'the discoverer of America'*.

A number of systems not only extract keywords from the question analysis to use as query terms, but also further modify the query according to the question type (Bouma et al., 2006; Monz, 2003a,b; Agichtein et al., 2001). On the hand, this can be done by adding supplementary query terms. For example, adding unit terms like *'metre'* or *'kilometre'* to the query for a *'how long'* question, will ensure that only documents containing some measure are found. On the other hand, weights can be set for the different query terms. This can be used to ensure that query terms that have been identified as central to the question are actually present in the retrieved documents.

2.2.2.3 Candidate Document Selection

The query generated from the question analysis is sent to the IR module, which returns the candidate documents best fitting the query. These documents are most likely to contain an answer to the question. Only the candidate documents are then processed using deeper methods to pinpoint the actual answer within the document. This step is also called document pre-fetching.

The linguistic enhancements used in IR that were mentioned above (2.1.1), especially query expansion, are usually also applied in the candidate document selection. Quite often, only passages likely to contain are retrieved rather than whole documents. This minimises the amount of processing to be carried out by the subsequent analysis steps. An overview of different passage retrieval techniques in QA is given in Clarke et al. (2006).

Recently, a number of QA systems have successfully made use of external knowledge sources (especially the Internet) for finding answers (Kaisser and Becker, 2005; Katz et al., 2005, 2004; Neumann and Xu, 2003; Buchholz and Daelemans, 2001). In general, this works as follows: Based on the question type analysis, keywords from the question are used to construct a query that is evaluated on the external knowledge source.

Most systems use the Internet as a source of external knowledge. They construct a number of high-precision patterns from the question and send them to Internet search engines, such as Google. For example, for the question *'When did James Dean die?'*, a pattern like *'James Dean died in *'* might be constructed. It is quite likely that a number of matches for such a pattern can be found somewhere on the Internet. Marc Light and his colleagues have shown in a study that if answer redundancy can be exploited, i. e., several possible answers can be investigated, more questions can be answered correctly (Light et al., 2001).

Other systems have experimented with using Internet databases, such as the Internet Movie Data Base (`http://www.imdb.org/`). Using a suitable interface, information like who directed a certain film can be found with high precision, as the database provides structured, table-like information.

The answers retrieved from the external sources is then combined with the query terms constructed from the question to identify documents in the document collection that contain the answer. This has been called answer projection.

This method generally increases answer recall and precision: On the one hand, an answer is more likely to be found, on the other hand, the correct answer can be more easily be identified if the system can rely either on redundancy or on structured information.

2.2.2.4 Candidate Document Analysis

Candidate documents retrieved by the candidate document selection step are often linguistically analysed as a preprocessing step for answer extraction.

Most systems run a Named Entity Recognition (NER), marking person and organisation names, dates etc. (e. g., Srihari and Li, 2000). Named entity recognition is often complemented by other IE techniques, such as relation extraction or recognising predefined events (Srihari et al., 2006).

Recently, this has been complemented by a fuller analysis, leading to complete syntactic structures and sometimes also predicate-argument structures

(Bouma et al., 2006; Katz and Lin, 2003; Van Durme et al., 2003; Kawahara et al., 2002; Elworthy, 2001; Harabagiu et al., 2001b; Hovy et al., 2001).

2.2.2.5 Answer Extraction

Using the results of the document analysis and of the question analysis as a basis, this step searches actual answers in the candidate documents. Many different techniques have been used for answer extraction. We will give an overview of the most important ones (see also Echihabi et al., 2006). Note that, generally, not a whole answer sentence, but only an answering phrase will be extracted, for example, a person name. Issues of answers and answering phrases (constituent answers) will be discussed in greater detail below (3.2.1).

Answer extraction has often been done by identifying a named entities in the candidate document whose type corresponds to the expected answer type (e. g., Abney et al., 2000). The expected answer type is identified from the question (cf. 2.2.2.2). It might be, for example, a person name or a date. Named entities are then marked during candidate document analysis. From these named entities, one that fits the expected answer type is selected as the most likely answer candidate, based on a ranking algorithm. Marc Light and his colleagues have shown that this approach will run into problems quite often, as candidate documents are likely to contain more than one entity of the expected answer type, making the decision difficult (Light et al., 2001).

A second approach to answer extraction associates question types with lists of answer patterns. These patterns can be quite complex. For example, an answer to question like *‘Who is the prime minister of X?’* may be found using a pattern like `[country name] "’s" [title] (CapWord CapWord)`. The answer patterns can be manually compiled, based on corpus inspection (Soubbotin, 2002; Soubbotin and Soubbotin, 2003). Patterns can also be learned automatically using large corpora and seed patterns (Echihabi et al., 2006; Ravichandran and Hovy, 2002).

Several systems have employed syntactic structures of questions and answers derived through parsing as a basis for identifying the answer (Bouma et al., 2006; Katz and Lin, 2003; Kawahara et al., 2002; Elworthy, 2001; Harabagiu et al., 2001b; Hovy et al., 2001). Different methods have been used for matching question and answer representations. Often, triples describing syntactic relations of the form <Head, Grammatical-Function, Dependent> are used for matching (cf., e. g., Katz and Lin, 2003). The more such triples match between question and answer, the better the answer candidate is. The answer phrase can be identified by using a ‘wild-card’ that matches anything to represent the question word.

One QA system has very successfully used inferencing over logical forms to find answers (Harabagiu et al., 2006; Moldovan et al., 2003b,a; Harabagiu et al., 2001b; Moldovan and Rus, 2001). It uses a full probabilistic parser to derive predicate-argument structures (called Logic Form by the authors) from both the question and from candidate passages retrieved by the previous processing steps. These logic forms are fed into a theorem prover. Linguistic knowledge is provided to the theorem prover as a set of axioms. These are mostly derived (semi-) automatically from WordNet by parsing the glosses given there in natural language and transferring them into Logic Forms (Extended WordNet, see also 4.2.1.2). In addition, so-called natural language axioms have been hand-coded. They axiomatise, for example, that both NP heads in an apposition share the same argument, viz. that they refer to the same object. If the logic prover can find a valid proof, the candidate passage contains a valid answer to the question and the answer phrase can be extracted. The idea of using automated reasoning for answering questions will be described in more detail in 3.3.

2.2.2.6 Response Generation

The best answer found by the previous processing steps is returned to the user. Most systems present a snippet from the text based on the results of the answer extraction. Only a few systems actually perform a linguistic answer generation (such as Vargas-Vera and Motta, 2004).

2.2.3 Discussion

In this section, we have given an overview of QA systems. We have shown the development of the research area and described the most important techniques used in QA.

Work on Question Answering has focussed on different aspects of this complex problem (cf. Nyberg et al., 2004):

Especially in the first QA tracks, QA was often treated as a special form of IR: Instead of returning documents or passages, researchers hoped to improve IR methods so that they would be able to pinpoint not only passages, but rather single words as answers to questions (Cormack et al., 2000).

An alternative approach was to rely on IE techniques, especially for answer extraction: By marking NEs in candidate documents and matching the expected answer type against the NEs, exact answers can be found (Abney et al., 2000).

The currently best-performing systems combine IR for candidate document selection with Natural Language Processing techniques, especially using syntactic structures in matching (Harabagiu et al., 2006; Cui et al., 2005).

Other successful approaches have used external knowledge sources, either the Internet or (semi-) structured data sources. By searching answers on the Internet, they have made use of answer redundancy: Finding an answer on the Internet is more likely than on a limited document collection. Besides, the correct answer is usually found more often than spurious answer, allowing to focus on frequent ones (Kaisser and Becker, 2005).

By searching answers in structured knowledge bases such as the Internet Movie Database, systems have exploited the structure of the database to improve precision (Katz et al., 2005, 2004).

We suggest an approach to QA that uses structured linguistic information throughout, especially for answer searching. We will further motivate this approach in the following section (2.3) and develop it in the following chapters.

2.3 Research Questions

In the previous section, we have described the components used in a typical QA system. We have shown that most current systems use techniques from IR for searching. Linguistic methods are mainly used for question analysis and during answer extraction. Most systems use relatively shallow methods, like Named Entity Recognition and pattern matching.

In this section, we will discuss potential shortcomings of ‘information-poor’ approaches to QA. We identify low precision as an important potential problem: A QA system using little or no linguistic information will often return wrong answers. We suggest that by systematically using structured linguistic information as a basis for answer searching, this problem can be tackled. We will develop linguistically informed QA from this starting-point in the following chapters and show that it can be practically implemented.

However, using more constraints in answer extraction will potentially hurt answer recall and increase answering times, if additional linguistic processing has to be performed during candidate document processing. These possible drawbacks must therefore be taken into consideration. This leads to the question if efficient methods for answer searching can be developed that work without an IR module for document pre-fetching.

2.3.1 Improving Answer Precision

We note that IR systems work well with limited linguistic information, especially stop-word removal and stemming (cf. 2.1.1). This is demonstrated by the success of Internet search engines. It has proven hard to improve IR systems by adding further linguistic information.

Does that also hold for QA systems? There is general agreement that finding answers (instead of documents or passages) requires additional linguistic information (e. g., Srihari et al., 2006; Hovy et al., 2001). The retrieval models of IR systems work well for retrieving documents or passages, but pinpointing exact answers is beyond their scope. The discussion of linguistic phenomena related to questions and answers (chapter 3) will show that identifying answers for questions is a complex linguistic task. Approximating it without sufficient linguistic information, especially structural information, is therefore not feasible.

We have described above (2.2.2.5) that most QA systems do not use exact matching of questions and answers, but rather some 'loose' matching techniques. The simplest ones are based on identifying a named entity of the type expected as an answer and pattern matches. Thus, the approaches have no or only little information about linguistic structures. They can therefore not distinguish whether words in the question and in the potential answer stand in the same linguistic relation. Consequently, a potential problem of these approaches is that of low precision, i. e., they return wrong answers that seem to match the question, but in fact provide no answer.

Consider the following example. It was taken from a website[4] where Ulf Hermjakob has collected several examples of funny wrong answers of the Webclopedia QA system.

(2.1) Where do lobsters like to live?
on a Canadian airline
[The answer was extracted from the following text:] First-class passengers **on a Canadian airline** will have to **live** with **lobster** and filet mignon now that pate de foie gras is being dropped from the menu, an animal rights group said.

Similar examples of precision errors are listed in Katz and Lin (2003). The authors discuss unrestricted matching of questions and answers as a source of precision errors and suggest that these can be solved by selectively using syntax structures of questions and answers.

Taking a look at the results from the QA track at the TREC shared competition also supports the impression that precision problems are not generally solved: TREC (2004a) lists the judgements for all answers to factoid questions returned by the participating systems. Of all answers, 77 % were judged to be wrong.

[4]`http://www.isi.edu/natural-language/projects/webclopedia/humor.html`

```
1990s
Albanians
BLOODS
Bateau Ivre
Blood (2 times)
Bloods (5 times)
Crip
Crips
Heard
Nickelodeon
Stanley ‘‘ Tookie ’’ Williams
Vice Lords
Vision
Visitors to the FlyPlaya Website will see the words ‘C’z
Young Gangster
a bright red Mickey Mouse sweatshirt
a long time
gang
many things
members
named Crip
the 14th century B.C
xxxxxxxxxxxxx
```

Figure 2.2: Answers Returned by Participating Systems for the TREC 2004 QA Track Question 1.2: ‘*What does the name [Crips] mean or come from?*’

When further looking at the different answers, it is also interesting to see that answers often do not seem to have anything to do with the question. As an example, consider fig. 2.2 that shows the answers returned for Q 1.2 in TREC 2004: ‘*What does the name [Crips] mean or come from?*’

The first research question of this thesis is therefore, how answer precision for QA systems can be improved by using structured linguistic information.

We suggest that linguistically informed QA can be used to address this problem. It uses syntactic structures of questions and answers and identifies answers by matching such structures.

2.3.2 Maintaining High Answer Recall

Using additional linguistic constraints during answer extraction will potentially hurt recall, i. e., fewer correct answers will be found: If the structure of question and potential answer differ, the answer will be disregarded, even though it is in fact correct. This is discussed in Katz and Lin (2003).

Therefore, the second research question is whether a method can be found that improves answer precision without unduly reducing answer recall. We suggest that this can be done by using a combination of different linguistic resources during answer searching (chapters 5, 7).

This combination of linguistic resources provides a handle to overcome a dependency on answer redundance: If the system can identify an answer even though it differs in wording from the question through the systematic use of combined linguistic information, it does not have to rely on finding this answer several times and in different formulations (or in structured external databases). This is especially interesting for small document collections or special interest questions.

2.3.3 Answering Times

A second consequence of using additional linguistic information is that it will lead to an increase in processing time. Techniques like Named Entity Recognition or pattern matching can be implemented very efficiently. Full syntactic parsing, however, is more time consuming.

If linguistic processing steps are carried out during candidate document analysis, this may lead to unacceptably long answering times of the system. This will especially be a problem if a QA system is to be used for on-line processing, in particular for interactive QA.

A third research question is therefore if and how linguistic processing can be shifted into the document preprocessing phase. By performing as much of the linguistic processing off-line, storing the results and using these stored representations for the on-line question answering task, answering times that are suitable for interactive QA systems can be reached.

2.3.4 Efficient Structured Searching

This leads to the fourth research question. As described above, virtually all current QA systems use an IR module for retrieving candidate documents from their document collection before extracting answers from these candidate documents. This means that questions must be translated into a keyword-based query to an IR system. As mentioned above, this translation is far from simple (see, e. g., Bouma et al., 2006; Monz, 2003a).

The fourth research question, then, is whether it is possible to use a more direct method of answer searching and thus do without an IR module for answer searching. Ideally, such a search method would make use of linguistic structures derived from the document collection in off-line processing.

We will develop a method that allows to compute linguistic representation of the texts in the document collection off-line, store them in a database and efficiently retrieve structures that represent potential answers to a given question (5).

2.4 Conclusions

In this chapter, we have introduced different types of information access systems. The focus was on Question Answering systems. We have also described Information Retrieval systems, Natural Language Interfaces to Databases and Information Extraction systems and shown the distinctions between the different system types.

We have shown that current QA systems typically use IR engines for searching candidate documents and that linguistic information is mostly used in question analysis and in answer extraction. However, often only relatively shallow methods are used, and then only selectively.

We have suggested that using structured linguistic information throughout the question answering process can help to overcome the potentially low precision of ‘knowledge-poor’ approaches.

In the following chapter, we will develop linguistically informed question answering, starting from a summary of phenomena related with questions and answers in human interaction. We derive a notion of indirect answerhood as a means of describing the relation between a question and a text that contains an answer. This relation of indirect answerhood is then further characterised with examples from a corpus study of questions and answers. It will be formally specified in chapter 5.

Chapter 3

Linguistically Informed Question Answering

In this chapter, we will outline linguistically informed Question Answering as an approach to Question Answering. The focus of our attention will be on developing a linguistically motivated method for finding answers in text. This method will be based on indirect answerhood, a relation that holds exactly between questions and texts that contain a (possibly indirect) answer to them.

In 3.1, we will sketch the framework for linguistically informed QA: We will characterise a linguistically informed QA system, especially with respect to a usage scenario. So far, no full, worked-out description of a framework for QA exists in the literature (cf. De Boni, 2004).

In 3.2, we will investigate work on the linguistics of questions and answers from a syntactic, semantic and pragmatic viewpoint. This rich body of literature has not yet been systematically related to QA systems. We identify a number of directly relevant findings on the nature of questions and especially on what is expected as an answer to a certain question and integrate them into our framework. We focus on semantic approaches to questions and answers and especially the concept of indirect answer (an answer that does not directly answer a question, but requires additional inferences, Higginbotham and May, 1981). We will use this idea as the basis for our method of finding answers in text collections: As the text collection of a QA system will almost never contain direct answers to users' questions, further inference steps will, in general, be needed.

In 3.3, we discuss the question answering based on structured semantic information. While semantic approaches to describing questions and answers (especially the partition approach, Groenendijk and Stokhof, 1997) would provide a good theoretical basis for QA, they cannot directly be utilised to build actual systems: We show that approaches to QA based on full semantic representations in some logical language (e. g., Burhans, 2002) are not suitable for large scale QA applications (Marcu and Popescu, 2005).

In 3.4, we examine work in the recent PASCAL Recognising Textual Entailment Challenges (Dagan et al., 2005). We find that its concept of textual entailment is partly relevant to QA, but that does not carry over directly.

In 3.5, we sketch indirect answerhood as a relation between texts and questions. In order to get an overview of possible differences between questions and indirect answers that need to be accounted for by the relation of indirect answerhood, we conducted a corpus study of questions and texts containing answers from past QA competitions. We identify a number of syntactic and lexical semantic variations as sources of inferences and give examples. We also show examples of complex inferences that cannot be modelled by indirect answerhood.

In chapter 5, we will define a relation of indirect answerhood between structured linguistic text representation that approximates the relation of indirect answerhood between texts specified in this chapter.

3.1 Developing a Framework for Question Answering

In this section, we will first motivate the development of a framework for QA. We will then informally describe the overall setting of linguistically informed QA and derive a first list of desiderata that will go into the framework.

3.1.1 Motivation

While Question Answering has received considerable interest over the past few years, with several dozens of implemented QA systems worldwide, there has been little work devoted to the theoretical foundations of QA, especially concerning its linguistic aspects. The evaluation framework that has been defined for the first QA track at the TREC conferences (Voorhees, 2000) is generally used as a basis and is often referred to as the ‘standard’ framework for open-domain QA. This seems somewhat surprising, given that the TREC QA framework’s definition is only rather sketchy and that it was defined as a framework for evaluating QA systems as a shared task (with comparability of the results as main object, Voorhees, 2001).

In contrast, there exist – at least semi-formally – worked-out models for information retrieval (cf. Baeza-Yates and Ribieiro-Neto, 1999, see also 2.1.1). Recent, more theoretically oriented work on QA has only produced partial models (De Boni, 2004; Burhans, 2002).[1] There are, of course, generic descriptions of QA systems (Hirschman and Gaizauskas, 2001) and a large number of individual system descriptions (for example in the TREC proceedings, cf. also 2.2.2). But so far, there exists no more formalised description along the lines of the IR model (Baeza-Yates and Ribieiro-Neto, 1999).

This lack of a formal background in QA has been pointed out by Marco De Boni in his recent Ph. D. thesis (De Boni, 2004):

> [T]here is a significant lack of theoretical understanding of QA systems and consequently a considerable amount of confusion about their aims and evaluation. [. . .] The study of question answering systems appears to have made slow progress and current QA systems are no more than prototypes, being able to answer only the simplest of questions . . . and even then only with low accuracy. [. . .] While there is agreement amongst researchers on the generic aim of QA systems (presenting an *answer* to a *question* as opposed to *a set of documents* associated with a *query*) little work has been done on clarifying the problem beyond the establishment of a standard evaluation framework for QA, the Text Retrieval Conference (TREC) QA track. . . [. . .] Another limiting factor has been that most current research has either aimed at solving the engineering problem of building and improving systems capable of achieving high scores within this framework without questioning the solidity of the framework, or in looking at "future directions" but without having clarified the problem setting and what directions this should lead to. (De Boni, 2004, 14–15, his emphases)

While we would consider De Boni's rather scathing criticism of the research in QA exaggerated, we would still agree with him that some effort needs to go into fundamental research in QA. We think that it is worthwhile looking into research in related areas, especially the research on questions and answers in linguistics. Integrating a number of findings from different research directions into one framework will help to develop a better understanding of QA.

In this chapter, we will describe the framework for linguistically informed QA that we develop in this thesis. As the starting-point, we investigate literature on phenomena related to questions and answers, especially in syntax and

[1]Wendy Lehnert's early work on QA (Lehnert, 1978) provides a number of insights, especially an often-cited question typology (cf. 3.2.3.1). It is, however, centred around the idea of modelling a cognitive process (couched in terms of Roger Shank's scripts) so that they only partly generalise.

lexical semantics. We identify the relation of indirect answerhood that holds between a question and a text that contains an answer as the key concept for finding answers in documents in linguistically informed QA. In chapter 5, we show how indirect answerhood can be approximated as matching syntactic dependency structures extended with lexical semantic information. We thus treat QA as a challenge for computational linguistics here, rather than one of information retrieval.

This is different from the approach taken by many researchers in QA: In our overview (2.2.1), we have shown that – after much initial interest from computational linguistics in the 1970's –, techniques from IR complemented by some shallow natural language processing modules such as Named Entity Recognition have been used in most QA systems. Only recently advanced linguistic methods and resources have (re-) appeared on the scene.

We will be little interested in issues of user modelling. While we will show that these are important issues that need to be addressed in a 'perfect' QA system (as, e. g., the *goals* of the user should be taken into account when presenting even relatively simple answers, cf. 3.2.3), these issues are beyond the scope of our current work. We will come back to this point in some more detail in 3.2.4.

We will mainly focus on what has been called traditional, 'factoid' QA. Our impression is that, in spite of the success rates in the latest TREC conferences, it is still worth investigating the underlying issues, especially the linguistic bases, for current QA systems. Under the auspices of DARPA, research in QA is currently moving towards more complex tasks (as described in 2.2.1.3). As these efforts build on current, TREC-style systems, we think that it is still important to better understand the bases.

3.1.2 Setting the Stage

We will build up our description of a framework for QA in several steps. We start by a initial description of the scenario and setting. As a starting point, we have chosen the following definition of QA by Mark Maybury:

> *Question Answering* (*QA*) is an interactive human computer process that encompasses understanding a user information need, typically expressed in a natural language query; retrieving relevant documents, data or knowledge from selected sources; extracting, qualifying and prioritizing available answers from these sources; and presenting and explaining responses in an effective manner. (Maybury, 2004b, 3, his emphasis)

Maybury's definition is rather broad, so we will first specify a number of points more closely. Let us start by defining the underlying scenario in which the QA system is employed.

In our scenario, a user with an information need interacts with the QA system to (partially) satisfy that information need. The interaction is limited to a terminal-style exchange, i. e., the user types her/his input to the system by keyboard, while the system's responses appear on screen. Both the user input and the system's responses are formulated in natural language. The user input will consist of questions and the systems responses of answers to these questions. The QA system uses a document collection to find suitable answers for the user's questions. Upon request, the QA system can display the original document from which it has drawn some answer and also a justification that describes why it considers its response a correct answer to the user's question. The interaction may take the form of an information seeking dialogue where the user asks a number of questions relating to different aspects of a topic. We will further clarify a number of these points, especially by pointing out what our framework does *not* include.

We will now summarise a number of additional assumptions that we make.

Textual Question Answering. We will only be interested in textual question answering. We do not mean to imply by this that we do not consider QA on other media than text a possible application of question answering or that it is any less interesting. We simply mean to focus our description. There is no reason why it could (and should) not suitably be expanded. Note that this especially means the following sub-issues:

- Spoken Language Input (cf. also 2.2.2)
- Non-textual documents (e. g., graphics, movies, sounds etc.)
- Spoken Language Output
- Non-textual output (e. g., graphics, graphs, tables etc.)

Document Collection. A collection of text documents forms the main source from which the QA system can draw information. We will for now assume that this collection is static and does not change over time. That means that no new documents are added to the collection and that the user cannot select only parts of the collection. We will further assume that the collection contains only text documents that are written in one natural language (such as German). Thus, our framework will only describe monolingual QA.

User. A user interacts with the QA system in the form of a query session. Such a query session consists of a sequence of questions that the user puts to the system and the system's replies that together form a dialogue. We assume that the user has an information need, i. e., that (s)he lacks certain information and wants the system to provide this information. Thus, we follow Wendy Lehnert's implicit principle for QA:

> Questions are asked to draw attention to gaps in the questioner's knowledge state; questions are answered to fill those gaps. (Lehnert, 1978, 254)

This principle entails that exam questions (where one would like to assume that the examiners know the answers to questions) or a system evaluation are not really covered by the framework. We will assume (again following Lehnert, loc. cit.) that the principle still applies, thus making no practical difference.

Questions. The user enters questions expressing (parts of) her/his information need in natural language to the system. We will assume that the natural language employed is the same as that of the document collection (thus we exclude cross-lingual QA). What is to be considered a question will be investigated in more detail below.

Answer Search. The system searches for information that can form an answer to the user's question in the document collection. As stressed in 2.1.1, it is essential for a document collection of a non-trivial size that it is indexed offline to allow efficient searching (2.1.1).

Answer Presentation. If the answer search has produced a result, the resultant information is presented to the user in the form of an answer to her/his question. What is to be considered an answer is investigated in greater detail below. It is worth noting here that, generally, the document collection will not contain a *direct* answer to a user's question and that therefore information from the collection must be recognised as an *indirect* answer and must be reformulated to become a direct answer. This will be discussed below (3.2.2.2, 6.4.3).

Simulating Human Question-Answering. As a starting point, we will assume that the QA system attempts to simulate a human question answerer. This assumption will be explained and discussed below (3.2, 3.2.4).

We will now start our review of related work with an overview of work on questions and answers in linguistics. Based on the reported findings, we will update the description of the framework.

3.2 Questions and Answers

Questions and answers have been extensively studied by researchers in logic and linguistics. We will give an overview of prominent issues discussed in the literature and then relate them to question answering systems in general and to our approach, more specifically.

While work on questions and answers has an obvious bearing on question answering, it is important to keep in mind that the descriptions reported here almost exclusively have human communication as their subject. These accounts describe questions and answers as they are observable in human communication and try to explain their relationship (i. e., why is a certain utterance acceptable as an answer to a specific question). Thus, they cannot be taken as a direct basis for the (human-computer) QA process.

We will find (cf. 3.2.4) that all of these approaches are concerned with the relationship between a given question and a given answer candidate and that they have little to say on the question of *finding* or *constructing* answers – which, however, is *the* central point in question answering. Question answering systems must *find* information containing the answer in their document collection and, based on that, *give* a suitable answer. Therefore, the different parts of the overall process, namely question analysis, answer finding and answer presentation, must be distinguished (cf. also 3.1.2), even though they are, of course, strongly interrelated and can sometimes not wholly be separated.

Nevertheless, we believe that looking at questions and answers from a linguistic viewpoint is useful for the development of a comprehensive framework of QA: We assume that a QA system that generally simulates the behaviour of a *human* who answers questions will be a usable and helpful system. This assumption may turn out to be too simplistic: Experiences with voice user interfaces show that users of spoken dialogue systems sometimes prefer systems that do *not* try to simulate full 'human' behaviour but that behave more schematically (Cohen et al., 2004). However, there are so far no specific studies concerning the usability of QA systems, so we cannot fall back on any established findings regarding this assumption. We think that taking a simulation of a human question answerer as the starting point is a useful idea: Humans obviously engage in information seeking dialogues (sequences of questions and answers) to gather information, thus at least the basic idea will come natural to a human user of a QA system (see also 8.3.3).

We will look at questions and answers from the viewpoint of syntax, semantics and pragmatics in turn. Work in each of these disciplines has contributed insights into different aspects of the nature of questions on the one hand and of the relationship between questions and answers on the other hand.

Our primary interest is how we can identify answers to a given question. That is, given a question and a response, can we decide whether the response actually provides an answer and how can we distinguish it from other utterances that do not provide an answer.

In order to better distinguish the different linguistic layers of questions, we will use the following terminology, following Higginbotham (1996) and Groenendijk and Stokhof (1997):

Syntax. Questions are typically expressed by using sentences of a specific mood, namely *interrogative sentences* (or interrogatives for short). They differ from sentences in declarative mood (typically used to express statements, and thus especially answers) morpho-syntactically by word order, use of the question mark, use of interrogative pronouns or particles etc. For example, *'Who killed John F. Kennedy?'* is an interrogative.

Semantics. In the remainder of this chapter, we will generally use *question* to refer to the semantic contribution, viz. the meaning of an interrogative.

Pragmatics. We will use the term *interrogative act* to designate the speech act that asks a question, i. e., that is used by speakers to express their desire that the hearer provide some specific piece of information, viz. the answer. For example, *'Please tell me who murdered John F. Kennedy.'* is a directive sentence, but in uttering it, a speaker will typically perform an interrogative act.

3.2.1 Syntactic Aspects

In this section, we will describe phenomena related to questions and answers in syntax. We will especially sketch a general syntactic typology of questions, that is, list types of questions that differ syntactically.

We will then give an overview of questions and answers in German. This overview shows, on the one hand, the scope of syntactic question types that a German QA system needs to handle. On the other hand, our aim was to give readers not familiar with the German language an impressionist view of questions and answers in German.

3.2.1.1 Interrogatives and Declaratives

Sentences like (3.1a) and (3.1b) are said in traditional grammar to differ in their sentence mood (namely declarative and interrogative); this is morpho-syntactically marked by certain features (here: inverted word order, do-support, question mark). While the features themselves are largely language-dependent, corresponding interrogatives and declaratives exist in most languages (see also Groenendijk and Stokhof, 1997, 1058–1060)

(3.1) a. Lee Harvey Oswald killed John F. Kennedy.
b. Did Lee Harvey Oswald kill John F. Kennedy?

Different ways have been proposed for modelling these correspondences in a number of syntactic frameworks. In general, they assume that an underlying question-feature (either of the main verb or of an underlying deep structure) triggers suitable rules that influence the surface structure.

An analysis of interrogatives in Noam Chomsky's *Government and Binding* theory (GB, Chomsky, 1993) is given in Higginbotham (1996).

Two different accounts of interrogatives in Head-driven Phrase Structure Grammar (HPSG, Pollard and Sag, 1994) can be found in Egg (1998); Ginzburg and Sag (2000).

A simple syntactic notion of answerhood can be described as holding between two sentences that only differ in sentence mood, namely where one is in interrogative mood and the other one in declarative mood. However, this simple notion of answerhood is not sufficient for QA: In most cases a potential answer in the document collection will differ in more than sentence mood only from the question. Such differences will be discussed in the following.

3.2.1.2 Question Types

English, German and many other languages have three general question types.[2], which we will discuss here. More fine-grained, semantically motivated distinctions in question types will be taken up in 3.5.2.5.

***Wh*-Questions.** A *wh*-question is an interrogative sentence that contains a question word (in English for example *'who'* or *'when'*). Examples for the first group of questions are shown in (3.2) through (3.5). We will follow common practice and call them *wh*-questions. This term comes from the fact that interrogatives in this group contain a question word and that most question words in English start with the letters *wh* (*'who'*, *'what'*, *'which'*, *'when'* etc.). As the term *wh*-question is widely used, we will stick with it, even though it is neither wholly accurate (the English question word *'how'* does not start with *wh*) nor does it carry over to other languages, of course. Other frequently used terms

[2]Note that we will use the term question type and terms like *wh*-question here instead of type of interrogative, *wh*-interrogative etc. On the one hand, these terms are more familiar, on the other hand they seem to correspond fairly closely with the question (type) underlying the interrogative. Though this is not wholly consistent, we believe that there is little potential for confusion.

include constituent question (as it can be seen as expecting an answer typically expressed by one phrasal constituent, cf. (3.2a), 3.2.2.2) or term question (as the expected answer type would be a term in a predicate logic representation).

(3.2) Who killed John F. Kennedy?
  a. Lee Harvey Oswald. (NP)
  b. Lee Harvey Oswald killed John F. Kennedy. (Sentential answer)

(3.3) Where was John F. Kennedy killed?
  a. In Dallas, Texas. (location PP)
  b. (pointing) *There.* (adverb)

(3.4) Why was Lee Harvey Oswald arrested?
Because he shot John F. Kennedy. (reason, clausal answer)

(3.5) Which US president was killed in Dallas, Texas?
John F. Kennedy. (NP)

(3.6) How old was John F. Kennedy when he was killed?
46 years (old). (NP)

(3.7) Who killed which US president?
Lee Harvey Oswald (killed) John F. Kennedy and J. W. Booth (killed) Abraham Lincoln. (Sentential answer, verb ellipsis possible)

The following points are worth noting:

- Besides a short answer consisting of a single constituent, a full sentential answer is also possible (cf. (3.2a) vs. (3.2b)). We will return to this point in greater detail in 3.2.2.2. In our system implementation, users can select constituent or full sentential answers through setting user preferences (6.4).

- *Wh*-Questions can ask for NPs (3.2), but also for certain adverbials (3.3, 3.4). Questions of this type can take a range of syntactic constituents as answers, namely all adverbials of the suitable kind (such as local adverbials, phrased as a PP in (3.3a) and as an adverb modifier in (3.3b)). Note that a suitable answer may also be a whole text, we will return to this point in 3.2.2.5. In our model of indirect answerhood, we use frame roles (5.2.2.6) and additional semantic role relations (5.2.2.7) in matching questions and answers to identify suitable answer constituents.

- The *wh*-questions asking for NPs can be further sub-divided into *who*-questions and *which*-questions. While the former put little semantic restriction on possible answer NPs (an answer to a *who*-question typically designates some human being, *what* typically expects a non-human entity as its answer), the latter are used to further restrict answers semantically (cf. (3.2) vs. (3.5)). We will return to this distinction in 3.5.2.5.

- Another group of *wh*-questions is constructed in English with *how*+adjective (or adverb) as question phrase. They generally ask for a certain measure or number that is related to the adjective or adverb. In (3.6), e. g., *'how old'* is used to ask for an age. See also 3.5.2.5.
- *Wh*-Questions can generally contain more than one *wh*-phrase (3.7). These are called multiple *wh*-questions. As with single *wh*-questions, both a constituent answer and a sentential answer is possible. We will return to multiple *wh*-questions in greater detail below (3.2.2.2).

Yes/no-questions. The second group of questions ask for the truth or falsity of a proposition. We will refer to them as yes/no-questions, as – at least generally – the appropriate answer to them is either *'yes'* or *'no'*. As with *wh*-questions, the term is not uncontroversial, as not all languages possess distinct morphemes corresponding to *'yes'* and *'no'*, even though they do have this question type. Both in Latin and Mandarin, for example, a yes/no-question is answered either by echoing or negating the matrix verb of the question or by using a paraphrase (such as *'(non) ita (est)'*, *'(that is) (not) so'* in Latin or *'duì'*, *'(that is) correct'* in Mandarin).

Yes/no-questions do not contain a question word. However, some languages use question particles to mark yes/no-questions, such as *-ne, num* or *nonne* in Latin or *ma* in Mandarin. We do not consider these question particles as question words here.

Besides the answers *'yes'* or *'no'*, sentential answers are also possible (3.8b).

(3.8) Did Lee Harvey Oswald murder John F. Kennedy?

a. Yes.

b. (Yes,) Lee Harvey Oswald murdered John F. Kennedy.

Since yes/no-questions require different answers than *wh*-questions, they must be treated differently in a QA system. In our system, we handle yes/no-questions as follows: All sentences that match the question are considered as possible answers, irrespective of polarity (3.5.1.2). In a second step, we check whether question and answer have the same polarity (5.1.5). If the polarity differs, we first output *'no'*, otherwise *'yes'*. Then we output the whole answering sentence (6.4.3). The result is a answer as in (3.8b).

Alternative Questions. This type of questions has characteristics from both *wh*-questions and yes/no-questions in that they ask for a constituent like *wh*-questions, but they do not contain a question word, but rather provide a list of alternatives to the hearer to choose from.

(3.9) Did Lee Harvey Oswald or Jack Ruby shoot John F. Kennedy?
Lee Harvey Oswald.

In our system, we have currently not implemented any handling for alternative questions, as they are rarely used. In the evaluation based on 854 German question-answer pairs (7.2.1.2), for example, we found only two alternative questions (0.2 %).

3.2.1.3 Embedded Interrogatives

For each type of direct interrogative, there exists a corresponding indirect or embedded form (cf. the examples in (3.10)). Indirect interrogatives can form a clausal complement of a number of words (verbs such as *ask*, *wonder* and *tell*, nouns such as *question*, adjectives such as *doubtful* in English). The rules that govern the formation of such embedded interrogatives are largely language dependent.

(3.10) a. The question who killed John F. Kennedy has been reopened by Oliver Stone's movie.
(Embedded *who*-interrogative)

b. I wonder whether Fidel Castro had John F. Kennedy killed.
(Embedded yes/no=*whether*-interrogative)

The example (3.10b) shows an example of a question that is not expressed by a direct interrogative but by a declarative that asks a question (and thus an indirect speech act). We will come back to this possibility in 3.2.3.1. Note that an indirect question was used to form the sentence; this is often the case with indirect speech acts asking questions.

3.2.1.4 German as an Example

In this section, we give a short overview of syntactic phenomena directly concerned with questions in German. It is intended as an example instantiation of the general regularities that have been described above. For readers who are not familiar with German, it provides an impressionist view of the phenomena encountered in German questions – which are not to far different from those in English. The section is largely based on standard grammar books for German, namely Eisenberg (1999); Helbig and Buscha (1989); Duden (2005).

Direct Questions. Direct questions are marked with a question mark '?' in German, just as in English. The three main syntactic questions types described above all exist in German. They differ concerning word order.

In describing phenomena in word order for German, we will make use of the topological sentence model of German (Engel, 1970) as it provides a adequate account of the somewhat complex regularities in German.

Readers who are unfamiliar with the model are referred to Eisenberg (1999, 384–399); Engel (1970) where the model is discussed in detail. Here is a minimal overview: All German sentences are taken to consist of five so called topological 'fields', namely *Vorfeld* (fore field), left sentence bracket, *Mittelfeld* (middle field), right sentence bracket and *Nachfeld* (after field). Three sentence types are distinguished by the position of the *finite* verb (or the finite part of a complex verb) and whether the *Vorfeld* is obligatorily filled or empty: In verb-first sentences (V1), the *Vorfeld* must be empty and the finite verb is positioned in the left sentence bracket (verb fronting, used in directive sentences, yes/no-questions and alternative questions), in verb-second sentences (V2) the *Vorfeld* must be filled by exactly one (maximal) phrase and the finite verb fills the left sentence bracket (declaratives, *wh*-questions), while in verb-last sentences (VL), the finite verb is positioned in the right sentence bracket (all subordinate sentences).

Wh-Questions in German such as (3.11) are verb-second sentences. The *Vorfeld* is typically filled with a *wh*-phrase. In multiple *wh*-question, additional *wh*-phrases are positioned in the *Mittelfeld*. In contrast, both yes/no-questions (3.12) and alternative questions (3.13) are verb-first sentences (somewhat similar to verb-fronting in their English counterparts, but in German there is no do-support).

(3.11) *Wer hat John F. Kennedy ermordet?*
Who has John F. Kennedy murdered
Who has murdered John F. Kennedy?

(3.12) *Hat Lee Harvey Oswald John F. Kennedy ermordet?*
Has Lee Harvey Oswald John F. Kennedy murdered
Has Lee Harvey Oswald murdered John F. Kennedy?

(3.13) *Hat Lee Harvey Oswald oder Jack Ruby John F. Kennedy*
Has Lee Harvey Oswald or Jack Ruby John F. Kennedy
ermordet?
murdered?
Has Lee Harvey Oswald or Jack Ruby murdered John F. Kennedy?

Answers to direct questions in German have exactly the forms described above in general: Each question type can be answered by sentential answers. Besides, both *wh*-questions and alternative questions can be answered by a single constituent. In German as an overtly inflecting language, a phenomenon that is generally called answer congruency is especially noticeable, as bare NPs in the constituent answer must carry the same inflection (especially case) as the corresponding *wh*-phrase in the question. We will come back to this point in greater detail in 3.2.2.2.

Yes/no-questions can be answered by *'Ja'* (*yes*) oder *'Nein'* (*no*). It should be noted that when the question is overtly negated, *doch* is often used to assert the 'positive case', as shown in (3.14).

(3.14) a. *Hat es dir nicht gefallen?*
Has it to you not pleased
You didn't like it?

b. *Doch.*
(but yes)
Yes, I did.

A German QA system needs to identify the different question types. The PREDS parser that we use in our system implementation recognises *wh*-questions and yes/no-questions, using a topological parser and a list of *wh*-words (6.2).

Indirect Questions. The direct questions in German each have an indirect counterpart. These are syntactically different from the direct form in that they are all verb-last sentences (as all subordinate sentences in German). Both indirect yes/no-questions and indirect alternative questions are introduced by the conjunction *ob* (similar to *whether*).

(3.15) *Die Frage, wer John F. Kennedy ermordet hat, ist durch*
The question who John F. Kennedy murdered has is by
Oliver Stones Film neu gestellt worden.
Oliver Stone's movie newly put been
The question who killed John F. Kennedy has been reopened by Oliver Stone's movie.

(3.16) *Bitte sagen Sie mir, ob Fidel Castro John F. Kennedy hat*
Please tell you to me whether Fidel Castro John F. Kennedy has
töten lassen.
kill let
Please tell me whether Fidel Castro had John F. Kennedy killed.

Question Words. The following list shows the most important German question words. Most of them have direct equivalents in English (3.2.1.2), only the interrogative prepositional adverbs are slightly different.

Interrogative pronouns. *wer*$_{nom}$, *wessen*$_{gen}$, *wem*$_{dat}$, *wen*$_{acc}$: nominal interrogative pronoun for humans (*who*)
was$_{nom}$, *wessen*$_{gen}$, *wem*$_{dat}$, *was*$_{acc}$: nominal interrogative pronoun for non-human entities (*what*)
welcher$_{masc}$, *welche*$_{fem}$, *welches*$_{neut}$: interrogative pronoun as determiner (*which*), only nominative case listed here
was für ein$_{masc}$, *was für eine*$_{fem}$, *was für ein*$_{neut}$: interrogative pronoun as determiner (*which*), only nominative case listed here

Interrogative adverbs. Local: *wo* (*where*)

Temporal: *wann* (*when*)

Modal: *wie* (*how*), also used attributively with adjectives or adverbs: *wie alt* (*how old*), *wie oft* (*how often*)

Causal: *warum*, *weshalb*, *weswegen*, *wieso* (*why*)

Interrogative prepositional adverbs. A short form for a *wh*-phrase consisting of a preposition and (the suitable form of) *was*, becoming *wo* in the short form, for example: *mit* + *was* (*with what*) → womit
Similar: *worauf* (*on(to) what*), *woraus* (*out of what*), *wofür* (*for what*)

A German QA system must recognise these different question words and the syntactic constructions of *wh*-phrases associated with them. In the SQUIGGLI system, all the listed question words and '*wh*'-phrases are recognised by the parser.

3.2.1.5 Discussion

We have sketched a number of syntactic phenomena concerning questions and answers. We have first listed a number of more general phenomena and then given a short overview of phenomena in one particular language, namely German, as an illustrative example.

For building a QA system, a list of language-specific syntactic phenomena must be put together as a basis for defining a question analysis module: The question analysis module should (obviously) be able to analyse and categorise question input that exhibits the respective phenomena. We will come back to this point in some more detail below (3.2.4).

Syntactic accounts of questions do not have much to say about the respective *answers*, at least not beyond the general points mentioned above. We will therefore now turn to semantic approaches that introduce the notion of answerhood, i. e., the relation between a question and its (either true or possible) answer or answers.

3.2.2 Semantic Aspects

We will introduce four different general semantic approaches of questions and answers in this section. They have been glossed erotetic logic approach (Belnap and Steel, 1976), categorial approach (Hausser and Zaefferer, 1979; Keenan and Hull, 1973; Krifka, 2001), propositional approach (Hamblin, 1973; Karttunen, 1977; Karttunen and Peters, 1980) and partition approach (Groenendijk and Stokhof, 1984, 1997; Higginbotham, 1996), respectively. Similar to some of the syntactic aspects of questions and answers, most semantic aspects seem also to be language independent.

Work on the semantics of questions goes back to (at least) Hamblin (1958). Hamblin states that the predominating view at the time was that the semantic content of interrogatives and declaratives does not differ and that the only difference between them was pragmatically motivated. He challenges this view and suggests that a semantics of questions should be defined through their answers. In his informal paper, he discusses questions and answers and arrives at the following three, often-cited postulates. The different approaches are based (explicitly or implicitly) on one or more of these postulates.

> Postulate 1. An answer to a question is a statement. [. . .]
> Postulate 2. Knowing what counts as an answer is equivalent to knowing the question. [. . .]
> Postulate 3. The possible answers to a question are an exhaustive set of mutually exclusive possibilities. (Hamblin, 1958, 162–3)

The different approaches make reference or use of (one or more of) Hamblin's postulates, but only the partition approach can be said to fully incorporate all three.

All semantic approaches to questions and answer are centred around the following core idea (much simplified, of course). The ? operator notation used here is borrowed from Groenendijk and Stokhof (1997). It is used to mark a question, for example $?\phi$ represents the question whether the proposition ϕ is true or not.

- A yes/no-question receives $?\phi$ as a representation, where ϕ is a proposition contained in the question. By asking the question, the questioner wants to be asserted whether the extension of the proposition is true or false, i. e., the answer to the question is either either ϕ or $\neg\phi$. So, for example, (3.8) could be represented thus:

(3.17) a. ?kill′(lee_harvey_oswald′,john_f_kennedy′)
b. kill′(lee_harvey_oswald′,john_f_kennedy′)

- A *wh*-question can be represented as $?x_1 \ldots x_n\phi$ where the variables $x_1 \ldots x_n$ correspond to the *wh*-phrases in the question and occur free in ϕ. A (partial) answer would be given by ϕ where $a_1 \ldots a_n$ instantiate $x_1 \ldots x_n$, respectively, so that no free variables remain. Thus, an answer 'binds' each of the *wh*-phrases to an individual in the domain. For example, (3.2) could be represented like this:

(3.18) a. $?x_1$kill′(x_1,john_f_kennedy′)
b. kill′(lee_harvey_oswald′,john_f_kennedy′)

This core idea is – in some, more or less adapted form – present in all approaches.

This general notion of semantic answerhood could be used, in principle, as the core of a QA system in the following way: From the document collection and from the users' questions, semantic representations are derived. Finding answers is done by identifying, for each proposition expressed in the question, matching propositions expressed in the document collection representation. *Wh*-phrases are represented by 'wild-cards' that match anything (similar to variable matching in PROLOG). An approach along these lines is described in Allen (1995, esp. 414–419).

However, deriving semantic representations from general texts with broad coverage is beyond the scope of current Natural Language Processing systems. We will return to this issue below (3.3). Besides, this approach does not account for variations in wording between question and answer, such as the use of a synonymous expression.

3.2.2.1 Belnap's Erotetic Logic

Of several different works defining erotetic logic[3], we will only present Belnap and Steel (1976) here[4]. An overview and additional literature can be found in

[3]Erotetic: from Greek ἐρώτησις (question). This coinage is generally ascribed to Arthur N. Prior and Mary L. Prior.

[4]The work reported here was done exclusively by Nuel D. Belnap (cf. Belnap and Steel 1976, 1, footnote). We will therefore refer to the approach as Belnap's.

Harrah (1984); Egli and Schleichert (1976). Work on erotetic logic focusses on delivering an adequate logic formalism for describing questions (and answers), generally by extending first-order predicate logic or similar logic formalisms.

Belnap states that he defined erotetic logic not so much as a formal representation for natural language, but rather as a formal description of databases and similar systems. Thus, he is not concerned with suitably representing natural language phenomena, although he often adduces examples from natural language for ease of exposition (Belnap and Steel, 1976, 139–148).

We will describe Belnap's language here shortly, as we believe that the definition of the language and its interpretation shows in an interesting way how question and possible answers are related and how additional constraints on the answer (such as the number of answers to be given) can be incorporated.

In Belnap and Steel (1976), an extension (called simply L) to first-order predicate logic with equality (FOPL$_=$) is defined as a formal language for representing questions: Question representations consist of a request (ρ) and a subject (σ) part. Both request and subject representation are tuples. The subject part represents the question itself (question variables and question proposition), while the request part represents specifications regarding the requested answers, such as the requested number of answers. A question representation is marked by an initial '?'. Thus, a general question representation looks like this (Belnap and Steel, 1976, 21–43):

(3.19) $?\rho\sigma$, general question form
$\rho : (\mathbf{s\,c\,d})$, a triple (request part)
$\sigma : (C_1x_1,\ldots,C_rx_r,x_{r+1},\ldots,x_n//Ax_1\ldots x_n)$, a pair (subject part)

In the following, we will give a short description of the different symbols and their meaning, starting with the request part ρ:

s: Selection size specification (number of answers that is requested). This is given by the lower bound v and the upper bound u. Both must be non-zero integers with $v \leq u$. u can also be set to $-$, which stands for 'no upper limit'. (Belnap and Steel, 1976, 36–46)

c: Completeness claim specification. Either $-$ or $\forall$ (no vs. maximal completeness claim, Belnap and Steel, 1976, 46–60).

d: Distinctness claim specification. Either $-$ or $\neq$ (no distinctness claim vs. complete distinctness claim). If distinctness is required, then answers must be extensionally different (Belnap and Steel, 1976, 60–67).

The elements of the subject part σ can be described as follows:

C: Category constraints: x_i (see below) must be of category C_i (such as *x is integer*). Variables $x_{r+1}, \dots, x_n$ are unconstrained.

x: Question variables (queriables) in the question.

A: Formula, containing $x_1, \dots, x_n$ as variables that are free within the formula. They are bound in σ as queriables.

Equipped with these definitions, we can now give some semi-formal examples for question representations:

(3.20) a. Name all prime numbers between 10 and 20.

b. $?(\bar{}_1\ \forall\ \neq)(x$ is integer$//x$ is a prime between 10 and 20$)$

c. Give at least two examples of prime numbers between 10 and 20.

d. $?(\bar{}_2\ -\ \neq)(x$ is integer$//x$ is a prime between 10 and 20$)$

We will sketch now how the standard interpretation of FOPL$_=$ is extended to cover the interpretation of questions (and answers) relative to a model M: A subject part is associated with a so-called real M-alternative. Informally, this describes a possible 'single' direct answer (such as (the denotation of) *13 is a prime between 10 and 20* for (3.20a)) in the following way: It contains a variable assignment function that assigns a value to every queriable (while leaving bound variables in A unchanged). If a category constraint for a queriable is given, then the definition ensures that this constraint is observed, i. e., that x_i is of type C_i. The union of all real M-alternatives makes up the real M-range, i. e., (again informally speaking) the description of all possible direct answers. The M-range does not yet distinguish true from false answers, but only lists possible answers. For (3.20a), the M-range would contain the denotation of *x is a prime between 10 and 20* for *all x* that are integers. For a formal definition see Belnap and Steel (1976, 22–34).

A direct answer to such a question then consists of a conjunction $(S \wedge C \wedge D)$. Here, *S* is a selection drawn from (i. e., a subset of) the M-range of the subject, *C* is the completeness claim and *D* is the distinctness claim. To be a direct answer, the size of the selection *S* must lie within the range given by the selection size specification of the question's request part.

If a completeness claim is requested, then *C* must satisfy that claim. For the maximal completeness claim discussed here, *C* must list all 'counterexamples', i. e., the M-alternatives for which the extension in M is false. If no completeness claim is made, then $C = \emptyset$. Thus *S* and *C* together must exhaustively cover the M-range.

If a distinctness claim is requested, then D must contain the suitable claim, namely a conjunction of disjunctions of the form $((a_{i_1} \neq a_{j_1}) \vee \ldots \vee (a_{i_n} \neq a_{j_n}))$, that is, the claim that all alternatives be pairwise distinct. Otherwise, $D = \emptyset$.

Most questions in natural language are underspecified with regard to the specifications of Belnap's formal request part. Consider, for example, the question *'Who killed which US president?'* (3.7). This question neither specifies a selection size nor a completeness claim: Listing any number of US presidents and their respective killer would count as a direct answer. One generally has to resort to explicit directives as (3.20a) or (3.20c) to express the corresponding request.

Discussion. As mentioned, Belnap's erotetic logic was originally intended as a 'technical' language for describing, for example, database queries. However, a number of correspondences to natural language can be established.

As the examples (3.20a) and (3.20c) show, questions can differ with respect to the number of entities that the denotation of an answer should contain. There are ways in natural language to express 'selection size' (as Belnap calls it), such as *'Which* five *theories predict this fact correctly?'* or (3.20c) above. However, many questions are (more or less) underspecified with respect to this point.

Generally, the following cases are distinguished: The mention-one interpretation, the mention-all interpretation and the mention-some interpretation (sometimes also mention-n interpretations as the more specific case) ask the answerer to provide one example, an exhaustive answer or only some (n) true examples, respectively. Whether the different interpretations are to be treated as semantically different (i. e., they would receive distinct readings) or whether the difference should be accounted for only by pragmatics, is discussed in some detail in Groenendijk and Stokhof (1997, 1111-22, they finally opt for a semantic distinction). We will not further be concerned with this particular issue but note that questions can differ with respect to the expected number of entities that an answer should contain.

We currently make no attempt to automatically identify answer size requests because of the described difficulties in establishing it. In the system implementation, we allow users to explicitly select the maximum number of answers that should be returned (3.2.4).

The issue of complete answers has also been the subject of discussion. As we have seen, Belnap suggests that a completeness claim must be made by explicitly listing all 'negative examples'. We will return to this point when we discuss the closely related notion of exhaustive answers that is a central building block of the partition approach (cf. 3.2.2.4 below).

The question of distinctness seems to be linguistically less interesting. Belnap himself points out that questions and answers in natural language seem to generally assume distinctness and that the distinctness selection was mainly added with a view to artificial languages where it might be useful (Belnap and Steel, 1976, 62–3). We will therefore assume that when there are multiple entities in an answer to a question, these should be distinct, viz. that the distinctness requirement always holds.

Note, however, that ensuring distinctness poses a practical challenge to QA systems, as it requires full semantic representations, in general. We currently use a simplification and only treat answers as distinct that differ at the surface. We will discuss this point in more detail below (3.2.4).

3.2.2.2 Categorial Approach

The categorial approach is based on the observation that a *wh*-question can be answered by a constituent of the correct type (cf. 3.2.1.2). Work on the categorial approach (among others, Hausser and Zaefferer, 1979; Keenan and Hull, 1973; Krifka, 2001) therefore mirrors the difference in the syntactic question (and answer) type directly in their semantic type. A question is represented as a lambda term where the type of the lambda-term corresponds directly to that of a possible (constituent) answer. A question is thus a functor that takes the answers as its argument. Functional application then yields exactly the proposition that represents the *answered* question. Let us take a look at an example. Note that the example is simplified in that an extensional representation is used and the representation of proper names as individual constants (type e) is assumed.

(3.21) a. Who did Mary see?

b. $\langle e,t\rangle$: $\lambda x[\text{see}'(x)(\text{mary}')]$

c. John

d. e: john$'$

e. t: $\lambda x[\text{see}'(x)(\text{mary}')](\text{john}') = \text{see}'(\text{john}')(\text{mary}')$

(adapted from (Krifka, 2001, 288))

In the example, (3.21b) is taken to be the representation of (3.21a). The answer (3.21c) is represented by (3.21d). Functional application then yields the proposition that is conveyed through the answer (3.21e).

We will not further discuss yes/no-questions and their answers in the categorial approach. The general idea is that they are handled through ‘yes’ and ‘no’ denotating $\lambda p.p$ and $\lambda p.\neg p$, respectively, so that functional applications of the representation of yes/no-questions (type $\langle\langle t,t\rangle,t\rangle$) to them derives the ‘positive’ or negated version of the question representation. See the sources cited in the respective sections for the exact spell-out in the different approaches.

Keenan and Hull's Approach. The approach described in Keenan and Hull (1973) is close to the generic version described above. It assumes that for each question-answer-pair $\langle \mathrm{Q}, \mathrm{A} \rangle$ where Q is a *wh*-question that contains exactly one phrase of the type *which CN* (CN stands for a common noun) and A is an NP, it can be checked whether A is a true answer by inserting each individual in the extension of A into Q instead of the *which CN* phrase and checking the truth of the resulting proposition. If the proposition is true for all members in the extension of A, then the answer is true. If it is false for at least one member, then the answer is false. A third value (zero) is defined for a number of inconsistent cases.

Hausser and Zaefferer's Approach. Let us turn to a somewhat more complex example. It is taken from Hausser and Zaefferer (1979), where an extension to Montague grammar (Montague, 1974) is defined that allows the derivation of questions and answers in the Montagovian way (i. e., by deriving a syntactic structure by categorial rules and translating it in parallel into a representation based on intensional logic by corresponding semantic translation rules).

The functional application of the question representation to the answer representation is done in the following way: When a question is asked, a context variable Γ is instantiated with the question representation. The type of Γ depends on the question type (as above). Constituent answers such as (3.22c) are assumed to be of category S (sentence). This is achieved by a special rule that turns a single constituent into an answer of type S that is licenced by the full stop. In the example below, *'a dragon'* would be a term, that is turned into an answer by adding the full stop, as in (3.22c). The answer representation directly embeds Γ. As Γ is correctly instantiated in the question context, the functional application is 'built into' the answer representation.

(3.22) a. What does Mary imagine?

b. $\Gamma_{\mathrm{S/NP}} := \lambda \mathscr{P}_{\mathrm{S/NP}}.m^{*}(\hat{}\mathrm{imagine}'(\mathscr{P}))$

c. A dragon.

d. $\Gamma_{\mathrm{S/NP}}([\lambda P.\exists x[\mathrm{dragon}'(x) \wedge P(x)]]_{\mathrm{NP}})$

e. $m^{*}(\hat{}\mathrm{imagine}'(\lambda P.\exists x[\mathrm{dragon}'(x) \wedge Px]))$

f. Mary imagines a dragon.

(adapted from Hausser and Zaefferer, 1979, 345–7)

In the example, the question (3.22a) is represented by (3.22b), while the answer (3.22c) is represented by (3.22d). Instantiation of the context variable Γ (3.22b) and β-reduction then yields (3.22e). This is exactly a possible representations that one would obtain for (3.22f) which would be a possible sentential answer for (3.22a). In Hausser and Zaefferer's approach, this is the only relation between a constituent and a sentential answer. More specifically, they do not further account for the fact that (3.22f) would count as an answer for (3.22a), but other sentences would not.

Note that the approach would also allow to derive a second (*de re*) reading for the answer besides the *de re* answer given here. We will not further go into the details of this possibility.

Structured Meaning Approach. Questions and answers can be integrated into the structured meaning framework (von Stechow, 1991). The result has a distinct 'categorial flavour' (Reich, 2003; Krifka, 2001). Note that Reich's account takes structured meaning as a starting point but integrates it with the idea, taken from the propositional approach (cf. 3.2.2.3), that questions are denotated by sets of propositions. Reich assumes that question denotations are 'sets of structured propositions', thus dividing propositions into a focus and a background part as described here. We will not further pursue this point here and only note that through the use of *structured* propositions the approach retains a 'categorial flavour' akin to that of Krifka.

By using the structured meaning framework, the authors can account for a number of phenomena where focus interacts with the interpretation of questions and answers, mainly observable by accent patterns in spoken language: In natural language, focus is mainly expressed by devices such as accent (focussed constituents receive sentence accent) or by word order changes (topicalisation). Consequently, the findings described here would be central for spoken language dialogue systems where they could help to achieve a natural intonation or to distinguish between different question types. In a purely text-based system, these questions are less central. In a structured meaning representation (von Stechow, 1991), semantic representations of questions and answers consist of two parts, the background and the focus part. The following criterion defines the answerhood relation in the structured meaning framework (B stands for background, F for focus and R for question restriction):

> Criterion for congruent question-answer pair $Q – A$, where $[\![Q]\!] = \langle \mathrm{B}, \mathrm{R} \rangle$ and $[\![A]\!] = \langle \mathrm{B}', \mathrm{F} \rangle$: $\mathrm{B} = \mathrm{B}'$ and $\mathrm{F} \in \mathrm{R}$. Krifka (2001, 296)

Let us consider the following example. We will follow general practice here and mark focus-bearing constituents in the examples with ‘F’. Accent is marked by ´.

(3.23) a. Who did Mary see?
b. $\langle \lambda x[\text{see}'(x)(\text{m}')], \text{person}' \rangle$
c. Mary saw [Jóhn]$_F$.
d. $\langle \lambda x[\text{see}'(x)(\text{m}')], \text{j}' \rangle$, where $\text{j}' \in \text{person}'$
e. * [Máry]$_F$ saw John.
f. $\langle \lambda x[\text{see}'(\text{j}')(x)], \text{m}' \rangle$, where $\text{m}' \in \text{person}'$
(adapted from Krifka, 2001, 296)

Given Krifka’s answerhood criterion, this example shows how the structured meaning account can explain the observation that (3.23c) is a good answer for (3.23a), while (3.23e), where *Mary* is accented, is not: The representations (3.23a) and (3.23d) form a congruent question-answer pair under the answerhood criterion (identical background, focus is element of restriction), while (3.23a) and (3.23f) do not (background differs).

Criterion and example together also show why this approach is generally classed under the categorial approaches: The type of the answer focus differs depending on the focussed constituent of the answer; the question restriction is also dependent on the question type (it is here expressed as a set or – equivalently – a characteristic function). By functional application (here, B(F)), the ‘unfocussed’ answer proposition can always be derived from the structured representation.

In addition, Krifka further differentiates and extends the first, syntactic definition of multiple *wh*-questions above (cf. 3.2.1). He argues that only the structured meaning approach will handle all different types of multiple *wh*-questions correctly. We will not repeat his arguments here (cf. Krifka, 2001), but only list the most important phenomena he describes. The main issue is that multiple *wh*-questions of the different types seem to request different selection sizes (in Belnap’s terms), namely mention-one vs. mention-some/mention-all interpretations.

Matching Questions. Multiple *wh*-questions that ask for a list of ‘matching’ answers (mention-some or mention-all interpretation), as in the following example:

(3.24) a. A: Whó left whén?
b. B: Máry left at fóur, and Jóhn left at four-thírty.
Krifka (2001, 303, his (52))

This type of question seems to be the standard form of multiple *wh*-questions. It would also allow for a constituent answer, cf. (3.7) above.

Conjoined Questions. This type of multiple *wh*-question is derived from two (or more) conjoined questions. For the second (and following) questions, all identical material is elided. This type of question does not call for a list answer, but rather for a single (conjoined) answer (mention-one):

(3.25) a. A: Whó left, and whén?
b. B: Máry left, at fóur.
Krifka (2001, 304, his (54))]

Quiz Questions. This type is used only in quiz situations, and with a special intonation, resulting in a mention-one reading.

(3.26) Which assassin killed which US president in which city with which weapon in 1963?
Lee Harvey Oswald killed John F. Kennedy in Dallas, Texas, with a gun.

Echo questions. Echo questions are used to signal (real or feigned) misunderstanding, again giving rise to a mention-one interpretation.

(3.27) (Inaudible) shot (inaudible).
Whó shot whóm?

(3.28) Have you heard? Mary is actually going to marry John.
Mary is going to marry whóm?

We consider matching questions the most relevant type of multiple *wh*-questions for QA. Besides, automatically identifying the different types of multiple *wh*-questions (especially without clues from intonation) is difficult. We therefore currently treat all multiple *wh*-questions as (potential) matching questions and search for and present all possible answers. In this way, no potential answers are lost. We will shortly return to multiple *wh*-questions below (3.2.2.5).

Critique 1: Constituent Answers and Sentential Answers. As described above, the categorial approach is especially motivated by the fact that *wh*-questions can be answered by constituent answers. This assumption stands in contrast with Hamblin's first postulate cited above (p. 48), at least in its 'pure' form. Hamblin suggests that even if a constituent answer is used, it must still receive the interpretation of a full statement in the context of the question (Hamblin, 1958, 162). This is captured by Hausser and Zaefferer's rule for turning constituents into constituent answers by adding a full stop described above (3.2.2.2). However, Hausser and Zaefferer's approach cannot directly distinguish between sentences that answer a questions and others that do not.

Ingo Reich argues convincingly that one should rather assume that a 'canonical' answer to a *wh*-question is always a full sentence and that a constituent answer is derived from it by elision of 'superfluous' material (Reich, 2003). We will shortly present Reich's arguments for assuming that a constituent answer is in fact an elided version of a sentential answer and then present his rule that licences the correct elisions. It should be noted that the claim is not that it would be impossible to handle these issues in a categorial framework of questions and answers but rather that they could only be handled by assuming a number of additional mechanisms. Reich therefore argues that his solution, which covers all observations, is preferable due to its descriptive economy.

Sentence modification. Reich notes (Reich, 2003, 25) that constituent answers can be modified by sentence modifying adverbs, for example:

(3.29) Who will win the election?
Probably Bush.

This cannot easily be explained when a constituent answer is assumed, as that would not be of the required category S that a sentence modifying adverb (S/S) assumes. This problem can, in principle, be solved by type coercion mechanisms. However, these become increasingly complicated when multiple *wh*-questions are to be handled correctly (Reich, 2003, 25).

Note that this observation also presents an argument against Hausser and Zaefferer's 'full stop' rule: Here, one would have to assume that the constituent answer *first* combines with the full stop to form a sentential type and *then* with the sentence modifying adverb. That seems unlikely for (3.29) and impossible for '*Bush, probably.*' where syntactic material intervenes between the constituent answer and the full stop.

Case marking. The constituent answer must have the same case as the question pronoun. This is, of course, more noticeable in languages with overt case marking. Consider the following German example.

(3.30)	a.	[*Welchen*	*Studenten*]$_{\text{dat}}$	*hast*	*du*	*eine*	*Eins*	*gegeben?*
		(To) which	students	have	you	a	one	given?

b. To which students have you given an A?

c. [*Allen Studenten*]$_{\text{dat}}$.
(To) all students.

d. * [*Alle Studenten*]$_{\text{nom/acc}}$.
All students.

e. To all students.

f. ? All students.

In German, (3.30d), where the constituent has the wrong case, is plainly ungrammatical. In the English translation, we would judge that there is also at least a preference for (3.30e) over (3.30f) as an answer to (3.30b). Reich notes (Reich, 2003, 23) that the ungrammaticality of examples like (3.30d) can more easily explained when one assumes an elided sentential answer. Case assignment is generally assumed to be done through syntactic rules by the governing word. If one assumes a constituent answer, it is not obvious what word should govern this (independent) constituent.

Reflexive/reciprocal pronouns. This point is closely connected with the preceding one: Reich observes (Reich, 2003, 23–4) that a constituent answer may consist of a single reflexive or reciprocal pronoun. It is generally assumed that these are only licenced when *c*-commanded by their antecedent. It is difficult to see how *c*-command can be assumed for a single constituent.

(3.31) Who do John and Mary love?
Each other.

These points argue strongly for assuming sentential answers to *wh*-questions. But how then can constituent answers that seem to be a prevalent and even preferred way of answering *wh*-questions be accounted for? Reich suggests that these are derived from the full sentential answer form by elision. He observes that in a suitable answer to a *wh*-question, all constituents that correspond to a *wh*-phrase in the question must be focussed (and thus generally carry focus accent). The sentential answer in (3.31) could be rendered as follows (F again marks a focussed constituent):

(3.32) John and Mary love [each other]$_{\text{F}}$.

Reich then proposes a rule that states that from an ‘underlying’ sentential answer to a *wh*-question, all unfocussed material (and thus especially all material that does not correspond to a *wh*-phrase) can be elided. In the example, the elision of all unfocussed material from (3.32) leads exactly to the expected constituent answer in (3.31). (Reich, 2003, 26–8)

These different arguments seem to support the view that a ‘pure’ constituent view of answers cannot be sufficient, especially as it cannot explain why both constituent and sentential answers can be used to answer *wh*-questions. It rather seems necessary to assume an approach like Reich’s that allows to relate constituent answers and sentential answers and that accounts for the ‘sentential nature’ of constituent answers from a syntactic viewpoint. Note, however, that this also means that from a *processing* point of view (given a question and a constituent answer, decide whether the question is answered) some method that can handle the elision to arrive at a full semantic representation of the answer is required – which may, in practice, even bear some similarity to the solutions suggested above.

This point is also important with regard to answer presentation in QA systems. A systematic method for *generating* constituent answers from a full sentential answer (namely a sentence matching the question) can be specified: It works exactly by eliding all material from the answering sentence that is already present in the question. This is exactly the approach that we use (6.4.3.2). Note that this approach must be based on syntactic representations of questions and answers that allows a comparison between the two.

Critique 2: Proliferation of Types. The second argument that has been brought to bear against the categorial approach is that a large number of different question (and answer) types must be assumed: On the one hand, NP questions, adverbial questions and yes/no-questions are distinguished by type. On the other hand, for multiple *wh*-questions, each combination of *wh*-phrases must be assigned its correct type.

This disadvantage is especially marked when one considers embedded interrogatives (cf. 3.2.1.3): Most words that take embedded interrogatives as their complement do not further distinguish between different types of questions. When assuming a categorial approach to question semantics, one would be forced to postulate different readings of question-embedding verbs for every possible question type.

Note, however, that there is a class of words (‘emotive factives’ such as *‘be amazing/surprising’*) that take only embedded *wh*-questions as complements (cf. the ungrammaticality of *‘*It’s amazing whether they serve breakfast.*’) and another (‘dubitatives’ such as *‘to doubt’*, *‘to question’*) that only takes embedded *whether*-interrogatives (hence the ungrammaticality of *‘*I doubt what they serve for breakfast.*’. Karttunen admits that his (propositional) approach cannot account for this fact, either (Karttunen, 1977, 383–4).

Krifka suggests that this and related problems can be accounted for by lexical rules (similar to meaning postulates used in Montague grammar) and special interpretation rules (Krifka, 2001, 313–6). He shows how the problem can be addressed and that it is not unsolvable for a categorial approach. Proponents of the other approaches (especially the propositional one) have stressed that an account that can do without these additional ramifications would be preferable.

Discussion. The categorial approaches are built around the central idea that a number of phenomena concerning questions and answers can be elegantly explained by representing questions as (abstract) lambda terms whose type reflects that of question and answer so that the representation of an *answered* question can be derived by functional application of question representations to answer representations. It thus incorporates Hamblin's second postulate (cited above, p. 48) and, with reservations and not at all in the 'basic' approach by Keenan and Hull, the first one.

Two main points of criticism have been described above. These are both addressed by the propositional approach where questions are assumed to be denoted by the set of their possible (or sometimes true) answers. We will describe this approach in the next section.

3.2.2.3 Propositional Approach

In the propositional approach (Hamblin, 1973; Karttunen, 1977; Karttunen and Peters, 1980), it is assumed that questions are represented as the set of propositions that constitute possible answers to them. For declarative sentences, which are analysed as propositions, the answerhood relation then boils down to checking whether the representation of the answer is a member of the question representation, as that is exactly the set of propositions that are possible answers. Note that the constituent answers to *wh*-questions are not touched upon in any of the work reported here under the propositional and partition headings. It seems, however, reasonable to assume that they can be accounted for by an elision mechanism like Reich's that was summarised in 3.2.2.2. The problems of the categorial approach that were summarised in 3.2.2.2, namely the problem of explaining sentential answers and the need to assume a large number of different question types and, consequently, a large number of readings for question embedding words, do not extend to the propositional approach, as it assumes one single semantic type for all questions, namely sets of propositions.

Let us take a look at an example of how a question would be analysed in Lauri Karttunen's approach (Karttunen, 1977). Karttunen shows how the propositional approach can be integrated into the framework of Montague grammar

(Montague, 1974) by giving a set of categorial rules and semantic translation rules for different types of questions.

Note that a question is represented by its true answers at a given index in Karttunen's approach, not by all possible answers. This view has been challenged, among others by James Higginbotham. In Higginbotham (1996), he gives an account of what changes would need to be made to Karttunen's question representations to fix this (and a number of other problems), while noting that changing the derivation process that assigns the representation to a question would not be straightforward (Higginbotham, 1996, 368–71). We will not further pursue this question here.

(3.33) a. Who sleeps?
b. $\lambda p.\exists x[p = \hat{}\text{sleep}'(x) \wedge \check{}p]$
c. $\{\hat{}\text{sleep}'(\text{mary}'), \hat{}\text{sleep}'(\text{john}')\}$
(adapted from Karttunen, 1977, 394

Note that (3.33) has been adapted: Karttunen *only* gives definitions for embedded questions. He assumes some process that ensures that a direct question ϕ receives a translation that explicitly encodes the interrogative act that could be paraphrased as *I ask you (to tell me) ϕ* (Karttunen, 1977, 383). In (3.33), we assume a direct representation.

The question (3.33a) translates into (3.33b): The answers are constrained to be propositions of the form $\hat{}\text{sleep}'(x)$ and their extension must be true at the given index ($\check{}p$). This is just the set of propositions at the given index that form a true answer to the question. In a world where John and Mary sleep (and where no one else sleeps), this would be the set of propositions given in 3.33c, which would be representations of the answers *'Mary sleeps.'* and *'John sleeps.'*, respectively.

It should be noted that it is not formally spelled out how the actual matching of question representation and answer representation can be done. Hamblin describes it as follows:

> Semantically, an answer to a question on a given reading is any statement whose denotation-set on a suitable reading is contained in that of the question. (Hamblin, 1973, 52)

Hamblin also notes that there may be additional syntactic well-formedness conditions for QA pairs (ibid.). In our opinion, some additional refinement would be needed. Consider the (conjoined) proposition ($\hat{}\text{sleep}'(\text{mary}') \wedge \hat{}\text{sleep}'(\text{john}')$, 'that John sleeps and that Mary sleeps') that we would consider a good answer to (3.33b) and that would yet not be a (simple) subset of (3.33c). We think that the actual answerhood relation would thus have to be spelled out in some more detail (cf. also Shan and ten Cate, 2002).

Critique. The main criticism against the propositional approach is that it does not explain indirect answers to questions. James Higginbotham and Robert May characterise an indirect answer as one that excludes possible other answers only through auxiliary beliefs of the dialogue partners (Higginbotham and May, 1981, 42). For example, if questioner and answerer agree in the belief that *‘If Mary talks, then John sleeps.’*, then *‘Mary is talking.’* should count as possible indirect answer to (3.33a): Both hearer and answerer believe that it is not possible that Mary talks and John is awake, so by assuring that Mary talks, the question whether or not John sleeps is resolved.

This observation cannot be accounted for, if the only possible answer propositions at the given index are enumerated by a set such as (3.33c).

Note that Markus Egg has shown how ideas from the partition approach that allows indirect answers can be integrated into the propositional approach (Egg, 1998, Appendix A). As the result is very similar to the partition approach itself (cf. 3.2.2.4), we will not discuss this possibility here any further.

Discussion. The propositional approach assigns uniform representations to different question types, namely sets of propositions that count as possible answers. It thus incorporates both the first and the second Hamblin postulate (p. 48).

It does not allow indirect answers to questions, as only propositions that directly answer the question are predicted as possible answers. The same applies for the different categorial approaches (3.2.2.2): answerhood is only predicted to hold if question and answer representation are equal (modulo variable bindings). This gap is filled by the partition approach, which will be described in the following section.

3.2.2.4 Partition Approach

The probably most influential recent approach to the semantics of questions and answers is the partition approach (Groenendijk and Stokhof, 1997, 1984; Higginbotham, 1996; Aloni, 2001). It is based upon a possible-worlds semantics. Partitions are sets of possible worlds (indices). They are used to represent the meaning of questions; we will describe the details in the following.

The semantic definition of questions and answers in the partition approach is based on the notion of possible exhaustive answers. Such an exhaustive answer for the question *‘Who sleeps?’* would, for example, record for every individual whether or not they sleep. However, this is assumed to be too strong a notion of answerhood to represent typical natural language answers. Therefore, a pragmatically motivated definition of partial answerhood is added. It allows to

It is raining.
It is not raining.

Is it raining?

No one walks.
Only a walks.
Only b walks.
Only c walks.
Only a and b walk.
Only a and c walk.
⋮
Everyone (=a, b and c) walks.

Who walks?

Figure 3.1: Partitions for two questions (adapted from Groenendijk and Stokhof, 1984, 146)

capture the fact that any reply *excluding at least one possible answer* will count as a partial answer. This allows to explain a number of phenomena connected with questions and answers, among them that of indirect answers that the categorial and propositional approaches cannot handle. We will only shortly and informally sketch the approach here; Groenendijk and Stokhof (1984); Higginbotham (1996) give formal accounts.

Partitions. As mentioned above, the partition approach extends a possible-worlds semantic, defined as usual. A question is assumed to partition the logical space. That means that every index in a given model is assigned to *exactly one* partition (set of indices). How this partitioning is done, is best shown by the two examples in fig. 3.1. The intuition is that the partitions correspond to the (complete) set of all possible exhaustive answers.

The left-hand side of the figure shows the two partitions for the question *'Is it raining?'*: One partition contains the indices where the extension of the proposition *that it is raining* is true (i. e., it is raining in each of these worlds), the other those where the extension is false (i. e., it is not raining). For yes/no-questions, there are exactly two partitions.

The right-hand side of fig. 3.1 sketches the possible partitions for the *wh*-question *'Who walks?'* ($?x.walk'(x)$, cf. 3.2.2) in a model with exactly three individuals, *a*, *b* and *c*. Here, the partitions are distinguished by which individuals the extension of the *walk* predicate contains: The first partition contains all indices where it is empty, i. e., where no one walks. It is followed by the partitions in which exactly one individual walks, then the possible combinations of two individuals and then the indices where all three individuals are walking.

The sense of a question is the set of partitions, constructed as sketched above to partition all indices exactly according to the possible exhaustive answers. All indices at which the extension of the question proposition are the same exactly make up one partition.

Answers. A reply to a question is then evaluated in the following way: A semantic representation of the reply is derived. The representation is then used to select indices with which its extension is compatible. Thus, in general a number of indices will be excluded as not compatible with the given reply. The reply is said to give an exhaustive answer when it only 'leaves' indices that all lie within the same partition, i. e., when only one single partition is left. This would, e. g., be the case for the answer *'Only a (and no one else) walks.'* for the right-hand side of fig. 3.1.

As mentioned above, this does not seem a natural answer in most contexts: Typically, the questioner will be content with a partial answer. Partial answerhood can be elegantly accounted for in the partition approach. The requirement that a reply must exclude all but one partition is simply changed to the following: To be a partial answer, a reply must exclude *at least one* partition. The intuition behind this is as follows: By excluding one partition, the answerer has provided information to the questioner that is pertinent to the question, as the questioner now can exclude one possible answer. An answer like *'a is not walking.'* would be considered a partial answer to the question, as it excludes all partitions in which a *is* walking.

To distinguish a true from a false answer, one simply has to check whether the 'real' world (represented simply by its index) lies within the set of indices that is compatible with the answer. If so, the answer is true, otherwise false.

This approach can also explain indirect answers: As answerhood is defined via (in)compatibility with sets of worlds, direct and indirect answers can, in fact, not even be distinguished. Consider an indirect answer like *'It rains.'* if it is known that *a* never walks if it rains: As this answer is incompatible with all worlds where *a* walks, it excludes a number of partitions from fig. 3.1 (namely all where *a* walks) and thus forms a (partial) answer. This 'built-in inference' also allows to directly explain that *'b is driving.'* would count as an indirect partial answer (that is, when both questioner and answerer agree on the fact that someone who is currently driving cannot possibly at the same time be walking).

We consider this point as very important, as the introduction of indirect answerhood allows to systematically integrate inferences as a way of arriving at answers into the framework: An indirect answer can be dissimilar from the question, additional knowledge in the form of inference rules (axioms) can be adduced to arrive at a direct answer. We will use this idea as the core for our QA framework (cf. 3.2.4.2).

Note that it may, in general, be necessary to explicitly model the questioners and the answerers knowledge and beliefs and especially their shared knowledge. Imagine the answerer replying with *'B geht.'* (German for *'B walks.'*). This may count as an answer to the question above, but only if the questioner knows German and the answerer knows the fact. In Groenendijk and Stokhof (1984), the basic account given here is extended to explicitly handle this by modelling doxastic and epistemic sets (to represent beliefs and knowledge, respectively). See also Groenendijk (1999), where in addition to 'pure' answerhood, some formalisation of the Gricean Maxims (see 3.2.3.2 below) is used to describe a number of phenomena in an interrogation scenario (chosen as an example for a more or less formalised scenario with clearly distributed roles).

We will concentrate on indirect answers that can be resolved by reference to linguistic knowledge. For example, we will assume that *'Lee Harvey Oswald slew John F. Kennedy.'* to (indirectly) answer the question *'Who murdered John F. Kennedy?'*. We assume that both questioner and answerer know the meaning of *'slay'* and *'murder'* and therefore that the two are synonymous, so that the necessary inference can be drawn by both questioner and answerer. We will not be concerned with the question of how believes could be modelled, but rather take linguistic knowledge as safe common ground.

Note also that Groenendijk and Stokhof use a more constrained, pragmatic definition of indirect answers (Groenendijk and Stokhof, 1984, 162–165): An indirect answer must *not* enable the questioner to directly infer an answer. It rather provides him with information that may help in finding the answer, for example by asking someone else for different information. For example, if the question *'Does a walk?'* is answered by A with *'a always walks if b walks. [But I have no idea whether that is currently the case.]'*, the questioner may next ask someone else (B) *'Does b walk?'*. If B answers with *'yes'*, the questioner can then infer that *a* also walks. We will not use this restricted definition of an indirect answer, but rather the more general one described above.

Comparing Answers. Groenendijk and Stokhof also give a definition that allows to compare two different answers for a given question (Groenendijk and Stokhof, 1997, 1095–6). The core of this definition is to gauge the informativeness of an answer with respect to the question: If one answer excludes exactly all the partitions that the other answer excludes and then at least one more, it is considered more informative (as it restricts the remaining possibilities more strongly).

Answers that are equal with respect to the partitions that they exclude, a (weaker) form of comparison based on entailment between the answers themselves is used. Two answers whose partitions do not stand in some subset relation cannot be compared.

Choice Interpretations. It has been shown that the basic version of the partition approach, though a powerful tool, still has certain intrinsic problems that make extensions necessary. Higginbotham (Higginbotham, 1996, 376–379) suggests that a further type raising is necessary to uniformly handle certain cases of quantifying into questions with partial answers. The question *'Where can I find two screwdrivers?'*, for example, has a reading that can be answered by (the answerer) arbitrarily choosing exactly two from all screwdrivers and then giving their whereabouts, generally referred to as *choice-reading*. Higginbotham accounts for these cases by raising the type of answers to sets of partitions through a rule that builds up these sets according to the semantics of the quantifier involved.

Higginbotham defines partitions somewhat differently in his approach to start with: He does not assume them to be relations between indices, but rather exhaustive sets of sets of proposition: Each of its elements states for every individual in the domain whether or not the question proposition holds for it, together they exactly exhaust the space of possibilities for the question proposition. For details see Higginbotham (1996). These different types of partitions can directly be translated into each other.

The representation of the questions *'Where can I find two screwdrivers?'* would thus form sets describing, for any possible combination of two screwdrivers, the possible answer to the question. A relevant answer can then be given by restricting the set of possible partitions for *any one* of these sets. Note that questions with more than one quantifier require repeated type raising in this approach.

Questions of Identity. Maria Aloni notes that the partition approach runs into problems when questions of identity are to be handled (Aloni, 2001). Consider her following example:

(3.34) Who is Eduard? ($?x\, x = e$)
Aloni (2001, 7)

The partition approaches assume (and need to assume, cf. Aloni, 2001, 6–9) rigid designators, i. e., individual constants that do not change over indices as the basis for setting up the partitions. Thus, from (3.34), only a single possible partition is ‘generated’, as only Eduard (e) is identical to himself. In Groenendijk and Stokhof’s account, such a question (viz. one that forms only a single partition) would be considered tautological and thus semantically empty. To solve the problem, Aloni introduces what she calls conceptual covers, i. e., individual concepts that satisfy a number of prerequisites, especially that they totally cover the individuals at every index. Partitions are then formed using conceptual covers instead of the individual constants. Intuitively, the effect is that, by using different covers, different ways of referring to the individuals are provided and thus that different perspectives of a situation can be formally described. This adds the possibility of referring to an individual by *any* scheme that uniquely identifies it – which then allows to identify proper (non-trivial) answers to identity questions as 3.34. For details see Aloni (2001). We will return to phenomena that have to do with identity, but also with predication in questions and answers below (3.5.2.4, 3.5.2.5).

Discussion. The partition approach can be seen as a way of spelling out all three of Hamblin’s postulates (see p. 48), as Groenendijk and Stokhof stress, cf. Groenendijk and Stokhof (1997, 1076-8). It especially integrates the second postulate, which requires that the possible answers of a question be an exhaustive and mutually exclusive set of possibilities. This is an important difference from the proposition set approach. We would therefore not like to follow Krifka who subsumes Groenendijk and Stokhof’s partition approach under the proposition set approaches, cf. Krifka (2001, 291–2). Partitions are exactly a formal spell-out of this idea.

The description that the partition approaches arrive at have a number of desirable properties, namely:

- They are based on the well-defined notion of exhaustive answers to a question,
- Partial answers to question are defined on this basis,
- Both direct and indirect answers can be distinguished from other replies to a question,
- Answerhood is defined independently from the truth or falsity of a question at a given index,
- The truth or falsity of an answer can simply be checked,
- With suitable extensions, questions of identity can also be handled and

- The uniform representation of questions as partitions allows to define semantically interesting relations between them such as entailment or equivalence (via inclusion or equivalence of the respective sets of partitions) similar to the corresponding notions for statements.

Still, a number of issues remain open, as Groenendijk and Stokhof point out (Groenendijk and Stokhof, 1997, 1108–22). Most of these have to do with adequacy of description:

First, Groenendijk and Stokhof note that the notion of assuming one single true exhaustive answer may be too strong. The partition approach is based on the central idea of exhaustive, mutually exclusive answers that together cover the whole space of possible answers, each represented by a partition. Groenendijk and Stokhof call this a strongly exhaustive approach and distinguish it from a weakly exhaustive approach: While in a weakly exhaustive approach an exhaustive answer includes all positive examples, in a strongly exhaustive approach an exhaustive answer must additionally give an exhaustiveness claim (e. g., by listing all positive examples and by asserting that no other positive examples exist). Groenendijk and Stokhof show that an exhaustiveness claim requires that the domain is completely known. They state that by simply listing all positive and all negative examples, no strong exhaustiveness claim can be made. Thus, Belnap's completeness claim (cf. 3.2.2.1) does not constitute a strong exhaustiveness claim. This is especially important for the truth conditions of embedded questions, Groenendijk and Stokhof claim. The proposition expressed by *'that John knows who comes to the party'*, for example, should only become true at indices where John does in fact strongly exhaustively know which persons in the domain come to the party (and not only based on his beliefs). For details see Groenendijk and Stokhof (1997, 1109–11).

Groenendijk and Stokhof list a number of reasons why a distinct mention-some interpretation (cf. 3.2.2.1) should be assumed, which is, however, incompatible with the strong exhaustiveness of the partition approach and closer in spirit to Karttunen's approach, cf. 3.2.2.3. One important reason is that the mention-some interpretation of questions seems to be incompatible with negative answers. For details, see (Groenendijk and Stokhof, 1997, 1111–3).

Then, there are a number of phenomena on the border between semantics and pragmatics, such as the issues of presuppositions of questions and context-dependency of questions and answers (both will be addressed below in 3.2.3, see also Groenendijk and Stokhof, 1997, 1119–21).

This leads Groenendijk and Stokhof to suggest that only an integrated and flexible approach can properly handle all issues: Different representations for questions and answers must be assumed, depending on phenomena to be covered in a specific question (or answer). Type-coercion rules would map between the different representations (Groenendijk and Stokhof, 1997, 1115–20). For example, a question would receive both a mention-some *and* a mention-all representation, from which the one better fitting the current context is then selected. Handling such phenomena ascribed to pragmatics could be done via a close integration, possibly in a framework of dynamic semantics (Groenendijk and Stokhof, 1997, 1120–2)

As mentioned above, to correctly handle focus phenomena, one might want to extend this integrated approach further by adding mechanisms from the structured meaning approach or a similar framework. For an approach along these lines (a combination of the structured meaning framework with proposition sets for handling questions and answers), see Reich (2003).

3.2.2.5 Additional Issues

We will shortly address three additional issues that are discussed in different work on the semantics of questions, namely the distinction between 'open' questions and 'informative' questions, the question whether and how *wh*-phrases interact with quantifiers and other scope-bearing material and the assumption that questions carry general presuppositions.

Open Questions. Groenendijk and Stokhof introduce an important distinction, namely that between (what they call) informative and open questions (Groenendijk and Stokhof, 1997, 1108–9). They note that the partition approach (as all other approaches discussed here) does not cover all questions in natural language but only a limited subclass that they call informative questions. Informative questions are characterised by the fact that all possible direct answers can be directly constructed from such a question. In the partition approach this is, for example, done by exhaustively partitioning the given indices based on the extension of the proposition contained in the question (8.3.4).

Open questions are different in that possible answers cannot be 'read off' from the question. Groenendijk and Stokhof's example is the question *'What are questions?'*. They claim that there is no 'simple', 'pre-set' answer to this question. An open question rather seems to call for a creative process. An answer may, for example, be given by a whole essay rather than a single proposition. Groenendijk and Stokhof also note that the notion of truth or falsity of an answer to an open question often seems to make little sense, and that other concepts (such as 'good' or 'helpful') seem more intuitive to describe such answers. They come to the conclusion that so far no useful theory of open questions has been established.

There is currently a growing interest in open questions in QA (cf. 2.2.1.3, Burger et al., 2001; Maybury, 2002). There is currently no agreed-upon scheme how such open questions should be answered.

We would also like to point out that it is difficult to identify such questions at all: First, it seems to us that there is no clear-cut distinction between informative and open questions as Groenendijk and Stokhof imply. There are certainly questions that seem purely informative (*'On what date was John F. Kennedy killed?'*), while others seem purely open (*'What are questions?'*). But there are other questions that seem to lie more or less in the middle (maybe a question such as *'Why is Christopher Columbus famous?'*, that may be answered in 'informative mode' by *'Because he discovered America.'* but where one could also write a whole essay).

This leads us to out second point, namely that it does not seem possible to find out from the question itself whether or not it is an open question. There seems to be a certain tendency, however, that so called definition questions (*'What are questions?'*, *'Who was Christopher Columbus?'*), *why*-question (*'Why is English the official language in the US?'*), *'how'*-question (*'How do I construct an atomic bomb?'*) and *what-if*-questions (*'What would happen if Switzerland would declare war on Island?'*) rather ask open (and thus more challenging) questions (cf. Burger et al., 2001; Maybury, 2002).

We will restrict ourselves to informative questions. We will present some thoughts on using linguistically informed QA for answering open questions by breaking down certain types of open questions (for example, biographical questions) into simpler questions and joining the answers below (8.3.4).

Questions and Quantification. It has been noted in work on the semantics of questions that there seems to be a certain interaction between question phrases and quantifiers within questions. These phenomena have sometimes been accounted for by assuming a quantifier-like status for *wh*-phrases (see especially Karttunen, 1977; Karttunen and Peters, 1980; Higginbotham, 1996). In the literature, they are generally referred to as 'quantifying into questions'. Consider the following example:

(3.35) What did everybody say?
Higginbotham (1996, 376, his (68))

Two readings seem to be available for this question, namely one where everybody said the same thing (wide scope of *'what'*) and a second reading that asks the answerer to list all (contextually relevant) persons and tell, for each of them, what it is that (s)he said.

This second reading is closely related to the so-called pair-list reading (Krifka's matching questions), as in the following example:

(3.36) Who said what?
Higginbotham (1996, 377, his (72))

Higginbotham remarks that (3.35) and (3.36) share the same complete answers, but differ in their partial answers. He shows this by the different truth conditions of the respective embedded versions:

(3.37) a. I have some information about who said what.
b. I have some information on what everybody said.
Higginbotham (1996, 377, his (73) and (74))

He claims that, for a speaker to (truthfully) assert (3.37b), (s)he must know at least one utterance from everybody, for (3.37a) can be truthfully asserted when knowing nothing at all about what one (or more) person(s) said.

It has therefore been suggested that by treating *wh*-phrases like quantifiers, the same mechanisms can be employed to account for ambiguities as that exhibited by (3.35), cf. Karttunen (1977); Karttunen and Peters (1980). Thus, *wh*-phrases are assumed to carry scope and to interact with other scope-bearing material to produce scopal ambiguities.

Other authors have pursued a different approach. They suggest that cases like these that involve quantification or pair-list readings should rather be accounted for by a functional (or relational) approach. The first work in this vein was done by Elisabet Engdahl (Engdahl, 1986). Engdahl adduces a number of arguments against the 'quantifier' solution. Her suggestion is to assume that answers characterise (and questions ask for) functions. The arity of the functions depends on the number of quantifiers and *wh*-phrases in the question; each of them 'binds' one place in the respective function. As Ginzburg and Sag remark, this is somewhat similar to Skolemisation, cf. Ginzburg and Sag (2000, 153–4). In (3.35), for example, an answer could be given by a function that provides a mapping from all (relevant) individuals to their respective utterances.

Thus, the answerer can define the function by listing every individual in its domain and the respective value of the function. This corresponds to a pair-list answer. Engdahl accounts for the first reading of (3.35) through the answerer providing a constant function (everybody said the same thing). In addition, her approach allows an interesting third possibility, namely that of so-called functional answers that 'intensionally' define the function. This allows explaining why, for example, '*Everybody said what they considered most important.*' would count as a good answer. Other approaches cannot explain this type of answer at all. Note that the predominant example in the literature, used also by Engdahl, is *'Whom does every Englishman admire most?'* –*'His mother.'*.

A number of other authors employ different versions of the functional approach, cf. Krifka (2001); Ginzburg and Sag (2000); Groenendijk and Stokhof (1997). The following example (3.38) shows a functional interpretation in Krifka's structured meaning approach (thus, a focus and background component is given, cf. 3.2.2.2).

(3.38) a. Who read what?

b. i. $\text{FUN(R)} = \lambda f \lambda x[x \in \text{DOM}(f) \rightarrow R(\langle x, f(x)\rangle)]$, the set of functions f such that every x in the domain of f stands in R-relation to $f(x)$

ii. $\text{FUN}'(A \times B)$ = the set of functions from A to B

c. $\langle \text{FUN}(\lambda\langle x,y\rangle[\text{read}'(y)(x)]), \text{FUN}'(\text{person}' \times \text{thing}')\rangle$, where $\text{FUN}(\lambda\langle x,y\rangle[\text{read}'(y)(x)]) = \lambda f \forall x[x \in \text{DOM}(f) \rightarrow \text{read}'(f(x))(x)]$, the set of functions f such that every x in the domain of f read $f(x)$, and $\text{FUN}'(\text{person}' \times \text{thing}')$ = the set of functions from PERSON to THING.

d. Mary *'Die Kinder der Finsternis'*, and John *'Das Totenschiff'*.

e. $f : \{M, J\} \rightarrow \{KF, TS\}$,
$M \rightarrow KF$
$J \rightarrow TS$

(Cited after Krifka, 2001, 312, his (87)–(90)

Note that, again, we are more interested in the different phenomena that have been described than the proposed solutions. Let us thus keep the three possibilities, namely 'everybody said the same' (constant function), 'list for everyone what they said' (pair-list) and 'summarise' (intension) in mind. We will return to them below (3.2.4).

Questions and Presuppositions. There is a discussion in the literature about questions and presuppositions (Groenendijk and Stokhof, 1997; Ginzburg, 1995a; Belnap and Steel, 1976; Keenan and Hull, 1973).

They are especially used in some approaches to describe selection size requests (cf. 3.2.2.1). If the correct answer does not satisfy the selection size request, that is assumed to lead to presupposition failure and thus to the question not having a (true) answer. The following example is taken from Higginbotham and May (1981).

(3.39) a. Which person went to the store?

b. # John and Mary went to the store.
Higginbotham and May (Cited after 1981, 43–44)

The question expressed by (3.39a) is assumed to carry the presupposition that not more than one person went to the store. The answer (3.39b) violates this presupposition and is thus incompatible with the question.

While some researchers assume that presuppositions are associated with questions (Belnap and Steel, 1976; Keenan and Hull, 1973, especially), others have argued that it is rather pragmatic aspects that play a rôle (Groenendijk and Stokhof, 1997; Ginzburg, 1995a).

We will not further be concerned with this discussion here. We note, however, that there are cases in which the questioner seems to make assumptions about the answer (especially regarding selection size). If these assumptions are wrong, then a cooperative answerer should explicitly address the fact. For example, one might respond to (3.39a) with '*John and Mary* both *went to the store.*' See also 3.2.3.3.

3.2.3 Pragmatic Aspects

In this section, we will summarise some aspects of questions and answers that are associated with pragmatics.

The first point to make is that interrogative acts cannot only be performed by the use of interrogative sentences. In fact, the utterance of sentences in all different sentence moods may be taken, in the right context, to perform an interrogative act. On the other hand, not all uses of interrogatives are actually posing a question (consider, for example, rhetorical questions). Such indirect speech acts are extensively discussed in speech act theory (e. g., Searle 1969).

Then, we will shortly introduce the Gricean Conversational Maxims (Grice, 1989). These maxims (and the underlying cooperative principle) are assumed to 'rule' all human interactions and allow, by specialisation of the maxims, to derive specific requirements for conversational settings. We will summarise Webber (1986), which deals with cooperative responses in the context of NLIDB systems.

The third important point is that the question of what forms a good answer to a certain question is dependent on what one might call situational factors, such as the (assumed) degree of information on (or familiarity with) the question's topic that questioner and answerer possess, or the (perceived) goals of the questioner in asking the question. We will introduce work by Jonathan Ginzburg where he points out that these are essential to a fuller account of questions and answers and that they cannot be properly addressed by any of the semantic accounts described in the previous section.

3.2.3.1 Questions and Speech Acts

As we already noted above, questions can not only be formulated as interrogatives, but also by sentences in other sentence moods. (Re-) consider the following examples. All of these can be taken (at least in the proper circumstances) to express a question:

(3.40) Please tell me whether Fidel Castro had John F. Kennedy killed. (Directive matrix sentences, =3.10b)

(3.41) I wish I knew who murdered John F. Kennedy. (optative matrix sentence)

(3.42) I want to know who murdered John F. Kennedy. (declarative matrix sentence)

This often-observed disparity between the sentence mood and the intention of the speaker has been one of the subjects of study in speech act theory. In other words: How can the fact be accounted for that questions are not always expressed by interrogatives, statements always by declaratives etc. We will only shortly sketch here how John Searle's early and influential theory accounts for these points (Searle, 1969).

Speech Acts. Searle assumes that every speech act consists of several simultaneous acts. We will only be concerned with the so-called illocutionary act that forms the central aspect of the overall speech act. It is assumed to consist of the illocutionary force F and the proposition P of the utterance, and be of the form $F(P)$.[5] The illocutionary force determines the overall type of the illocution, for example a statement (*'It is the case that P.'*) or a question (*'Is it the case that P?'*). There are a number of devices indicating illocutionary force, i. e., indicators in the utterance that help to determine the intended force. Sentence mood (and the surface phenomena associated with it, such as word order, mood of the matrix verb, use of interrogatives etc., cf. 3.2.1) is an important indicator. In cases where a question is expressed by an interrogative, the force of the illocution is that of a question, indicated by the suitable indicators, allowing the hearer to infer that a question is indeed asked.

There are, however, cases where the performed illocution and the illocution indicated at the surface level of the utterance differ. This kind of speech act is called indirect speech act. In such cases, the hearer must note that the indicated

[5]Note that when assuming a non-propositional semantics of questions (as all the semantic approaches above did, cf. 3.2.2), this must be adapted accordingly. See Groenendijk and Stokhof (1997, 1073–4) for a discussion.

illocution is not really intended and infer the ‘proper’ one. Such indirect speech acts are shown by the examples (3.40) through (3.42) above.

It should be noted that questions are often analysed as a subtype of directives in speech act theory, namely as a request that the hearer perform a future speech act which is an answer to the question. This could be paraphrased as *‘I (hereby) ask you to answer (the question) Q’* (Groenendijk and Stokhof, 1997, 1070). We have assumed a separate type of speech act, namely an interrogative act, for convenience’s sake, cf. 3.2, see also D’Andrade and Wish (1985).

From this description, it follows directly that not all utterances featuring an interrogative need to be interrogative acts, i. e., ask a question. We will introduce two frequent conventional uses of interrogatives in indirect speech acts: rhetorical questions and polite requests.

Rhetorical questions are a commonly used device where an interrogative is used to make an assertion. In the narrow sense of the term, a rhetorical question asserts the *negation* of the underlying proposition. Consider the following example:

(3.43) Who could deny that education is important?
Implied assertion: There is *nobody* who could deny that education is important.

Polite requests can be formulated using interrogatives in a number of languages. They express a directive:

(3.44) Could you pass the salt?
Implied directive: Please pass the salt.

It is thus not sufficient to only look at interrogatives to properly recognise and analyse questions, as other kinds of utterances may be used to transport questions. Interrogatives, on the other hand, must be examined for possible indirect uses, such as rhetorical questions or polite requests. However, we currently restrict system input to interrogatives.

Questions in Speech Act Classifications. A number of typologies of speech acts have been suggested. They extend and often modify the ‘basic’ categories of Searle (1969) and provide more fine-grained classes.

These typologies have mainly been used to describe and annotate speech acts in human interaction, e. g., in doctor-patient conversation. For an overview of different speech act theories and their respective classifications, see D’Andrade and Wish (1985).

In the typologies, interrogative acts are further differentiated, mostly according to the questioner’s intention in asking them. The most interesting types of interrogative acts from a viewpoint of QA are those that request information, evaluation and interpretation from the answerer (cf. D'Andrade and Wish, 1985, 240, 245). Other types, such as requests for confirmation or clarification, may be needed in full dialogue-style QA. We do not expect several others, such as requests for sympathy or action, in the context of human-computer QA. Requests for information are the type of interrogatives most directly connected with QA; we will therefore concentrate on these.

Wendy Lehnert gives a classification of questions in the context of QA (more exactly, QA for story understanding). We repeat her list of (mostly pragmatic) question types here and give one of her examples for each type (Lehnert, 1978, 52–77):

1. Causal antecedent (*'Why did John go to New York?'*)
2. Goal Orientation (*'For what purpose did John take the book?'*)
3. Enablement (*'How was John able to eat?'*)
4. Causal Consequent (*'What happened when John left?'*)
5. Verification (*'Did John leave?'*)
6. Disjunctive (*'Was John or Mary here?'*)
7. Instrumental/Procedural (*'How did John go to New York?'*)
8. Concept Completion (*'What did John eat?'*)
9. Expectational (*'Why didn't John go to New York?'*)
10. Judgmental (*'What should John do to keep Mary from leaving?'*)
11. Quantification (*'How many people are here?'*)
12. Feature specification (*'What colour are John's eyes?'*)
13. Request (*'Would you pass the salt?'*)

Lehnert argues that automatically identifying the type of a given question is far from being straightforward, as it often includes reasoning about the questioner's goals. Her suggested solution, which is mainly based on question representations in the form of Conceptual Dependency networks, is not usable in practical QA systems. We will discuss this issue below (3.3.1).

3.2.3.2 The Gricean Conversational Maxims

In his influential work in pragmatics, Paul Grice investigates general rules (or maxims) that are normally followed by all participants in a conversation (Grice, 1989)[6]. We will recapitulate the central points here. It should be noted that Grice's work does not make any special reference to questions and answers. But as they form contributions to a conversation they certainly fall within the scope of the work reported here.

Grice starts from what he calls the Cooperative Principle. He states that all participants in a conversation should observe this general principle.

> Make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged. (Grice, 1989, 26)

He continues to further elucidate this general principle by breaking it down into the following maxims that should be observed for any contribution to a conversation:

> [Maxims of Quantity:]
>
> 1. Make your contribution as informative as required (for the current purposes of the exchange).
>
> 2. Do not make your contribution more informative than is required.
>
> [Maxims of Quality:]
>
> 1. Do not say what you believe to be false.
>
> 2. Do not say that for which you lack adequate evidence.
>
> [Maxim of Relation:]
>
> Be relevant.
>
> [Maxims of Manner:]
>
> 1. Avoid obscurity of expression.
>
> 2. Avoid ambiguity.
>
> 3. Be brief (avoid unnecessary prolixity).
>
> 4. Be orderly.

[6]The part of this work that we report here was originally a series of lectures by Grice at Harvard University in 1967 (cf. Grice, 1989, v), first published as Herbert Paul Grice. 'Logic and Conversation'. P. Cole and J. Morgan (eds.). *Syntax and Semantics. Vol. 3: Speech Acts*. Academic Press, New York, 1975.

(Cited after Grice, 1989, 26–7, several omissions not marked, headings in square brackets added by me)

These maxims thus provide general guidelines for contributions to a conversation. They apply, of course, directly to answers to questions (as they form contributions to a conversation). Thus, when designing a QA system that simulates human question answering behaviour, these maxims should be taken into account. We will describe in the following section how types of responses for a cooperative NLIDB system can be derived.

3.2.3.3 Responses

Cooperative replies to questions have been investigated in the context of Natural Language Interfaces to Databases (NLIDB, cf. 2.1.2) in Webber (1986). Bonnie Webber gives a number of examples for (more or less) cooperative reactions that an answerer can show given a question in a certain situation. While she does not in all cases explicitly refer to the Gricean Maxims described above, this paper can be taken to spell out the cooperative principle underlying the Gricean maxims for many cases that can arise in the context of NLIDB.

One central concept of the work is that of a response. A response is a cooperative reply to a question that the user has posed. A response is taken to subsume all answers, but also all other helpful remarks that the answerer (here, the NLIDB system) can make.

We think that it is important to draw the attention to the fact that a system should be as cooperative as possible. We will repeat Webber's (non-exhaustive) list of what respondents may include into response for what reasons here.

1. The answer,
2. Additional information (or an offer to provide information) that R[espondent] believes Q[uestioner] may need in order to fulfill Q's plan,
3. Information (or an offer to provide information) that R believes Q may need to satisfy his goal (at some level), where the plan that R believes Q to have will not fulfill that goal,
4. Additional information that R believes *justifies* or *explains* that response so that Q will accept it,
5. Additional information or graphical presentation that R believes will clarify the response, so that Q will understand it,
6. Information intended to correct a misconception that Q reveals through his question, that interferes with R's ability to

answer that question or with Q's ability to correctly interpret R's answer,

7. Information intended to correct a misconception that Q reveals through his question, that doesn't interfere with R's ability to answer that question or with Q's ability to correctly interpret that answer, but that R feels responsible for correcting,

   [...]

10. Information as to R's attitude towards the answer,
11. Rejection of the question (often accompanied by support for such a seemingly uncooperative action).

Webber (1986, 379–81, my additions, her italics)

This definition of response helps to distinguish it from a direct answer on the one hand (1. above) and from 'unhelpful' replies to a question. We believe that this addition is useful in defining proper system reactions to a question. A system should especially react with the proper response whenever a question implicature turns out to be wrong: A cooperative system should not just ignore such a case but rather comment on it. We will return to this issue below (3.2.4.3).

3.2.3.4 Resolvedness Conditions of Questions and Answers

Jonathan Ginzburg observes that answers as they are predicted by the semantic accounts of questions and answers would in many cases not satisfy questioners or, in other words, not resolve their question. He proposes a solution in the framework of situation semantics (Ginzburg, 1996, 1995a,b). We should point out here that Ginzburg himself classes his approach as a semantic one. We have moved it to this section as it especially addresses points that have traditionally been studied in pragmatics. Given that the recent semantic theories, especially the different dynamic flavours in semantics, have moved the border forwards into traditional territory of pragmatics, this should not be taken as a strong statement (cf. also Groenendijk and Stokhof, 1997, 1120-1122 on that point). We will summarise some of Ginzburg's observation without going too deeply into his proposed solution, as that requires a considerable technical apparatus.

The Problem of Enumerating All Possible Answers. Ginzburg sets out by noting that a representation of questions that is based on their possible answers (as all semantic accounts described above are, recall that they all subscribe to –

at least – Hamblin's second postulate, cited on p. 48) runs into difficulties when answers are not based on identifying one or more 'simple' individuals. His example is the following:

(3.45) What is the word for "relaxation" in Chukotian?
Ginzburg (1996, 400, his (33))

Ginzburg comes to the conclusion that it would be implausible to model an answer based on alternatives when the questioner has no or little knowledge of the Chukotian language, as (s)he cannot conceive a possible answer, let alone enumerate all possible answers. We are, however, less certain than Ginzburg that (3.45) cannot be represented in semantic terms. Intuitively, we would expect someone who has just asked (3.45) to be prepared to take anything that they would consider a word, i. e., any (spoken) sequence of speech sounds or (written) sequence of characters as a possible answer. We cannot see why one cannot, for example, in the partition approach described above, form partitions based on an enumeration of all *conceivable* words, based either on some transliteration or on some inventory of language sounds. This would, if one would not set some arbitrary upper boundary for the length of expected representation, lead to infinitely many partitions. This does not, as far as we can see, pose any general problem to the partition approaches. Where we would agree with Ginzburg is, that this raises the question whether the semantic approaches are *psychologically* plausible: One would not like to assume that both questioner and answerer need to mentally represent the question by a huge or even infinite enumeration of possible answers. But even then, one could imagine the partitioning not to be based on an actual enumeration of the partitions but rather on some finite description that can unambiguously enumerate them. Note also that none of the work in semantics makes any claim about psychological plausibility.

Context Dependency of Answers. Far more importantly, Ginzburg notes that what makes an answer a *good* answer is highly dependent on the context or situation. He observes that especially the questioners' knowledge and their goals in asking the question need to be taken into account to gauge whether or not a certain answer will satisfy them, or – in Ginzburg's terms – resolves their question. He especially challenges the notion of an exhaustive answer (viz. answering a question by reciting a list of individuals) that seems to underlie the semantic approaches. He presents example (3.46) which he embeds in the following setting: A politician and a scientist visit a research institute. They are both not familiar with the people working there. They attend a number of lectures and afterwards approach the director and ask (3.46a). If the director gives a ‘semantically good’ answer by enumerating the individuals, both questioners, Ginzburg argues, will be disappointed and react with something like (3.46c) when recounting the scene. Recall that in Groenendijk and Stokhof’s account, this answer (when accompanied by an exhaustiveness claim such as *‘And that’s all the people who were there.’*) would be the best (most informative) possible answer (as it is maximally exhaustive, cf. Groenendijk and Stokhof, 1997, 1094–6).

A local scientist, on the other hand, may be happy with the answer and summarise the situation as in (3.46d). Both other questioners would require a different kind of answer, for example, (3.46e) instead of (3.46b) would probably satisfy the politician. Note that for Ginzburg this point is a semantic one (and not purely pragmatic), as the truth conditions of the ‘situation assessment’ given as an embedded interrogative in (3.46d) crucially depend on which answer is given (e. g., (3.46b) vs. (3.46e)) and thus on what Ginzburg calls the resolvedness of the (embedded) question. For Ginzburg, this proves that question resolution cannot be accounted as a purely pragmatic notion as it influences the (semantic) truth conditions in this example.

(3.46) a. Q: Who has been attending these talks?
b. The director: (Provides list of names)
c. I asked the director who had been attending the talks. She didn’t really tell me. All she did was recite a list of names, none of which meant much to me.
d. The director was asked about who had been attending the talks and she told us.
e. [Querier is the high ranking EC politician.] The director: A number of linguists and psychologists.

Ginzburg (1996, 400, his (34a–d) and (35a), respectively)

It should be noted that most of the semantic accounts described above would actually *allow* (3.46e) as an answer to (3.46a), albeit only a partial one. What they cannot account for is why one should prefer an answer like (3.46e) over (3.46b) in the given context.

Answer Granularity. The next point has been referred to as the fine-grainedness of an answer (Groenendijk and Stokhof, 1997, 1121). It is especially noticeable when answering to questions concerning time and place. Consider the following example (Ginzburg gives a number of additional, similar examples, cf. Ginzburg (1996).):

(3.47) a. Where is Deerfield, Illinois?

b. 87° 54’ longitude, 42° 12’ latitude.
c. Near Lake Michigan, about 20 miles north of Chicago.
d. Next to Highland Park.
e. On the planet Earth.
(adapted from Lehnert, 1978, 11)

One can readily imagine different situations where question (3.47a) is most appropriately answered by any of (3.47b) (say, two cartographers talking) through (3.47e) (say, a Science Fiction movie). This shows that, while any of these answers is *true* in all of these situations, in general only one of them will be satisfactory in a given context. Again, this cannot be accounted for by the semantic approaches.

Ginzburg’s Representation of Questions. We will only briefly describe how Ginzburg intends to handle questions and answers. He does so in a framework of situation semantics (see Ginzburg, 1996, 1995a,b for more details). The important point is that in this framework, one can make explicit reference to situations (comparable to possible worlds in intensional logic, but assumed to have an inner structure) and to a certain framework of reference (here used to explicitly model the questioner). Ginzburg’s definition of resolvedness[7] looks as follows:

A fact τ RESOLVES $(s?\mu)$ relative to a mental situation *ms* iff
1. Semantic condition: τ is a fact of s that potentially resolves μ
2. Agent relativization: $\tau \Rightarrow_{ms}$ Goal-content(ms) (Intuitively: τ entails the goal represented in the mental situation *ms* relative to the inferential capabilities encoded in *ms*.)
Ginzburg (1996, 407, his (51))

This should be read as follows:

τ represents a resolution to the question, i. e., a satisfactory answer.

$(s?\mu)$ represents the question. Situation s is the representation of the situation in which the question is asked and μ is its semantic representation (Ginzburg assumes a lambda-term somewhat similar to the subject part of Belnap’s question representation, cf. 3.2.2.1).

[7]For indirect answers, Ginzburg defines an additional – weaker – notion of partial resolvedness. He argues that this differentiation is important as a questioner who receives an indirect answer cannot truthfully claim to *know the answer* to the question, but rather only to *have some information* concerning the question or to know the answer to a certain extent. For details see Ginzburg (1995a,b).

Potential resolvedness is defined in Ginzburg (1995a, 471–6). Intuitively, it means that τ entails – for a *wh*-question – some positive example, i. e., entails that the extension of the question proposition contains some individual, and that this fact could not be inferred without adding τ or it entails that the extension of the question proposition is empty.

$\tau \Rightarrow_{ms}$ **Goal-content(ms)** means that in *ms*, i. e., the mental state of the questioner, some fact can be inferred that resolves the currently held goal of the questioner (for more details, see Ginzburg, 1995a, 499–504)

With this definition, Ginzburg allows explicit reference to be made to a) the situation, b) the questioner's knowledge (by referring to the inferences that are possible in *ms* by $\Rightarrow_{ms}$) and c) the questioner's goals (Goal-content(ms)).

When comparing this to Groenendijk and Stokhof's suggested solution, one notices that the explicit representation of the questioner's goal and the requirement that it be resolved by an answer is the main difference: Ginzburg's situation can at least partly be described in terms of indices in intensional logic, the questioner's knowledge is modelled as the questioner's epistemic and doxastic sets (cf. 3.2.2.4) but Groenendijk and Stokhof do not explicitly model the questioner's goal.

Note that this rather works like an additional 'filter' over *possible* answers: All answers in (3.46) and (3.47) would be considered possible (though possibly indirect) answers under Groenendijk and Stokhof's partition approach. Ginzburg's approach additionally provides a filter (or rather a handle for gauging the questioner's preferences concerning the different answers).

3.2.3.5 Discussion

In this section we have discussed a number of issues subsumed as pragmatic aspects of questions and answers. On the one hand, we have broadened the definition of questions by showing that not only interrogatives can be used to express them and explained this fact by introducing the notion of indirect speech acts.

Then, we have summarised work that shows that a specific answer to a question may fail to satisfy the questioner – even though it is a possible and true and maybe even the most informative answer (according to Groenendijk and Stokhof's account, that is). Ginzburg has shown that the amount of satisfaction that an answer provides depends on the questioners (especially their prior knowledge and their goals). He introduces the additional criterion of resolvedness and shows how this may be accounted for by adding an explicit user model containing, among other things, the user's goals.

The overall consequence seems to be that only an integrated approach along the lines carefully argued for in Groenendijk and Stokhof (1997, 1108–22), which unifies different semantic approaches and pragmatic aspects, can fully account for all phenomena described in the literature on questions and answers. Groenendijk and Stokhof show convincingly that neither a purely semantic nor a purely pragmatic account can achieve this.

We will show below, however, that basing QA on structured semantic representations (let alone pragmatic modelling) of texts and questions is currently beyond the scope of natural language processing systems. We will therefore suggest a approximation based mainly on syntactic and lexical semantic information.

3.2.4 Updating the Question Answering Framework

We will now return to the framework of linguistically informed QA (3.1.2). We will further refine the characterisation of question analysis, answer finding and answer presentation in several respects.

We have described a number of phenomena related to questions and answers in this section. As we have pointed out, the work that we have reported is concerned with questions and answers in human communication. In this section, we will re-examine them in the context of linguistically informed QA.

3.2.4.1 Question Analysis

In the description of linguistically informed QA above (3.1), we have stated that the users can enter questions expressed in natural language. In the light of our review of work on questions and answers, we need to refine and elaborate this statement.

Questions and other Requests. In our first description of the framework (3.1.2), we just stated that users can enter questions to the system. As described above, questions will mostly take the form of interrogatives (3.2.1, 3.2.3). However, we have shown that questions may be expressed in other syntactic forms (namely as an indirect speech act, 3.2.3.1).

We will currently restrict possible system input to questions in the form of direct interrogatives (3.2.1).

Syntactic Question Types. Of the three syntactic question types described above (3.2.1), a QA system needs at least to handle *wh*-questions. We additionally include yes/no-questions (6.4.3.2).

Handling alternative questions is currently not covered by linguistically informed QA.

Interrogative Analysis. Interrogatives generally exhibit a number of syntactic features that distinguish them from sentences in other sentence moods. We have listed the most important constructions for English (3.2.1) and for German (3.2.1.4). For German, we have also given an overview of the question words.

Especially the different subtypes of *wh*-phrases need to be handled. For English (and German) these are *wh*-phrases with nominal interrogative pronouns (*'who'*), interrogative pronouns as determiners (*'which company'*), interrogative adverbs (*'when'*) and *how*+adjective/adverb-phrases. We will describe below how the most important types can be handled in linguistically informed QA (3.5.2.5, 5.1.4).

Note that the issue of question parsing must not be underestimated: Many available parsers for natural language lack support for interrogative structures. Statistical parsers, for example, often do not handle them due to their scarcity in general corpora. Parsers must, therefore, often be adapted for handling questions; statistical parsers may need to be re-trained, cf. Hovy et al. (2001, 4–5).

Requested Number of Answers. As pointed out above, many questions have more than one answer. Question may contain an explicit request to return a certain number of answers (or a request to return all answers, cf. 3.2.2.1). We currently do not integrate any means of automatically recognising the number of requested answers. In the system implementation, users can explicitly set the (maximal) number of answers that is to be returned (6.4.5).

Multiple *Wh*-Questions. We have introduced multiple *wh*-questions of the matching (pair/tuple-list) type (3.2.2.2). They form an interesting type of questions as they allow to compactly ask for 'list-like' information.

Linguistically informed QA allows handling matching *wh*-questions; it will match and return all possible answers (3.5.2.5, 5.1.4).

We will return to the presentation of the answers for multiple *wh*-questions below.

Open Questions. We have discussed a number of issues of open questions above (3.2.2.5). We explicitly exclude open questions from the scope of linguistically informed QA as we specify it here. We have shown above that open questions rather call for producing a text, probably in the style of an essay than returning a short answer. We will further discuss open questions and summary-style answers in 8.3.4.

Quantifying into Questions. We have introduced a number of phenomena above that are related to the interplay of quantifiers and other scope bearing material with question words, generally referred to by ‘quantifying into questions’. To correctly handle these phenomena, the QA system needs to properly treat the ‘underlying’ problems: Scope bearing linguistic material in questions must be recognised and then correctly evaluated in answer finding.

This may be difficult in general, however: As shown above, quantifiers in *wh*-questions may especially introduce wide-scope and narrow-scope readings of the quantifier (generally associated with tuple-list answers and with single answers respectively, cf. 3.2.2.5). Under the functional analysis, they would be represented as constant functions and non-constant functions, respectively. Consider the following questions (3.48a) and (3.48c), that might be paraphrased as (3.48b) and (3.48d), respectively. This example shows how complex the derivation of the correct reading for a question containing an (implicit) quantification may be.

(3.48)
a. What was the annual GNP of the US 2000–2005?
b. For each year from 2000 to 2005, list the Gross National Product of the US!
c. What was the overall GNP of the US 2000–2005?
d. Name the sum of the Gross National Product of the US in the period from 2000 to 2005!

We will therefore exclude questions with quantification from the our current work: As we do not opt for a full semantic representation as the basis for answer finding (3.3), but rather a matching of questions and answer representations that is based on syntactic structures extended with lexical semantic information (5.1), correctly handling such phenomena is beyond the scope of our approach. As many such questions can either be broken down into a sequence of questions (e. g., (3.48a) can be broken down into five questions, one for each year) or paraphrased by a multiple *wh*-question (cf. (3.35) vs. (3.36)), we think that this restriction is acceptable.

3.2.4.2 Answer Finding

As we have already pointed out in the introduction to this chapter, the work on questions and answers reported in this chapter has not really been concerned with the issue of answer finding. The accounts subsumed under semantics and pragmatics ‘only’ define an answerhood relation: Given a question *and* a reply, they show how to find out whether or not the reply provides an answer to the question. The issue of how an answer can be found in a body of knowledge is not addressed.

We can, however, use one concept that was introduced in 3.2.2.4 as a basis for defining the answer finding process, namely that of an indirect answer. Intuitively, an indirect answer is a reply that allows the questioner to arrive at a ‘proper’ direct answer by applying additional inferences. The notion of indirect answer needs, of course, to be defined more rigidly. We will do so in 5.1 below.

The obvious but important point that needs to be made here is the following: It will almost never be the case that one can find a direct answer to a given question in a document collection. Consider the following example (=3.2).

(3.49) a. Who killed John F. Kennedy?
b. Lee Harvey Oswald killed John F. Kennedy.

Here, (3.49b) may be considered as the canonical, sentential direct answer to the question (3.49a). A user of a QA system would expect this answer or the related constituent answer *‘Lee Harvey Oswald’* to be returned as an answer for (3.49a). However, the chances that *exactly* this text string is found in a document collection must be considered extremely small. Therefore, it is necessary that a QA system is able to find not only direct answers but also – far more importantly – *indirect answers* to questions.

To be useful as a basis for answer finding in QA, we must, of course, define indirect answers more precisely. For now, we will just assume that a text passage contains an indirect answer to a given question if someone who has asked the question and is given the text passage as a reply, will typically accept it as containing an answer. As a test, we might afterwards just ask the questioner *‘Does that answer your question?’*. A positive reply will then indicate that we have indeed given (at least) an indirect answer.[8] Consider the following example.

(3.50) a. A: Who killed John F. Kennedy?
b. B: [Some passage recounting Kennedy’s murder that names Lee Harvey Oswald as the assassin]
Does that answer your question?

[8]This test is, of course, inspired by Ginzburg’s ‘semantic’ answerhood test, cf. 3.2.3.4.

c. A: Well, you might have said ‘Lee Harvey Oswald’ straight away, but yes, thanks.
d. B: [Some passage recounting Kennedy’s murder that leaves out Oswald altogether]
   Does that answer your question?
e. A: No. You have told me something about his murder, but you haven’t told me who did it.
f. B: [Some passage recounting Kennedy’s parentage and childhood years]
   Does that answer your question?
g. A: No. That was not relevant at all!

In this example, only (3.50b) actually answers the question. Thus, (3.50c) might be a fairly typical reply to our test question. For all other cases, the questioner would be forced to negate the test question.

From a slightly different perspective, this means that it is necessary, for finding answers to users’ questions to a QA system, to draw inferences based on the given document collection to arrive at suitable answers.[9] A system that cannot use any inferences at all will only be able to answer a very small portion of questions, as the likelihood of finding a direct answer to a question in a document collection (i. e., without any inferences) is generally small.

In the next section, we will argue that we consider using a full semantic representation of questions and answers as a basis for QA not to be a viable option at the moment (3.3). We will therefore develop a restricted approximation of indirect answerhood that allows inferences on the basis of syntactic and certain lexical semantic variations (3.5). We argue that this approximation is useful, on the one hand, and restrictive enough to be practically useful, on the other hand.

3.2.4.3 Answer Presentation

A number of conclusions about what would form a possible answer and how it should best be presented can be drawn from the work reported above.

Answer Fusion. Whenever two or more answers are found to a question, the QA system should try to fuse them and present them as compactly as possible. There are three issues involved here: As described above, multiple answers to a question should be distinct (cf. 3.2.2.1). On the other hand, different answers may, in practice, be due to inconsistencies or contradictory information in the underlying document collection. For multiple

[9]Additional knowledge is, of course, necessary for drawing inferences, namely some form of inference rules.

answers for *wh*-questions with quantifiers, additional possibilities for answer fusion may be available. We will go into the three issues in turn.

Distinctness. Multiple answers to a question should be distinct (3.2.2.1). From this, it directly follows that answers that are essentially equivalent should not be presented as different answers, but rather conflated. As our approach is syntax-based, we can only recognise distinctness (or equivalence) at the text surface. For now, we ensure that no two answers that are equal (i. e., that have identical surface strings) are output as different answers. This also applies to cases where two answers refer to the same entity and this has been established by anaphora resolution (3.5.2.4).

This is a simplification, of course, as this does not exclude all cases where two answers are equivalent but differ in wording. Note that, in fact, the problem is even greater: For recognising the (non-) distinctness of answers, full language understanding would be required, as different ways of phrasing an answer need not even use similar wording. In a partition approach such as Groenendijk and Stokhof's, the distinctness of answers could be inferred from the fact that the intersection of the partitions representing the separate answers would be empty (this is true for exhaustive answers, for partial answers, their intersection would only have to differ from their union, i. e., at least one partition is excluded). This might also allow to handle complex cases, like local inclusion, e. g., the answers *'In the US'* and *'Near Chicago'* for the 'Deerfield question' (3.47), as the respective partitions for the partial answer would not conflict.

Conflicting Answers. An answer to a question should be truthful, as required by the Gricean maxims (3.2.3.2). A QA system will in general assume the veradicity of the information in its document collection, i. e., that it is true. The system can only give true answers relative to this collection. If more than one answer is presented, conflicting answers should be marked and the user should be alerted to the conflict.

Consider the following example that shows that detecting conflicts is far from trivial. Remember that Elizabeth Taylor and Richard Burton were married twice. Thus, we have the unusual case that a question for the date of their marriage (3.51a) should return two answers, ideally like (3.51b). However, only with additional world knowledge can this case be distinguished from one with conflicting information in the text collection that should lead to an answer like (3.51e).

(3.51) a. When did Elizabeth Taylor marry Richard Burton?

b. 1964, for the first time, 1975, for the second time.

c. ? I have conflicting information. One source says 1964, the other 1975.

d. ? June 1964, for the first time, July 1964, for the second time.

e. I have conflicting information. One source says June 1964, the other July 1964.

Having such openly conflicting information in a text collection seems unlikely at first. However, in the 2004 TREC, two different answers were found (by different participants, though) in the text collection (cf. 3.5) and judged as correct (in fact, the second one is correct):

(3.52) a. In fact, when star James Dean was killed in the auto crash on May 5, 1955, he was on his way to visit his new friend, Monty Roberts. (NYT19980619.0287)

b. On Sept. 30, 1955, Dean was on his way to a California auto rally and to look at a farm with Monty when he was killed in a traffic accident. (NYT19990604.0100)

Since the recognition and resolution of conflicts in answers is an extremely demanding task in general, we will exclude it from the linguistically informed QA for now.

Multiple Answers to *Wh*-Questions with Quantification. We have described above that there are three possible ways in which multiple answers to *wh*-questions with quantification (including multiple *wh*-questions) may be presented (3.2.2.5). These correspond to characteristic underlying functions in a functional view of such answers described above. The important point here is that in a QA system, this is mainly a point of answer presentation. The optimal way of presentation can, in general, be identified by analysing the list of answers. We will distinguish three cases:

Constant. If all answers are equal (or very similar), they may be presented as a single answer:

(3.53) How did Abraham Lincoln and John F. Kennedy die?
They were (both) murdered.

List Answer. If all answers are distinct, they may be presented as a list answer (default case):

(3.54) Who killed which US president?
Lee Harvey Oswald (killed) John F. Kennedy.
J. W. Booth (killed) Abraham Lincoln.

Intensional Answer. This case is, of course, difficult to handle automatically. It requires that (assuming an underlying function for now) the function is recognised and that it can suitably be verbalised. In most cases, however, this is best only issued as additional information, together with the default list answer:

(3.55) What profits did Allianz AG and Siemens AG make in 2005?
They made record profits: Allianz earned 4.38 thousand million €, Siemens 3.4 thousand million €.

We currently use the ‘list answer’ case for all multiple *wh*-questions. That means that each possible answer is listed separately, without any attempt to fuse information.

Number of Answers. As mentioned above, there are linguistic devices for explicitly requesting one, *n* or all answers to a question (3.2.2.1). We currently do not automatically handle such answer-size requests. However, we allow the users to manually set the (maximum) number of answers to be returned (6.4.5).

As described above, we cannot safely recognise distinctness of answers in our syntax-based approach. This also means that the counting of answers will refer to ‘string-distinct’ answers, as described above.

Exhaustiveness. As mentioned above (3.2.2.1), questioners will generally find it helpful when answerers signal whether or not their answer was exhaustive. In a QA system, exhaustiveness can in general only be claimed relative to the number of answers that the system has actually found in its document collection. Exhaustiveness interacts with the required number of answers (see above). In our system implementation, we generate an informative output like *‘n/no additional answers found.’*

Justification. The Gricean maxims state that a speaker should not say anything for which (s)he lacks evidence (3.2.3.2). This point corresponds with 4. and 5. from Bonnie Webber’s list of cooperative responses cited above (3.2.3.3). Our system can show either the sentence(s) minimally containing the answer from the original document or the whole document. It can also present a justification of its inferences in natural language (cf. 6.4.6).

Short Answers. As discussed above, *wh*-questions can, in general, best be answered by constituent answers, yes/no-questions by yes or no and alternative questions by one of the alternative constituents. A QA system must therefore minimally provide these options. In addition, it can also offer sentential answers, as individual users might prefer these: From a full sentential answer, it is easier to gauge the reliability of the answer. In our system, the user can manually switch between constituent answers and full sentential answers and also influence the answer verbosity (6.4.3.2).

Answer Congruency. When constituent answers are used, they must be congruent with the corresponding *wh*-phrase. We assume Reich's analysis here, namely that a constituent answer is an elided form of the sentential answer (3.2.2.2). In a QA system, this means that when a constituent answer is used, it must be in a surface form that would allow to insert it into the (declarative form of the) interrogative. It must therefore be properly inflected in overtly inflecting languages such as German. But it also means that for adverbial questions phrases, the answer must be an adverbial, not, for example, a bare NP. This is best shown by an example from the TREC 2004 QA track[10]:

(3.56) a. How did James Dean die? (Q 4.3)
b. * Car crash. / James Dean died car crash.
c. In a car crash. / James Dean died in a car crash.

For indirect answers, this may require that the answer constituent that is found in the document is changed to show the correct morphological form. We achieve this in our system by using an answer generation module that correctly inflects the answer based on the requirements from the question structure (6.4.3.2).

Cooperative Responses. We have described above that a cooperative system should, in cases where no answer can be found, generate a suitable response (3.2.3.3).

Consider the following example (3.57). Here, the system should detect that the question implies the existence of a) an individual called John F. Kennedy and b) that this individual was murdered. Thus, in case that the relevant information is missing, the system could generate helpful replies such as (3.57c) or (3.57d) instead of (3.57b), possibly saving the user a large amount of frustrating re-formulation attempts.

[10]Note that the NP answer (3.56b) that we judge ungrammatical was accepted in the official judgment.

(3.57) a. When was John F. Kennedy murdered?
b. I don’t know.
c. Sorry, I do not have any information about John F. Kennedy at all.
d. Sorry, I do not have any information about any murder at all.

We have integrated this sort of response into our answer generation module (6.3.5).

User Modelling. We have described above that the formulation of an answer should take the questioner into account in several respects (3.2.3.4), most importantly the questioners’ (perceived) prior knowledge and their goals, especially their goals in asking the question. Let us assume, for now, that we have a suitable user model at our disposal to influence the answers given by the system to the user (the ‘questioner’). In a QA system, with its separation between answer finding and answer presentation, this can generally be dealt with by answer selection and answer reformulation:

By answer selection we mean filtering out answers that were found in the document collection but that are not useful according to the user model, e. g., an answer that the user is assumed to already know or an answer that has already been given by the system earlier.

Answer reformulation concerns answers that are found in the document collection that have the wrong ‘abstraction level’ or answer granularity (again, as determined by the user model). Remember the examples cited above, namely (3.46) and (3.47): In the first one, different questioners were supposed to require different ‘functional descriptions’ when asking who attended a certain talk (such as *‘A number of scientists attended the talk.’*), in the second one, different possible ways of locating Deerfield, Illinois, were given.

Answer reformulation would require that the system uses additional knowledge to generate possible reformulations of the found answer. For the latter example (the ‘Deerfield example’), this might, e. g., be a geographical database and a number of suitable templates: As an answer for the question *‘Where is X?’*, answers like *‘in Y’*, with *Y* a region, a state or a description of the region of the earth (*‘In south-east Asia’*) or *‘near Z’* could be generated. From these, all answers making reference to places (probably) unknown to the user (again, according to the user model) would be filtered, leaving only promising answers.

For the former example (the 'talk attendees example'), additional knowledge (probably from the document collection) would have to be used to generate a suitable 'functional' description: Instead of a list of names of individuals, a 'functional' description would make use of some (salient) property that the individuals share. Note the parallels with 'intensional answers' to multiple *wh*-questions described above and also with approaches to generating definite descriptions (where, by contrast, some minimal *distinguishing* feature in a group of individuals is searched, cf. Krahmer et al., 2003). Debra Thomas Burhans defines a method for generating 'generic' answers in a (toy) system based on automated theorem proving that describe (roughly) the most generic concept in an underlying concept hierarchy that subsumes the individuals 'answering' the question (Burhans, 2002, 72–88).

While these two short descriptions show how, in principle, one could go about to solve two cases of answer reformulation, finding a general method for re-formulating answers according to the users' knowledge and goals seems a far more demanding task and one that is beyond the scope of our current work.

In addition, managing and automatically building full user models for general purposes and all domains is not a solved problem yet. Marco De Boni discusses a number of different approaches (De Boni, 2004, 150–6) and comes to the conclusion that current approaches require too much knowledge that needs to be encoded manually and that the complex reasoning processes involved make such approaches too unwieldy for general, unlimited applications.

We will therefore not further pursue the issue of user modelling for QA here. We note, however, that in an interactive QA system (i. e., a system that allows the users to lead an information seeking dialogue, cf. 6.4.3.2) these problems seem less acute, as the users can employ follow-up questions in case they are not satisfied with the answer. If, for example, the QA system finds and outputs the answer *'Next to Highland Park.'* for the question where Deerfield is located, and the user is not satisfied with that, (s)he may try *'And where is Highland Park?'*, the system might then find a more satisfactory answer like *'20 miles from Chicago'*.

3.3 Question Answering Based on Structured Semantic Information

In the previous section, we have discussed a number of phenomena concerned with questions and answers in natural language. We have pointed out that the approaches discussed in this connection in the literature are concerned with checking the answerhood relation between specific given pairs of questions and answers. Therefore they do not lend themselves directly to finding answers in large text collections. We will now turn to the issue of how finding answers in text collections can best be done.

In 2.2.2, we have described that the task of finding answer candidates is delegated, in most current QA systems, to an Information Retrieval module. The IR module is used to retrieve passages that are likely to hold an answer to a given question. The retrieved texts are then further processed to extract the answer itself, based on patterns, word overlap methods or, for the most advanced systems, theorem proving using logical representations of the candidate texts (Moldovan et al., 2003b).

As described above (2.1.1), IR systems typically compute the relevance of documents based on the overlap of search terms and document terms. Very often, both the search terms and the index terms are further refined (when stemming or query expansion is used, for example), but in principle, matching is mostly done on unstructured bags of words, derived from the text surface.

We have shown above that semantic and also pragmatic considerations need to be taken into account to find out whether a given question is answered by a given reply, i. e., to define the relation of answerhood. The approaches that can – arguably – describe the widest range of phenomena related to answerhood, namely the partition approach (3.2.2.4) and Ginzburg's approach using resolvedness conditions (3.2.3.4), are based on semantic representations of questions and replies: In a framework using semantic representations and well-defined methods for computing inferences from them, answerhood can be represented in a natural way.

We will explore the idea of using semantic representations and automated reasoning as a basis for QA systems in this section: By using representations that can be directly used for inferencing (preferrably a well-understood semantics based on model-theoretic logic) and by providing a sufficiently sophisticated knowledge base of inference rules, answer finding could be implemented in a both comparatively simple and very flexible way.

This approach has been followed by a number of researchers. We will summarise some work in this area in the first part of this section: First, we describe work on QA based on componential lexical semantics (Lehnert, 1978). We will then review work on QA by automated reasoning (Burhans, 2002; Friedland et al., 2004a; Gardent and Jacquey, 2003).

However, none of the approaches has so far resulted in systems with broad, general coverage. We will discuss some of the reasons; most of them are related to the fact that the problem of deriving suitable and useful meaning representations from general texts is complex and far from being solved.

In the conclusion, we will propose that an intermediate level of abstraction, namely between shallow bag of words and deep full meaning representations, should be used for practical QA systems.

3.3.1 Question Answering Based on Componential Semantics

In her influential work on Question Answering, Wendy Lehnert has argued that full understanding of the underlying text is required for answering questions about it (Lehnert, 1978): Only if a QA system can reason about the contents and especially about the goals, reasons and consequences connected with the events described it can provide suitable answers. She describes a QA system that can answer questions about small everyday stories. Her work has therefore been subsumed under the heading ‘QA in story comprehension’ (cf. Hirschman and Gaizauskas, 2001, 280–1).

As a basis for story understanding, Lehnert uses a representation of the story and questions based on Roger Schank’s work on scripts and plans (Schank and Abelson, 1977), which in turn builds on his earlier work on Conceptual Dependency (CD) networks (Schank, 1973, 1972).

The representations are made up from basic meaning concepts (such as physical transfer, PTRANS) connected by basic relations called dependencies (such as Agent or Cause). A representation is thus given by a network where basic concepts (semantic primitives) are connected through a number of relations. One important guiding thought is to keep the inventory of concepts and dependencies small and to represent complex events by a small sub-network of primitive concepts that make up the representation of the event. Thus, a text word is typically represented by a number of basic concepts that are interlinked to make up the different events and entities that are understood when the word is used. As words are represented as a combination of the contributing basic components, the approach has been called componential lexical semantics (cf. Miller, 1998a, xvi; Fellbaum, 1998b, 92–94).

Note that it is a central tenet of Schank’s work that the CD representation also encodes meaning that is only ‘implicit’ in the corresponding text:

> Conceptual Dependency (henceforth CD) is a theory of the representation of the meaning of sentences. The basic axiom of the theory is:
>
> A For any two sentences that are identical in meaning, regardless of language, there should only be one representation.
>
> The above axiom has an important corollary that derives from it.
>
> B Any information in a sentence that is implicit must be made explicit in the representation of the meaning of that sentence.
>
> Schank and Abelson (1977, 11)

Question answering is done in Lehnert's approach by deriving CD representations from the text and from the user's questions. By using an additional question typology that is based on a combination of syntactic, semantic and pragmatic features, the question representations may be further interpreted, e. g., by recognising a polite request like *'Do you know the time?'* as a request to specify the time rather than a question requiring a yes/no-answer (cf. 3.2.3.1). Finding answers is then done by matching the network (graph) representing the question within the text representation.

One appealing feature of this approach is that it reduces additional inferencing, as the CD representations provide a high level of abstraction: Most differences in surface wording will not play a rôle when the texts are represented by primitive concepts. Complex events are described in terms of basic sub-events, greatly facilitating the matching of question and answer representations, even if the surface wording differs markedly.

It has become clear, however, that this approach does not scale up. Wendy Lehnert has shown that it can successfully be employed for toy examples. However, there are (at least) three main issues where the approach runs into problems:

Defining the Primitives. It has turned out that defining the stipulated primitives underlying language is very hard. Breaking down words into their basic components works well for prototypical examples, but tends to run into trouble when wider coverage is aimed for.

Translation. The approach relies on the assumption that the CD representation carries the full meaning (including implicit parts of the meaning) of text and question. In spite of great advances in natural language processing over the last decades, deriving such representations from general texts goes far beyond the capabilities of current language processing systems.

Inference Engine. It is not clear how inferencing on CD representations can be consistently defined: In contrast with formal logic representations, where notions like entailment and equivalence are well-defined, comparable definitions for CD representations are missing. Moreover, the complexity of the required inference mechanism makes in computationally intractable.

The following quote by George A. Miller summarises the initial enthusiasm for componential semantics in the 1970's and a gradual sobering-up over the 1980's:

> Roger Schank and his colleagues were building language-processing systems having small vocabularies for well-defined topics, where word-meanings were represented by a few hundred LISP programs, but it was becoming clear even in 1985 that this approach would have trouble scaling up. [. . .] Analyzing a word's meaning into semantic components that can be captured in LISP code is a form of componential lexical semantics. That is to say, componential semantics approaches the meaning of a word in much the same way it approaches the meaning of a sentence: the meaning of a sentence should be decomposable into the meanings of its constituents, and the meaning of a word should be similarly decomposable into certain semantic primitives, or conceptual atoms. Philip N. Johnson-Laird and I had explored componential semantics with much enthusiasm in our 1976 book, Language and Perception, but in 1985 we still did not have any definite list of the conceptual atoms and it was beginning to look as if, whatever other virtues componential lexical semantics might have, it was not the best theory for natural language processing by computers.
> (Miller, 1998a, xvi)

3.3.2 Question Answering Using Knowledge Representation and Reasoning

In the previous description of work on QA using componential semantic representations, we have pointed out that one of the weaknesses of the approach is the lack of usable definitions of the inferences involved. One obvious solution of this problem is the use of well-understood representations based on formal logics, for which a well-defined model-theoretic interpretation, including entailments, exists.

We will now describe work on QA by automated reasoning, most of it carried out by researchers in Artificial Intelligence (AI) specialising in Knowledge Representation and Reasoning (KR&R, Russell and Norvig, 1995, 151–334).

3.3.2.1 Knowledge Representation

Knowledge Representation and Reasoning systems generally use a formal language to represent the knowledge that the system can draw upon and provide an inferencing mechanism that allows deriving new facts from this knowledge base.

Different formal languages have been used for knowledge representation. After early systems that often used proprietary languages lacking well defined inferencing facilities (e. g., KL-ONE, Brachman and Schmolze, 1985), most approaches today use either first-order predicate logic (FOPL) or a language based on FOPL with higher-order extensions, or (a dialect of) description logic (DL, Baader et al., 2003). The choice of representation language is generally influenced by the conflicting requirements of using a formalism that is as expressive as possible, on the one hand, but allows efficient processing of inferences, on the other hand. We will return to this discussion below.

In work on KR&R, knowledge is assumed to be made up from a set of general rules (axioms), often called Terminology Box (or T-Box for short) and known facts in the form of assertions, called the Assertion Box (A-Box). This general division of labour also holds for QA systems in work of KR&R: The knowledge to be queried is represented as assertions derived either automatically or manually from some source of knowledge. The rule knowledge needed for inferencing is provided by an additional external resource, for example a computational ontology such as Cyc (4.4.1, Matuszek et al., 2006), possibly embellished with additional inference rules.

It should be noted that most approaches in KR&R are only marginally interested in the process of deriving the knowledge representation from natural language input or in generating natural language answers. These processes are either taken for granted (but cf. 3.3.3.2) or it is assumed that the system is directly operated by knowledge engineers communicating with it in the system's knowledge representation language (Friedland et al., 2004a; Green, 1969). In general, it is assumed that both the knowledge base and the questions are available in a suitable representation language and that answers are to be returned in the same language.

3.3.2.2 Reasoning

The idea to use automated reasoning systems to answer questions is not particularly new: The first systems based on this idea were built in the late 1960's and early 1970's (Green, 1969; Luckham and Nilsson, 1971). A recent system is described in Burhans (2002).

Knowledge Representation and Reasoning systems use automated reasoning systems; most often theorem provers are used. General theorem provers take a knowledge base and a theorem as input and return either true or false, depending on whether the theorem follows from the knowledge base or not (Bläsius and Bürckert, 1992). Research over the past decades has produced a number of efficient theorem provers for different formal logics.

The best-known systems for FOPL include Otter (McCune, 2003), Waldmeister (Hillenbrand, 2003; Buch and Hillenbrand, 1996), Blicksem (de Nivelle, 1998), and SPASS (Weidenbach et al., 2002). RACE and its successor RACER (Haarslev and Möller, 2001a,b) are probably the widest-used systems for DL.

We will now describe how QA through theorem proving can be done in principle (Burhans, 2002, 2–5). Note that this task differs from the one of testing answerhood: Instead of checking, for a given question and a given reply, whether answerhood holds, the theorem prover returns possible answers to a question representation from a given knowledge base. The theorem prover thus takes over the task of searching for answers. As it can draw on the information in the knowledge base and thus combine facts derived from the textual knowledge source with general inference rules, finding indirect answers is 'built in': This approach will automatically use inferences to arrive at a suitable answer representation.

Yes/No-Questions. From the facts (inference rules and assertions) that are available to the system, a knowledge base $\mathscr{K}$ is constructed in a suitable formal language (we will assume first order predicate logic, FOPL, here). It is crucial that $\mathscr{K}$ is consistent, as otherwise automated reasoning will produce false answers (*ex falso quodlibet*).

This knowledge base $\mathscr{K}$ is fed into the theorem prover. A question can then be put to the system in the form of a FOPL proposition p, that is, the proposition 'underlying' the question. Theorem provers typically use resolution by refutation ('proof by negation', cf. Eisinger and Ohlbach, 1992): The proposition p is first negated ($\neg p$) and the prover attempts to prove that $\mathscr{K} \cup \{\neg p\}$ is inconsistent to show that p follows from $\mathscr{K}$.

With this simple approach, only answers to yes/no-questions can be found, as the proof will only return a yes/no answer: *False* (i. e., $\mathscr{K} \cup \{\neg p\}$ is inconsistent) is the result of a successful refutation, meaning that the proposition p *does* follow from $\mathscr{K}$, i. e., the answer to the question is *yes*, or the result that no refutation can be found, meaning that the proposition p does *not* follow from $\mathscr{K}$, i. e., the answer to the question is *no*. Of course, these answers hold relative to $\mathscr{K}$. That means: The veridicity of $\mathscr{K}$ must be assumed (only true facts in $\mathscr{K}$) and a closed-world assumption must be made (all true facts in $\mathscr{K}$), otherwise the answer in the case when no refutation is found is rather *'I don't know'* than *no*.

***Wh*-Questions.** The approach can be extended to handle answers to *wh*-questions by letting the theorem prover output individuals answering the question in the following way: Consider, for example, the question *'Who is sleeping?'*, that could be represented by an intermediate representation like $?x\,\mathrm{sleep}'(x)$ (using the notation from Groenendijk and Stokhof, 1997).

To find answers, i. e., suitable variable bindings for x, an existentially quantified and negated version is derived; for the example, this is $\neg\exists x.\mathrm{sleep}'(x)$. This negated version is then used as above in a refutation resolution. That has the effect of making the theorem prover disprove the statement that there is no one who sleeps. If it can be concluded from the knowledge base that there is (at least) one individual who sleeps, then the prover will find this counterexample.

The prover must be extended so that it returns not only *false* (i. e., successful refutation), but also the satisfying individual. In cases where more than one answer is to be found, the control strategy of the prover can be further changed so that it will be able to find all such individuals and not only the first one: Finding one counterexample is, of course, sufficient as a refutation of the input proposition. Therefore, general theorem provers will stop after finding the first counterexample.

Note that, equivalently, answers could be found using a model generator instead of a theorem prover. The task would then be to find a minimal model that, for a *wh*-question $?x_1 \ldots x_n\phi$, satisfies $\mathscr{K} \cup ?x_1 \ldots x_n\phi$ (again, the notation is borrowed from Groenendijk and Stokhof, 1997), i. e., a set of domain entities and a variable assignment for $x_1 \ldots x_n$ that makes
$lbracket\phi]\!] = 1$. Until recently, theorem provers were far more efficient than model generators. Therefore, theorem provers have been more widely used than model generators. For additional discussion, see Bos and Markert (2006); Blackburn and Bos (2003).

Answers. Based on the satisfying individuals, an answer can be generated. In a general QA system, this should be presented in natural language, of course.

Therefore, a suitable answer generator that takes FOPL as input would have to be integrated, especially for indirect answers.

As an interesting additional feature, one could present the proof steps (or a suitable natural language description) to the user in order to justify the derived answer (cf. Burhans, 2002): Burhans's dissertation is built on the idea that, in fact, *every* proof step in such a proof provides not only justification but at least an indirect, partial answer to the question. Intuitively, this could be compared with giving 'hints' that eventually lead the questioner to work out the answer. This seems, indeed, closely related to the idea of indirect answers described above. Whether users of an actual system would find these 'hints' useful may be another question, as often the 'mechanical' reasoning of a theorem prover is considered tedious and 'unnatural' by (untrained) human users.

3.3.2.3 A Large-Scale Experiment: The HALO Project

We will now summarise a recent, large-scale pilot study designed to examine the current state-of-the-art of QA in KR&R and its possibilities in the HALO project (Friedland et al., 2004a,b). Three participating organisations with established expertise in the area of KR&R, namely Cycorp Inc, Austin, TX, Ontoprise GmbH, Karlsruhe, Germany, and SRI International, Menlo Park, CA (with support from Boeing Phantom Works, Seattle, WA), were given the task to first manually encode, in a knowledge representation formalism of their choice, the information contained in 70 pages of a Advanced Placement[11] chemistry textbook, and then use a reasoning system to answer previously unseen complex questions based on the derived knowledge base.

Each team had four months to encode the knowledge. The teams employed knowledge engineers, in the case of SRI counselled by chemists, who worked together to derive an optimal representation of the facts. When the task was set, it was estimated that the text contained about 100 'major rules' of chemistry so that a sufficient number of non-trivial questions could be asked.

After this knowledge engineering phase, the teams had to freeze their respective systems and received a set of 100 questions altogether (50 multiple choice questions, 50 essay-style questions). The essay-style questions were quite demanding, as the following example shows: 'Pure water is a poor conductor of electricity, yet ordinary tap water is a good conductor. Account for this difference.' (Friedland et al., 2004a, 15).

The teams had another two weeks to encode the questions in their KR language. Then, the participants' systems were run on the respective question representations by a third party; the resulting answers (systems were required to

[11]Advanced Placement is a college entry exam in the US, cf. `http://apcentral.collegeboard.com/`.

produce natural language answers and justification for their answers) were then graded independently by three chemistry lecturers for correctness and for comprehensibility.

Scoring was done through an agreed-upon definition of ‘answer nuggets’ (i. e., information that had to be contained in a correct answer and in an acceptable justification), each worth one point. Results were reported to be surprisingly good: For the multiple choice questions, the best systems received around 70 % of the possible points for correctness and around 50 % for justification. For the more difficult essay-style questions, the best system still achieved around 30 to 40 % correctness with 20 % of the possible points for answer justification. This is even more surprising, as the mean scores for human testees are reported to be only slightly higher (Friedland et al., 2004a).

However, these good results must be put into perspective by considering the effort made to achieve them: Setting up the test took the teams nearly half a year each. Unfortunately, no exact figures (such as a number of person months) are given for the respective work-effort that was spent[12]. One interesting figure is cited, however, namely the estimate that encoding the knowledge from the 70 pages of chemistry text book cost an average of 10 000 US Dollars *per page* for each team (Friedland et al., 2004a, 22). When considering this figure, it should also be recalled that each of the teams already had an up-and-running KR&R system and a rich general ontology when they started the knowledge representation process.

Another interesting fact is the overall running time reported for the systems for answering the whole set of 100 questions, which ranges from two hours for the fastest system to twelve hours for the slowest. This means that the systems need an average of between one minute and seven minutes to answer one question. This seems to indicate that the KR&R approach to QA currently does not seem promising for interactive QA systems, especially when larger amounts of knowledge to be searched are involved (remember that the experiment was based upon information from 70 pages of a chemistry textbook).

Two interesting conclusions can be drawn from this study: First, it indicates an upper bound of coverage for QA using current KR&R techniques for complex, essay-style questions, that seems to lie around the 40 % mark. This figure, of course, must be handled with the due amount of caution, as tasks of this kind are extremely hard to compare. A failure analysis for all systems has shown that there is a large number of sources for errors; most problems seem to arise through difficulties in knowledge encoding, through insufficient expressivity of

[12]It is only stated that SRI used ‘four chemists to help with the knowledge formation process’ (Friedland et al., 2004a, 7).

the used languages and from unconvincing natural language generation facilities (Friedland et al., 2004b).

Second, scalability of this type of system seems still to be comparatively low even after decades of research and in spite of the existence of comparatively large repositories of knowledge encoded in formal ontologies (cf. also 4.4).

3.3.2.4 Discussion

We have discussed work on QA using Knowledge Representation and Reasoning as the basis. We have shown that the idea of employing KR&R for QA is appealing: By using a meaning representation and an inferencing mechanism, answers can be easily found in principle. One important advantage of the approach is that all relevant knowledge is assumed to be present in a formal language and thus accessible for reasoning. That means that the approach is geared towards finding indirect answers, as inferencing is used for answer finding in any case.

The task of finding answers, even indirect answers, can thus be defined more simply than checking answerhood for a given reply: We have shown above that to account for indirect answers, additional effort is required.

We have also noted that most researchers in the KR&R traditions are generally not interested in the question of automatically deriving knowledge from texts in natural language and/or generating natural language output. These issues are either skirted or assumed not to be relevant for the task. We will take up the discussion of deriving broad-coverage full meaning representations from general texts in the next section where we will come to the conclusion that it is still beyond the state of the art of current natural language processing systems.

We have additionally noted the large amount of labour needed to set up and maintain knowledge bases suitable for the reasoning process required for broad-coverage QA and the comparatively high processing time involved; we will also touch upon these points again in the following section.

3.3.3 Question Answering Based on Full Semantic Representations for Natural Language: Some Issues

In the previous section we have shown that QA based on Knowledge Representation and Reasoning has a number of advantages, especially the natural integration of an inference mechanism and the resulting possibility of employing explicitly encoded knowledge for finding indirect answers to questions.

Thus, using a full meaning representation and inferencing could form an ideal basis for a natural language QA system. By combining a system that automatically derives semantic representations from texts with a suitable KR&R

system, powerful QA systems could be built: This could be seen as using a KR&R system as a back-end for the natural language QA system. In such a combined system, both the document collection and the questions to the system would be translated into a semantic representation based on a formal representation language such as FOPL. By combining the information from the text documents with a suitable general knowledge base (holding linguistic and general world knowledge), the knowledge required for answering questions on facts contained in the document collection would then be readily accessible.

In spite of the advantages of using full semantic representations as the basis for QA, there are a number of issues that make this approach impractical for building actual large-scale natural language QA systems. We will highlight on some of the issues in this section.

We will start by discussing the difficulty of choosing an adequate representation formalism: While the representation formalism should be expressive as possible (to appropriately represent natural language expressions), the more expressive types of formal logic are computationally intractable in general.

We will then turn to the issue of automatically deriving full semantic representations from natural language texts. While this can be done for small examples (cf., e. g., Blackburn and Bos, 2006; Allen, 1995), the problem is still far from being solved for general texts.

We will conclude by pointing out that, even though ontologies defining tens of thousands of concepts are available, these knowledge resources still do not fully allow general reasoning about comparatively simple matters of daily life, such as shopping or playing tennis.

3.3.3.1 Representation Formalism: Issues of Expressivity and Complexity

First, we will discuss a number of issues related to computational complexity and representational adequacy. While it is, on the one hand, desirable to use expressive representation formalisms (such as intensional logic, cf. Montague, 1974) to capture phenomena in natural language, more expressive formalisms are, on the other hand, computationally more complex, making them unsuitable as a basis for large-scale, real systems.

Complexity Issues. For representing natural language, many semanticists have employed higher-order logics, most commonly intensional logic (cf. Montague, 1974). Intensional logic allows to naturally capture natural language phenomena related to intensionality, modality, tense and aspect and others (cf. Blackburn and Bos, 2003). These, especially intensionality, are not only relevant from the point of view of theoretical linguists, but are actually needed to capture central aspects of the meaning of – seemingly – simple texts (see, e. g., Condoravdi et al., 2003, who show that phenomena related with intensionality are commonplace in technical documentation). Thus, from a viewpoint of natural language semantics, using an expressive formalism such as intensional logic is desirable.

For actual, large-scale implementations, however, the use of higher-order logics does not seem suitable, as higher-order logics are incomplete, i. e., there are formulæ whose general validity cannot be shown in any calculus (Eisinger and Ohlbach, 1992; Gödel, 2006). This makes them unattractive for use in computer systems.

To overcome this problem, methods have been suggested that allow using FOPL representations as a useful approximation to a ‘full’ representation for natural language (cf., e. g., Blackburn and Bos, 2003): Approximations of higher-order representations can be defined in FOPL by adding suitable axioms constraining the models (van Benthem and Doets, 1983). While this approximation is, of course, less expressive than the higher-order logic, it can still be useful (Blackburn and Bos, 2003).

Currently, Description Logics (DL, Baader et al., 2003) are often discussed as an alternative formalism with even lower expressional power than FOPL but several advantages: A number of description logic formalisms (note that quite a number of different, more or less similar formalisms are generally lumped under the heading of Description Logic) can be efficiently processed. Especially in the context of the Semantic Web (Berners-Lee et al., 2001), different knowledge representation formalisms that can be translated into description logics have been discussed (such as OWL, W3C, 2004b; DAML, DAML, 2001 and RDF, W3C, 2000)

Undecidability of FOPL. Even when ‘only’ FOPL representations of natural language are used, this will result in computationally intractable systems, in general: Alan Turing has shown that FOPL is undecidable (Turing, 1936). The proof problem ($\mathscr{K} \models ?p$) is semi-decidable: If a proof exists, then it can be found in finite time, but if no proof exists the search for a proof will, in general, not terminate.

In practice, this means that one has to set a time limit for the search for answers, as otherwise the system may never ‘come back’. If, however, no proof (and consequently, no answer) has been found within the time limit, it does not follow that no answer exists in the knowledge base: There may exist one that simply has not been found yet.

Efficiency of Automated Reasoning Systems. As described above, it is provably impossible to construct a QA system that, based on an automated theorem prover for FOPL finds an answer in finite time in general.

The last few years, however, have seen an enormous improvement in the performance of practical theorem provers. Using clever search optimisations and indexing techniques, these systems can now find proofs in knowledge bases containing hundreds of thousands of facts efficiently (see, e. g, Hillenbrand, 2003). This is even true when FOPL with equality is used, which generated efficiency problems for 'older' systems (Blackburn and Bos, 2003).

Still, using theorem provers to control the amount of information required to represent large document collections with hundreds of thousands of documents (and thus millions of facts) and handle knowledge bases with hundreds of thousands of rules still seems beyond the scope of current systems. Johan Bos states, for example, that feeding 'only' the data contained in WordNet (cf. 4.2.1) into a theorem prover will already lead to efficiency problems:

> Of course, to compile the entire WordNet into first-order logic and give that to a theorem prover won't bring us anywhere – theorem provers are designed to deal with mathematical problems, they are not good at dealing with huge chunks of formulas.
> Bos (2006, 8)

Bos suggests using an additional step to extract only potentially relevant information from WordNet for inferencing purposes. This approach, however, is not compatible with the idea of using automated reasoning for finding answers.

Also recall the processing times for a relatively small knowledge source reported above in connection to the systems in the HALO project (3.3.2.3), where average processing times for complex answers of several minutes were given. These figures show that processing the knowledge as required for question answering constitutes an especially hard challenge for automatic reasoning systems.

It should also be noted that even optimised Description Logic provers may run into difficulties when processing knowledge bases of the size discussed here: In Haarslev and Möller (2001b), experiments are reported with checking large T-Boxes ('large' meaning about 100 000 individuals here) for consistency with a given A-Box. Even on their optimised system, these checks still took about 20 minutes to run.

3.3.3.2 Automatically Deriving Structured Semantic Information from Natural Language

There is general agreement that deriving full semantic representations from general natural language texts is currently beyond the state of the art of natural language processing, at least outside closely restricted domains and/or text types (cf., e. g., Marcu and Popescu, 2005; Monz and de Rijke, 2001).[13] We will review the arguments brought forward in some more detail.

It should be noted that all arguments are focussed on *broad, general* coverage. The general assumption is that, given sufficient resources, it would be possible to find representations (and methods for automatically deriving them) for each given natural language sentence, which are suitable for the purpose at hand – even though the representation may not have theoretically desirable properties such as being linguistically 'convincing' in every detail or being generalisable to related phenomena. However, these methods will not generalise well enough to cover a sufficient portion of sentences in general texts. Thus, the problem does seem to be not so much one of depth, but rather one of broad coverage: An enormous challenge in building 'real' systems, which are to correctly handle large amounts of texts (cf. Chanod et al., 1994).

Representational Adequacy. There is generally no agreed-upon 'standard' way of representation for a wide range of natural language phenomena. Most representation formalisms have a number of drawbacks, because they will be unsuitable for certain purposes. Among other things, this includes issues of granularity. As a simple example, just consider how a compound like *'stock price'* should best be represented in FOPL. Should it be rendered as one predicate (stock_price′)? Then, additional knowledge would be necessary to relate stock_price′ to its super-ordinate price′. Representing it along the lines $\text{price}'(x) \wedge \text{of}'(x,y) \wedge \text{stock}'(y)$ relates it more closely to price′, but may have other disadvantages (cf. Marcu and Popescu, 2005, 90).

This overall uncertainty is captured by the following quote, where Daniel Marcu and Ana-Maria Popescu – somewhat provocatively – state that to appreciate the difficulties one actually has to really attempt doing so:

> The reader is strongly encouraged to try to write FOL [First-Order (Predicate) Logic] formulas to express the meaning of the first few sentences in one of today's top New York Times stories… It is the best exercise for understanding the limits of using FOL as a vehicle

[13] A recent series of papers by Johan Bos and his colleagues (Bos, 2006; Bos et al., 2004), in which a wide coverage system is described that can derive semantic representations for a wide range of input texts will be discussed later in this section.

for natural language semantic interpretation.
Marcu and Popescu (2005, 91, my explanation)

Note that this ties in with the difficulty of the task of representing information from a text book in a formal representation language and the associated high cost reported in the HALO pilot study (3.3.2.3): Even though the participating groups have shown that, in principle, it is possible to extract and represent the contained knowledge (even if only manually), the required effort was enormous.

Ambiguity. Another point that is touched upon in Monz and de Rijke (2001) is the issue of ambiguity: Nearly every single natural language sentence is ambiguous at some linguistic level (cf. Gardent and Webber, 2001). As it is mostly not possible to resolve these ambiguities directly, they would in general need to be overtly expressed in a full representation. This, however, would result in a blow-up of the representation size, as that may grow exponentially with the input size (cf. Monz and de Rijke, 2001, 63). This general problem has been addressed by a number of formalisms using underspecified semantics that keeps the size of the representation small while preserving the ambiguities (such as Minimal Recursion Semantics, MRS, Copestake et al., 2005, Constraint Language for Lambda Structures, CLLS, Egg et al., 2001, 1998, Underspecified Discourse Representation Structures, UDRS, Reyle, 1993, Underspecified Semantic Description Language, USDL, Pinkal, 1995). Note that no general theorem prover currently supports reasoning on such underspecified representations.

Consistency. We have mentioned above (3.3.2.2) that the knowledge base $\mathscr{K}$ representing the document collection and the inferential knowledge available to the system needs to be consistent: Finding an answer using a theorem prover relies on the assumption that adding the negation of the question representation introduces the *only* inconsistency. If $\mathscr{K}$ itself is inconsistent, then every question would receive a positive answer, as running the theorem prover will detect this inconsistency.

While this is a simple enough requirement in theory, ensuring consistency of an automatically derived (or manually built) knowledge base is far from easy, as the processed documents may contain inconsistencies. While it would be, in general, possible to avoid global inconsistencies by assigning pieces of knowledge to separate contexts and thus keep them apart (provided that the representation formalism supports this), ensuring that knowledge at different places in the collection can be combined for reasoning is then of course far from being simple.

Rules in the Cyc ontology (cf. 4.4.1, Matuszek et al., 2006), for example, are strictly contextualised: They are grouped into so-called micro-theories; knowledge stored in different micro-theories may only be brought together if there is an explicit accessibility relation between the two micro-theories involved. However, this requires an additional effort while encoding the knowledge, on the one hand, and may prevent correct inferences from being drawn, on the other hand, due to accessibility relations that were forgotten during knowledge engineering.

The same general problem also applies for facts derived automatically from text documents: Automatically detecting and properly handling inconsistencies poses a serious problem.

Coverage and Robustness. While the issues described so far are especially related to the 'proper' way of representing natural language, this point is related with the actual natural language tools. In spite of recent advances in natural language processing, there are still no parsers that can derive a 'deep' semantic representation from arbitrary natural language input.

The probably most important reasons are a lack of both broad coverage resources (lexicon, syntax, semantics) and of robustness in processing (e. g., one typographic error in the input sentence will often cause the parsing process to fail). Recent approaches have tried to overcome this problem at least in part by fall-back strategies or a combination of 'shallow' and 'deep' parsing strategies (cf., e. g., Callmeier et al., 2004).

Instead of deriving and working on full semantic representations (as in the textbook approaches described in Blackburn and Bos, 2006; Allen, 1995), most current systems use partial representations tailored for the task at hand, such as Moldovan et al. (2003a).

In a series of recent papers (Bos, 2006; Bos et al., 2004), Johan Bos and his colleagues describe a wide-coverage system. The system is based on a Combinatory Categorial Grammar (CCG) parser. The authors report that the system can derive semantic representations automatically for a substantial number of general texts:

> We demonstrate that well-formed semantic representations can be produced for over 97 % of the sentences in unseen WSJ text. We believe this is a major step towards wide-coverage semantic interpretation, one of the key objectives of the field of NLP.
> Bos et al. (2004)

It turns out, however, that there has so far been no full evaluation of the derived representations. It is therefore far from clear how correct or suitable these representations are. In a recent paper, Johan Bos has reported an evaluation that is based on the automatic generation of semantic models from the representations and so-called mini WordNets (i. e., only those parts of the lexical hierarchy of WordNet for which words are present in the input text; for WordNet see 4.2.1). He found that for 88 % of his test texts he could successfully generate such a model, meaning that WordNet knowledge and derived semantic representation were consistent (Bos, 2006).

However, though it is important that the derived representations are consistent (see the discussion above), this is only one aspect: It does not follow that the representations are actually useful semantic representations. This remains a matter of future research.

3.3.3.3 Axiomatising Knowledge

Axiomatising inference knowledge, both natural language knowledge and world knowledge, is an additional problem. The translation of natural language into a formal logic representation is only the first step towards a QA system based on automatised reasoning. In order to be able to reason with this representation, the system needs to have access to a knowledge base of natural language knowledge and world knowledge in the form of knowledge axioms. A system must, for example, have access to the knowledge that a statement like *'A acquired B.'* is informationally equivalent to *'B was sold to A.'*, or rather, that the respective derived representations are equivalent. Thus it needs knowledge axioms that allow inferring the equality of the respective representations. The additional knowledge needs to be made available to the system in a way that is consistent with the natural language representation, of course: A suitable mapping from natural language representations to concepts in the knowledge base must exist.

It is comparatively difficult to describe and enumerate the required knowledge exactly; for humans it comes so natural to employ it that it is hard to pin down exactly and externalise. It seems that both linguistic knowledge and world knowledge are needed and that they are interlinked to a degree that they cannot be patently separated (cf., e. g., Kay, 1989; Fillmore, 1982).

To date, no comprehensive repository of all the knowledge that would be required for general inferences in a formal language exists. We will discuss two kinds of possible sources of partial knowledge below, namely the lexical databases WordNet (4.2) and FrameNet (4.3) and the generic ontologies Cyc (4.4.1) and SUMO (4.4.2). As we will see, coverage of all these resources is still comparatively limited, even though many person years of diligent work have gone into each of them.

3.3.4 Discussion

Delegating the task of finding answers to questions in a QA system to an automated reasoning system based on ‘deep’ semantic representations of both document collection and question seems an ideal solution: It naturally supports the need for reasoning to derive a direct answer to the users’ questions from the document collection that will, in general, contain only indirect answers.

In this section we have shown a number of practical objections why (given the current state of the art in Natural Language Processing and Knowledge Representation and Reasoning) having a QA system rely solely on automated reasoning based on full semantic representations for finding answers to the questions does not seem an adequate approach: Techniques for deriving deep semantic representations from texts are not yet robust and general enough to allow processing arbitrary texts. Then, even given the recent advances in efficiency of theorem provers, they are not (yet) suitable tools for searching the representations of huge document collections for answers. Moreover, comprehensive repositories of general knowledge in a formal language are still not available today.

When considering the greater picture, a pattern begins to emerge: We have argued above that approaches to QA based only on bags-of-words or simple surface pattern matching have a number of drawbacks, especially lack of precision, and also (when not using query expansion techniques) lack of recall.

Approaches based on full semantic representations were shown to potentially remedy these disadvantages when answer finding is done on abstract representations using reasoning. We have seen, however, that these approaches are not generally usable and that resources (both the required large knowledge and suitable NLP tools) are still missing.

We have therefore decided to aim for middle ground: By using text and question representations that abstract over a number of surface phenomena (such as word order or syntactic and lexical semantic variation), we arrive at a level of abstraction that allows finding answers that are quite different from the questions in terms of surface words. At the same time we stay within the limits of current robust NLP techniques, in not requiring too abstract representations.

3.4 Recognising Textual Entailment

In this section, we will summarise related work on Recognising Textual Entailment. Textual Entailment has recently grown into an area of research in its own right, mainly centred around the newly created shared-task competition

'Recognising Textual Entailment' (RTE, Bar-Haim et al., 2006; Dagan et al., 2005). The RTE challenge has been conducted by the Pascal network of excellence, which is an EU-funded research cluster that supports, among other things, a number of language resources and evaluations. So far, there have been two rounds of the challenge in 2005 and 2006, respectively.

We show that the task and the relation of textual entailment is related to indirect answerhood. We will summarise work done in the context of the challenge and show that, while it has produced some insights (especially a discussion on the need for 'uncertain' inferences and the use of world-knowledge) and some successful systems, no systematic description of the required inferencing techniques and of the involved knowledge has been presented.

3.4.1 The Textual Entailment Task and its Relevance for Question Answering

Participants in the RTE challenge have to prepare a computer system that decides, for two given text fragments, whether the one can be inferred from the other[14]:

> We consider an applied notion of textual entailment, defined as a directional relation between two text fragments, termed t – the entailing text, and h – the entailed text. We say that t entails h if, typically, a human reading t would infer that h is most likely true. This somewhat informal definition is based on (and assumes) common human understanding of language as well as common background knowledge. Textual entailment recognition is the task of deciding, given t and h, whether t entails h.
> Bar-Haim et al. (2006, 1)

For example, given the text '*Israel's prime minister, Ariel Sharon, visited Prague.*' and the hypothesis sentence '*Ariel Sharon is Israel's prime minister.*', the system should return true (i. e., textual entailment holds between t and h). Participants receive a number of training examples, where each pair is marked manually by the organisers for textual entailment, in order to set up and test their systems. In the challenge itself, participants let their systems automatically annotate previously unseen examples of the same kind for textual entailment.

The task seems to be quite difficult to solve: In the first round, the best systems reached an accuracy of 60 % (with a trivial baseline of 50 %: test examples

[14]We cite the definition as given for the second instalment of the challenge in 2006, RTE2. It is virtually identical with that of the first RTE, however, the stress on '*most likely* true' has been added (cf. Dagan et al., 2005), following discussion, see, e. g., Zaenen et al. (2005).

are, by design, equally divided into positive and negative examples, cf. Dagan et al., 2005). The second challenge has seen two systems performing markedly better, namely around the 75 % level (Bar-Haim et al., 2006).

The organisers explicitly stress the relevance of the tasks for question answering:

> It seems that major inferences, as needed by multiple applications, can indeed be cast in terms of textual entailment. For example, a QA system has to identify texts that entail a hypothesized answer. Given the question "What does Peugeot manufacture?", the text "Chrétien visited Peugeot's newly renovated car factory" entails the hypothesized answer form "Peugeot manufactures cars".
> Dagan et al. (2005, 2)

Both training and test examples were drawn from a number of different sources (cf. Bar-Haim et al., 2006; Dagan et al., 2005, 3–4). Among these sources were pairs of questions and text fragments containing an answer (either judged correct or incorrect) from the QA shared tasks CLEF and TREC (cf. 2.2.1). For the RTE challenge, interrogatives were changed to declaratives and other minor changes were made.

For example, from the question '*Who is Ariel Sharon?*' and the text containing the candidate answer '*Israel's Prime Minister, Ariel Sharon, visited Prague.*', the hypothesis '*Ariel Sharon is Israel's Prime Minister.*' is derived.

This example shows that the textual entailment relation is closely related to the relation of indirect answerhood that we have singled out as a central concept for linguistically informed QA (3.2.4).

The challenge has fostered a number of research papers; we will summarise some of the results. We start by looking at three papers that discuss the task of textual entailment and relate it to 'classical' semantic and pragmatic relations. We then give an overview of the participating systems and the methods that they use for modelling the textual entailment relation.

3.4.2 Circumscribing the Task

Zaenen et al. (2005) was presented at an ACL 2005 workshop on RTE. The authors suggest that the relation of textual entailment is actually made up from different relations, namely (semantic) entailment proper, conventional implicatures (Karttunen and Peters, 1979) and conversational implicatures (Grice, 1989). The authors suggest that the 'textual entailment' should better be renamed 'textual inference' to clearly distinguish it from the well-defined and narrower notion of semantic entailment. We will follow this suggestion and use the term 'textual inference' from now on.

The authors cite several examples for the different types of inference from the RTE 1 data. While they are willing to allow both cases of semantic entailment and conventional implicature (especially cases exhibited by certain appositive constructions, Zaenen et al., 2005, 33) as candidates for textual inference, they are less convinced about cases of conversational implicatures. They argue, for example, that from *'Green cards are becoming more difficult to obtain.'*, one cannot generally infer *'Green card is now difficult to receive.'* (Zaenen et al., 2005, 35). The conclusion in the second sentence may be conversationally implicated, but it could be easily cancelled by a sentence like *'But it's still no real problem to get one.'*.

Thus, cases that exhibit 'only' conversational implicatures should not be marked as cases of entailment, but rather using a different label, as they are likely to be controversial. In fact, several papers by participants in the challenge list disputable examples where the authors disagree with the 'official' entailment judgment – generally, of course, in cases where their own system differed from the official judgment, e. g., Bayer et al. (2005); Bos and Markert (2005); Newman et al. (2005).

Zaenen *et al.* also note that the amounts and proportions of linguistic knowledge and world knowledge involved in different examples from the RTE data differ substantially and suggest that additional annotation might help to clarify controversial cases. The authors report that they have tested their suggestions, in that they have taken part in defining a more detailed annotation scheme and annotated a smaller corpus similar to the RTE corpus in the context of the AQUAINT Knowledge-Based Evaluation (Crouch et al., 2006, 2005), especially including a distinction between *strict* and *plausible* inferences and between inferences whose (main) source is linguistic knowledge and others whose main source is world knowledge. The authors clarify that they are currently mainly interested in linguistic inferences (cf. Crouch et al., 2006).

In an answer to Zaenen et al. (2005), Christopher Manning defends both the RTE challenge itself and (with some small reservations) also the way it has been set up. Manning argues strongly against reducing the RTE challenge (and the study of textual inference) to clear-cut cases needing only linguistic knowledge to be resolved. In his opinion, the RTE challenge opens an interesting research area where, ideally, research in Knowledge Representation and Reasoning (KR&R) and Natural Language Processing could (re-) converge; he states that research in KR&R got side-tracked when they started to ignore tasks involving real-life language data as applications (cf. Manning, 2006, 5; see also 3.3).

Manning argues for keeping cases that do not exhibit semantic entailment but one of the 'weaker' forms, especially conversational implicature: Future applications in Information Access will, in his opinion, need to make use of exactly this sort of inferences that are not certain, but very likely. Among other reasons, he reports that annotators have difficulties with more complex annotation schemes for inferences, introducing more controversial cases instead of removing any (Manning, 2006, 10).

A case of uncertain inference that he especially draws attention to is the use of reported speech with constructions such as *according to* and modals like *may*. While these constructions are assumed by most semantic analyses to be opaque, i. e., the truth or falsity of the embedded propositions cannot be inferred, people tend to 'believe' the embedded statements, generally quite successfully gauging the certainty of their belief based on the (perceived) trustworthiness of the source of the statement.

Finally, Manning suggests that the RTE task could be improved if not only very short text snippets, but rather short passages were used as text part of the text-hypotheses pairs: Most controversial cases would become clearer, he argues, if more context was provided and some possible, but unlikely readings could be excluded.

Crouch et al. (2006) reacts to Manning's criticisms by stating that with Zaenen et al. (2005), the authors did not intend to have examples involving cases exhibiting conversational implicatures or heavy use of world knowledge removed from textual inference tasks. Their suggestion was rather to use a clearer and more detailed annotation scheme.

In our opinion, characterising the RTE task would probably become easier when it was seen in the context of specific applications: A system for recognising textual inference could, as described by the initiators of the challenge (cf. Dagan et al., 2005), form part of a larger system, such as a QA system. In the context of such a system, it would be easier to define which sorts of inferences should or should not be integrated from the perspective of the end-user. However, for an evaluation that is not contextualised, this is harder to define.

There are – at least – two important points to be gleaned from this discussion: First, people seem to allow different types of inferences when asked if some fact follows from a text (that is, when asked about possible inferences). Semantic entailment and conventional implicature can account for a substantial part of cases. However, in addition, people also accept uncertain inferences, based, for example, on conversational implicatures (unless they are explicitly cancelled) or on reported speech attributed to trustworthy sources.

Second, people will use both linguistic knowledge and world knowledge for textual inferences.

A useful definition of indirect answerhood (which is closely related to textual inference) should accommodate these points.

3.4.3 System Descriptions

We will now turn to the descriptions of systems that have participated in the RTE competitions. We are especially interested in the question how systems model the textual entailment relation, as such a model could provide insights for modelling the (similar) relation of indirect answerhood.

While in the first RTE challenge, the best performing systems relied on word-based measures of semantic similarity learned from the Internet (cf. Dagan et al., 2005), the second round showed that several systems using deeper analyses (especially syntactic matching and logical inferences) outperform knowledge-poor systems significantly (Bar-Haim et al., 2006): Given the current task and data set, there seems to be an upper limit of 60 % accuracy for systems using only lexical and no structural information (Bar-Haim et al., 2006). This indicates that trying to model textual inference (and thus also indirect answerhood) using word-based methods alone is not promising.

We will describe three systems here that have reached high accuracy scores in the evaluation and give an overview of the linguistic resources that they use. Several other RTE systems will be described below, where we describe NLP systems that use graph matching (5.2.1).

The predominant approach used for systems participating in the RTE 2 challenge employs matching structured representations of text and hypothesis. Different similarity measures are utilised in matching. These measures often include conceptual similarity measures derived from WordNet or similar sources and structural similarity measures between syntax structures derived by parsers. The different measures are combined by machine learners, which are trained with the training data provided by the organisers.

This general strategy is exemplified by the best-performing system in the RTE 2 challenge, which combines information from many different sources (Hickl et al., 2006): It especially uses a combination of syntactic information with lexical semantic information from WordNet and PropBank, enhanced with paraphrase information collected automatically from the Internet; the different features are combined by machine learning techniques. This system mainly differs from the other systems by the amount of training data used: In order to get better results for machine learning, the authors automatically collected about 200 000 additional training examples from the Internet. They used a combination of headline and first sentence in news-stories as positive examples and sentences with NE overlap, but little other word overlap as negative examples. The authors report that training on this additional corpus boosted their system's accuracy score by almost 10 % on the evaluation data.

The system that came in second employs logical inferencing using the theorem prover Cogex (described in Moldovan et al., 2003a; Cogex is derived from the general theorem prover Otter, McCune, 2003). It draws on extended WordNet (cf. 4.2.1.2), a set of manually coded natural language axioms (e. g., translating natural language conjunctions into logical conjuncts), manually coded semantic axioms (containing, e. g., a meaning postulate for intersective adjectives) and temporal axioms as sources of knowledge (Tatu et al., 2006). Unfortunately, the axioms are only sketchily described and few examples are given.

Bos and Markert (2006) describes a system that utilises semantic representations in the style of Discourse Representation Structure (DRS, Kamp and Reyle, 1993) derived automatically from both text and hypothesis as input for automated reasoning (using the wide-coverage parser described above, cf. 3.3.3.1, Bos et al., 2004). As a knowledge base, the authors employ hyponymy and synonymy information from WordNet and a small set of general rules, manually constructed to cover the training examples of the RTE task, spanning, among others, possessives, active/passive alternations, but also more specific knowledge, such as ‘if A is the wife of B, then A is married to B and *vice versa*’.

Two conclusions can be drawn from these system descriptions: First, linguistic information is needed for modelling textual inference, especially structural information and also information about lexical semantic similarity and paraphrases.

Second, the descriptions of the systems are not detailed and also not systematic enough to derive a model for indirect answerhood from them. In the next section, we will sketch a model of indirect answerhood and list phenomena that are relevant, with examples from a corpus study of questions and answers.

3.5 Indirect Answers

We have identified indirect answerhood as the core concept for finding answers in linguistically informed QA (3.2.4.2).

While this notion is theoretically attractive, we have shown that a computational implementation based on a full semantic representations of document collections and questions currently does not offer a workable solution (3.3).

We have also pointed out that the textual inference relation, as specified in the Recognising Textual Entailment challenge, is related to that of indirect answerhood (3.4). However, work on the RTE has not yet produced a model that we could directly take over. There is also no specification of phenomena that it comprises; it is defined empirically.

We will therefore develop a notion of indirect answerhood. It will be formally defined in chapter 5. In this section, we will first characterise the relation. As we cannot base the relation upon full structured semantic representations of questions and document collection (3.3), we want to approximate ‘full’ indirect answerhood. The aim is to arrive at a definition of the indirect answerhood relation that is, on the one hand, useful for QA (i. e., it covers a broad range of inferences that are relevant for QA) and, on the other hand, efficiently tractable. Therefore, we will take a closer look, in this section, at types inferences in indirect answerhood and phenomena that give rise to them as a basis for defining indirect answerhood.

We will first split the relation of indirect answerhood into the two relations of textual inference and direct answerhood (3.5.1), taking our lead from Groenendijk and Stokhof (1997). We will then list types of inferences that should be accounted for by the two relations and relate the different types to linguistic phenomena. This list is illustrated with examples from a corpus study of questions and answers from past TREC and CLEF QA competitions (3.5.2.1).

3.5.1 Specification

Before turning to the list inference types that need to be accounted for by a relation of indirect answerhood, we will first describe this relation somewhat more closely.

We split the relation of indirect answerhood into the two relations of textual inference and direct answerhood. This will be described in 3.5.1.1. We will introduce the general idea with an example first.

(3.58) Text: *John F. Kennedy was assassinated by Lee Harvey Oswald in Dallas, Texas.*
Question: *Who killed John F. Kennedy?*
Answer: *Lee Harvey Oswald killed John F. Kennedy.*

Despite the obvious differences in surface wording, a human reader and speaker of English can, when given *Text* and *Question*, easily produce *Answer*. We assume that the relation of indirect answerhood holds between *Text* and *Question* and that of direct answerhood between *Question* and *Answer*. This is the case, because we can account for all differences in surface wording by linguistically motivated inference steps as follows:

- The verb *assassinate* is mapped onto its hypernym (generalisation) *kill*: From the fact that the hypernym relation holds, we can conclude that whenever *A assassinates B*, it necessarily follows that *A kills B*. (Textual inference, lexical semantics)

- The active voice of the question is mapped onto the passive voice of the text; this preserves meaning. (Textual inference, syntactic mapping)

- Every component of the question is mapped onto the text; the 'superfluous' material in the text, namely *in Dallas, Texas* can be ignored[15]. (Direct answerhood, inclusion)

The inference steps together produce *Answer*, which in turn can be mapped onto *Question* using the direct answerhood relation: *Question* and *Answer* differ only in their sentence mood, after mapping *Lee Harvey Oswald* onto *who*, which is possible given the identity of the syntactic type (subject NP) and the semantic type (person).

We will specify the relations of textual inference and direct answerhood (3.5.1.1), before giving some additional details concerning negation (3.5.1.2) and uncertain answers (3.5.1.3).

3.5.1.1 Textual Inference and Direct Answerhood

We will first characterise the indirect answerhood over surface texts, without making assumptions about underlying structures. Given a natural language text τ and a natural language question ε (both concepts are to be defined presently), the indirect answerhood relation is to hold exactly if the text contains an (indirect) answer to the question.

For ease of exposition, we break down the definition of the relation into two parts (taking our lead from Groenendijk and Stokhof, 1997), namely the relation of direct answerhood and a number of inference steps that transform the text τ into a suitable form. Indirect answerhood can then be semi-formally characterised as follows:

Indirect Answerhood as Textual Inference and Direct Answerhood. A natural language text τ indirectly answers a natural language question ε iff there is at least one series of inference steps $\tau \rightarrow \tau' \rightarrow \ldots \rightarrow \tau^*$ (where $\rightarrow$ is the relation of textual inference) so that the direct answerhood relation $\tau^* \rightsquigarrow \varepsilon$ holds (where $\rightsquigarrow$ is the direct answerhood relation between a text and a question).

Of course, textual inference and direct answerhood cannot adequately be defined on the text surface alone; in chapter 5, we will give a definition that is based on the matching of linguistic structures.

The textual inference relation $\rightarrow$ is defined as a relation between two natural language texts τ and τ' that holds iff τ' can be inferred from τ. Textual inference

[15]This simplification needs, of course to be investigated more closely. Leaving out negations, for example, would seriously flaw the answer, as then the answerhood relation no longer holds. We will return to this point below.

is, for example, assumed to hold when τ' can be derived from τ *exactly* by replacing one word in τ by a synonym or if a sentence in τ only differs from a corresponding one in τ' in that one is in active voice and the other in passive voice.

It should be noted that this specification of the indirect answerhood relation is not to be understood as a real series of transformations on the text surface that has to be performed. We use it to describe the intuition that the relation can be broken down into a sequence of elementary steps, without claiming that it has any linguistic, especially psycho-linguistic, reality. We will therefore not further be concerned with questions such as defining a minimal sequence of inference steps etc.

The similarity between this definition and the definition of textual entailment of the Recognising Textual Entailment challenge (RTE, cf. 3.4) is obvious. However, our focus here is on the basic mapping steps that *together* allow to establish that one piece of text can be inferred from the other. The textual entailment relation of the RTE, by contrast, looks at the *overall* relation.

The direct answerhood relation $\rightsquigarrow$ holds between a text τ and a question ε iff the following hold:

- τ and ε are structurally equivalent but
- τ is in declarative mood and ε in interrogative mood and
- each *wh*-phrase contained in ε (if any) has a corresponding phrasal counterpart in τ that is of the correct syntactic and semantic type.

In general, this will be a relation of direct, but partial answerhood in the sense of Groenendijk and Stokhof. We will assume that ε is always in the form of an interrogative. As described above (3.2.4), questions can, of course, be expressed by other means, but we will restrict ourselves to interrogatives.

3.5.1.2 Answers with Inverted Polarity

Negation plays an important rôle for inference. *‘Lee Harvey Oswald did not murder John F. Kennedy.’*, for example, must *not* be taken to entail *‘Lee Harvey Oswald murdered John F. Kennedy.’*

Negation in texts has therefore sometimes been used as an argument for employing deeper linguistic methods (e. g., Harabagiu and Lacatusu, 2004; Attardi et al., 2002 on negation in QA, cf. also, e. g., Delmonte et al., 2005, on handling negation in RTE). So far, the standard solution has been to discard possible answers containing negation.

We would like to suggest a different view in the light of of the work on questions and answers discussed above: Several semantic approaches explicitly allow answers that consist of negated forms of the proposition contained in the question (notably the partition approaches, Groenendijk and Stokhof, 1997, and, even more explicitly, Higginbotham, 1996, who builds up partitions from non-negated and negated versions of the question proposition instantiated for all individuals, cf. 3.2.2.4). This makes sense in that negated answers are generally good answers, as the following example shows:

(3.59) Who is coming to the party?
Tom is not coming./Not Tom./Tom isn't.

Answers whose polarity differ from that of the question must be considered for QA, i. e., negative answers to positive questions and positive answers to negated questions. We will call these answers 'answers with inverse polarity'. However, one important point needs to be kept in mind: When the system gives an answer with inverse polarity, this difference must be properly expressed.

The key point is to define the relation of direct answerhood in such a way that it allows matching questions and potential answers both of the same and of different polarities. In cases where the polarity differs, however, this fact must be properly reported to the user. Note that the relation of textual inference must not be changed; the recognition of differing polarities can be done during the matching of question and (direct) answer.

We suggest a solution that consists of complementing the matching of question and answer with an answer checking step (5.1.5) that is used to mark answer with inverse polarity. Such answer will be presented to the user with a suitable warning message. In our implementation, for example, the system will issue a warning and always output the whole answer sentence, not only a constituent answer (6.4.3.2).

3.5.1.3 Uncertain Answers

We have discussed that textual inference encompasses not only 'proper' semantic entailment, but also 'uncertain' inferences (called 'strict' and 'plausible' inferences in Crouch et al., 2006, 2005; cf. 3.4.2).

The following types of uncertain answers can be distinguished. Note that these different types are heterogeneous, we will subsume them all under the term uncertain answers.

Answers in Opaque Contexts. Possible answers may be embedded in semantically opaque contexts, especially in reported speech.

Answers Containing Modals. If an answer contains a modal verb or modal particle not present in the question, it is considered uncertain. This case is similar to the previous one.

Answers Based on Uncertain Inferences. Answers may be identified where indirect answerhood between the question and the uses uncertain inference steps. For example, a textual inference step may consist in replacing a word by its hypernym (see below). In this case, strict entailment does not hold, but the inference is still likely.

Note that different linguistic expressions interact in complex ways. For example, negation often changes the direction of entailment (Nairn et al., 2006). Deep linguistic processing would be needed to distinguish certain and uncertain answers. Instead of discarding all answers that contain any ‘dangerous’ material (such as *verba dicendi* or modal verbs), we currently identify *potentially* uncertain answers and mark them accordingly.

If both certain and uncertain answers are found, certain answers are preferred. In cases where no certain answer is available, the system may present uncertain ones. However, these will be output together with a warning, explaining that and why this answer is to be treated with care.

Uncertain answers will be identified using an additional answer checking step, similar to negation (5.1.5).

Note that we still refer to the underlying relation as one of textual inference. With the admission of ‘uncertain’ inferences, it might be argued that it is rather a relation of relevance than of actual inference. However, we think that the main contribution is still one of certain or likely inference (as human users see it), and that, thus, the name is still justified.

3.5.2 Relevant Phenomena

In the previous section, we have specified the indirect answerhood relation, which we have identified as the central concept of linguistically informed QA (3.5.1). We have split the relation of indirect answerhood into the two relations of textual inference and direct answerhood.

We will now further characterise these two relations: So far, we have described how they interact to relate questions to texts that provide indirect answers for them. We will now list types of textual inferences that are necessary (3.5.2.2 through 3.5.2.4), on the one hand, and further characterise how different question types can be accounted for (3.5.2.5), on the other hand (e. g., how *which* COMMON NOUN phrases can be handled).

We will especially classify types of inferences by linguistic phenomena with which they are related. For example, if two texts differ only by the fact that one word is replaced by a synonym, then the relation of textual inference holds between the two.

The list of phenomena was compiled with the help of a corpus study of question-answer pairs and the texts from which the answers were extracted from the TREC 2004 and CLEF 2003 QA tracks.

Our starting point in investigating the difference between questions and answers at the surface level is to systematically look at language variability: One abstract underlying meaning can be expressed by a wide range of different (equivalent) surface realisations (variants). These systematic variations and the regularities that govern them have been widely studied in linguistics. For example, a sentence in the active voice corresponds to another one in the passive voice as it carries essentially the same meaning.

We will group the types of inferences by the related linguistic phenomena. The different types will be illustrated with examples from the corpus. Note that in most cases more than one inference step is necessary to account for the examples. We will not describe all required inferences for the examples, but focus on one inference step.

3.5.2.1 Corpus Study

The list of inference types in this section was drawn from a corpus study. As the corpus we used question-answer pairs and the texts containing the respective answers from the TREC 2004 and CLEF 2003 QA tracks (cf. 2.2.1). We selected about 100 question-answer pairs from each collection and went through them to find what types of inference steps are required to arrive at the answer when both the question and the text containing the answer are given.

We summarise what we found to be the most important types of inference that would be needed to account for the examples that we looked through. For each phenomenon, we present at least one clear-cut example.

Note that it is not always possible to unequivocally identify all necessary inference steps. First, without limiting oneself to specific resources (without, for example, using WordNet to check what lexical relations WordNet suggests hold between two words), it is not always possible to correctly identify which specific relations hold between word pairs in the question and in the answer. Second, we found that it is quite easy to overlook necessary inferencing steps, as drawing inferences from texts comes very natural. We have therefore restricted ourselves to a manageable inventory of types of inferences related to well-known linguistic phenomena, such as syntactic variation and lexical semantic relations and concentrated on clear-cut cases.

We have used the following sources: The list of TREC 2004 QA track questions (TREC, 2004b), the corresponding judgments for factoid questions (i. e., the list of all answers returned by participating systems together with the marking of the evaluators, TREC, 2004a), the documents from the LDC AQUAINT collection[16], the QA@CLEF 2003 cross language (German) track question collection (CLEF, 2003) and the CLEF document collection[17]. The examples used in this section are cited with the respective competition's question number. For documents, we cite the unique document identifiers as assigned by the conference organisers.

We manually 'expanded' some questions from the TREC collection for ease of readability. In the 2004 QA track, several questions concerning one subject (target) are grouped together (Voorhees, 2005). The target is explicitly given and the questions often make only anaphoric reference to it. We have manually replaced such anaphors with their 'full form'. Consider, e. g., example (3.60) below: The question (TREC 2004 Q 3.1) is taken from TREC 2004 target 3, namely *'Hale Bopp comet'*. In (3.60) we have modified the original question *'When was the comet discovered?'* using the target to read *'When was the Hale Bopp comet discovered?'*.

Note that we do not address the question how actual systems taking part in the different competitions have in fact managed to arrive at a certain answer. As described above (2.2.2), with very few exceptions, the participating systems do not try to establish a full inference chain, but rather use other techniques (pattern matching, use of external knowledge sources etc., cf. 2.2.2).

3.5.2.2 Syntax and Morphology

In the following, we will look at meaning-preserving morpho-syntactic variants. Morphological changes that are necessitated by syntactic differences will be assumed to be covered by the respective syntactic variation and will not be discussed separately.

Each of the different types of syntactic variations discussed here figured in 7–10 % of the question-text pairs in the corpus.

Diatheses. We have mentioned active/passive as an example of syntactic variation. Corresponding sentences in the active and passive voice should be treated as equivalent. Note that more inference steps are necessary to derive a direct answer from the text. Especially anaphora play an important rôle in this sentence, as it must be established that *'it'* refers to *'the comet'* which, in turn, refers to *'the comet Hale-Bopp'*, cf. also 3.5.2.4.

[16]Available through the Linguistic Data Consortium, LDC, Catalog number LDC2002T31: `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002T31`

[17]Made available through the CLEF consortium, `http://www.clef-campaign.org/`

(3.60) When **was** the Hale Bopp comet **discovered**? (TREC 2004, Q 3.1)
The comet Hale-Bopp… The comet was named after its two observers – two amateur astronomers in the United States who **discovered** it on July 22, 1995. (XIE19960105.0039)
July 22 , 1995

Besides the active/passive diathesis, there are a number of additional, language specific diatheses. Among the more prominent ones in English is dative shift (cf. '*John gave the book to Mary.*' vs. '*John gave Mary the book.*'). However, we have not found any examples of these types of diathesis in the corpus. In Katz and Levin (1988), some examples are listed and described as important for question answering.

Possessives. Another common variation is the systematic relation between (pre-nominal) possessives and (post-nominal) *of*-PPs in English (PP_{von} in German), as exhibited in the following example.

(3.61) Who is the **AARP's top official or CEO**? (TREC 2004 Q 5.4)
Horace Deets, **president and co-CEO of the AARP**, addresses a National Press Club luncheon on the senior advocacy group's legislative priorities and plans for the future. (NYT20000908.0136)
Horace Deets

Note that the underlying relation must not be one of possession in the strict sense of the word; rather, a number of different relations may be similarly expressed. Most of them allow the use of either a possessive or a post-nominal PP.

Compounds. Compounding is widely used in German. As German compounds are written in one word (in English, most compounds start out being written in two orthographic words, as in the example), a full morphologic analysis is needed to recognise them reliably.

(3.62) Who is the **AARP's top official or CEO**? (TREC 2004 Q 5.4)
A 23-city, 20-state bus trip featuring **AARP President** Tess Canja and other volunteers that will start in Philadelphia during the Republican National Convention and end two weeks later in Los Angeles during the Democratic National Convention. (NYT20000726.0100)
Tess Canja

This English example shows that the compound '*AARP President*' is considered equivalent with '*AARP's president*' (and thus also with its hypernym '*AARP's top official*').

(3.63)

In welchem	*Monat*	*begann*	*das*	***Treibstoffembargo***	*der*	*U.N.*
In which	month	began	the	fuel embargo	of	the UN

gegen	*Haiti?*
against	Haiti?

In which month did the fuel embargo of the UN against Haiti begin? (CLEF 2003 C GER 0071)

Bereits	*im*	*Juni*	*vergangenen*	*Jahres*	*hatte*	*die*	*UNO*	*ein*	***Öl-***	***und***
Already	in	June	last	year	had	the	UNO	a	oil	and

Waffenembargo	*gegen*	*Haiti*	*verhängt.*
weapons embargo	against	Haiti	imposed.

The UN had already imposed an embargo on oil and weapons against Haiti last June. (FR940508-000315)

im	*Juni*
in	June

This somewhat more complex example in German shows a case where a compound written in one word (*‘Treibstoffembargo’*) must be matched against a split compound (*‘Öl- und Waffenembargo’*). This is only possible if the split compound is correctly analysed (i. e., as *‘Ölembargo und Waffenembargo’*) and if additionally the hypernymy of *‘Treibstoff’* (*fuel*) and *‘Öl’* (*oil*) is taken into account (see below).

3.5.2.3 Lexical Semantics

We now turn to language variation and inferences at the level of lexical semantics.

We assume that the textual inference relation holds between texts where one word is replaced by another, related one. In principle, all kinds of lexical semantic relations between words can give rise to such inferences. For example, the several dozen, quite heterogeneous lexical relations described in Mel’čuk and Zholkovsky (1988) could all form part in accounting for textual inferences. However, we will concentrate in this section on ‘classical’ lexical semantic relations between words, such as synonymy or antonymy.

Note that we will not be concerned with changes in texts that are necessitated by different surface realisations of semantic roles. We assume, for example, that textual inference holds between two texts where the word *‘buy’* is replaced by the inverse *‘sell’*. We will not be concerned here with the fact that the buyer is expressed, e. g., as the subject of *‘buy’* but as a PP_{to} of *‘sell’*. We will return to this discussion of semantic roles below, cf. 5.2.2.6.

Word Senses. Throughout this section, we will make use of one fundamental simplification: In many cases, we will be talking about a 'word', when we actually only mean one sense of that word. Cruse introduces the term lexical unit for the latter usage (Cruse, 1986, 49–83). For example when stating that *'party'* is a hyponym of *'organisation'*, we will ignore the fact that this is, of course, only true for one lexical unit, namely when the sense of *'party'* that may be paraphrased as *'political party'* is concerned, but not, e. g., the *'festivity'* sense. We think that this simplification should not lead to any confusion.

However, this issue is related to a practical problem, namely that of word sense disambiguation: The different senses of a word cannot be directly distinguished at the text surface. We will return to the issue of word sense disambiguation (or rather, to the reasons why we do *not* employ it) in chapter 6.

Multi Word Expressions. We will also not go into the issues specific to multi word expressions (MWEs, Villada Moirón, 2005; Sag et al., 2001; Nunberg et al., 1994; Cruse, 1986, 37–45).

We assume that MWEs can take part in all inference and equivalence relations described in this section and will not specifically mention them. For example, we would consider a text containing a form of *'give someone the sack'* as synonymous to and thus equivalent with one that only differs from it by replacing the MWE by its synonym *'dismiss someone'*, as described below. We will describe how MWEs are recognised in our system based on information from GermaNet in 6.2.8.

Synonymy. Texts that only differ by replacing a word with a synonym should be considered equivalent for the purposes of the textual inference relation. This type of inferences played a rôle in approximately 10 % of the examples in the corpus.

In the following example, *'found'* must be replaced with its synonym *'establish'* to allow matching question and answer.

(3.64) When was the International Finance Corporation (IFC) **established**? (TREC 2004 Q 45.1)
Founded in 1956, the IFC has committed more than 23.9 billion dollars of its own funds and arranged 17 billion dollars in syndications and underwritings for 2.067 companies in 134 developing countries, according to the sources. (XIE19981108.0129)
1956

Hyponymy. We assume that textual inference holds between texts that only differ in that one word is replaced by a hypernym. Note that in this case *entailment* does not generally hold. We will discuss this issue below.

Here is a TREC example:

(3.65) Who founded the Black Panthers **organization**? (TREC 2004 Q 8.1)
The Black Panther **Party** was founded in 1966 by Seale and Huey Newton, who met as students at Oakland's Merritt Junior College and were working at a city anti-poverty center. (APW19990124.0072)
Huey Newton

In this example, the question has *'organization'*, whereas the text has its hyponym *'party'*. To arrive at the answer, *'party'* can be replaced by its hypernym, *'organization'*, because proper entailment holds.

We also assume textual inference to hold in cases where a word is replaced by a *hyponym*. In these cases, proper entailment does not hold in general.

(3.66) How fast does the Concorde **fly**? (TREC 2004 Q 20.4)
The Concorde flies at some 55,000 feet, nearly twice ordinary passenger jets, and **travels** at nearly 1,350 miles per hour, twice the speed of sound. (NYT20000725.0342)
1,350 miles per hour

In this case, the text does not properly entail the answer to the question: *'Travel'* would have to be replaced by its *hyponym 'fly'*. One would still like to consider this a possible (and even likely) answer, even though one can imagine contexts in which such an inference would be wrong, such as *'The Concorde flies at some 55,000 feet and travels at nearly 20 miles per hour max. when taxiing at the airport.'* which should definitely *not* trigger the (unguarded) answer *'20 miles per hour'*.

A second example is (3.67): An executive director is a hypernym of CEO (there are other executive directors, e. g., the CFO or the chairman of the board). Again, the answer is relevant.

(3.67) Who is the **CEO** of the AARP? (TREC 2004 Q 5.4)
"If we are over the hill, we certainly have picked up steam going downhill," said Horace Deets, the former Catholic **priest-turned-executive director** of the 34 million-member AARP. (NYT20000519.0335)
Horace Deets

In the corpus, answers that involved inferences related to the replacement of words with hyponyms and hypernyms were both relatively frequent; they were present in slightly more than 10 % of the question-text pairs.

Note that recognising whether or not proper entailment holds is far from easy: A number of linguistic constructions (negation, quantification and conditionals) *reverse* the direction of the entailment (cf., e. g., Zaenen et al., 2005).

This behaviour is related to the upward and downward monotonicity of the contexts: Contexts not containing any monotonicity-changing constructions behave upwardly monotonic; 'broadening' the sentence meaning – for example by replacing a word with a hypernym or by removing restricting modifiers from the sentence – leads to a sentence that is always true if the original sentence is true. However, negation and several other linguistic constructions inverse the direction: By introducing a negation into a context, it becomes downward monotonic (cf. also Zaenen et al., 2005).

Note that in a downward monotonic context, proper entailment holds when a word is replaced by a *hyponym*. For example, *'Jack Ruby did not kill John F. Kennedy.'* entails *'Jack Ruby did not murder John F. Kennedy.'* (*'murder'* is a hyponym of *'kill'*).

Therefore, answers involving textual inferences related to the replacement of a word with a hyponym or a hypernym both additionally need to be checked for the presence of monotonicity-changing constructions, especially for negation, and marked accordingly (3.2.3.3).

Inverse Relations. An inverse lexical relation between two words each describing a relation (such as *'in front of'* and *'behind'*, *'teach'* and *'learn from'* and *'give'* and *'receive'*) is said to hold if for all objects A and B for which the first relation holds between A and B, the second relation holds between B and A. That means that the two relations are logically equivalent, but use either A or B as a 'reference point'. Thus, *'A teaches B'* can be said to express the same underlying relation as *'B learns from A'* from the different perspectives of A and B, respectively (Cruse, 1986, 231–243).

We assume that the textual inference relation holds between two texts that differ only in that a word is replaced by its inverse, i. e., a word with which it stands in the inverse relation. To be able to use the inverse relations, however, we must not only replace the words that stand in the inverse relation with each other but also change the syntactic relations with the arguments accordingly. Thus, *'A sold X to B.'* is equivalent to *'B bought X from A.'* but *not* to *'A bought X from B.'*.

(3.68) Who was Horus's **mother**? (TREC 2004 Q 14.3)
It also hosted statues of Amon's wife, Mut, the goddess Isis, her husband, Osiris, and their **son** Horus. (XIE19990713.0042)
Isis

Most family relationships can be expressed by inverse relations, such as mother/father/parent and son/daughter/child. Here, from the stated fact that Horus is the son of Isis (and Osiris), and from the fact that Isis is a female (a goddess), it can be inferred that Isis is, indeed, Horus's mother.

We found a relatively small number of examples for this case in the corpus, most of them related to family relations.

Derivation. We assume that the textual inference relation holds between two texts if they differ by replacing a word with another one derived from it or vice versa. By 'derived' we mean the words differ in part of speech but have essentially the same meaning.

We not only include morphologically or etymologically related words (*'read'*, *'reader'*), but also words that are semantically similar, but morphologically and etymologically unrelated. For example, we employ the relation of 'pertainymy' from GermaNet (4.2.1), which holds between certain adjectives and words that they pertain to, such as *'dental'* and *'tooth'*.

A quite large number (about 20 %) of answers in the corpus involved a textual inference related to the replacement of a word with a derived word.

(3.69) Who did minstrel Al Jolson **marry**? (TREC 2004 Q 30.3)
Sam Harris, seen most recently on Broadway in "The Life," will play Al Jolson in a new musical based on the singer's life called "The Jazz Singer." [...] Harris describes the piece as a full-scale biography of the famed showman, from his height in the mid-1920s through his work on the first Hollywood talkie, his pursuit and **marriage** of Ruby Keeler and gradual decline in the 1930s and '40s.
(NYT19980604.0236)
Ruby Keeler

Here, we would like to assume equivalence between the verb *'marry'* in the question and the noun *'marriage'* in the text. Note that the accusative object of the verb and the PP$_{of}$ (*'who'* vs. *'Ruby Keeler'*) correspond, as does the subject of the verb (*'minstrel Al Jolson'*) and the possessive pronoun of the noun (*'his'*, referring to Jolson).

3.5.2.4 Anaphora

Anaphora play an important rôle in finding answers for questions in text: In the corpus study, cases where finding an answer involved the proper resolution of anaphora were frequent: Anaphoric references had to be resolved to correctly answer questions in the corpus in about 15 % of the cases.

There are a number of different forms of anaphors. We will describe the most important ones here.[18] For a more comprehensive overview see, e. g., Mitkov (2002). Anaphora resolution will also be taken up in greater detail in 6.2.9.

We first discuss anaphora that establish coreference, before turning to bridging anaphora where the relationship between anaphor and antecedent is more complex.

Pronominal Anaphors. Pronouns that may be used anaphorically include third person personal pronouns[19], relative pronouns, demonstrative pronouns, possessive pronouns etc. The following example contains a relative pronoun (*'which'*) that anaphorically refers to *'the Hale-Bopp comet'*. This anaphor must be resolved to correctly answer the question.

(3.70) How often does the Hale Bopp comet approach the earth? (TREC 2004 Q 3.2)
The simultaneous arrival of the **Hale-Bopp comet**, **which** only approaches the earth once every 3,000 years, lent the eclipse a rare status that was shared by nearly 90,000 onlookers. (XIE19970310.0220)
once every 3,000 years

We assume textual inference to hold between two texts if coreference is established for an anaphoric pronoun and it can thus be linked to its antecedent, making a direct match possible.

Definite Noun Phrases. Definite noun phrase anaphora refer back to an already mentioned entity in a similar way as pronominal anaphora, but in general they carry additional information about that entity. Consider the following example.

(3.71) What rank did Eileen Marie Collins reach? (TREC 2004 Q 54.9)
Eileen Collins likes to use a little model of a space shuttle when explaining to her littlest fan where she'll be sitting when Columbia blasts off this week. [. . .] **NASA's first female commander** laughs as she recalls how her daughter once asked: "Mommy, have you ever been to the moon?" [. . .] **The 42-year-old Air Force colonel** will

[18]We will be talking about anaphora (backward referring) here. Most observations also hold, *mutatis mutandis* for cataphora (forward referring). As cataphora are generally rarer than anaphora and use much the same devices, we will not further distinguish them here.

[19]First and second person pronouns are usually assumed to be used deictically, cf. Mitkov (2002, 20–1).

be responsible for four other astronauts, three of them older men. (APW19990717.0096)
colonel

Here, both definite NPs *'NASA's first female commander'* and *'the 42-year-old Air Force colonel'* anaphorically refer to *'Eileen Collins'*. But they not only provide a connection like pronominal anaphora; they also provide further information (namely, that Eileen Collins is with NASA, that she's NASA's first female commander, that she is a commander, that she is 42 years old etc.).

Again, we assume that textual inference holds between two texts when coreference resolution allows to link a definite description to its antecedent so that a direct match becomes possible.

Proper Name Anaphora. Proper names like *'Franz Kafka'* are generally assumed to refer (more or less) unambiguously to an individual or entity. However, it is useful to assume that they also form anaphoric chains in texts, especially as it is possible to use short forms to take up such a name.

(3.72) Where was Franz Kafka born? (TREC 2004 Q 22.2)
City councilors named a square after Prague's literary son, **Franz Kafka**, on Tuesday despite earlier protest from the district's mayor who said the move would have terrified the writer. [...] **Kafka** was born in Prague in 1883 and wrote in his native German language. (APW20000425.0204)
Prague

The example shows that a once introduced individual (here, Franz Kafka) can be anaphorically referred to by a shortened version of her or his proper name (here, the last name, Kafka).

Thus, the indirect answerhood relation must provide means of matching an answer to a question where the answer uses a short form of a named entity and this short form can be resolved to the long form used in the question through coreference in the answer text.

Bridging Anaphora. So far, we have only discussed anaphora that establish coreference between anaphor and antecedent. Bridging anaphora introduce a more complex relationship between anaphor and antecedent (often called anchor), such as set-membership or part-of (Gardent et al., 2003; Vieira and Poesio, 2000).[20] Bridging anaphors are definites that are not coreferent with an

[20]By bridging anaphora we do *not* refer to anaphora that establish coreference but where anaphor and antecedent have a different lexical head, as some authors, including Vieira and Poesio, do.

antecedent, but that stand in some relation to an entity mentioned earlier. Consider the following example:

(3.73) What type of plant does the boll weevil damage? (TREC 2004 Q 63.1)
Since the boll weevil crossed from Mexico into Brownsville, Texas, in 1892, American **cotton** growers and entomologists have spent a century battling the pest, and at long last victory seems at hand. As surely as the weevil spread across the southeastern quarter of the country, destroying billions of dollars in **crops** and thousands of jobs, a scientific counterattack that began in North Carolina in 1978 is slowly beating the varmint back. (NYT19980715.0128)
cotton

This example shows a quite complex case of bridging: Here, a relation between crop and the anchor cotton needs to be established. The relation between the two cannot be straightforwardly accounted for in terms of ‘classical’ semantic relations; it might be paraphrased something like this: Cotton is a agriculturally used plant and as such plants yields crops, thus it can be inferred that cotton crops are indeed meant.

Thus, resolving bridging anaphora may provide an important source of information for indirect answerhood. The anaphora resolution module that we use currently does not support resolving bridging anaphora (cf. 6.2.9); thus, we cannot make use of this sort of information.

3.5.2.5 Direct Answerhood Phenomena

We have split the indirect answerhood relation into that of textual inference described so far and that of direct answerhood. We will now characterise the relation of direct answerhood.

The relation of direct answerhood holds between interrogatives and declaratives where all *wh*-phrases (if present in the question) are semantically compatible with the corresponding phrases in the answer (3.5.1.1).

Consider the following example.

(3.74) *Who* **founded the Muslim Brotherhood**? (TREC 2004 Q 61.1)
The Muslim Brotherhood is outlawed and many of its leaders are behind bars. [. . .] **The brotherhood**, **founded** *by Hassan el-Banna* in Egypt in 1928, grew into a vast movement with tens of thousands of supporters in Egypt and branches throughout the Arab world.

Thus, we only use the term bridging anaphora for a subset of the definition used by Vieira and Poesio (2000, 557–559).

(APW19981229.0694)
Hassan el-Banna

We assume that by textual inference we can derive the (possible) direct answer *‘Hassan el-Banna founded the Muslim Brotherhood.’* from the text above. Among the steps that need to be carried out are: Coreference resolution (*‘the Muslim Brotherhood’*, *‘the brotherhood’*), extraction of participle clause and passive-to-active transformation. Note that, for the moment, we simple assume that the modifiers *in Egypt* and *in 1928* can be ignored; we will come back to this point presently.

Direct answerhood between *‘Who founded the Muslim Brotherhood?’* and *‘Hassan el-Banna founded the Muslim Brotherhood.’* is assumed to hold: The question has the *wh*-phrase *‘who’* as subject and the answer has *‘Hassan el-Banna’* and both are semantically compatible (*‘who’* is assumed to ask for a person).

It must thus be possible to map the whole question onto the answer. Constituent answers can then be derived in the following way: Phrases that correspond to *wh*-phrases in the answer can be used as constituent answers (cf. Reich’s explanation of constituent answers, 3.2.2.2). This is exactly how we produce short answers (cf. 6.4.3.2).

We will now further characterise the direct answerhood relation for different syntactic question types.

Adverbial Questions. Adverbial questions containing question adverbs such as *‘when’* or *‘where’* have already been mentioned above (cf. 3.2.1.2). Answers to these questions are syntactically relatively unconstrained: Any adverbial of the correct semantic type (such as location, time or reason) can, in principle, answer such a question, for example NPs, PPs, adverbs or whole clauses. In the following example, *‘in Jacksonville’* must be identified as locational PP in order to licence it as an answer to the *wh*-phrase *‘where’*.

(3.75) **Where** was Fred Durst born? (TREC 2004 Q 2.4)
Limp Bizkit lead singer Fred Durst did a lot before he hit the big time. [...] Born **in Jacksonville, Fla.,** Durst grew up in Gastonia, N.C., where his love of hip-hop music and break dancing made him an outcast.
(APW19990704.0024)
in Jacksonville

Of the TREC 2004 questions, about 35 % of the factoid questions were adverbial questions.

‘*What/which* COMMON NOUN’ Questions. For questions that contain *wh*-phrases of the form *‘what/which* COMMON NOUN’, we assume that *wh*-phrase and corresponding answer phrase match exactly if the head of the answer is a hyponym of the common noun in the *wh*-phrase. It need not be a direct hyponym in general but may be any descendant of the common noun (when regarding hyponym relations as forming a hierarchy tree, cf. Cruse, 1986, 112–135). Consider the following example:

(3.76) **What kind of animal** is an agouti? (TREC 2004 Q 7.1)
Birds of every size and color, skinks (a type of lizard), agoutis (rabbit-sized **nocturnal rodents**), deer and, of course, the innumerable monkeys.
(NYT19990311.0093)
nocturnal rodents

In the example, direct answerhood holds, as *‘(kind of) animal’* is a hypernym of *‘rodent’* (though not a close one if one assumes a hyponymy hierarchy that, for animals, is structured similar to the biologic taxonomy).

In the TREC 2004 corpus, we found about 20 % of the questions to be of the *‘what* COMMON NOUN’ type.

‘*How* ADJECTIVE’ Questions. Another special kind of questions has been mentioned above, namely that of *‘how* ADJECTIVE’ questions (3.2.1.2). These questions ask for a measure or degree of an underlying dimension. For example, *‘how many’* questions ask for a number in general, *‘how long’* for a length (either of space or time) etc. Thus, in general, the ‘underlying’ concept of the adjective must be derived to identify suitable answer phrases. For measurements, e. g., it is necessary to allow phrases whose head is an appropriate unit for that measurement (such as *‘metre’*, *‘m’*, *‘centimetre’*, *‘kilometre’*, *‘light year’*).

The following simple example shows a number question using *‘how many’*.

(3.77) **How many kibbutzim** are there now? (TREC 2004 Q 19.5)
There are now **275 kibbutz communities** in Israel, scattering throughout the country from the Golan Heights in the north to the Red Sea in the south. (XIE19980421.0039)
275

Questions starting with *‘how* ADJECTIVE’ made up about 15 % of the questions in TREC 2004.

Definition Questions. There is a type of questions that has been called definitional questions (Blair-Goldensohn et al., 2003). They generally take the form *'Who/what is X?'* or similar (cf. (3.76)).

The simplest possibility of matching a definition questions onto an answer is when the answer context is also definitional, that is when it uses a predicative construction. This is shown by the following example.

(3.78) **What kind of a particle** is a quark? (TREC 2004 Q 38.1)
Its crucial component is the "strange quark" – one of six types of quarks, which are **the tiniest building blocks of matter**. (XIE19980113.0177)
tiniest building blocks of matter

We will describe two additional, relatively simple, sources of definitional knowledge (cf. Blair-Goldensohn et al., 2003, for a number of additional patterns): We assume that both definite NP anaphora and appositions can be used as sources of definitional knowledge. *'John F. Kennedy, the first catholic to become US-president. . . '* is assumed to conventionally imply *'John F. Kennedy was the first catholic to become US-president.'* (Karttunen and Zaenen, 2005; Zaenen et al., 2005; Karttunen and Peters, 1979).

The first possibility is to answer the definitional question with a definite description that (uniquely) identifies the subject of the question. This means that we interpret the *'be'* of the question as a *be of identity* (cf., e. g., Kamp and Reyle, 1993, 257–279).

We can use coreference relations as sources of such descriptions: If coreference between the subject of the question and a definite description is established, this definite description can be used to answer the question – and vice versa.

In addition to definite descriptions that were mentioned above (3.5.2.4), such coreference relations are also generally assumed to be introduced by appositions with definite NPs (Mitkov, 2002, 5–6). Consider the following example:

(3.79) **Who** is the CEO of the publishing company Conde Nast? (TREC 2004 Q 53.1)
After all, as Fortune put it recently, the president and CEO of Conde Nast, **Steven Florio**, "turned a mildly profitable property into one of the greatest money pits in American magazine history."
(NYT19980710.0094)
Steven Florio

Here, the entity that answers to the definite description *'the CEO of the publishing company Conde Nast'* is searched. As *'Steven Florio'* is coreferent with the matching phrase *‘the president and CEO of Conde Nast’*, he is a good answer to the question.

The second possibility is to make use of appositions with indefinite NPs. Different from those with definite NPs, they are rather assumed to introduce predication – thus directly parallel to the second reading of *‘be’*, namely as *be of predication* (cf., e. g., Kamp and Reyle, 1993, 257–279).

Consider the following example, that asks for a definition for *‘boll weevil’*. Note that the *wh*-phrase forms the predicative here (inverted word order).

(3.80) **What kind of insect** is a boll weevil? (TREC 2004 Q 63.1)
Boll weevils, **beetles** that destroy cotton, are proliferating because cash-strapped farmers cannot pay for eradication.
(APW19990311.0091)
beetles

Information Appositions as sources of information played a rôle in more than 15 % of the question-answer pairs in the corpus.

Uncertain Answers. We have mentioned above that uncertain answers are often useful, as they provide relevant information. The following example shows an answer that is contained in reported speech (and therefore in an opaque context).

(3.81) What do practitioners of Wicca worship? (TREC 2004 Q 32.1)
The inch-thick chaplain handbook includes a five-page primer on Wicca, **described as** “a reconstruction of the Nature worship of tribal Europe.”
(APW19990617.0023)
Nature

3.5.2.6 World Knowledge

So far, we have described textual inferences that can be systematically related to different kinds of linguistic variation. There are, however, more complex kinds of inferences that are, nevertheless, also useful for QA.

In addition to primarily linguistic knowledge (such as the information that two words are synonymous), these complex kinds of inference involve ‘world knowledge’. Note that these two cannot systematically be distinguished (cf. Zaenen et al., 2005; Kay, 1989). We use the term world knowledge here to subsume knowledge that is not obviously linguistic.

We give a few examples of questions and texts, from which the correct answer could only be extracted using such complex inferences. We mainly aim to show the limits of indirect answerhood as we have specified it here: The examples described in the following are beyond the scope of indirect answerhood.

(3.82) *What conflict* did the USS Constitution **distinguish** herself in? (TREC 2004 Q 42.2)
Just a few blocks and you're back on the water, across the Charlestown Bridge from the USS Constitution, which first launched in 1797 and **gained its nickname**, "Old Ironsides," *in the War of 1812*.
(NYT19990830.0147)
War of 1812

The first example may be considered as exhibiting 'only' extended linguistic knowledge rather than world knowledge. Here, the following inferences would be needed: A person or entity may gain a nickname for famous (or infamous) deeds (among other things, many nicknames are assigned on the basis of personal features or traits, cf. Roman names such as Brutus=brute, Crassus=paunchy). So it is likely that someone (or something) who gains a nickname in an event has distinguished himself/herself/itself in this event. Note that the answer is thus not properly entailed, though relevant (cf. 3.5.1.3).

For cases like this one, automatically acquired paraphrases might provide an alternative solution. In recent work on paraphrase acquisition, different researchers have managed to automatically extract phrasal paraphrases from the Internet using seed patterns. For example, the system described in Lin and Pantel (2001a,b) found for the phrase *X is the author of Y* possible paraphrases like *Y is co-authored by X* or *X tells story in Y*. Using such a system, it might well be possible to systematically find that *X distinguished him/her/itself in Y* can be loosely paraphrased as *X gained a nickname in Y*.

However, while this approach seems very promising for some cases, it is hard to see how it would succeed for the following ones, see also 8.3.1.2.

(3.83) *What year* was **the movie "Wall Street"** released? (TREC 2004 Q 23.3)
Douglas won his first Academy award in 1975 for producing "One flew Over the Cuckoo's Nest," which won five awards including best picture. He received a best actor Oscar *in 1987* for his role as Gordon Gekko in **"Wall Street"**. (XIE19980731.0279)
1987

Here, knowledge is required about the world of movies, including the Academy of Motion Picture Arts and Sciences Academy Awards® (nicknamed Oscar®) fairly well to allow the above answer. The knowledge and reasoning required goes something like this: Oscars are awards that are, among other things, given to actors who play a rôle in a film (‘best actor’). They are awarded in January for films that were released in the previous year (see the Academy’s web site, `http://www.oscars.org`). Thus, if Michael Douglas received an Oscar for a rôle in ‘Wall Street’ (we need also to assume that this is actually referring to the movie, and not, e. g., a theatre play) in year X, then the film must have been released the year *before* that. Thus, the proper answer to be inferred from the article should have been 1986, not 1987. As it happens, the answer is still correct: The article is just not accurate, as Douglas actually *received* his Oscar in January 1988, cf. the Internet Movie Database, `http://www.imdb.org/`.

Note that this very predictable temporal relation between an award and the feat for which it is awarded is peculiar to the Oscar and similar awards. For other prizes, such as, say, the Nobel Prize, this relationship does not hold at all. Consider the following example:

(3.84) *Who* **discovered prions**? (TREC 2004 Q 10.2)
Recent winners of the Nobel Prize in medicine or physiology, and their research: [. . .] 1997: *Stanley B. Prusiner*, United States, **discovery of prions**, an infectious agent at the heart of several forms of brain-wasting disease. (APW19981012.0310)
Stanley B. Prusiner

Here, one could *not* infer the year of the discovery from the fact that the prize was awarded to Prusiner in 1997 (except, of course that it must have taken place before 1997), as Nobel prizes may be (and tend to be) awarded years after the research for which they are awarded has been conducted.

A lot of information is implicitly encoded in the table-like text: Stanley B. Prusiner *is* a recent winner of a Nobel Prize in medicine or physiology. He has won the prize in 1997. He works (or has been born, or both) in the USA. His research has lead to *his* discovery of prions. This discovery is the (main) reason for his being awarded the mentioned Nobel Prize. Though the main ‘chunks’ of information are there, the relations between them are implicit only and need to be inferred from different sources: The ‘table heading’ (the first line) gives directions regarding the interpretation of the ‘rows’ of the table. It allows, e. g., the inference that Prusiner is a Nobel Prize winner. Others (e. g., using an appositive to state the country of birth or the nationality or the residence of a person) are more or less conventionalised. In general, this sort of knowledge and inference is especially hard to capture, as there are few ‘anchors’ that this knowledge seems to be tied to.

The next example shows some more interaction of (peripheral) linguistic knowledge and world knowledge.

(3.85) *Who* was *President of the United States* at the time of **the Teapot Dome scandal**? (TREC 2004 Q 41.2)
And even the reputation of the nation's most scandal-tarred and least respected chief executive is on the mend. Yes, to judge from some new historical accounts, *Warren G. Harding* is on the comeback trail. The Ohio Republican, who died in office 75 years ago this month, has long been cast as a venal, sybaritic, glad-handing boob who left behind an administration lousy with corruption, epitomized by **the Teapot Dome scandal**, in which oil barons bribed his interior secretary. (NYT19980824.0101)
Warren G. Harding

Here, one must first infer that Warren G. Harding was a president of the US. This is suggested by the cataphor *'the nation's most scandal-tarred and least respected chief executive'*: That *'nation'* refers to the US must here be inferred, namely from the fact that in an article by a US paper it can be seen as a 'larger situation definite description' (cf. Poesio et al., 2004), as in this article there is no possible overt anaphoric antecedent. From *'[Harding] has long been cast'* which signals the report of a judgment about him the objective facts must be extracted, namely that before he *'left [it] behind'*, he must have lead an administration (because it is the duty of the US president to lead the federal administration), that if this administration is *'lousy with corruption [that is] epitomized by [a] scandal'* then it must be held responsible for the corruption and the scandal (whether by commission or by omission) and thus this very scandal must have occurred during some contextually relevant time, especially before this administration was *'left behind'*.

The important thing to take away from this section with its examples is that a QA system would, ideally, need to draw inferences at a high cognitive level, indeed. While we consider it possible to code a comparatively high amount of the core linguistic knowledge involved, it is beyond the scope of current computer systems to *systematically* encode knowledge of the sort reported in this section: Each of these useful pieces of knowledge can be represented. However, arriving at a comprehensive representation seems a challenge for the future. We will return to the issue of knowledge representation in the context of computational ontologies below in 4.4.

Recently a number of large scale experiments in automatic paraphrase acquisition from very large corpora, especially the Internet, have kindled hopes that it may be possible to automatically acquire at least frequent patterns that may be used to extend linguistic knowledge for NLP systems.

3.5.3 Discussion

In this section we have characterised the relation of indirect answerhood that forms the core of linguistically informed QA.

We have specified the relation of indirect answerhood as a compound of the relations of textual inference and of direct answerhood. We have listed types of textual inferences and direct answerhood, related with different linguistic phenomena. We have used a corpus study of questions answers and underlying texts put together from past TREC and CLEF results to identify and illustrate the different types. By grouping phenomena that need to be handled to match questions and answers in these corpora by linguistic phenomena, we arrive at a characterisation of the relation of indirect answerhood and the kind of linguistic knowledge that needs to go into it.

A more formal definition of the indirect answerhood relation between text and question representations based on dependency structures extended with lexical semantic information will be presented in chapter 5.

The relation of indirect answerhood, as we have just characterised it, is based on syntax; it also includes local inferences based on lexical semantic information, but it does not allow full semantic inferences. We have described above that this full inferencing capability is an important part of the more advanced semantic approaches to describing answerhood.

Let us therefore take stock and recapitulate what sort of phenomena related with answerhood we can tackle with indirect answerhood as specified in this section.

One important point is that direct answerhood as it stands assumes that a question representation can be embedded into a corresponding answer representation. Thus, it does not support information fusion: Information that is not contained in one sentence of the text cannot be matched against a question sentence. This defines a sentence as an the domain in which textual inferences can be drawn (modulo anaphora).

Sentences differing only by syntactic variations, such as word order, diatheses, variations of PP and other constructions, can be mapped onto each other. As only variation that do not alter the meaning are used, indirect answerhood is not prone to precision problems like bag-of-words-based approaches. For example, it will not make the mistake, when given the question ‘*What do frogs eat?*’ to return the sentence ‘*Alligators eat. . . frogs.*’ as a possible answer (the example is taken from Katz and Lin, 2003).

Indirect answerhood also allows to match questions and answers that differ by the replacement of words related by lexical semantic relations such as synonymy.

Due to the locality of the approach, it will not be able to handle more complex inferences, especially one involving quantification. For example, it can only find an answer to questions like *'Who won two Nobel prizes?'* if a sentence in the document collection happens to explicitly contain that information, such as *'Marie Curie was the only woman ever to win two Nobel prizes.'*.

Answering open questions (i. e, giving essay-style answers, cf. 3.2.2.5) is beyond the scope of indirect answerhood. Thus, especially *'why'* or *'how-to'* questions can only be answered if the answer is explicitly given in the document collection.

Basing QA on full meaning representations and automated reasoning is currently not a viable solution. We have therefore suggested to use a more constrained approximation of indirect answerhood to keep it manageable. The discussion above has shown that this imposes systematic limitations on what answers can be found. Still, a large number of interesting cases can be handled.

3.6 Conclusions

In this chapter, we have introduced linguistically informed question answering.

First, we noted that there is little work that connects theoretical findings in linguistics and logic on questions and answers, especially the relation of answerhood that holds exactly between questions and their respective answers, with practical QA system implementations.

We have reviewed work on questions and answers in linguistics to extract an overview of phenomena related to answering natural language questions. We have found that a full account of questions and answers needs to handle semantic phenomena (especially when indirect answers, i. e., answers that require additional inferences, are concerned) and probably even needs to take pragmatics into account (in the form of a representation of the questioner's goals).

We have listed, however, a number of reasons why full meaning representations of texts and automatic reasoning as a basis for a QA system currently seems not workable. The most salient of these were the lack of any systems that are able to produce full semantic representations of general texts, the difficulty of keeping a semantic representation of large amounts of texts free from inconsistencies (which is necessary because only consistent knowledge bases can be used for inferencing) and the lack of reasoning systems that are efficient enough to handle the number of facts and rules that would result from representing large document collection and knowledge bases.

We have then turned to characterising a relation of indirect answerhood (compounded from the relations of textual inference and direct answerhood) that allows local syntactic and lexical semantic inferences and thus provides a basis for linguistically informed QA. We have especially listed different types of textual inferences and related them to linguistic phenomena, based on a corpus study from past TREC and CLEF competitions. This description was centred on syntactic and lexical-semantic phenomena.

We have assumed that the indirect answerhood relation holds between a text representation and a question exactly if the text contains an answer to the question that can be inferred by a sequence of inference steps. We have derived a number of generic inference rules from linguistic variation and ‘classical’ lexical semantic relations.

In the next chapters we will describe different linguistic resources that can serve as sources for local inference rules and then turn to the question of how the relation of indirect answerhood can be actually employed to form the core of a linguistically informed QA system.

Chapter 4

Resources

In chapter 3, we have introduced indirect answerhood as the central concept of linguistically informed QA. We have specified indirect answerhood as a relation between texts and questions and we have given examples. In this chapter we will present several linguistic resources as possible sources of linguistic information for indirect answerhood. In chapter 5, we will then define indirect answerhood as a relation over syntactic dependency structures extended with lexical semantic information and show how information from linguistic resources can be integrated in practice.

In 4.1, we will describe Dependency Grammar (Tesnière, 1980) and Partially Resolved Dependency Structures, the syntactic dependency representation that we employ (PREDS, Braun, 2003; Fliedner, 2004a). We have specified the PREDS so that they are especially suited for being used in our definition of indirect answerhood.

In 4.2 and 4.3, we will give an overview of two lexical databases that we have used as knowledge sources, namely WordNet (Fellbaum, 1998c), or rather its German 'offspring', GermaNet (Kunze and Lemnitzer, 2002b), and FrameNet (Fillmore et al., 2003a).

These resources assign words in natural language to abstract concepts (synonym sets in WordNet, frames in FrameNet) and list different lexical semantic relations between those, such as synonymy in WordNet or the subframe relation in FrameNet. We make use of information on both concepts and relations in our definition of indirect answerhood. FrameNet additionally provides information on semantic roles, which is also employed.

Other lexical resources could also be used as sources of information for linguistically informed QA. Especially PropBank (Palmer et al., 2005) and the Prague Dependency Treebank (PDT, Böhmová et al., 2003) provide interesting information: The PropBank adds a semantic annotation level (namely normalised predicate-argument structures) to the Penn Treebank. In the PDT, a corpus of Czech has manually been annotated with morphological information, syntactic dependency structures and thematic roles in three separate but related layers. However, as the annotation is for English and Czech, respectively, we cannot directly take over any information for our German system.

As an example for different knowledge sources, we will then turn to general, non-linguistic ontologies (4.4). We will describe two resources, namely Cyc (Matuszek et al., 2006) and SUMO (Niles and Pease, 2001). Both ontologies define general concepts and also concepts in special domains. Both also contain a large number of axioms, expressed in formal languages based on predicate logic. These axioms represent the greater part of the information encoded in the two ontologies. Therefore, this sort of knowledge source does not fit our approach of linguistically informed QA: We cannot directly integrate knowledge represented in a predicate logic formalism, so that only a small part of the information in such ontologies could actually be used. Additionally, there are currently no mappings available from German words to the concepts, neither for Cyc nor for SUMO.

4.1 Dependency Grammar and Dependency Structures

We use Partially Resolved Dependency Structures (PReDS) to represent the syntactic structures of texts. We have chosen dependency structures as the basis of the text representations explicitly because they provide a relatively high level of abstraction over surface differences (see below).

PReDS are based on work on Dependency Grammar (Tesnière, 1980; Mel'čuk, 1988), but differ from the dependency structures defined by these authors in several respects. In this section, we will describe Dependency Grammar in general (4.1.1) and PReDS in particular (4.1.2). We will also motivate the choice of dependency structures over phrase structures for representation.

4.1.1 Dependency Grammar

Dependency Grammar is a valency-oriented grammar model that describes syntactic structures of sentences in terms of hierarchical dependency relations between their words. A dependency relation is a directed relation between a governing element (head) and a dependent element (dependent). The main verb of a sentence, for example, is considered as a head, on which arguments and

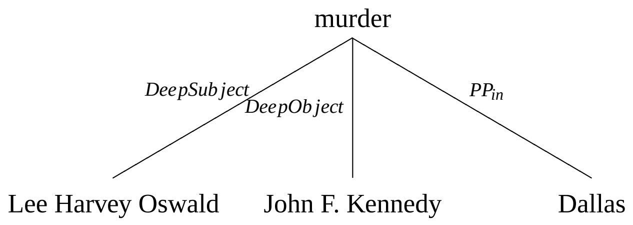


Figure 4.1: PREDS for 'Lee Harvey Oswald has murdered John F. Kennedy in Dallas.' (Simplified)

modifiers depend. A sentence is represented as a tree structure where the words correspond to nodes that are connected by edges representing the dependency relations. Figure 4.1 shows an example for a dependency structure (more precisely, a PREDS).

In Dependency Grammar, word order is typically not represented; dependency is used as central structuring criterion. Dependency structures directly express valencies: All arguments (and also modifiers) of a word are represented as its direct dependents in the structure. Rules in Dependency Grammar are typically specified by listing the arguments that a word requires. We describe additional features of dependency structures in 4.1.2.

Dependency Grammar is generally attributed to Lucien Tesnière (Tesnière, 1980). He sets out the general principles and gives exemplary dependency representations for a number of phenomena from different languages. More recent work on dependency grammars in linguistics includes Mel'čuk (1988).

In computational linguistics, dependency grammar has received a growing interest over the last years. Both the Negra and the Tiger corpus of German newspaper texts use a syntactic annotation format that combines features from dependency grammar and phrase structure grammars (Brants et al., 1999, 2002).

The analytical (syntactic) level of the Prague Dependency Treebank also uses dependency structures for annotation (Böhmová et al., 2003).

Ralph Debusmann has introduced Extensible Dependency Grammar (XDG) in his recent Ph. D. thesis (Debusmann, 2006). In contrast with most other Dependency Grammar formalisms, XDG allows grammars to be modularly extended, e. g., for representing word order constraints. Extensible Dependency Grammar is based on earlier work on Topological Dependency Grammars (Duchier and Debusmann, 2001). In Debusmann et al. (2004), the use of XDG as a syntax-semantics interface is described.

4.1.2 Partially Resolved Dependency Structures

We will now describe PREDS as a special form of dependency structures. PREDS are used to represent the syntactic structure of German sentences. We will point out important differences from dependency structures defined in the literature on Dependency Grammar, especially from the ones defined in Tesnière (1980).

Note that PREDS allow the representation of underspecified syntactic structures (hence *partially resolved*); this feature is especially used for representing attachment of PPs in an underspecified way. We will discuss this characteristic below (6.2.7.1).

PREDS are tree structures specified as follows: Each content word in the text is represented by a node in the dependency tree. The nodes are connected through edges that correspond to a directed relation of dependency: If two words are syntactically related, it is assumed that one is dependent upon the other. This dependent becomes a child node of the governing head node. Both arguments and modifiers of a word are assumed to be its dependents. For example, the main verb of a sentence will be represented as the root node of the corresponding dependency structure, with all arguments and modifiers as direct children. In fig. 4.1, the main verb *'murder'* is accordingly represented by the root node and subject, object and PP modifier as its children. This is in accordance with generic dependency structures.

Nodes in PREDS are labelled with the lemma of the word that they correspond to. They may also contain additional information. We especially encode morpho-syntactic information such as number, tense and information about articles into the nodes (see also below). This information is not used by the indirect answerhood relation, but only during parsing and for answer generation. We therefore do not show it in examples such as 4.1.

The edges of the tree structure are labelled by the specific dependency relation. We use grammatical functions in PREDS. Note that we additionally normalise over active and passive: We identify the 'underlying' deep subject and object and label them as Deep Subject (DSubj) and Deep Object (DObj), respectively. This is shown in fig. 4.3. The PREDS do not encode information on word order.

Most function words are not represented in PREDS; rather, the information that they carry is integrated into the node of a content word or into an edge label. In this, we go further than other dependency representations where, for example, articles or modal words are represented by separate nodes (Tesnière, 1980). We will describe these differences in the following.

Prepositional phrases are treated as follows in PREDS: The preposition is not represented by a node, but integrated into the edge label. This is shown by the PP *'in Dallas'* in fig. 5.1: The preposition *'in'* is not represented by a node, but integrated into the edge label PP_{in}.

Most other function words are represented as additional information in a suitable content word node. In fig. 5.1, *'has murdered'* is collapsed into a single node, for example. Rather than keeping the auxiliary verb, the information conveyed through it (past tense) is added as a feature to the main verb node (such features are not shown in fig. 5.1). We handle modals and other function words, such as articles, analogously.

Conjunctions, however, are represented as a normal node governing the coordinated elements. This differs from Tesnière's treatment, who introduces a special edge type to express junction (*jonction*, Tesnière, 1980, chs. 38,39).[1]

Coreferences are expressed by additional coreference edges between words: To represent the coreference relation between an anaphoric expression (such as a anaphoric pronoun) and its antecedent, a corresponding antecedence edge is added to the PREDS structure. We will describe coreference in more detail below.

The simplifications are necessary in order to allow simple matching of structures: Typical constituent structures will differ much more in structure for relatively similar sentences. Compare for example the two constituent structures in fig. 4.2 for the active sentence '*Lee Harvey Oswald murdered John F. Kennedy.*' and the passive sentence '*John F. Kennedy was murdered by Lee Harvey Oswald.*' with the corresponding PREDS structures in fig. 4.3: The PREDS are equivalent. The constituent structures, in contrast, differ substantially. Using dependency structures (and especially PREDS), we can define textual inferences far more simply, namely as relabellings of local tree structures, while relating constituent structures to each other would require far more complex mapping operations.

Note that this applies for all simplifications described above: By representing articles, prepositions, auxiliary verbs and modal verbs not as single nodes but integrating them into nodes and edge labels, we arrive at simpler structures that can be matched more easily. The definition of indirect answerhood that we will give in chapter 5 employs PREDS as text representations and relies on these simplifications.

[1]Note that for storing coordinated structures in the database, direct links from the governor to each conjunct are added, so that each of the conjoined elements can be found in addition to the conjoined structure when matching questions and answers, cf. 5.2.2.8.

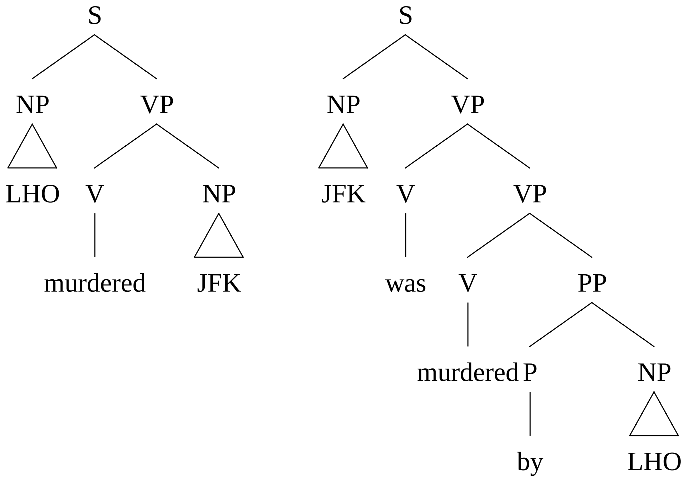


Figure 4.2: Basic Constituent Structures for Active and Passive Sentence: *'Lee Harvey Oswald murdered John F. Kennedy.'* vs. *'John F. Kennedy was murdered by Lee Harvey Oswald.'*

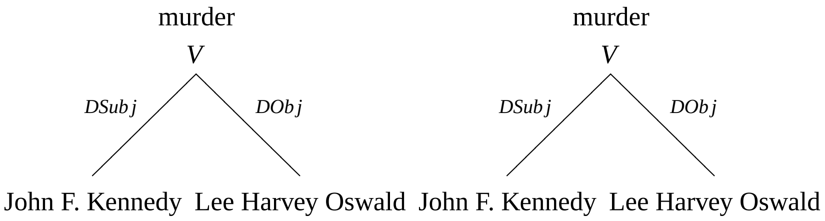


Figure 4.3: PREDS for Active and Passive Sentence: *'Lee Harvey Oswald murdered John F. Kennedy.'* vs. *'John F. Kennedy was murdered by Lee Harvey Oswald.'*

4.2 WordNet and GermaNet

As one important resource of lexical semantic information, we use the lexical semantic resource GermaNet, a German version of WordNet. We will introduce both WordNet and GermaNet in this section. As GermaNet largely builds upon the theoretical work (and also on the software implementation) done within the WordNet project, we will start with describing WordNet and then present GermaNet, highlighting on important differences from WordNet.

4.2.1 WordNet

We will first describe 'standard' WordNet (Fellbaum, 1998c), which is generally used in computational linguistics applications, before shortly turning to the so-called extended WordNet (XWN, Moldovan and Novischi, 2002) that has been derived from WordNet through further semi-automatic analysis of the WordNet resource itself.

4.2.1.1 Princeton WordNet

Since its first creation by George A. Miller, Christiane Fellbaum and their colleagues at Princeton University in 1985, WordNet has become the probably most-used lexical semantic resource in computational linguistics and natural language processing. Originally designed as a study in psycholinguistics, it was soon discovered as an important source of information on lexical semantics. For an overview of the genesis and history of WordNet, see Miller (1998a).

In natural language processing, WordNet is used as a structured lexical semantic database that describes words and semantic relations between them. Here are a number of the most important design principles:

Synonym Sets. The basic relation in WordNet is that of synonymy. The definition of synonymy that is used here is quite general: It does not require that synonyms can replace each other in *every* linguistic context (as, e. g., Cruse, 1986, 88), but rather that there is at least one general context in which they can replace each other. Words (or rather word senses, see below) are grouped into sets of synonyms (in WordNet jargon called synsets, Miller, 1998b, 23–4).[2]

Words and Word Senses. Words are the basic unit upon which WordNet is built. For polysemous words, different word senses are assumed and distinguished. This is done by assigning each word sense to a different

[2]Note that the notion of discrete word senses defined through synonymy has been challenged almost from the outset. See, e. g., Kilgarriff (1997); Buitelaar (1998).

synset; for better distinction, the different senses are additionally numbered. Relations between words distinguish between word senses; they are thus defined between word/meaning pairs (Fellbaum, 1998a; Cruse, 1986).

Lexical Hierarchy. The synsets are used as the basis for constructing a lexical hierarchy, using the hypernymy relation between the synsets. Note that for verbs, a relation called troponymy is used instead. A verb V1 is a troponym of a verb V2 if they fit the following formula: *To V1 is to V2 in some particular manner.* A troponym thus specialises its superordinate with regard to manner (Fellbaum, 1998b). When we talk about hypernymy in the remainder of the section, that is supposed to mean troponymy when verbs are concerned.

Multiple inheritance is allowed, that is, a synset can have more than one hypernym (Miller, 1998b, 34–37). WordNet contains a considerable number of proper names (such as *George Washington* or *Edinburgh*). Recently, efforts have been undertaken to distinguish between classes and instances within the noun hierarchy, a distinction that has so far been missing (Miller and Hristea, 2006). However, reliably annotating this distinction has proven to be more difficult than expected.

Parts of Speech. The part of speech of the words is used in WordNet as a main distinction; four separate hierarchies are formed for nouns, verbs, adjectives and adverbs, respectively. The four hierarchies are partly interlinked through different derivation relations (cf. tab. 4.1, 3.5.2.3, Cruse, 1986, 130–3).

The hierarchies for adjectives and adverbs differ slightly from the others in that they use *antonymy* as the basic relation for defining synsets (thus *'light'* is grouped into two different synsets; this distinction is based on the difference in meaning that can be observed by the different associated antonyms *'heavy'* and *'difficult'*). Instead of (close) synonymy, the (looser) notion of similarity is used in addition to the antonymy relation. (Miller, 1998c)

Other Semantic Relations. In addition to the basic relations of synonymy and hypernymy (and its inverse, hyponymy), several additional semantic relations are defined in WordNet. Several of the corresponding 'classic' lexical semantic relations have been mentioned earlier. Table 4.1 shows the most important relations in WordNet with examples, see also 3.5.2.3. This table is based upon Miller (1998b,c) and Fellbaum (1998b). Note

Relation	Applicable for parts of speech	Example
Synonymy	noun, verb, adjective (similarity), adverb (similarity)	violin, fiddle
Hyper/hyponymy	noun, verb (troponymy)	violin, string
Antonymy	noun, verb, adjective, adverb	light, heavy; light, dark
Mero/holonymy	noun	air bag, car (component); professor, faculty (member); tree, sapwood/heart wood (stuff)
Entailment	verb	snore, sleep
Cause	verb	frighten, fear
Derivation	all	rent, rental
Derivation from noun (pertainymy)	adjective	dental, tooth
Participle_of	adjective	breaking, break
Association	all	rouble, Russia
Attribute	noun, adjective	small, size

Table 4.1: Semantic Relations in WordNet

Part of Speech	Synsets	Lemmata	Senses	Avg. # of senses	
				Synset	Lemma
noun	81 426	117 097	145 104	1.78	1.24
verb	13 650	11 488	24 890	1.82	2.17
adjective	22 141	18 877	31 302	1.41	1.66
adverb	3 644	4 601	5 720	1.57	1.24
sum	117 597	155 327	207 017	1.76	1.33

Table 4.2: Some Database Statistics for WordNet 2.1

that, while hypernymy is defined as a relation over synsets, all other semantic relations are defined as relations between word senses, as they may not hold for all element in a synset. Meronymy/Holonymy relations (part-of and its inverse relation) are differentiated into the three sub-relations that hold between to word senses A and B in the following cases: a) A is a component part of B, b) A is a member of B and c) A is the stuff that B is made from (Miller, 1998b, 37–9).[3]

Valency Information. WordNet provides only a very limited amount of valency information for verbs, namely markers for intransitive, transitive and ditransitive frames of verbs and *PP* as a marker subsuming all PP arguments. (Fellbaum, 1998a, 11)

Glosses and Examples. WordNet provides glosses and examples written by the lexicographers. A gloss gives a short, lexicon-style definition of a concept; examples are used to further illustrate a certain concept. (Miller, 1998a, xx-xxi)

The latest publicly available version of WordNet at the time of writing is version 2.1 of March 2005. Table 4.2 gives some statistics about its size.

[3]In Priss (1998), the relations of hyponymy, synonymy and meronymy in WordNet have been (partly) formalised. From the (mathematically plausible) definition of the relations, some interesting insights about the actual instantiations in WordNet can be gained, up to suggestions for correcting WordNet relations that violate some theoretical property of the mathematical model of the relations, such as the transitivity of (some of the) holonymy relations. However, to our knowledge no large scale 'bug fixing' of WordNet based on these findings has so far been carried out.

4.2.1.2 Extended WordNet

Extended WordNet (XWN) has been derived from WordNet mostly using semi-automatic annotation and disambiguation procedures by Sanda Harabagiu, Dan Moldovan and their colleagues (Moldovan and Novischi, 2002; Mihalcea and Moldovan, 2001; Harabagiu et al., 1999; Harabagiu and Moldovan, 1998).

The most important source of information are the glosses in WordNet written by the lexicographers. The glosses are first semantically disambiguated (using WordNet senses, cf. Mihalcea and Moldovan, 2001). These disambiguated glosses are then used to define an additional 'semantic' relation, holding between two word senses if the first is used in the gloss of the second. This additional relation increases the connectivity of the WordNet network.

In a next step, similarity paths between all WordNet concepts are defined *including* the new relation of *A is used in the gloss of B* and its inverse *B's gloss uses A*. Each such WordNet relation (such as hyponymy, cause etc.) is assigned a constant weight; for each path an overall weight can then be computed by adding the weights of its components (Moldovan and Novischi, 2002). This method is used to derive a similarity measure between concepts that is 'denser' and provides more realistic values than other, similar measures. We will describe other word similarity measures defined on the basis of WordNet in 5.2.2.2. These measures can, for example, be used as a basis for defining an information retrieval model based on word similarity (cf. 2.1.1).

In addition, all WordNet glosses are automatically translated into so-called logical forms (LF). Logical forms provide a comparatively simple semantically oriented representation: Each word is represented by a predicate made up from its lemma, its part of speech and an identifier for the (disambiguated) word sense. Verb predicates receive three arguments, bound to different variables, the first representing the eventuality corresponding to the predicate instance, the second is bound to its syntactic subject and the third to its syntactic object (Davidsonian style, cf. Davidson, 1980). By variable sharing, these are connected to the predicate instances representing the respective syntactic objects in the sentence. Other syntactic material, such as prepositions and conjunctions are expressed by suitable predicates[4]. An example is shown in fig. 4.4. Logical Forms can be used as a source of inference rules. This implicit knowledge from WordNet has been used in the QA system by Southern Methodist University and Language Computer Corporation: Logical forms are derived from passages containing potential answers and fed to a theorem prover that uses the LFs derived from WordNet as axioms to 'prove' answers (cf. 3.3.2.2, Moldovan et al., 2003a).

[4]Note that no logic operators are used. Natural language conjunctions like *'or'* are translated into a *predicate or* that can only be interpreted through suitable axioms.

Tennis, Lawn Tennis
Gloss: a game played with rackets by two or four players who hit a ball back and forth over a net that divides a tennis court
game:n#2(*x*2) & play:v#2(*e*1,*x*1,*x*2) & with(*e*1,*x*3) & racket:n#4(*x*3) & by(*e*1,*x*1) & or(*x*1,*x*3,*x*4) & two:n#1(*x*3) & four:n#1(*x*4) & player:n#1(*x*1) & hit:v#1(*e*2,*x*1,*x*5) & ball:n#1(*x*5) & back_and_forth:r#1(*e*2) & over(*e*2,*x*6) & net:n#5(*x*6) & divide:v#5(*e*3,*x*6,*x*7) & tennis_court:n#1(*x*7)

Figure 4.4: Extended WordNet: Example for a Logical Form Derived Automatically from a WordNet Gloss for the Synset *Tennis, Lawn Tennis* (Adapted from Harabagiu et al., 1999, their Table 2)

Note that while WordNet itself is a relational lexical semantic resource (cf. Miller, 1998a, xvi; Fellbaum, 1998b, 92–94) and as such uses words as basic units and is interested in the relations between them, Extended WordNet makes a move towards adding some compositional lexical semantic information to WordNet.

4.2.2 GermaNet

GermaNet is a lexical database of German designed mainly along the lines of WordNet (Kunze and Lemnitzer, 2002a,b; Kunze and Wagner, 1999; Hamp and Feldweg, 1997). In this section, we will briefly describe GermaNet, especially pointing out conceptual differences between GermaNet and WordNet. These are the main points:

Linguistic Motivation. As described above, WordNet has originally been defined as a psycholinguistic model and only later been ‘discovered’ in computational linguistics (4.2.1). GermaNet has been intended as a tool in (computational) linguistics from the outset, slanting design decisions towards practical applications (Kunze and Wagner, 1999, 9). GermaNet was built using much the same principles as WordNet, but it was built from scratch, i. e., without making use of existing WordNet concepts, employing German text corpora (mostly newspaper texts, Hamp and Feldweg, 1997).

Use of Artificial Concepts. WordNet almost exclusively uses concepts that can be expressed by ‘real’ English words. This quite often leads to conceptual gaps in the hierarchy where some synsets seem to have some least general common subsuming concept which cannot be represented in the hierarchy, as no single word exists to describe it in English. In GermaNet, an attempt was undertaken to generate a more conceptually balanced hierarchy by adding synsets in these cases that are labelled with an artificial description of the 'missing' concept (Kunze and Wagner, 1999, 9).

Differences in Relations. Not all relations that are defined in WordNet are also present in GermaNet. Table 4.3 shows the relations defined in GermaNet. Note that several of the relations cross part of speech boundaries. While relations in WordNet span the whole hierarchy fairly evenly, the relations in GermaNet often only cover small portions of the hierarchy; many links that one would expect are simply missing. This is reflects the much differing numbers of instances for the different relations shown in tab. 4.3. The relatively low number of instantiated relations somewhat diminishes the usefulness of GermaNet as a resource for QA, of course. We will come back to this point in 8.3.1.1.

Hierarchy for Adjectives and Adverbs. In GermaNet, not only nouns and verbs, but also adjectives and adverbs are ordered in a hierarchy defined by the hyponymy relation (Kunze and Wagner, 1999, 9). In WordNet, only similarity is defined for these and no attempt at building a hierarchy for them is made, cf. 4.2.1.

Avoidance of Polysemy. One guiding design principle in GermaNet was to avoid polysemy as much as possible, i. e., not to introduce more senses for a word than absolutely necessary.[5] Thus, the word senses used in GermaNet are generally coarser than those in WordNet, leading to an average of 1.15 word senses per lemma, compared with 1.33 for English WordNet. This is partly due to the observation that too fine-grained distinctions of word senses pose a problem for NLP tools (Kunze and Wagner, 1999, 9). The different polysemy figures (especially marked for verbs) can be found in tabs. 4.2 and 4.4.

Different Verb Valency Description. Verb valencies in GermaNet are described in a notation that was based on the CELEX notations (Gulikers et al., 1990) and thus more fine-grained than those in English WordNet (Kunze and Wagner, 1999, 9). We will return to verb valencies in GermaNet presently.

[5]See, e. g., Buitelaar (1998) who argues against the somewhat arbitrary and often too fine-grained sense distinctions in WordNet.

Relation	Parts of Speech	Number of Instances
Hypernymy	all	45 444
Holonymy	nouns	3 302
Pertainymy	across pos (lemma)	1 667
Antonymy	all (lemma)	1 515
Association	across pos	978
Meronymy	nouns	725
Causation	across pos	234
ParticipleOf	adjective→verb (lemma)	220
Entailment	nouns, verbs	14

Table 4.3: Relations in GermaNet (May 2003 Version)

Part of Speech	Synsets	Lemmata	Senses	Avg. number of senses	
				per Synset	per Lemma
noun	28 093	37 812	41 086	1.47	1.09
verb	8 855	8 061	12 123	1.37	1.50
adjective	5 170	7 149	7 860	1.52	1.10
adverb	2	2	2	1.00	1.00
sum	42 120	53 024	61 071	1.45	1.15

Table 4.4: Some GermaNet Statistics (May 2003 Version)

Only One Meronymy Relation. Instead of three different meronymy relations as in WordNet, only one, more coarse-grained relation is used in GermaNet (Kunze and Lemnitzer, 2002b).

We used the GermaNet version of May 2003.[6] It comprises some 42 000 synsets with a total of 53 000 lemmata (61 000 different word senses). Table 4.4 shows detailed figures for the different parts of speech.

It was briefly mentioned above that GermaNet contains a list of possible valency frames for verbs. Such valency information would, in principle, be very useful for our approach as it could be used in the definition of local inference to describe systematic changes in valency when replacing a verb with, say, one of its synonyms and also to distinguish different usages of a verb. Unfortunately, we found that we could not use the valency information encoded in GermaNet,

[6] In summer 2006, the new version 5.0 of GermaNet was released. This new version has not yet been integrated into our system.

as they are (still) too coarse-grained, on the one hand, and often do not help to distinguish different syntactic readings of a verb, on the other hand. We will return to this point below (5.2.2.5).

About 15 000 concepts from the GermaNet hierarchy have been integrated into EuroWordNet (Vossen, 1998), a multilingual lexical database linking WordNet resources for the following eight European languages: English, Dutch, Spanish, Italian, Czech, Estonian, French, German. Integration is done via an inter-lingual index that abstractly describes concepts based on English WordNet; concepts from the different databases are linked into the interlingua, enabling the transfer of concepts between languages (Kunze and Wagner, 1999).

In principle, this sort of mapping could be used to relate German lemmata to other resources (such as FrameNet, 4.3, or ontologies like SUMO, 4.4.2) through English WordNet, as mappings to and from WordNet to these resources exist. The part of GermaNet integrated into EuroWordNet has turned out to be too small, however, to provide a realistic means of tapping into other lexical semantic resources via English WordNet.

Of the information contained in GermaNet, we have mainly used the synonymy and hypernymy information. Several of the other relations were also used in our system, but turned out not to be useful, as the number of defined instances was quite small (cf. tab. 4.3). We will describe the extraction of the information from GermaNet in more detail in 5.2.2.5.

4.3 FrameNet

In this section, we will describe the FrameNet project (Ruppenhofer et al., 2006; Fillmore et al., 2003a; Baker et al., 1998) and the lexical semantic resource developed within the project and then turn to a German FrameNet project, namely the Salsa project (Burchardt et al., 2006b; Erk et al., 2003a,b), where a corpus of German texts is annotated with FrameNet information.

We will then describe which frame resources we have used for the SQUIGGLI system.

4.3.1 English FrameNet

Within the FrameNet project, a lexical semantic resource is developed that defines frames as abstract descriptions of prototypical situations and objects, specifies roles for the participants, lists words that are associated with these frames, links the frames through different frame-to-frame relations and gives annotated examples from corpora (Ruppenhofer et al., 2006; Fillmore et al., 2003a; Baker et al., 1998).[7] The FrameNet project is carried out at the University of Berkeley, CA, in Charles Fillmore's group.

4.3.1.1 Frame Semantics

FrameNet is based on the theoretical work in frame semantics, which is concerned mainly with the description of frames as (hypothetical) cognitive representations associated with language use (Fillmore, 1982, 1985, 1977, 1976)[8]. In frame semantics, meaning is described through frames: A frame represents a scene, that is a prototypical situation or concept, including information on the objects and participants (typically) involved in it (Fillmore, 1977).

One example of such a frame is a commercial transaction, where some goods are exchanged between a seller and a buyer for money. A word (or rather a lexical unit, i. e., a sense of a word, cf., e. g., Cruse, 1986), such as *sell* is said to *evoke* the corresponding COMMERCIAL_TRANSACTION frame when used in language (it is called the frame evoking element):

> Particular words or speech formulas, or particular grammar choices, are associated in memory with particular frames, in such a way that exposure to the linguistic form in an appropriate context activates in the perceiver's mind the particular frame – activation of the frame, by turn, enhancing access to the other linguistic material that is associated with the same frame. (Fillmore, 1976, 25)

Frames provide information about the participating entities through so-called frame elements; frame elements can thus be seen as abstract semantic role labels. Frame elements are frame-specific: No attempt is made to break them down to a small, general inventory of thematic roles. In the commercial transaction, for example, a SELLER, a BUYER, some GOODS changing hands and some MONEY are (typically) involved.

This specificity of roles for frames is a major difference from other work on semantic roles: Work on thematic roles (also θ-roles, Jackendoff, 1972) and 'deep case' (Fillmore, 1968), for example, assume that there is a small general inventory of semantic roles. Examples of thematic roles are AGENT and PATIENT. These thematic roles specify the participating entities in events at an abstract semantic level. They are each associated with conceptual properties.

[7]We will focus on applications of FrameNet in computational linguistics. For its use in lexicography, see Atkins et al. (2003b).

[8]This work is, in turn, based on Charles Fillmore's earlier work on case grammar (Fillmore, 1968), where a small inventory of 'deep cases' (such as agentive or instrumental) is assumed from which 'surface cases' are derived via transformations.

Vodafone bought Mannesmann		Mannesmann was bought by Vodafone	
Frame	COMMERCE_BUY	Frame	COMMERCE_BUY
FEE	bought	FEE	was bought
Buyer	Vodafone	Buyer	Vodafone
Goods	Mannesmann	Goods	Mannesmann

Figure 4.5: Frame Annotation Example: ‘*Vodafone bought Mannesmann.*’ and ‘*Mannesmann was bought by Vodafone.*’

Vodafone bought Mannesmann.		Vodafone acquired Mannesmann.	
Frame	COMMERCE_BUY	Frame	COMMERCE_BUY
FEE	bought	FEE	acquired
Buyer	Vodafone	Buyer	Vodafone
Goods	Mannesmann	Goods	Mannesmann

Figure 4.6: Frame Annotation Example: ‘*Vodafone bought Mannesmann.*’ and ‘*Mannesmann acquired Vodafone.*’

For example, the AGENT is an animate entity initiating or carrying out the action under discussion. Thematic roles generically describe the participants of an event according to their respective properties, irrespective of the predicate. They map onto grammatical functions and syntactic case through rules. For example, the AGENT will typically be expressed by a subject in sentences in active voice.

It has proven difficult, however, to consistently describe semantic and syntactic regularities through a small set of semantic roles and mapping rules. FrameNet uses semantic roles (namely frame elements) that are dependent on the frame. Thus, the problem of reducing semantic roles to a small consistent set is avoided.

When used for annotation, frames abstract over syntactic variations such as active and passive. Figure 4.5 shows that the frame annotation is equal for an active sentence and its passive counterpart; fig. 4.6 shows that it is also equal for two sentences that are synonymous in meaning, but employ different verbs.

This abstraction over differences in surface realisations makes frames especially useful for defining indirect answerhood: Syntactic differences related with syntactic variations and also with the replacement of a word with semantically similar one can be abstracted over.

Fillmore (1985) described how frames can be used as a (lexical) semantic representation and outlines how they could be interpreted. This frame semantics is further motivated and illustrated in Fillmore (1982); a more detailed overview of its use in the FrameNet project can be found in Fillmore et al. (2003a). Fillmore and Baker (2001) presents an example where (part of) a short news story is translated into a frame semantics representation.

4.3.1.2 FrameNet Annotation

The FrameNet database that is produced by the Berkeley FrameNet project lists frames, their respective frame elements and lexical units that evoke the frame. For each frame, a number of manually annotated example sentences is given: In the example sentence, the parts of the sentence corresponding to the frame evoking element and the frame elements are marked and labelled with the frame element, a phrase type (such as NP) and a grammatical function (such as object). Example sentences are mostly taken from the British National Corpus (BNC, `http://www.natcorp.ox.ac.uk/`). Annotation proceeds frame by frame: First, all instances of all frame evoking elements associated with a frame are selected from the corpus, then an attempt is made to filter difficult material (unclear or untypical cases, Atkins et al., 2003a). The guiding idea is that annotation provides uncontroversial illustrative examples for the frame under consideration. For a more detailed overview of the corpus annotation, see Ruppenhofer et al. (2006). The annotation of one example frame is described in Fillmore et al. (2003b). The FrameNet database can be accessed via the Internet under `http://framenet.icsi.berkeley.edu/`.

4.3.1.3 Frame-to-Frame Relations

The FrameNet database defines a number of relations between frames (the so-called frame-to-frame relations), that partly correspond to the ‘classical’ semantic relations (cf. 3.5.2.3). Note that there is – as far as we are aware – currently no worked-out formalisation of the frame-to-frame relations, making it difficult to exactly specify and distinguish them: As the annotation within the FrameNet project proceeds, the frame-to-frame relations in the database tend to change and with them, the definitions of the relations themselves. We used the informal characterisations in Ruppenhofer et al. (2006) and also Ruppenhofer et al. (2005) as the basis of our description.

Inheritance. A frame that ‘inherits’ another frame is a specialisation of that frame. It is thus similar to the ‘classical’ hyponymy relation in lexical semantics (however, frame relations are relations between frames, not lexical units, Ruppenhofer et al., 2006, 104–6). For example, EXECUTION inherits KILLING. All frame elements of the parent frame must also be defined for the child frame (either associated with the same selectional preferences or more specific ones).

Subframe. Complex frames (generally describing a whole scenario) are made up from smaller, distinctive subframes. Subframes generally have a specific temporal sequence. Frame elements of the subframes can generally be identified with frame elements of the superordinate frame. COMMERCIAL_TRANSACTION, for example, has COMMERCE_GOODS_TRANSFER and COMMERCE_MONEY_TRANSFER as subframes, indicating that both take place in the 'larger' scenario (Ruppenhofer et al., 2006, 108). Some experiments have been undertaken to further specify the relations of the subframes to each other (Narayanan et al., 2002), but this sort of information is not available for the majority of frames.

Causative-of and Inchoative-of. These two relations generally link stative frames with inchoative and causative frames built upon them (Ruppenhofer et al., 2006, 110). For example, the frame CAUSE_CHANGE_OF_SCALAR_POSITION is causative of CHANGE_POSITION_ON_A_SCALE (i. e., some agent effecting some change; typical frame evoking elements are *raise* vs. *rise*) which in turn is inchoative-of POSITION_ON_A_SCALE (that describes a transition between states, for example from *low* to *high*).

Using. If a frame 'uses' another frame then that other frame provides a more schematic view of the situation. For example, COMMERCE_BUY uses COMMERCE_GOODS_TRANSFER, indicating that an act of buying includes the act of some goods changing hands (Ruppenhofer et al., 2006, 110). This relation will be in large parts superseded by the new relation Perspective-on (from release 1.3 on, Ruppenhofer et al., 2006, 106–8).

See also. This relation is mainly useful for lexicographers. It was added as a possibility of pointing from one frame to another, somehow related frame where none of the other relations holds (Ruppenhofer et al., 2006, 111). The OPERATE_VEHICLE frame, e. g., points to the RIDE_VEHICLE frame, as both are similar, but need to be distinguished. The nature of this relation is so general that we have found it not to be useful for automatic processing.

In a recent case study, Aljoscha Burchardt, Anette Frank and Manfred Pinkal have shown how frame-to-frame relations can, in principle, be combined with information from deep parsing to yield a partial meaning representation (Burchardt et al., 2005b).

4.3.1.4 Using FrameNet for Linguistically Informed Question Answering

To make use of FrameNet, we induce relations holding between lexical units from the assignment of lexical units to frames and from the relations holding between those frames. Given the description of the resource and Fillmore's definition cited above that states that 'activation of the frame, by turn, enhancing access to the other linguistic material that is associated with the same frame' (Fillmore, 1976, 25), this seems justified.

As frames describe abstract prototypical situations, the FrameNet annotation abstracts over a number of surface phenomena in the underlying text: Details like sentence voice or pragmatic features like register or dialect are not represented. For a number of constructions like transparent nouns (*kind of X*), support verb constructions (where a semantically – more or less – empty verb shares arguments with its frame evoking object, such as *take decision* vs. *decide*) or non-literal uses (idioms or metaphors), will receive a suitable frame annotation (Fillmore and Sato, 2002). In fact, a notion of paraphrasability underlies the definition of frames and the assignment of frame evoking elements to them (Ruppenhofer et al., 2006, 11–18). Thus, FrameNet structures provide an interesting level of abstraction in many respects.

However, some information is lost through the abstraction: Note that, for example, antonyms are often assumed to evoke the same frame with no means of distinguishing the 'polarity': Adjectives like *good* and *bad* both evoke the frame DESIRABILITY. Thus, a QA system based on our approach that uses FrameNet data as a resource needs to be able to additionally check potential answer representations for mismatches in the polarity of question and answer, cf. 5.1.5.

Of the FrameNet relations, we have used all except for the *See also* relation and the *Precedes* relation (the latter was only defined in release 1.3). However, when transferring frame-to-frame relations into our relations of local inference, care must be taken that the relations are properly represented. When using the inheritance frame relation like a hyponymy relation, for example, its inverse does not give rise to entailment, but only to 'uncertain' inference (3.5.1.3); the inverse of the causative relation behaves similarly. We will describe the extraction of the relational information in more detail in 5.2.2.6.

All in all, the FrameNet data can be expected to eventually provide a very useful basis for building (partial) semantic representations of texts, roughly along the lines described in Fillmore (1985). By highlighting on prototypical situations (represented as frames) and linking them through different frame-to-frame relations, a representation at a level of granularity can be derived from texts that allows local inferences as required for QA, as described above (cf. 3.5). The idea of using FrameNet data as a source of knowledge for inferences somewhat similar to ours is also explored in Chang et al. (2002a,b, preliminary ideas) and in Narayanan and Harabagiu (2004a,b, first experiments with a QA system using FrameNet data).

Table 4.5 shows detailed figures on the size of the FrameNet database that we used, namely release 1.2 of December 2004. This version defines 609 frames with a total of 8 764 frame evoking elements (7 198 different lemmata) for English (see also tab. 4.5). In July 2006, release 1.3 became available that contains mappings from about 10 500 words (8 500 tokens) to 750 frames. We did not update to the new release, in order to remain compatible with the first release of the German FrameNet corpus annotated in the Salsa project (4.3.2).

Note that these figures cannot be directly compared with those reported for WordNet (4.2) and GermaNet (4.4): FrameNet is mainly interested in semantic roles and thus argument-bearing words. Thus, a lexical coverage for about 10 000 words is sufficient to account for a substantial amount of the argument-bearing words in texts. In Fillmore and Baker (2001), for example, the annotation of a short newspaper text is described that shows that a relatively small number of frames and associated frame evoking elements suffices to represent a text.

It is reasonable to assume a partition of labour between frame semantic representations and other lexical semantic resources (especially WordNet/GermaNet) in linguistically informed QA: Frame representations provide information on argument-bearing words and semantic roles, while word nets contribute information on lexical concepts and their relations, especially for nouns.

However, a wider coverage is still needed, especially a coverage of more different domains would be advantageous. This applies to German FrameNets in particular, where the coverage is currently more limited (cf. 4.3.2).

We therefore believe that, for some time to come, even with continuing effort in the FrameNet project, FrameNet data can only provide a limited coverage and needs to be complemented with other information in practical systems. Our approach shows one systematic way of doing so for QA. We will take up this discussion in the evaluation (7.2.2.1) and the conclusions section (8.3.1.1).

4.3.2 German FrameNet: The SALSA Project

In the Salsa project (*SAarbrücken Lexical Semantics Annotation and analysis,* `http://www.coli.uni-saarland.de/projects/salsa/`), a Ger-

Part of Speech	Lemmata	FEEs	Avg. # of frames per lemma
noun	3 404	3 679	1.08
verb	2 265	3 412	1.38
adjective	1 411	1 546	1.10
preposition	61	70	1.15
adverb	46	46	1.00
numerals	10	10	1.00
interjections	1	1	1.00
total	7 198	8 764	1.22

Table 4.5: Some FrameNet Statistics (Release 1.2, December 2004, 609 frames)

man text corpus is annotated with FrameNet structures as a computational linguistics resource (Burchardt et al., 2006b; Erk et al., 2003a,b). The Salsa project is carried out at Saarland University, Saarbrücken, in Manfred Pinkal's group. We will shortly describe the effort here, especially focussing on differences from the English FrameNet project.

By annotating a German corpus with FrameNet information[9], a resource is created that is useful in – at least – three ways: As a training corpus for tools for automatic FrameNet annotation of German texts (cf. Erk and Padó, 2006; Baldewein et al., 2004), for research in language similarities and differences (especially from a FrameNet perspective, Padó and Lapata, 2005) and as a source for research in syntactic and lexical semantic phenomena of German. We are interested in the first application, namely using the annotated corpus as a source for grammar induction. We have used the corpus to semi-automatically derive a grammar for our FrameNet construction module that builds FrameNet structures on top of the dependency structures derived by our parser (cf. 6.2.10). We use this grammar to map German lexical units in the input text to FrameNet frames and thus to add a FrameNet annotation level to our text representations.

The corpus that is annotated with frame structures is the Tiger corpus, a German newspaper corpus that has been manually annotated with syntactic structures in the Tiger project (Brants et al., 2002). The Tiger corpus comprises some 40 000 sentences from the German daily paper Frankfurter Rundschau,

[9]Note that, ideally, FrameNet structures would be completely language independent, as they describe prototypical situations rather than language-specific details. In Burchardt et al. (2006b), a few cases are cited where frames from English FrameNet cannot directly be used to represent German texts, as some peculiarity of the English language has crept into the design of the frame.

manually annotated with morphological information and syntactic structures. The syntactic annotation uses fairly theory-independent combination of phrase structures and dependency structures, employing crossing edges to account for the relatively free word order in German (Brants et al., 1999).

4.3.2.1 Frame Annotation in the Salsa Project

A frame annotation is added to sentences from this corpus by selecting nodes in the syntax tree corresponding to frame evoking elements and frame elements, respectively, and labelling them with the frame or frame element in question. Each sentence is annotated by two annotators (student assistants), in case of differences two further annotators (project researchers) adjudicate the annotation (Burchardt et al., 2006b). A specialised annotation tool is used to facilitate the process (Burchardt et al., 2006a).

In the Salsa project, annotation is done lexical unit by lexical unit. This is an important difference from the Berkeley FrameNet, where annotation proceeds frame by frame. Besides, not only ‘vanilla’ cases are selected from the corpus; rather, the aim here is at full annotation. In consequence, annotators potentially have to choose from a number of different possible frames for each sentence for polysemous target predicates.

This exhaustive corpus annotation carries a number of special challenges (Ellsworth et al., 2004): On the one hand, for about a third of the annotated sentences, no suitable FrameNet frame has yet been defined. These gaps in FrameNet’s coverage are reported to the Berkeley FrameNet project. The plan is to eventually fill these gaps (Burchardt et al., 2006b).

Then, there are a number of cases that exhibit different degrees of ‘limited compositionality’, namely support verb constructions and idiomatic and metaphoric expressions. Here, an attempt is made to annotate both ‘non-literal’ and literal meanings as a basis for further research. Relatively often, no clear decisions can be taken. In these cases, parts of the annotation can be left underspecified (Burchardt et al., 2006b).

We have mentioned above the possibility of using the Salsa corpus to train software tools for automatically deriving FrameNet structures from German texts. A first software package for automated FrameNet annotation (cf. 6.2.10.1) both for English and German has been developed in the project (Erk and Padó, 2006). The tools of the package have successfully been trained with the German FrameNet annotation. Among other things, they will help to speed up the annotation process.

Currently, the Salsa corpus contains close to 20 000 annotated sentences for 476 German predicates using 252 FrameNet frames and 373 new proto frames

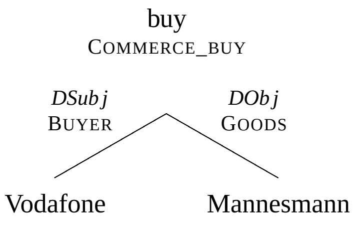


Figure 4.7: Dependency Structure Extended with Frame Annotation for ‘*Vodafone Bought Mannesmann.*’

(628 in total). Annotation was almost exclusively done for verb predicates, only extended by a handful of deverbal nouns.

These figures show that, currently, grammars induced from the Salsa corpus will only reach a limited coverage. In Burchardt et al. (2005a), an interesting method is described for overcoming this data sparseness: For words that are not covered by the FrameNet annotation, the most similar words that *are* known are searched using a WordNet-based similarity measure. A preliminary evaluation indicates that the precision of this method lies around 40 %. When the method is used as a basis for grammar induction, it may be necessary to manually correct the results.

4.3.3 Frame Resources Used in SQUIGGLI

We combine information from the Berkeley FrameNet database and from the annotated Salsa corpus in the following way: We extracted information on frames and frame-to-frame relations from the English FrameNet database. Together, they are used as a source for textual inferences. This will be described in more detail in chapter 5.

We have induced frame annotation rules from the Salsa corpus. These rules are used to assign frame annotations to text representations (see below). The interface between the FrameNet data and the German frame annotation is their use of the same frame structures.

For both texts in the document collection and questions, we derive syntactic dependency structures. These are extended with a frame annotation: The frame annotation module uses a set of rules that match parts of dependency structures to generate suitable frame representations. Figure 4.7 shows an example: The

Part of Speech	Frames	Lemmata
verb	175	418
noun	36	94
adjective	17	12
total	228	524

Table 4.6: Coverage of Frame Assignment Rules

dependency structure of the sentence is extended by additional frame information (in CAPS).

The rules used to add frame structures to the PREDS were partly induced from the Salsa corpus, partly written manually. We used the adjudicated data prepared for the first corpus release (scheduled for spring of 2007) for induction.

We did the extraction semi-automatically, using a Ruby script[10] to extract information on the syntactic 'paths' in the underlying Tiger annotation between the constituent marked as frame evoking element and the different frame elements. The tool converts these paths into suitable grammatical functions, such as subject, object or different PPs. From this information, mappings are derived using a standard template.

We could thus induce lexical entries for 355 lexical units, almost exclusively verbs, for 201 frames, altogether (not counting new frames that have not yet been added to the FrameNet hierarchy, as no frame relations are currently available for those).

We extended the rules automatically derived from the Salsa corpus with a set of manually written rules. These were written to cover frames, especially from the business domain, for a small set of about 100 German newspaper articles.

We manually added selection preferences for frame elements based on GermaNet predicates to a number of rules. For example, for a BUYER of COMMERCE_BUY, we add a preference for a hyponym of *person* in GermaNet. These preferences will be described in more detail below (6.2.10).

Table 4.6 shows the current numbers of frame assignment rules for different parts of speech. These figures show that coverage is limited so far. Even this limited frame information was useful for finding answers: In the evaluation, we found that our system could find about 16 % of all answers only through

[10]We would like to thank Katrin Erk and Sebastian Padó of the Salsa project for providing the script.

the use of frame information (7.2.2.1). This indicates that frame information (especially the information on frame elements) forms an important resource for linguistically informed QA.

4.4 Ontologies

In this chapter, we have concentrated so far on lexical databases as possible sources of knowledge. We will now shortly introduce two formal ontologies, namely Cyc (Matuszek et al., 2006) and SUMO (Niles and Pease, 2001), and discuss their potential as sources of information for linguistically informed QA.

We will follow common practice in computer science and AI and regard a formal ontology as a repository of conceptual knowledge: They list abstract concepts and relate them to each other through different relations. There is often a mapping from/to natural language concepts, but ontology and natural language lexicon are mostly treated as separate resources, which are related via relations from (natural language) words to concepts in the ontology (cf., e. g., the detailed model in Nirenburg and Raskin, 2004, esp. 170–245).

It should be noted that, in practice, there often is a considerable overlap between knowledge contained in linguistic resources like WordNet and 'technical' ontologies like the ones described here. However, the focus is different: In technical ontologies it lies on the abstract concepts and their relation in the world, while in lexical resources words and their linguistic relations are focussed.

We will shortly introduce a number of concepts and terms used in ontologies. We use the term ontology here in the rather narrow sense employed in computer science and artificial intelligence (AI), and not the more general (and far more controversial) sense employed in philosophy (cf. Guarino, 1998); our description follows Russell and Norvig (1995, 217–264).

Individuals/Instances. An ontology can enumerate a set of entities called individuals or instances. There is generally no restriction of what can be regarded as such an entity (including moments in time, ideas etc.).

Categories. A category (also often called concept or class) provides a name for a set of individuals that share a certain property. Categories form the basis for allowing abstract reasoning.

Attributes. Each individual has a set of attributes, consisting of name and value, describing it. Generally, the assignment to categories is done on the basis of similarity with regard to a certain attribute.

Relations. Relations between individuals (and especially between concepts) are used to organise the ontology. Most ontologies use the *is-a* relation as the basic relation to construct a taxonomic hierarchy of concepts (naturally leading to sub-concepts and super-concepts). Additional relations may be defined.

Note that under this definition, both the WordNet and the FrameNet resource can be regarded as special linguistic ontologies; however, we will continue to distinguish between these lexical ontologies and formal ontologies.

We will now shortly describe two ontologies that are designed under technical considerations.

Both ontologies encode their knowledge in a formal language (namely CycL and KIF, both based on first-order predicate logic, see below). This formal language is used to define both relations and additional knowledge. In fact, the larger part of the knowledge residing in the respective ontologies is encoded not as general relations such as the *is-a* relation, but rather by special axioms representing additional constraints over the categories.

Making the knowledge encoded in the ontologies available for QA would be interesting. However, utilising this knowledge is not straightforward, as the axioms contain inconsistencies and often are too rigidly defined to be useful in natural language processing (Suchanek, 2005). Besides, transferring the knowledge to textual inference rules as would be required in our approach of linguistically informed QA.

Moreover, there are currently no mappings from German words to concepts in either Cyc or SUMO[11]. Therefore, we could not directly employ these ontologies in our system.

4.4.1 Cyc

Cyc is the world's largest general purpose computational ontology with a knowledge base containing currently over 250 000 concepts and more than two million 'facts' (rules and assertions, Matuszek et al., 2006). It is estimated that 900 person years have gone into building Cyc over the last 20 years.

Rules are expressed in CycL, a language based on first-order predicate logic with some higher-order language extensions (such as the possibilities to quantify over functions and use predicates as arguments of functions, Matuszek et al., 2006; Ramachandran et al., 2005).

The taxonomy of concepts is built up using a number of special relations expressed in CycL (especially *is-a* and *generalises*). A large part of Cyc's rules is made up by constraints over the arguments of predicates. The three assertions

[11]Such a mapping does exist for WordNet synsets to Cyc and SUMO concepts, Pease and Niles (2002).

in the following example, e. g., are used to constrain the relation *biological-Mother* and ensure that nobody can be their own mother, that an entity that has a *biologicalMother* must be of the type animal (or of a subtype) and that the *biologicalMother* itself must be a female animal. The constraints are 'enforced' by a set of axioms; for example, there is an axiom that asserts that for every instance of a relation whenever an assertion with the predicate *arg1Isa* exists that the first argument of the instance is actually of the required type.

```
(isa biologicalMother IrreflexiveBinaryPredicate)
(arg1Isa biologicalMother Animal)
(arg2Isa biologicalMother FemaleAnimal)
```
Matuszek et al. (2006, 2)

The Cyc ontology is conceptually divided into three description levels, namely the upper, middle and lower ontology. The upper ontology contains a small number of very general, universal concepts such as relationship types, the middle ontology 'everyday knowledge' and the lower ontology specialised, heavily domain-dependent knowledge. (Matuszek et al., 2006)

Assertions in Cyc are encapsulated in contexts (in Cyc jargon called 'microtheories', Cycorp, 2002; Guha and Lenat, 1994). This allows the knowledge base to hold global inconsistencies: In reasoning, only necessary contexts are activated, each of them guaranteed to be free of inconsistencies. Through additional, manually set links in the knowledge base, other contexts may be activated in addition, also guaranteed to introduce no conflicts into the reasoning process. It is claimed that through this strategy the knowledge base can be kept manageable (as it must not be globally consistent).

Cyc was only available under commercial licences until recently. Since spring 2006, Cyc is available free of charge for research purposes as Research-Cyc, `http://research.cyc.com/`. Besides, a small subpart is also generally available free of charge as OpenCyc, `http://www.opencyc.org/`.

4.4.2 Suggested Upper Merged Ontology: SUMO

The Suggested Upper Merged Ontology (SUMO[12]) has been constructed by merging, unifying and extending a number of smaller existing ontologies (Niles and Pease, 2001). It is currently the largest computational ontology available to the general public. In many respects, it is a common effort, as it is largely based on a lively ongoing discussion on a dedicated email list.

[12]`http://www.ontologyportal.org/`

SUMO currently contains some 20 000 concepts – for all sub-domains combined – and relates them to each other. Most concepts describe general, 'everyday' entities, but also linguistic concepts (Farrar et al., 2002), geographic names or financial terms.

One interesting feature of SUMO is that, similar to Cyc, its concepts are linked to axioms described in the Knowledge Interchange Format language (KIF, Genesereth and Fikes, 1992). KIF is a language that allows expressing (a superset of) first-order predicate logic and was defined as a standard format for formalising and exchanging knowledge.

Axioms generally define restrictions or constraints on concepts. In Niles and Pease (2001), for example, the concept 'collection' – a football team, for example, would be (a subclass of) such a collection – is introduced together with an axiom postulating that a collection must not be empty, i. e., have at least one member.

4.5 Conclusions

In this chapter, we have introduced Dependency Grammar and PREDS, the dependency structures that we use as the 'core' of our text representations.

We have further introduced two lexical semantic resources, namely WordNet (or rather GermaNet) and FrameNet, which we have used as sources of knowledge for QA.

Both WordNet and FrameNet can be used as sources of data, especially of lexical semantic relations, (cf. Fellbaum, 1998b, 92–94, Miller, 1998a, xvi): Most information about the concepts contained in their databases is expressed as relations between the concepts. They do not attempt to exhaustively describe the information by breaking down the concepts into smaller sub-concepts as the componential approaches to lexical semantics would.[13]

We have given an overview over the relations encoded in WordNet (and GermaNet) and FrameNet. As described above (3.5), these relations are used as the basis for linguistically motivated local inferences in our approach to QA. We will return to the question which of these relations we have actually used and how we have translated them into our local inference rules in 5.2.

We have also described formal ontologies, namely Cyc and SUMO, as possible knowledge bases. We have noted that with both Cyc and SUMO, most knowledge resides in axioms. We could not directly integrate either of these

[13]Even though the FrameNet relations of Using and Subframe can be seen as introducing means of de-composing complex frames. However, the aim is not to *exhaustively* describe a frame through its components.

ontologies into our approach, especially as there are currently no direct mappings from German words to the respective concepts.

In the next chapter (5) we will describe in more detail how our notion of indirect answerhood can actually be translated into a efficient algorithm as a basis for QA and how linguistic knowledge from GermaNet and FrameNet can be integrated before turning to a description of our system in chapter 6.

Chapter 5

Matching Structured Representations of Questions and Answers

In chapter 3, we have developed the notion of indirect answerhood as the core of linguistically informed QA. So far, we have only specified this relation informally between surface texts. We will now turn to the question of how this relation can be defined and used in a practical QA system.

In 5.1, we will define indirect answerhood (and its components, namely textual inference and direct answerhood) as a relation between text representations. These text representations are based on syntactic dependency tree structures and extended by lexical semantic information, especially information derived from the lexical resources GermaNet and FrameNet (chapter 4). Textual inferences will be defined as relabelling of the text representations and direct answerhood as an embedding of question representations in text representations. Both are controlled by rules in a linguistic knowledge base.

In 5.2, we will turn to the question how linguistic information from different sources can be integrated into the linguistic knowledge base and thus be utilised for indirect answerhood. We will show how indirect answerhood phenomena that we have identified as relevant in our corpus study above (3.5.2) can be practically modelled.

In 5.3, we will define an efficient search algorithm based on the relation of indirect answerhood. The algorithm is based on a tree matching algorithm (more exactly, unordered path inclusion, Kilpeläinen, 1992). We describe the extensions needed to integrate the textual inferences and direct answerhood relations into the search algorithm.

In 5.4, we will further modify the search algorithm so that it makes use of a generic relational database system. This allows us to easily store structured text representation derived during pre-processing and search answers by systematically deriving suitable database queries from question representations.

5.1 Defining Indirect Answerhood

In 3.5, we have described linguistic phenomena that need to be accounted for when defining indirect answerhood. We have sketched the relationship on text surface structures. This is, of course, not a suitable level of representation for automatic systems. We will define an indirect answerhood relation on the basis of linguistic representations.

We base the definition on syntactic dependency trees extended with lexical semantic information. The relations of textual inference and direct answerhood are represented as a relabelling and a matching operation on these tree structures, respectively.

The relation of indirect answerhood between text representations models indirect answerhood on texts: We have introduced indirect answerhood on texts as the central concept of linguistically informed QA. Since a model of indirect answerhood based on structured semantic representations is not practically feasible, we have decided to use a model based syntactic representations extended with lexical semantic information. The definition of indirect answerhood given in this chapter approximates the ‘full’ relation of indirect answerhood between texts. Note that we will call the relation between text representations indirect answerhood and only distinguish it from the ‘full’ relation on texts where the distinction is not clear from the context.

The definition of indirect answerhood that we give is quite abstract. It is especially not language-specific: Language-specific information is encoded in the text representations and in the linguistic knowledge base, but not in the relabelling and matching operations on the text representations. We will further specify the text representations and describe an instantiation of the linguistic knowledge base for German, as we have used it in SQUIGGLI in 5.2.

We will start by introducing relevance values that are used to model relevance judgments (5.1.1). We will define how texts are represented next (5.1.2), then the notion of local inference between two such representations (5.1.3), what it means if one representation stands in the relation of direct answerhood to another (5.1.4) and how answers with inverse polarity and uncertain answers can be automatically marked (5.1.5).

5.1.1 Relevance

We integrate a measure of relevance into the definition of indirect answerhood. This relevance measure models the relevance of an answer (or rather, an answer representation) for a given question (its representation). It can be interpreted as a 'degree of answerhood'. This measure is computed recursively when checking for indirect answerhood. It is mainly based on the confidence of local inferences; information about the relevance of each local inference step is stored in a linguistic knowledge base for each possible textual inference step.

We can simply handle uncertain inferences (3.5) by using the relevance value of local inferencing steps to express confidence in the inference step: By assigning lower relevance values to uncertain inferences, we can ensure that they are dispreferred, whenever more likely (and thus more relevant) inferences (and hence, answers) can be found.

We can also compare different answers that are found for a question with each other based on their relative relevance. This opens different possibilities for presenting the found answers to the user: The system can either present only the highest-ranked answer or the top *n* answers. Or it can present all answers that are ranked over some relevance threshold. We will return to this discussion below when we describe the user interface of the SQUIGGLI system (6.4.5).

In addition, we use this measure to prune the search for answer representations; this will be discussed in detail in 5.3.2.3.

Relevance measures are computed as follows: A basic relevance value is assigned to each textual inference step. Whenever a word is replaced by another one (or rather, when a relabelling step is performed on text representations, see below), an individual confidence value is assigned to this replacement. For a sequence of replacements, the product of the relevance values of the contributing steps is used as the overall relevance value.

This definition is quite flexible and could accommodate a number of different models of relevance, depending on how the relevance values for the different inference steps are instantiated. We currently use a general measure of (semantic) similarity as the basis for defining the confidence values for the local inferencing steps. This will be described in 5.2, where we show in more detail how we build up the linguistic knowledge base from the available resources (especially GermaNet and FrameNet, cf. chapter 4). It is also discussed in Fliedner (2005); note that in this paper, we used the term 'generalised similarity measure' (GSM) instead of relevance. The term relevance, however, is more suitable for describing indirect answerhood than the more constricted notion of similarity, cf. the discussion above (3.5).

Relevance is a frequently used concept in Information Retrieval (cf. 2.1.1, see also Baeza-Yates and Ribieiro-Neto, 1999, 19–34). Quite generally, a retrieval model attempts to describe (or predict) the expected relevance judgments of users through a ranking function that assigns a rank to a document relative to a given query. The computed measures are generally used to rank different matching documents.

Relevance in QA, in contrast, is a relatively new subject. Some QA systems use internal measures and heuristics to select the best answer(s) found in the document collection (cf. 2.2.2), but these are generally presented as a mere technical detail rather than a linguistically relevant part of the process and not further specified.

Relevance in QA is the central subject of Marco De Boni's Ph. D. thesis (De Boni, 2004): He suggests that answerhood should not be defined in terms of a yes/no-decision but rather as a relation between a question and *all possible text fragments*. A QA system would then, given a question – at least in principle – rank all text fragments in its underlying document collection for relevance and present the most relevant fragment as an answer. De Boni suggests that the overall relevance measure should be computed by combining a number of different independent measures (such as word overlap). It turns out, however, that the details are not sufficiently worked out and too closely linked to De Boni's own QA system to be useful in the general case.

5.1.2 Text Representations

We define indirect answerhood as a relation between text representations. These text representations are based on syntactic dependency structures. The dependency structures are extended with lexical semantic information. We use a German version of FrameNet and GermaNet (the German version of WordNet) as sources of lexical semantic information (chapter 4).

The text representation consists of a set of syntactic dependency structures, where each sentence corresponds to one dependency structure. We will assume a broad definition of sentence that includes, for example, headlines and similar material, not necessarily containing a verb. A dependency structure is based upon a labelled tree structure. A text representation is a set of such dependency trees, i. e., a forest of dependency trees. We have described the PReDS dependency structures in 4.1. Note that the use of dependency structures and the different simplifications of the structures described above are fundamental for our following definition of indirect answerhood.

PReDS form the core of the text representations. They are extended with lexical semantic information. This information is represented as additional node and edge labels in the dependency structures: Each node and each edge may carry an arbitrary number of labels; the definition (def. 5.2) therefore uses a

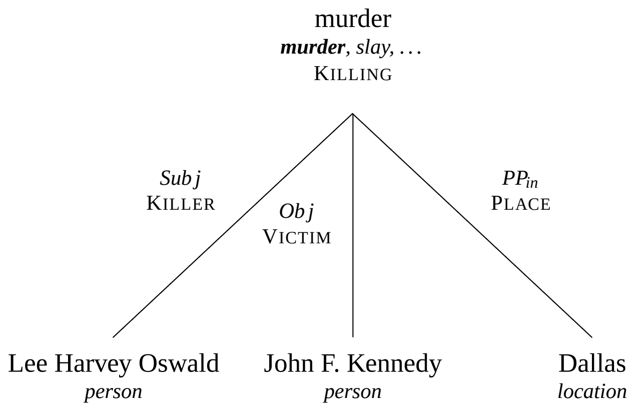


Figure 5.1: Text Representation for *‘Lee Harvey Oswald has murdered John F. Kennedy in Dallas.’*

labelling relation instead of the more common labelling function. Dependency trees contains both syntactic and semantic information in the labels.

We do not formally distinguish between syntactic and semantic labelling, but we assume that the labelling alphabets may consist of two (or more) separate subsets containing syntactic and lexical semantic labels. Syntactic labelling includes word stems as node labels and grammatical functions as edge labels; semantic labels represent lexical semantic predicates as node labels and semantic role labels as edge labels. Figure 5.1 shows a simple example.

Each node and edge in the example has different labels, expressing information from different linguistic levels. In the figures, the linguistic levels are distinguished by different fonts: For syntactic information, i. e., word stems and grammatical functions, we use a roman font, for WordNet information, *italics*, and for frame information, CAPS.

In addition to the dependency trees, we assume an additional type of edges between nodes to express both coreference (3.5.2.4) and predication (3.5.2.5). These edges can link nodes both within single trees and also nodes of different trees of one text representation. Note that these additional edges may, in general, destroy the ‘tree-ness’ of the text representations. However, we will not use this additional edge type in the final algorithm for matching question and answer representations, rather, it will be compiled out to keep representations manageable. We will describe this in greater detail below.

We will start with an auxiliary definition, namely that of the labelling alphabets (or labels) that we use. It defines node and edge labels. Each edge label is compounded by a node label and an ‘atomic edge label’, such that each edge label contains a node label as its ‘prefix’.

Definition 5.1 (Labels) *The labels are a triple $\mathscr{L}$, $\mathscr{L} = \langle L_N, L_A, L_E \rangle$:*

L_N *is a non-empty set (the node labels).*

L_A *is a non-empty set (the atomic edge labels).*

L_E *is a non-empty set (the edge labels). Edge labels are defined as pairs $\langle l_N, l_A \rangle$, where $l_N \in L_N, l_A \in L_A$. In general, we will write $l_N.l_A$ as a short form.*

Note that we will assume that the labels remain constant for all following definitions, that is the same labelling alphabets are used.

Definition 5.2 (Text Representation) *A text representation is a seven-tuple T, $T = \langle V, E, \mathscr{L}, P, R, C, W \rangle$:*

V *is a non-empty, finite set of nodes.*

E *is the set of edges, $E \subseteq V \times V$. V and E form a forest of trees.*

$\mathscr{L} = \langle L_N, L_A, L_E \rangle$ *(the labels).*

P *is a node labelling relation, $P \subseteq V \times L_N$.*

R *is an edge labelling relation, $R \subseteq E \times L_E$. For every $\langle \langle u, v \rangle, l_N.l_E \rangle \in R$: $\langle u, l_N \rangle \in P$.*

C *is a set of additional edges between nodes, $C \subseteq V \times V$ (the equivalence edges).*

$W : V \mapsto \{0, 1\}$ *(the wh-phrase marking)*

The relation C is used to represent the additional edges for coreference and predication; the function W is used to mark *wh*-phrases in questions (cf. 5.1.4).

This text representation is to be derived from a given text by an appropriate parsing relation, defined as follows.

Definition 5.3 (Parsing) *Text representations T as defined above are derived from texts τ by a parsing relation f : $f \subseteq T \times \tau$.*

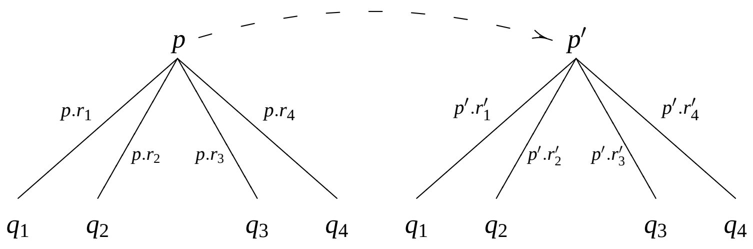


Figure 5.2: Local Inference as Relabelling of Elementary Trees

This definition assumes that more than one text representation can be derived from a text, in general. This is realistic, as current parsers cannot resolve all ambiguities in natural language texts.

We assume that the indirect answerhood relation between a question and a text holds, if at least one text representation of the former stands in the indirect answerhood relation with at least one text representation of the latter.

It is therefore important to keep the number of representations small for practical implementations. In the SQUIGGLI system, we heuristically select a best parse from different alternative parses.

5.1.3 Textual Inferences

We define textual inference uniformly as local relabellings of elementary trees using a 'linguistic knowledge base' that encodes, for instance, relations such as synonymy.

As domain for the relabelling operation, we define elementary trees. An elementary tree consists of a node and the outgoing edges (if any). Thus, relabelling is a very local operation: It cannot make reference, for example, to any child nodes. This keeps the operation as simple and computationally tractable.

Figure 5.2 shows an (abstract) relabelling of an elementary tree: The node label p is changed to p', the labels of the outgoing edges r_1 through r_4 are changed to r'_1 through r'_4, respectively. Note that the daughter nodes labelled q_1 through q_4 are not touched by this relabelling at all. The relabelling must be licenced through a relabelling rule; information about permissible relabelling steps (and thus about possible local inferences) comes from the linguistic knowledge base.

The linguistic knowledge base consists (mainly) of relations between the two labelling alphabets that are used in the text representations. These relations describe relabellings of elementary trees as just described, namely a relation between two node labels and corresponding relations between edge labels. For the relabelling shown in fig. 5.2, e. g., the former would include the relabelling relation between node labels p and p', the latter that between the edge labels $r_1, \ldots, r_4$ and $r'_1, \ldots, r'_4$.

These relabelling relations correspond to textual inferences of the represented texts. Thus, whenever the relabelling of an elementary tree of a text representation is licenced by the linguistic knowledge base, the inference relation holds between the text representations and therefore also between the texts.

Definition 5.4 (Linguistic Knowledge Base) *A linguistic knowledge base is a triple $\mathscr{K} = \langle \mathscr{L}, \mathscr{P}, \mathscr{Q} \rangle$, defined as follows:*

$\mathscr{L} = \langle L_N, L_A, L_E \rangle$ *(the labels).*

$\mathscr{P}$ *is a finite set of possible relabellings of the form $\langle \langle l_N, l'_N, r_1 \rangle, \mathscr{R} \rangle$ where*

> $l_N \in L_N, l'_N \in L_N, r_1 \in \mathbb{R}$ *(the node relabelling), r_1 is a real (the relevance value of the node relabelling)*
>
> $\mathscr{R} \subseteq L_E \times L_E \times \mathbb{R}$ *(the possible edge relabellings). We require that every element of $\mathscr{R}$ is of the form $\langle l.l_E, l'.l'_E, r_2 \rangle$, where $l = l_N, l' = l'_N$. r_2 is a real (the relevance value of the edge relabelling).*

$\mathscr{Q} \subseteq L_N \times L_N \times \mathbb{R}$ *(wh-phrase matching relation, with relevance value).*

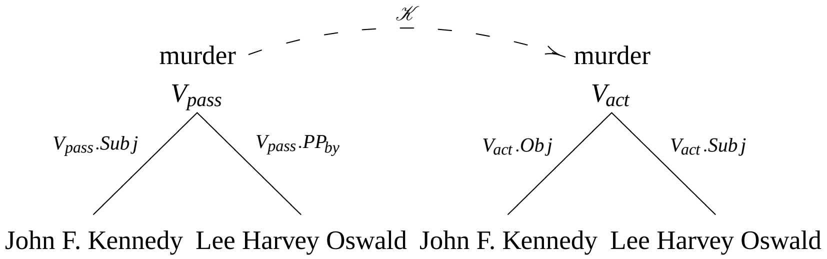
The set $\mathscr{P}$ represents the possible local relabellings: For every possible relabelling (every inference), it contains one structured element that exactly describes the relabelling as an edge relabelling and a set of relabellings of the outgoing edges; for every node relabelling and every edge relabelling, a distinct relevance value is defined.

Note that the definition of $\mathscr{R}$ ensures that all edge labels are compatible with the label of their respective mother node, both before and after relabelling.

The *wh*-phrase matching relation will be used in def. 5.7 for direct answerhood (see below).

We will now turn to the definition of the relabelling operation itself, where the relabelling step must be licenced by the linguistic knowledge base.

The relevance of a local inference step (the relabelling of a local elementary tree) is defined as the product of all relabellings (both node and edge relabellings) that are used.



Licencing rule in $\mathscr{P}$:
$\langle\langle V_{pass}, V_{act}\rangle, \{\langle V_{pass}.Subj, V_{act}.Obj\rangle, \langle V_{pass}.PP_{by}, V_{act}.Subj\rangle\}\rangle$.

Figure 5.3: Inference: From Passive to Active Sentence

Definition 5.5 (Local Inference) *Local inference is defined as relabelling of elementary subtrees:*

Let $T = \langle V, E, \mathscr{L}, P, R, C, W\rangle$, $T' = \langle V', E', \mathscr{L}, P', R', C', W'\rangle$ be text representations as defined above.

Then T' can be obtained from T by local inference with relevance $r \in \mathbb{R}$ relative to labels $\mathscr{L} = \langle L_N, L_A, L_E\rangle$ and a knowledge base $\mathscr{K} = \langle \mathscr{L}, \mathscr{P}, \mathscr{Q}\rangle$ which we write as $T \xrightarrow{r}_{\mathscr{K}} T'$, iff:

$V = V', E = E', C = C', W = W'$ and

there is a $\langle\langle l_N, l'_N, r_1\rangle, \mathscr{R}\rangle \in \mathscr{P}$ so that

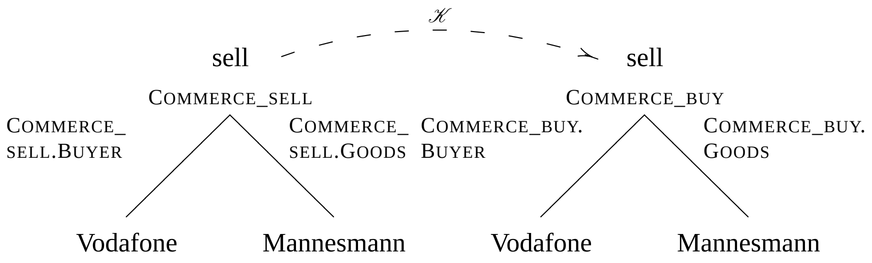
P is like P', except that exactly one $\langle u, l_N\rangle \in P$ is replaced by $\langle u', l'_N\rangle \in P', u = u'$ and

R is like R' except that for every v, $\langle u, v\rangle \in E$ there is a $\langle u, v, l_E\rangle \in R$ that is replaced by $\langle u', v', l'_E\rangle \in R', u = u', v = v'$ with $\langle l_E, l'_E, r_2\rangle \in \mathscr{R}$.

The relevance value r is computed as the product of the node relabelling r_1 and all used edge relabellings r_2.

Note that this definition requires the structure of both the ‘input’ structure and the ‘output’ structure to be exactly identical; only one elementary tree (node and outgoing edges) is relabelled.

Let us consider two examples for such a relabelling/inference. We will first give an example for the inference (or rather, equivalence) between the active



Licencing rule in $\mathscr{P}$:
$\langle\langle$COMMERCE_SELL, COMMERCE_BUY$\rangle$,
$\{\langle$COMMERCE_SELL.BUYER, COMMERCE_BUY.BUYER$\rangle$,
$\langle$COMMERCE_SELL.GOODS, COMMERCE_BUY.GOODS$\rangle\}\rangle$

Figure 5.4: FrameNet Information as Source of Inference

and passive version of a sentence (fig. 5.3). The figure shows the text representations for '*John F. Kennedy was murdered by Lee Harvey Oswald.*' and '*Lee Harvey Oswald murdered John F. Kennedy.*' If the linguistic knowledge base contains the given relations between active and passive verb nodes and the corresponding edge labels then the relabelling is licenced and thus the relation of textual inference holds between the two. No relevance values are given here (nor in the following examples) to keep them simpler. Note that in the dependency representation that we actually use (the PREDS), this relabelling between active and passive would not be needed, as we normalise over active and passive during parsing. We use this example to show that such a relabelling can be very simply defined (6.2).

The second example (fig. 5.4) shows how an underlying inverse relation between two lexical semantic predicates translates into textual inference between two text representations (and therefore also between the corresponding texts) through FrameNet information: Here, the knowledge base contains the information that FrameNet COMMERCE_SELL and FrameNet COMMERCE_BUY are related, as are the corresponding frame elements (semantic roles) of BUYER and GOODS. This would permit to find that '*Mannesmann was sold to Vodafone.*' provides an answer for '*Who bought Mannesmann?*', as the frame representations can be mapped onto each other.

Note that the relabelling operation is limited to one linguistic level: The word stem *sell* of the syntactic level is not changed in this example when the

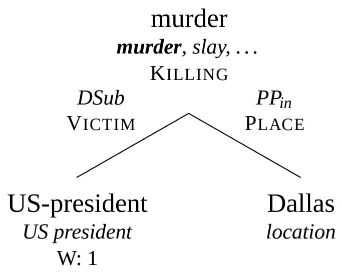


Figure 5.5: Text representation for the question *'Which US-president was killed in Dallas?'*

frame label changes. While this means that different representation levels may become, in a sense, incompatible during inferencing (for example, *sell* has no direct relation with the FrameNet frame COMMERCE_BUY), this independence of levels is an important advantage of this definition: It allows to find mappings based on local inferences on one level only.

As before, we can now define textual inference simply as a finite series of inference steps.

Definition 5.6 (Textual Inference) *The textual inference relation holds between two texts τ and τ^* relative to a given knowledge base $\mathscr{K}$ and a parsing relation f iff the following relation holds between any two of their text representations $\exists T, f(\tau, T)$ and $\exists T^*, f(\tau^*, T^*)$:*
There is a series of local inference steps $T \xrightarrow{r'}_{\mathscr{K}} T' \ldots \xrightarrow{r^*}_{\mathscr{K}} T^*$. $r = r' \times \ldots \times r^*$.

The relevance value of a textual inference relation is computed as the product of those of the different inference steps.

5.1.4 Direct Answerhood

Questions are considered as texts, they can therefore receive text representations as defined above. However, we need to make several additional assumptions regarding the representation of *wh*-phrases in questions.

A *wh*-question is represented by a text representation T as defined above, with the following changes (cf. also 3.5.2.5), some of them shown in fig. 5.5.

Marking *Wh*-phrases. The node $u \in V$ representing the head of the *wh*-phrase is marked, $W(u) = 1$; $W(v) = 0$ for all other nodes $v \in V, v \neq u$ (cf. def. 5.2). In fig. 5.5, the phrase *'which US-president'* is accordingly marked as a question phrase.

Question pronouns. *Wh*-phrases consisting of a single question word (such as *'who'*) are labelled with a suitable sortal predicate $l_N \in L_N$, e. g., *person* for the question pronoun *'who'*.

***Which/what Common Noun* phrases.** The node $u \in V$ representing the Common Noun is marked with $W(u) = 1$. The question determiner *'which'*/*'what'* is *not* represented by a node in V. In fig. 5.5, the phrase *'which US-president'* is marked as a question phrase, while the question word *'which'* is not further represented. The label is marked by the expected semantic type of the answer (viz. *US president*).

***How Adjective* phrases.** The node $u \in V$ representing Adjective is marked with $W(u) = 1$. It is further labelled with a suitable predicate $l_N \in L_N$, e. g., *length* for the question phrase *'how long'*. This suitable predicate is that predicate in the hierarchy defined by $\mathscr{Q}$ that is the most specific predicate subsuming the labels of the heads of all possible corresponding answer phrases. The question word *'how'* is *not* represented by a node in V. Marking the head is sufficient to mark it as a *wh*-phrase, the syntactic marker is not required.

Now we can turn to the definition of direct answerhood. We define the relation of direct answerhood to hold between a text representation and a question representation when the question can be 'embedded' into the text (i. e., when it forms a subtree of the text representation) and when all phrases that are marked as *wh*-phrases in the question have a corresponding answering phrase whose head is subsumed semantically by the head of the *wh*-phrase.

We will start with the ancillary definition of subtree matching: Given a question and an answer representation and a root node for each of them it recursively checks whether the question representation can be mapped onto the answer representation. The check starts with the root node.

The following cases must be distinguished: If a node u' in the question representation is marked as a *wh*-node, then it matches only nodes in the answer representation that are labelled by a synonym or hyponym of at least one label of u' with respect to the $\mathscr{Q}$ relation. Nodes in the question representation that are not marked as a *wh*-node match nodes in the answer representation if the two nodes have at least one identical label.

For each node in the question representation, all outgoing edges must map onto one in the answer representation. This mapping is possible if the edge labels are identical and if the child nodes can be matched. The child nodes match if one of the two following conditions holds: Either, the subtrees rooted in the respective nodes reached via the edge match directly (recursion step), or, by following a coreference link in the answer representation, a different, but coreferent subtree can be found that matches the question subtree.

From the definition of relevance values for each element of $\mathscr{Q}$ the relevance values of the matching of question and answer is computed.

Definition 5.7 (Subtree Matching) *Given two text representations* $T = \langle V, E, \mathscr{L}, P, R, C, W\rangle$ *(answer representation) and* $T' = \langle V', E', \mathscr{L}, P', R', C', W'\rangle$ *(question representation), two subtrees rooted in* u *and* u'*, respectively, match with relevance value* $r \in \mathbb{R}$ *relative to labels* $\mathscr{L}$*,* $\mathscr{L} = \langle L_N, L_A, L_E\rangle$ *and a knowledge base* $\mathscr{K} = \langle \mathscr{L}, \mathscr{P}, \mathscr{Q}\rangle$ *iff*

$$\exists \langle u, l_N\rangle \in P, \exists \langle u', l'_N\rangle \in P', \begin{cases} \langle l_N, l'_N, r_1\rangle \in \mathscr{Q}, \text{ if } W(u') = 1 \\ l_N = l'_N, \text{ else} \end{cases} \quad \text{and}$$

$\forall v', \langle u', v'\rangle \in E'$*:*

$\exists v \in V$ *such that* $\langle u, v\rangle \in E, \langle u', v', l'_N.l'_E\rangle \in R', \langle u, v, l_N.l_E\rangle \in R, l'_E = l_E$ *and*

either the subtrees rooted in v *and* v' *match with relevance* r_2 *or*

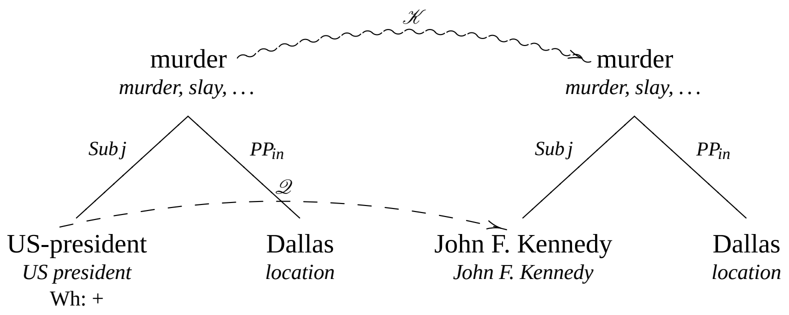
$\exists w \in V, \langle w, v\rangle \in C$ *such that the subtrees rooted in* w *and* v' *match with relevance* r_2*.*

r is computed as the product of r_1 (if defined) and the relevance values for all subtree matches r_2.

Now the definition of Answerhood becomes quite straightforward: The relation holds exactly if, for the single root node in the question representation, some node in the answer representation can be found so that the respective subtrees match as just defined.

Definition 5.8 (Direct Answerhood) *The direct answerhood relation* $T \overset{r}{\rightsquigarrow}_{\mathscr{K}} T'$ *with relevance* $r \in \mathbb{R}$ *relative to labels* $\mathscr{L}$*,* $\mathscr{L} = \langle L_N, L_A, L_E\rangle$ *and to a knowledge base* $\mathscr{K} = \langle \mathscr{L}, \mathscr{P}, \mathscr{Q}\rangle$ *holds between two text representations* $T = \langle V, E, \mathscr{L}, P, R, C, W\rangle$ *(answer representation) and* $T' = \langle V', E', \mathscr{L}, P', R', C', W'\rangle$ *(question representation) iff*

$v' \in V'$ *is the single root in* T' *and*



Licencing rule in $\mathscr{Q}$: $\langle US\,president, John\,F.\,Kennedy\rangle$.

Figure 5.6: Matching the representations for *'Which US-president was murdered in Dallas?'* and *'John F. Kennedy was murdered in Dallas.'*

there is a $v \in V$ such that the subtrees rooted in v and in v' match for T and T' with relevance r.

An example for such a match is shown in fig. 5.6. Note that matching is licenced through a rule in the knowledge base that defines *John F. Kennedy* to be a hyponym of *US president*; all other nodes and edges match due to equality of labels.

The following definition connects the relations of direct answerhood between text representations with that between two texts (via the parser).

Definition 5.9 (Direct Answerhood (Texts)) *The relation of direct answerhood holds between two texts τ and τ' with relevance value $r \in \mathbb{R}$ relative to a given knowledge base $\mathscr{K}$ and a parsing relation f iff the following relation holds between any two of their text representations $\exists T, f(\tau, T)$ and $\exists T', f(\tau', T')$, $T \overset{r}{\rightsquigarrow}_{\mathscr{K}} T'$.*

5.1.5 Answer Checking

We have argued above that the definition of direct answerhood includes answers with inverse polarity (3.5.1.2) and uncertain answers (3.5.1.3). We will propose virtually identical methods for identifying these types of answers in this section. For the purposes of this section, we will subsume both types under the term 'special answers'.

We do not want to discard special answers since they are (probably) relevant to the question. To reliably distinguish these special answers from others, deep linguistic processing would be required. As a first approximation, we identify answers that potentially fall into the category of special answers. In the system implementation, we present them to the user with a suitable warning.

Identification of special answers is done in an additional answer checking step. It checks the presence of negations and 'dangerous' material, respectively, in the representations of potential answers. Whenever a potential answer contains such material, it will be presented to the user only with a suitable message. We will introduce the method for answer checking first and then give an overview of the phenomena that need to be accounted for, as well as the associated 'lexical triggers'.

The following definition gives the proposed method for identifying answers with inverse polarity and uncertain answers. As the method is virtually identical for both types and only differs through the use of a set of negations and a set of 'potentially dangerous predicates', respectively, we have summarised the definition for both types.

Definition 5.10 (Answers with Inverse Polarity, Uncertain Answers)
Let $\mathscr{L} = \langle L_N, L_A, L_E \rangle$ be labels as defined above. Then τ' is an answer with inverse polarity (an uncertain answer) relative to a set of negations $N \subseteq L_N$ ('potentially dangerous predicates' $D \subseteq L_N$) iff

$\exists T, f(\tau, T)$ and $\exists T', f(\tau', T')$ so that the direct answerhood relation $T \rightsquigarrow_{\mathscr{K}} T'$ relative to the knowledge base $\mathscr{K}$ as defined above holds between two text representations $T = \langle V, E, \mathscr{L}, P, R, C, W \rangle$ (answer representation) and $T' = \langle V', E', L'_N, L'_E, P', R', C', W' \rangle$ (question representation) and

$\exists u' \in V', u \in V$ such that u' corresponds to u as in the definition of direct answerhood above and

$G = \langle V'', E'', \mathscr{L}, P'', R'', C'', W'' \rangle$ is the text representation containing u, with $V'' \subseteq V, E'' \subseteq E, P'' = P \cap V'' \times L_N, R'' = R \cap V'' \times V'' \times L_E, C'' = \emptyset, \forall u'' \in V'' : W''(u'') = 0$, V'' and E'' form the maximal tree containing u and

$\exists v'' \in V'', \exists l''_N \in N(\exists l''_N \in D) : \langle v'', l''_N \rangle \in V$ and $\neg \exists v' \in V', v'' \in V''$, so that v' corresponds to v'' through answerhood.

This check allows us to identify special answers, i. e., answers where negations or potentially 'dangerous' material, respectively, exist somewhere within the tree containing the answer. First, the tree in the text representation corresponding to the answer sentence (G) is isolated. Recall, that in general a text representation consists of a forest, with one tree representing a sentence. Then, this tree is searched for the existence of a negation or potentially dangerous predicate, respectively (defined as sets of node labels). Whenever such a predicate is found, the whole answer is marked as answer with inverse polarity or uncertain answer, respectively. Note that the predicate is not considered if it is contained both in the question and the answer: A negated question and a similarly negated answer would not be marked as answer with inverse polarity, for example.

Answer checking could be further refined. The definition above does not attempt to compute an 'overall polarity': Different 'dangerous' expressions will, in general, interact with each other. From *'There was never any doubt that Lee Harvey Oswald killed John F. Kennedy.'*, for example, it can be inferred that Lee Harvey Oswald killed John F. Kennedy, even though *'never'* is a negation and *'doubt that'* is considered potentially dangerous. In Nairn et al. (2006); Bobrow et al. (2005), an implication projection algorithm is suggested, i. e., a method for computing whether entailment holds for more complex sentences.

We will now give an overview of the phenomena that need to be handled and give examples of associated 'trigger words'. For a discussion of the different phenomena, see also Nairn et al. (2006); Karttunen and Zaenen (2005); Bobrow et al. (2005); Zaenen et al. (2005); Haghighi et al. (2005); Burchardt and Frank (2006)

Negation. If the sentence containing the answer is negated (and the question is not), it will be marked as answer with inverse polarity. Typical negation-introducing words include *'nicht'* (*not*), *'kein'* (*no, no-one*), *'nichts'* (*nothing*), but also adverbials such as *'nie(mals)'* (*never*), *'kaum'* (*hardly*), *'wenig'* (*little*).

Opaque Contexts. A number of expressions introduce 'opaque contexts', i. e., contexts that do not allow inferences about the facticity of the embedded expressions. These especially include expressions embedding reported speech, introduced, for example by verbs like *'sagen'* (*say*), *'berichten'* (*report*), *'behaupten'* (*mention*), nouns like *'Aussage'* (*statement*) and *'Behauptung'* (*assertion*) and a number of others, expressions reporting believes and attitudes like *'denken'* (*think*) and *'halten für'* (*consider*) and others, such as *'suchen'* (*search*) or *'planen'* (*plan*).

Modal Verbs. Modal verbs, such as *'wollen'* (*want to*), *'können'* (*can*) and *'dürfen'* (*may*), also need to be flagged. As we require them to be collapsed with the main verb (5.1.2), these need to be treated slightly different in answer checking, namely as a check whether or not the corresponding feature is present for a verb node or not.

This short overview already shows that several different expressions whose presence may indicate a special answer. We used the above expressions as ‘seeds’ and then extracted related words from GermaNet. This gave us a list of about 2 000 negations and potentially dangerous words.

During our experiments with SQUIGGLI and during evaluation we found that only a small number of answers were actually uncertain answers: Only a handful of cases were marked during evaluation (cf. 7.2.1.4). If this observation generalises for other data then uncertain answers are not very frequent in QA. In contrast, corresponding example sentences form an important part of the data in the Recognising Textual Entailment challenges. This suggests that the problem occurs less often in practical QA systems than might be expected from the RTE data.

5.2 The Linguistic Knowledge-Base

We will now relate the abstract definition of the indirect answerhood relation given in the last section more concretely to the different phenomena that we described as relevant for indirect answerhood (3.5) and show how those can be accommodated. This involves both questions of representation and normalisation, on the one hand, and questions of defining inference rules, i. e., instantiating the linguistic knowledge base, on the other hand.

We will start by looking at a number of related approaches for matching text representations, most of them defined in the context of the RTE challenge (5.2.1).

We will then take a closer look at the different linguistic levels of the text representations, describe what normalisations they use and – most importantly – what inference rules are defined for them and how the linguistic knowledge represented in the lexical resources GermaNet and FrameNet (4.2, 4.3) can be transferred into suitable relabelling relations licencing textual inferences (5.2.2).

5.2.1 Related Work: Matching Graph-Based Text Representations

Research on approaches to Information Access using structures based on linguistic representations has increased over the last years. It was at least partly spurred by interest in the idea of the Semantic Web, i. e., an additional ‘semantic’ annotation level that is to be added to Internet pages and allows intelligent agents to harvest information directly (Berners-Lee et al., 2001).

Recently, several approaches to Information Access based on matching graphs representing linguistic structures have been proposed. Especially the Recognising Textual Entailment challenge (RTE, cf. 3.4) has seen a number of approaches that model the task of textual inference by matching linguistic representations of text and hypothesis.

We will summarise several approaches, both from the RTE and from other sources, here. A number of additional approaches can be found in Magnini and Dagan (2006).

We will be mainly interested in three issues, namely what sorts of linguistic knowledge have been utilised by the different approaches (and how), how the matching process itself is defined and how efficiently it can be implemented.

5.2.1.1 Matching Semantic Networks Using Query Expansion for Question Answering

One approach to QA that is quite similar to ours is described in Hartrumpf (2004). The described system took part in the 2004 German QA@CLEF challenge.

In this approach, first the whole document collection is parsed to derive a lexical semantic representation in a graph format called MultiNet (Helbig and Gnörlich, 2002; Hartrumpf, 2001). This combines dependency-style structures containing semantic predicates as node labels and semantic roles (such as AGENT or LOCATION) as edge labels. These graphs are stored in a way that enables efficient search.

For questions input to the system, a similar MultiNet structure is derived. A number of 'expanded' versions are generated from this representation. This is done by applying a set of equivalence and inference rules based on a proprietary lexical resource called HagenLex, which contains MultiNet meaning descriptions and lexical semantic relations for a number of German concepts (Hartrumpf, 2003).

An example rule given in the paper postulates, for example, that if an event causes another event, the two events are temporally related. Thus, if a question to the system contains a reference to two events that are temporally related, the (reverse) application of the inference rule produces a version of the question where the two events are related by a causal (instead of a temporal) link.

Searching in the document collection is done for all expanded forms of the question. First, the network representations are broken down into dependency triples, labelled with the head concept, the daughter concept and the edge label for all dependency relations in the question representation. If all these triples can be matched in the graph representation of a sentence in the document collection, the answer node can be identified and a suitable answer can be generated.

Thus, the approach is quite similar to ours, as it also uses structured representations of the texts in the document collection and the questions. The structures contain dependency information and lexical semantic information, and matching is defined via a set of equivalence and inference rules.

Our approach differs in that it combines information from different sources (namely dependency structures, GermaNet and FrameNet) and query expansion and search are conducted in an interleaved fashion (see below, cf. 5.3.2.2).

5.2.1.2 Approaches Based on Approximate Tree Matching

Several approaches have been defined that are based upon approximate tree matching using edit operations (Zhang and Shasha, 1997). Matching two trees is defined as a series of editing operations that transform the first tree into the second. Most algorithms use the three operations of inserting, deleting and relabelling a node. These operations are generally associated with different costs; different algorithms exist for efficiently finding the 'cheapest' sequence of editing operations that allows matching the two trees. Most approaches define a – relative or absolute – threshold for the overall editing costs: Matches whose cost lies below the threshold are considered as 'good' matches, other matches are disregarded.

Dependency Tree Mapping for Question Answering. A syntax-oriented approach using tree matching to check potential answers in QA is described in Punyakanok et al. (2007). The authors use syntactic dependency structures (encoded as trees) without semantic information as a basis. Nodes in the tree are labelled with lemmata, while edges remain unlabelled. They state that their approach is to be used in a 'standard' QA system as a means of checking potential answer candidates as returned by an IR module through comparing the representations of question and answer candidates for structural similarity.

Structural similarity is defined through ordered tree matching. The authors report that the algorithm for finding the 'cheapest' transformation of one tree into the other is efficient, with its complexity bounded by the square of the product of the sizes of the two input trees (Punyakanok et al., 2007, 5). Note that the approach is not defined as a means of searching for potential answers, but only for checking answers found by other modules.

The authors also report the results of an experiment using the TREC 2002 QA track data (pairs of questions and answers), where the tree mapping method leads to an improvement in accuracy of 30–40 % when compared with a pure bag of words measure for computing similarity. This ties nicely with the intuition that checking for structural similarity in addition to pure word-overlap based similarity can help to improve accuracy.

It should be noted that this approach uses little linguistic knowledge: Relabelling (and thus replacing words) in the structures is not controlled by any measure of similarity; rather, a uniform cost is assigned to replacing one word with another (though the cost is defined to differ for content words and function words). Grammatical functions are not regarded at all, as the dependency trees are not edge-labelled.

Dependency Tree Mapping for RTE. A similar approach (in the context of the RTE challenge (3.4) is described in Kouylekov and Magnini (2006, 2005). The authors use tree matching of dependency structures for both text and hypothesis as a basis for detecting textual entailment.

To be able to use edge labels during tree matching, these were integrated into the node labels, as the standard tree matching algorithms do not support edge labels.

The authors experimented with different edit cost values: Costs for insertion were mostly based on the relevance of the inserted word (based on its inverse document frequency value computed over large corpora; idf, cf. 2.1.1), costs for substitution were a constant in cases where entailment between the word and its substitute could be found in WordNet (that is, for synonyms, hypernyms, pertainyms and entailed words) and infinite otherwise, while the cost for deletion was always set to zero. Different cost functions were combined using a machine learning approach.

5.2.1.3 Concept Graph Subsumption for Modelling Inferences

In the 2005 RTE challenge (3.4), the system of one participating group used a subsumption check on multi-layered text representations (called concept graphs) for modelling inferences (de Salvo Braz et al., 2005).

Concept graphs contain a phrase structure representation of the input text together with a lexical semantic annotation based on PropBank (Palmer et al., 2005). In order to check whether a given hypothesis can be inferred from a given text, concept graphs for both are derived. By using a set of manually defined rewrite rules, together with a larger number of paraphrase rules that were automatically acquired from large corpora (Lin and Pantel, 2001b) and hypernymy information from WordNet, the system tries to transform the text representation into the hypothesis representation. If this transformation succeeds it is concluded that the hypothesis can be inferred from the text. The rules that were used are not further specified.

The authors report their implementation to be efficient, even though they state that the underlying problem is NP-complete (de Salvo Braz et al., 2005, 4). They attribute this to the efficiency of the commercial software package that they employ for the subsumption check.

In the RTE evaluation, the system was in the mid-field of all participating systems. In an error analysis, the authors found most problems to be due to missing inference (rewrite) rules in their knowledge base.

5.2.1.4 Graph Matching for Modelling Inferences

Another approach to model the RTE problem is given in Haghighi et al. (2005). This approach uses comparatively rich graph representations for text and hypothesis as a basis for graph matching. The graphs represent dependency structures, additionally labelled with WordNet information and semantic role labels based on PropBank (Palmer et al., 2005). Additional links denote coreference relations. Thus, the structures are quite similar to the ones we use.

Matching of these structures is defined as graph matching based on cost functions for vertex matching and path matching. A number of different measures go into the different functions; the optimal weighting of the underlying features is learned using a machine learner. Costs for vertex matching are based on whether the two words are identical (same word form), whether they have the same lemma, whether they are WordNet synonyms or hyponyms, combined with a general WordNet similarity (cf. 5.2.2.2) and similarity based on a Latent Semantic Analysis measure. Costs for edge matching differ for exact matches, partial matches (of labels) and cases where a path of length greater than one must be collapsed into a path of length one to allow a match. The authors state that the graph match can be computed 'efficiently'.

In addition to the graph match, a check for possible 'non-entailment' is performed: Similar to the answer checking step (5.1.5), several phenomena such as negation and embedding in opaque contexts are checked and matching structures that contain 'dangerous' material are filtered.

Thus, the overall approach is quite similar to ours in several respects. However, it does not use a notion of local inference based on a licencing rule that controls node and edge relabelling like ours, it rather uses a more general similarity match.

5.2.1.5 Partial Matching of Graphs with LFG and FrameNet Information

Another approach based on the partial matching of linguistic graph structures to the RTE challenge is described in Burchardt and Frank (2006).

The authors derive graphs containing different levels of linguistic information, namely Lexical Functional Grammar structures (LFG, Bresnan, 2001), disambiguated WordNet predicates, FrameNet annotation, and SUMO predicates. Partial graph matches are searched for graphs of text and hypothesis: Node matching is defined for nodes labelled with the same or semantically related predicates (from one of the different lexical sources), edges are permitted to match if any of the edge labels (grammatical functions, frame elements etc.) matches.

From these partial matches and a number of additional dissimilarity matches (mainly identifying 'dangerous' material, such as negation, modals etc.), features are derived (such as the proportion of matched nodes) and fed into a machine learner to optimally combine them.

This approach uses a wide range of linguistic information (syntax structures and different lexical semantic information). Though connected graphs are preferred, the current approach also allows partial matches only.

5.2.1.6 Conceptual Graph Matching

In Montes-y-Gómez et al. (2001), an algorithm for matching so-called conceptual graphs is presented. The paper starts with the introduction of conceptual graphs: They can be seen as dependency structures, where vertices (nodes) are labelled with content word lemmata, which are regarded as semantic concepts, and edges are labelled with semantic roles such as AGENT or PATIENT. Both the concepts and the relations are grouped by two distinct subsumption relations, forming two hierarchies. The construction of these hierarchies or their derivation from existing resources is not described in the paper.

Matching two conceptual graphs is done by finding a third graph that subsumes both of them, which amounts to computing the overlap between the two. This is controlled by a similarity measure defined as two relations between two conceptual graphs based on the Dice coefficient, one for concept overlap and one for overlap of edges emanating from common concepts (for details, see Montes-y-Gómez et al., 2001, 107–109). Thus, the matching process differs from the one we use in that it does not require matching graphs to be connected: An overall similarity is computed based on how many elementary trees can be matched and with what respective similarity.

The matching process is described as closely related to the problem of establishing subgraph isomorphism (for two given graphs, establish whether one can be embedded in the other, cf. 5.3.1.2) and thus NP-complete (Garey and Johnson, 1979, 202). However, the authors claim that this is not a practical limitation of their approach (Montes-y-Gómez et al., 2001, 106–107). The authors say that their method would be applicable to Information Retrieval, but do not describe how their algorithm may be extended into a search algorithm that could actually be used for retrieving information from document collections.

5.2.1.7 Conclusions

We have shown that there are quite a number of different approaches utilising matching of linguistic tree (or graph) structures. Most of them have been developed recently for the RTE challenge. We will now look at differences from and similarities to our approach.

Linguistic Resources. The approaches described above differ quite substantially in the amount of linguistic information that is used during matching: All approaches use syntactic dependency structures derived from the texts to be compared. However, while a number of approaches solely rely on the similarity of the dependency structures (especially approaches that use tree edit distances with uniform costs, e. g., Punyakanok et al., 2007), others make use of additional information from different linguistic sources (e. g., de Salvo Braz et al., 2005).

A number of approaches employ manually written rules for mapping syntactically equivalent structures onto each other. Unfortunately, none of the descriptions gives a complete overview, but the most commonly cited examples include active/passive diathesis, possessive and PP_{of} constructions and predicative vs. appositive constructions (*‘A is the B’* vs. *‘A, the B, . . . ’*).

Information from WordNet is used by quite a number of systems, most often restricted to information about synonymy, to allow what is sometimes called a ‘semantic matching’ (e. g., Marsi et al., 2006; de Salvo Braz et al., 2005).

Only a handful of the approaches use other lexical semantic resources that provide information on role labelling, such as PropBank or FrameNet, namely Burchardt and Frank (2006); Haghighi et al. (2005); de Salvo Braz et al. (2005).

Another source of information that has sometimes been employed is provided by automatically acquired paraphrases (de Salvo Braz et al., 2005). These seem to provide quite a rich source of additional information, as they can capture likely local inferences that are, nevertheless, not commonly listed in manually constructed lexical resources.

Not all of the described systems address the problem of non-factive contexts. Semantic entailment does not generally hold between text and hypothesis (or question) if the tree (or graph) representation of the hypothesis can be embedded into that of the text: Phenomena such as negation or embedding in semantically opaque contexts cannot directly be handled by such an approach. Approaches that address the issue mostly use additional steps to identify such 'uncertain' inferences and then discard them (e. g., Burchardt and Frank, 2006; Haghighi et al., 2005).

Algorithms Used for Matching. First, it should be noted that almost all of the approaches described here try to solve the problem of textual inference (a few that of answerhood) for two given texts. Of course, this can usefully be employed in a 'standard' QA system for answer extraction from a (small) set of documents containing answer candidates, as returned by an Information Retrieval engine (cf. 2.2.2). Only the approach described in Hartrumpf (2004) is used for searching answers in large text collections.

Accordingly, techniques employed for finding matches quite often are computationally expensive (up to NP-complete algorithms). For relatively small representations of texts and hypotheses, this will generally not pose a problem. However, as soon as the approach were to be used for answer searching in text collections, it would turn into a serious efficiency problem. This high complexity stems from the fact that several algorithms used for matching are rather unconstrained.

In most cases, tree structure matching can be done more efficiently than matching of general graph structures. Ordered matching problems can be solved at least as efficiently as those of unordered structures; sometimes the ordered version of a problem is efficiently solvable while the unordered version is intractable.

Another way in which the used algorithms differ is whether or not they use parenthood in the pattern structure as a constraint. If that constraint is not used at all or only the weaker constraint of ancestry (comparable to considering the edges in the pattern as 'rubber bands' that can be stretched during matching), algorithms with higher complexities are required. Kilpeläinen (1992) provides an overview of different tree matching algorithms; several graph matching problems are described in Diestel (2005); Garey and Johnson (1979).

Besides the trade-off in efficiency (using more constraints during searching makes it possible to use more efficient algorithms), we also expect a general trade-off between precision and recall: Using a less-constrained algorithm for matching will generally increase recall, as it allows very dissimilar structures to match. On the other hand, using a type of matching that is little constrained will, in general, result in a loss of precision.

We use an algorithm based on tree matching through unordered path inclusion. This will be described in more detail below (5.3.2.2). Thus, we use a relatively constrained algorithm as the basis for searching. In doing so, we aim for high efficiency and high precision.

5.2.2 Linguistic Knowledge and Inference

In this section, we will further specify text representations and describe an instantiation of the linguistic knowledge base for German. We will give an overview of how the different linguistic levels interact and how information from different linguistic resources contributes concretely to instantiate a model of indirect answerhood for German.

First, we will take a closer look at two important assumptions that we have built into the definition of indirect answerhood, namely that of structural uniformity of different linguistic levels and how they can be utilised to increase robustness (5.2.2.1).

Next, we will add some remarks on how we have instantiated relevance values in general, namely by manually set, generic values (5.2.2.2).

We will then describe, for the different linguistic levels, the sorts of inferences that are supported at each level, and the sources from which this information is drawn, one by one, namely information from dependency structures (5.2.2.3), from named entities (5.2.2.4), information on lexical concepts from GermaNet (5.2.2.5), frame semantic information (5.2.2.6), information from additional semantic relations (5.2.2.7) and from coreferences and predicatives (5.2.2.8).

We will walk through a small example that shows how different linguistic levels can be combined during the matching process (5.2.2.9).

Note that linguistic information is used for the matching process in two principal ways. On the one hand, variants are normalised. For example, two sentences that differ only in word order will receive the same representations and thus be treated as equivalent for matching purposes. On the other hand, textual inference rules in the linguistic knowledge base let us systematically relate different structures to each other. For example, two sentences may be recognised as equivalent if they only differ by the substitution of a word by a synonym.

This section shows in an exemplary fashion how information from different linguistic resources can be integrated. The integration of information happens in a modular fashion: As long as the information can be represented as local relabelling rules (for textual inference) or matching rules (for direct answerhood), it can be directly incorporated. This modularity allows to easily tap new sources of information (such as information on paraphrases, cf. 8.3.1.2). In principle, it would also allow to port the approach to languages other than German.

5.2.2.1 Structural Uniformity and Robustness

We will start by shortly recapitulating and elaborating on two important properties of our definition of indirect answerhood: First, the definition assumes structural uniformity of the different linguistic levels. Second, a certain degree of robustness is built into the approach, as information from different linguistic sources (such as syntactic dependency and lexical semantic frame information) can be integrated into both the text and question representations and the overall matching process independently from each other.

Structural Uniformity. In the definition of indirect answerhood (5.1), we have presupposed that a syntactic dependency structure provides the complete structural skeleton and that lexical semantic information can be added as a supplementary layer of labelling. We will now re-appraise this assumption more critically.

Put in a nutshell, this assumption means that syntactic and lexical semantic structures will be essentially uniform: The syntactic head of a partial syntactic structure will also be the semantic head of the corresponding semantic structure; dependents of this head in the syntactic structure will also be dependents of the semantic head in the corresponding structure. This is in accord with Tesnière's original description of dependency structures (cf. Tesnière, 1980, chs. 20, 21).

Intuitively, the assumption makes sense: At the lexical semantic level, the predicate-argument structures corresponds to the head-dependent structure of the syntactic level.

For semantic representations in predicate logic, this structural uniformity would no longer hold: Syntactic heads and dependents and functors and arguments at the semantic level must not necessarily correspond to each other. For example, while we consider determiners and expressions like *'all'* as dependents at the syntactic level, they will be represented as quantifiers at the semantic level, which will be lifted to take scope over the whole expression. Similar differences may arise from the treatment of modal verbs (which we collapse with the main verb) or restrictive adjectives.

It may be necessary to weaken the assumption of structural uniformity when more complex structures are represented. However, as the assumption is built into the search algorithm that we use, a different, more complex search algorithm would then need to be used.

Robustness. We have defined local textual inferences and direct answerhood as relabelling and matching of elementary trees, respectively. Because the domains of the operations are elementary trees, individual relabelling and matching operations are completely independent from each other. The individual operations can therefore make use of information from different linguistic levels: The relabelling of one elementary tree can, e. g., be based on syntactic information, while the relabelling of another elementary tree within the same overall structure can be based on frame-semantic information.

In consequence, information must not be present for each level for different elementary trees: It is sufficient if one level is present and can be operated on.

Discussion. This combination of independence of the different linguistic levels and the locality of the operations leads to an increase in robustness: Information from different sources and at different levels can be combined. Missing information at one level does not mean that no overall match can be found: Neither of two text representations must be complete and correct at all levels.

For a practical system implementation, it is important that pieces of information from different levels combine and that they complement each other: Natural language processing systems will often not come up with complete structures.

We will slightly change the definition later (5.2.2.9) based on the assumption that an answer is more relevant if a match can be found for an elementary subtree at more than one linguistic level (for example, a match can be found both on the WordNet level and on the frame level). This change does not change the matches that are found, but only their relative relevance.

5.2.2.2 Instantiating Relevance Values

In 5.1, we have introduced relevance values; each relabelling and matching operation is associated with a relevance value. We will now describe which relevance values are assigned to different inference and matching rules.

The definition of indirect answerhood through relabelling allows to assign fine-grained individual relevance values to each relabelling – not only to the node relabellings, but also to the individual associated edge relabellings: It is, for example, possible to distinguish between relabellings of PP arguments with different prepositions depending on the node label.

Different relatedness and similarity measures, based either on lexical hierarchies like WordNet or on corpus studies have been proposed. We expect such measures to provide a good source of fine-grained relevance values (assuming that relatedness or similarity can be used to model relevance). We will give an overview of the most important measures proposed for English. Currently, no comparable data is available for German, so we use simpler generic values. We will describe this in more detail below.

Related Work: Word Similarity Measures. Similarity measures (or, more generally, relatedness) between linguistic predicates, especially for word senses in WordNet, has been intensively studied, and a number of different measures have been suggested.

In a recent paper, Alexander Budanitsky and Graeme Hirst have listed a number of the most important different measures of lexical semantic relatedness and evaluate their appropriateness for one particular task, namely the correction of malapropisms[1] (Budanitsky and Hirst, 2006). We will shortly review the most important approaches, mostly following the overview in Budanitsky and Hirst (2006).

Measures Based on Taxonomic Distance. Several methods have been suggested that directly derive a measure of relatedness of two concepts from a given taxonomy (hierarchy) of concepts (e. g., Leacock and Chodorow, 1998; Wu and Palmer, 1994). The simplest measure just uses the length of the shortest path between the two concepts in the hierarchy. More advanced measures use additional factors (such as the position of the concepts in the hierarchy; this is based on the observation that natural language-based hierarchies tend to be conceptually 'denser' towards the lower end). Most approaches make only use of the hyponymy-relation in WordNet, while some also employ other relations such as antonymy or meronymy (resulting in relatedness measures rather than similarity measures). For an overview see Budanitsky and Hirst (2006, 16–19) and also Patwardhan et al. (2003).

Information-content-based Approaches. A second set of approaches makes use of information extracted from corpora (see, e. g., Lin, 1998b; Jiang and Conrath, 1997; Resnik, 1995). The core idea is that the relatedness of two concepts can be defined as the extent to which they share information. This can be interpreted as the probability of encountering the most specific common subsumer of both concepts, employing a definition from information theory. These probabilities can be induced from sufficiently large corpora. See Budanitsky and Hirst (2006, 19–22) for an overview; Lin (1998b) for a discussion of the information-theoretic basis.

In two different evaluations (Budanitsky and Hirst, 2006; Patwardhan et al., 2003), a measure that combines information-content elements with a normalisation based on the underlying lexical hierarchy (Jiang and Conrath, 1997) generally performed as least as well as the other measures.

[1]Malapropisms are spelling errors that lead to real words that are semantically inappropriate, e. g., *'I made an entry in my dairy.'*, with *'dairy'* in lieu of *'diary'*. For an overview of algorithms for handling spelling error, see Fliedner (2004c, 2006a).

Relevance Values Based on Relation Type. The above summarisation shows that similarity (or relatedness) measures can be derived either from lexical hierarchies directly or – with generally better results – induced from large corpora.

We found that, while such data is readily available for the English WordNet hierarchy (cf. Pedersen et al., 2004), no comparable resource exists for either GermaNet or FrameNet (let alone German FrameNet).

We currently use manually set generic relevance values. For example, all synonymous expressions are currently assigned a relevance value of 1.0 (perfect match). We will describe the relevance values that we have used for the different relation types below.

These manually set relevance values could be replaced by empirically justified values, especially by values induced from corpora in the future. Note that all that is necessary for utilising this sort of knowledge is already present in the definition of indirect answerhood so that the information could be easily integrated.

While we think that experimenting with corpus-induced, finer-grained measures to instantiate the relevance values would be interesting, we would this not to substantially change the results but to rather have the effect of fine tuning, that is, help to differentiate between answer candidates that would tie for relevance with the current instantiations.

Restricting the Range. In 5.1, we have defined the relevance values as rational numbers. We have decided to further restrict the range and use only basic values between 0 and 1, where 0 indicates no relevance and 1 the highest possible relevance (perfect match).

The advantage of this restriction is that all relevance values will keep within this range, as only multiplication is used to combine relevance values. This makes direct comparison of the relevance values for matching different structures easy.

Used Relevance Values. We currently differentiate mainly between inference relations that do not change the underlying meaning and others, where the related representations are similar, but differ in meaning.

The former are assigned a relevance value of 1.0 to indicate equivalence. The relations include the following ones: Matches between identical lemmata and grammatical functions at the syntactic level, relations between GermaNet synonyms and between identical frame structures.

The latter (i e., similarity matches) are assigned a basic relevance value of 0.9. This applies for the following cases: Matches between different, but similar grammatical relations, matches between GermaNet hyponyms and hypernyms,

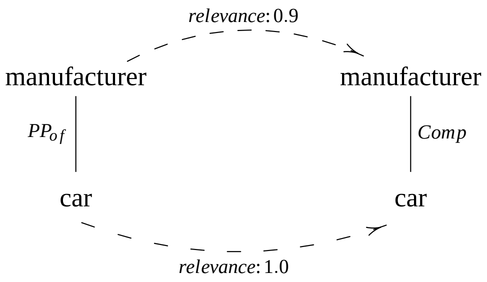


Figure 5.7: Example for a Match Between Dependency Structures: PP *'Manufacturer of Cars'* matches Compound Noun *'Car Manufacturer'*

antonyms, derivations and words related by the causation and entailment relations and inferences based on frame-to-frame relations, namely Inheritance, Subframe, Causative-of, Inchoative-of and Using.

5.2.2.3 Syntactic Dependency Information

Syntactic dependency structure is used for inferences and matching in two ways: On the one hand, a number of surface phenomena are abstracted over (such as active/passive or word order), permitting to match texts that have different surface forms but the same underlying meaning. On the other hand, we employ a number of inference rules to relate similar structures to each other.

Text Representations and Normalisations. As stated above, text representations are based upon syntactic dependency structures. For the system implementation, we have used PREDS (4.1.2). This choice is not essential. However, the use of dependency structures and the simplifications concerning the function words described above are necessary.

In addition to the points mentioned above, we represent an internal dependency structure for compound words in the following way: Each part of the compound receives a node of its own in the tree representation. The nodes are linked by *Comp(ound)* edges.

By representing the internal structure of compound words, their equivalence with paraphrases, such as NPs with the same head as the compound and postnominal modification by a PP or genitive modifier can be easily captured (cf. fig. 5.7, see also 3.5.2.2).

By using dependency structures as the core of the text representations, a high degree of abstraction is already attained. It should be noted that the PReDS are not *full* syntactic representations, as they do not express all kinds of linguistically relevant information in the structures: Auxiliary verbs, for example, carry information such as tense and mood; this information is not used in the process of finding answers. It is present in the structures, however, and can be used later in the QA process for generating answers, cf. 6.4.3.

Dependency structures carry, however, what we consider to be the core information content of the given text in a structured way.

Inferences. The representation of syntactic dependency structures normalises (or abstracts over) quite a number of surface variants in text. We add a small number of syntactic inference rules to the linguistic knowledge base to enable further matches between syntactically equivalent structures.

Compound Parts, PPs and Genitive Modifiers. We assume that relabellings are possible for several grammatical functions that are often used (more or less) interchangeably to connect nouns, namely the compound relation in noun compounds, PP relations and genitive modifiers. An example for a a match between compound part relation and PP_{of} is shown in fig. 5.7.

Argument Relabelling. To make up for the lack of a full-coverage lexicon containing valency information for German words (especially verbs), we have added general fall-back rules that permit mapping between different types of possible arguments. On the one hand, we can map between words that allow different argument realisations (such as English *compare with* and *compare to*). On the other hand, these rules are especially important when a word is replaced with another, semantically related one (for example a synonym): They may differ in the argument realisation, but GermaNet does not provide sufficient information on word valencies to account for this (we will come back to this point below, 5.2.2.5).

It is especially possible to map PPs with different prepositions onto each other with high relevance values. Mapping between other possible arguments is also possible, but only with lower relevance scores.

Underspecified Grammatical Functions. The parser that we use is sometimes not able to assign a unequivocal grammatical function to an NP argument. In these cases, an underspecified label (*NPArg*) is employed. By adding relabelling rules that map possible argument functions to these underspecified labels and vice versa, we ensure that matches are still possible.

5.2.2.4 Named Entities

Many types of named entities introduce special kinds of variations: In many cases, it is possible to refer to one entity in a number of different ways. For example, the following expressions can be used to refer to the same person: *John F. Kennedy*, *John Fitzgerald Kennedy* and *J. F. Kennedy*. In a QA system, it should be possible to use any of these different 'versions' interchangeably, both in questions and in the text collections: The system should still be able to find out which person is referred to.

For question answering, all these different versions (ideally, even ones containing mistakes such as misspellings) should match, i. e., the system should identify these different named entities as referring to the same person – at least with a certain probability.

We have therefore decided to utilise a method for robustly matching named entities. Relevance values are used as a means to express the degree of reliability of the match.

We will explain the method for proper names; other types of NEs are handled analogously. All named entities are represented by one elementary tree, where the 'components' are the children. Consider, for example fig. 5.8: *John*, *Fitzgerald* and *Kennedy* are children, the edges are labelled with the respective functions *FirstName*, *MiddleName* and *LastName*.

For matching two representations of person names, we assign different constant relevance scores to matches between the components. If present, the last name is currently required to match. For each additional matching part (such as the *FirstName* part), an additional constant is added to the overall relevance. For first and middle names, an initial is considered to match the 'spelled out' version of that name. Thus, *Fitzgerald* would be considered to match *F.* as middle name.

It should be noted that this method has a certain preference for recall over precision built in: Conflicting parts of a name do not exclude a match, thus *John F. Kennedy* and *Edward Kennedy* would be – wrongly – considered a possible match. However, as the assigned relevance score is considerably lower than, say, for matching *John F. Kennedy* and *J. F. Kennedy*, the latter candidate would be preferred over the former. We have decided to focus on the improvement in recall to allow matching inaccurate or erroneous versions of names.

5.2.2.5 GermaNet Information

We use a number of different lexical semantic relations from GermaNet as a source for textual inferences. Besides, hyponymy information from GermaNet is used to instantiate the relation $\mathcal{Q}$ of the linguistic knowledge base that is

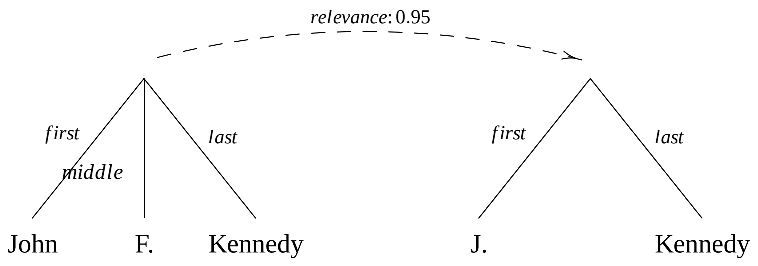


Figure 5.8: One Example for Matching Named Entities Using Relevance: Proper Names

employed for matching *wh*-phrases (especially *which/what* COMMON NOUN phrases) against answering phrases.

GermaNet Concepts as Node Labels. For all lemmata in the dependency structure that can be found in GermaNet, a GermaNet identifier is added as a node label. Currently, we do not disambiguate different word senses in GermaNet. In consequence, every node can be labelled with more than one GermaNet concept. Note also that we use lexical units rather than synsets as labels, as several of the semantic relations in GermaNet use lexical units as their basis (cf. 4.2). We therefore add inference rules to the knowledge base that relate all lexical units in one synset to each other. This permits the identification of synonyms.

Lexical Relations as Sources of Inference. GermaNet does not provide useful information on the arguments of words (different from FrameNet where information on frame elements as role labels is central). Thus, relabelling nodes with arguments (especially verb nodes) can only be approximated: When relabelling local trees based on some GermaNet relation, such as synonymy or hypernymy, we have no information available on how the labellings of the corresponding edges need to be changed. This is, of course, an unsatisfactory situation, as semantically related words may differ in their valency, especially concerning the appropriate prepositions for PP arguments.

We have described above the use of ‘uncertain’ matching of grammatical functions (5.2.2.3). This is used in conjunction with relabellings based on GermaNet information. When, for example, a node that carries a GermaNet concept as a label is relabelled with a synonym of that concept, the outgoing edges may be relabelled with any of the uncertain relabellings. Note that, as both conserving edge labelling and more likely relabellings (such as relabelling a PP using one preposition with another one using a different preposition) have higher relevance values, correct relabellings are, in general, preferred.

Information from the semantic relations is used for node relabelling in the following ways.

Synonymy. GermaNet concepts that belong to the same synonym set give rise to two inference rules (that is the concepts can mutually replace each other during relabelling).

Hyponymy. We allow relabellings that replace a concept either by one of its hyponyms or hypernyms (cf. 3.5.2.3). We have discussed above that these relabellings may produce uncertain inferences (depending on whether the context in which it is done is upward or downward monotonic).

Antonymy. Antonyms also introduce two inference relations. Matches involving antonymy need to be marked during the answer checking process (5.1.5).

Derivation. GermaNet contains only few derivation (pertainymy and participle-of) relations. We treat them like synonymy relations, that is, inferences in both directions are possible.

Causation, Entailment. As an experiment, we have added bidirectional inferences for both causation and entailment relations. Here, the 'backward' inference cannot, of course, be drawn in general. We reasoned that it might still provide relevant information (namely necessary, if not sufficient evidence): For example, GermaNet lists an entailment relation between *'gelingen'* (*succeed*) and *'versuchen'* (*try*). When asking if somebody succeeded in doing something, a user might be interested to learn (with a proper accompanying message) that he at least *tried* to do it. However, we could not observe a single relevant case in our experiments, since both relations are sparsely populated in GermaNet.

Association, Meronymy. After a few initial experiments, we did not use inferences based upon these two relations, as they turned out to produce only irrelevant results.

Figure 5.9 shows an example for a relabelling of *'take over'* with its direct hypernym *'buy'* (relevance 0.9). No edge labels are changed, in this example, so the overall relevance is 0.9.

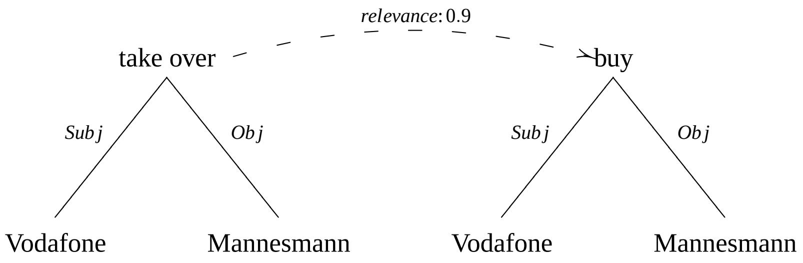


Figure 5.9: Relabelling Using GermaNet Information: *'Vodafone took over Mannesmann'* allows inferring *'Vodafone bought Mannesmann'*, overall relevance 0.9

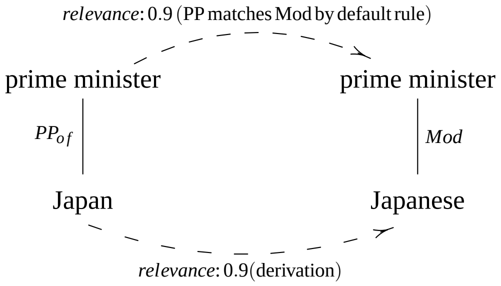

relevance: 0.9 (PP matches Mod by default rule)
prime minister
prime minister
PP_of
Mod
Japan
Japanese
relevance: 0.9(derivation)

Figure 5.10: GermaNet and Grammatical Function Relabelling: *'Prime Minister of Japan'* is equivalent with *'Japanese Premier'*, overall relevance 0.81

In fig. 5.10, two relabellings are shown that change the representation of *'prime minister of Japan'* into that of *'Japanese prime minister'*: *'Japan'* is replaced by *'Japanese'* (derivation, relevance 0.9), while the edge label is changed from PP_{of} to Mod (Modifier) by a syntactic rule (relevance also 0.9), thus the overall relevance is 0.81.

Hyponymy for Matching *Wh*-Phrases. From the GermaNet hyponymy and synonymy relations, we have directly extracted information for the relation $\mathscr{Q}$, that is the relation that needs to hold between the head of *which/what* COMMON NOUN and the associated concept of *how* ADJECTIVE phrases in questions and the head of the answering phrase (cf. 5.1.4).

5.2.2.6 Frame Information

We can systematically capture a number of additional language variation phenomena using frames as conceptual predicates (node labels), frame elements as role labels (edge labels) and frame-to-frame relations as sources for inferences (relabellings).

Frames. Frame structure, as in FrameNet, is especially well-suited as a source of lexical semantic knowledge for QA for several reasons:

Abstraction over Parts of Speech. Lexical units are grouped into frames irrespective of their parts of speech. This allows to easily map, e. g., two text fragments onto each other that carry essentially the same meaning, but where one is headed by a verb and the other by a noun, such as *'A bought B'* vs. *'(the) acquisition of B by A'*. In GermaNet, this mapping requires additional knowledge in the form of derivation relations (see above).

Semantic Role Labelling. By semantic role labelling ('frame elements'), syntactic variations are abstracted over: As such syntactic variants are mapped onto the same FrameNet representations, no additional relabelling mechanism is required.

Frame-to-Frame Relations. Frame-to-frame relations, as recorded in the FrameNet database, list correspondences between frame elements. *'A sold B to C.'* can be directly mapped onto *'C bought B from A.'*: The frames COMMERCE_SELL and COMMERCE_BUY are properly related, as are the participating frame elements BUYER, SELLER and GOODS.

For every subtree for which a FrameNet representation can be found (based on the lemma of the node and the argument realisation), the corresponding FrameNet labels will be added: The name of the frame is added as a supplementary label to the node corresponding to the frame evoking element; the edges are labelled with the corresponding frame elements. Thus, the subtree is annotated as representing an instance of the respective frame. An example is shown in fig. 5.11.

Note that frame structures are not fully disambiguated. Syntactic differences are used for disambiguation. For example, the reflexive use of a verb may be associated with a different frame than the intransitive one. In these cases, disambiguation is done. In other cases no disambiguation is performed, for example, where the correct frame can only be identified through sortal preferences on arguments.

Frame information provides an additional level of normalisation: syntactically different realisations, as, e. g., occasioned by dative shift, will receive the same FrameNet representation. For *'John gave the book to Mary.'* and *'John gave Mary the book.'*, a GIVING frame with the same frame elements is derived. In particular, *'Mary'* is identified as the RECIPIENT in both cases.

Frame Relations as Sources of Inferences. The FrameNet lexical database not only defines frames as abstract semantic predicates and frame elements as abstract semantics role labels, but also a hierarchy based upon different frame-to-frame relations defined both between frames and frame elements. We translate frame-to-frame relations directly into relabelling relations with corresponding relevance values.

We currently use all available FrameNet frame-to-frame relations, except for the SEE_ALSO relation, even though some of them are only rather vaguely defined (cf. 4.3). It is therefore not always possible to foresee whether or not using a frame-to-frame relation will or will not result in a valid inference relation. For example, when two words evoke the same frame, this does not mean that they stand in the classical synonymy relation: *'Good'* and *'bad'* both evoke the DESIRABILITY frame, even though they would be considered antonyms in terms of classical lexical relations.

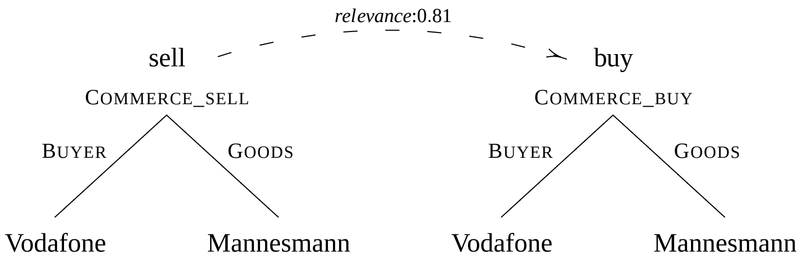
We have decided to exploit all frame-to-frame relations as sources of inferences. From the definition of the relations (cf. 4.3), we considered that this would in most cases produce interesting, if possibly sometimes unlikely inferences. We considered, however, that the additional step of answer checking should detect and properly mark those cases.

In our experiments, we did not observe any serious problems with this approach (cf. 7.2.2.3). This may to a large extent be due to the limited overall current coverage of the frame lexicon that we use (cf. 4.3.3). We expect clearer definitions of the relations to emerge together with growing coverage; at some point, it may turn out to be advisable to remove all but some core relations from consideration.

This is how the relations are currently utilised for inferences:

'Same Frame'. This is not strictly a frame-to-frame relation: Two subtrees labelled with the same frame (and frame elements) match during direct answer matching, simply because the labels are identical. No additional inference rules are required.

Inheritance. We treat inheritance like a classical hyponymy relation, that is, we use it to introduce inferences in both directions (cf. 3.5.2.3).



The required relabelling relations are derived from the following frame-to-frame relations:
COMMERCE_SELL uses COMMERCE_GOODS-TRANSFER (relevance 0.9), COMMERCE_GOODS-TRANSFER is used by COMMERCE_BUY (relevance 0.9)

Figure 5.11: Example for Relabelling Using Frame-to-Frame Relations: ‘*Mannesmann was sold to Vodafone’* is equivalent with ‘*Vodafone bought Mannesmann’*

Subframe. We considered this relation as describing a default inference: From the subframe relation between COMMERCIAL_TRANSACTION and COMMERCE_GOODS_TRANSFER, it seems plausible to assume that an instantiation of the former frame permits the inference that – by the same token – the latter frame is also instantiated. We therefore derived inference relations from the subframe relation; again, we instantiated both directions.

Causative-of and Inchoative-of. Both the causative-of and inchoative-of relation are also used bidirectionally.

Using. This relation is mostly used to relate similar frames using different perspectives to each other (4.3). We therefore considered it an important source of inferences; allowing, e. g., to relate (through the respective frames) word that stand in classical inverse relations. Again, both directions were used.

See also. As mentioned above, we did not use the see also relation, as it did not seem to provide generally useful information.

Figure 5.11 shows an example for a relabelling using frame-to-frame relations.

As the discussion in this section shows, FrameNet contains information that we consider very interesting for textual inferences and thus for QA: As a representation, it provides a useful level of abstraction over a number of surface phenomena, and it also allows the definition of inferences. However, we consider that using FrameNet as a source for textual inferences is currently somewhat hampered by two important shortcomings. First, the current coverage of FrameNet (especially for German FrameNet) is too small for practical applications. While we expect future versions of the FrameNet database to provide a far better coverage, the current version needs to be complemented with other resources (as we have done). Second, a clearer specification of the frame-to-frame relations (sometimes perhaps also a change) would help to more clearly identify cases in which they can be utilised in natural language processing.

5.2.2.7 Additional Role Relations

We utilise a number of additional role relation edges. These additional edges comprise the following: BENEFICIARY, CAUSE, CIRCUMSTANCES, COMPANY, CONDITION, DEGREE, DESTINATION, DURATION, INSTRUMENT, ITERATION, MEANS, PLACE, PURPOSE, SOURCE, TIME, TIME-END and TIME-START.

These additional edge labels are added in cases where the relation (that is semantic rather than syntactic) can be recognised with a sufficient degree of certainty, using a combination of syntactic patterns with GermaNet information. For example, TIME edges are added for subordinate sentences with the conjunction *'als'* (*when*) as well as $\mathrm{PP}_{\mathrm{in}}$ if the argument is a hyponym of *'Zeiteinheit'* (*unit of time*) in GermaNet, such as *'Vorjahr'* (*previous year*), *'Sommer'* (*summer*) or *'Mittelalter'* (*middle ages*).

Thus, the additional role labels can be seen as an additional level of normalisation that lies beyond the purely syntactic level. Such an additional role label is also added for adverbial question words such as *'wann'* (*when*) or *'wo'* (*where*). Thus, the additional edge labels allow to match adverbial questions representations against answer representations for quite different syntactic constructions (cf. 3.5.2.5).

5.2.2.8 Coreference and Predication

In the definition of indirect answerhood above (5.1), we have introduced additional edges that we have called coreference edges (represented by set C in defs. 5.2 and 5.8).

In practice, we use these edges not only to represent coreference, but also predication relations. We have introduced the notion of predication above and pointed out its importance, especially for definitional questions, such as *‘Who/what is X?’* (3.5.2.5).

A coreference edge will be added to the corresponding text representation for all nodes between which either coreference or predication is known to hold. We will continue to refer to these edges as coreference edges, but it should be kept in mind that they may also express predication.

Note that these additional predication edges allow us to draw additional textual inferences of a certain type: For a sentence like *‘Nanosoft, a US-based software company, produces the operating system TrapDoor.’*, the apposition will give rise to a predication relation between *Nanosoft* and *software company*. Thus, for a question like *‘Which company makes the operating system TrapDoor?’*, the representations can be matched, as the predication relation (represented as an additional edge) can be followed during answerhood matching, permitting the inference that *Nanosoft* is indeed a *company*.

In addition, the representation will also match definitional questions. In this example, the question *‘What is Nanosoft?’* would match the predication relation between *Nanosoft* and *software company*, giving rise to the answer *‘a US-based software company’*.

Such definitional questions are frequent both in shared-task evaluations like TREC and also in the evaluation data that we used (7.2.1.2). Making the predications in the text explicit is thus an important task of a QA system.

5.2.2.9 Merging the Levels

So far, we have assumed that, for each elementary tree in the question representation, a match of labels with the answer representation at one linguistic level is sufficient. We will now slightly change the relevance definition to reflect the assumption that whenever a match for an elementary tree can be found at more than one linguistic level, this match is strengthened, expressed by a higher relevance value.

We achieve this by ‘penalising’ elementary trees for which linguistic information at a certain level is present in the question representation, but not in the matching answer representation.

This is done as follows: First, a set of linguistic levels is defined (viz., a syntactic level including GermaNet information and a frame semantic level). Each label defined in the linguistic knowledge base is assigned to one of these linguistic levels. For every elementary tree for which no match can be found at *all* defined levels, a penalty factor will be multiplied in during matching question and answer representation.

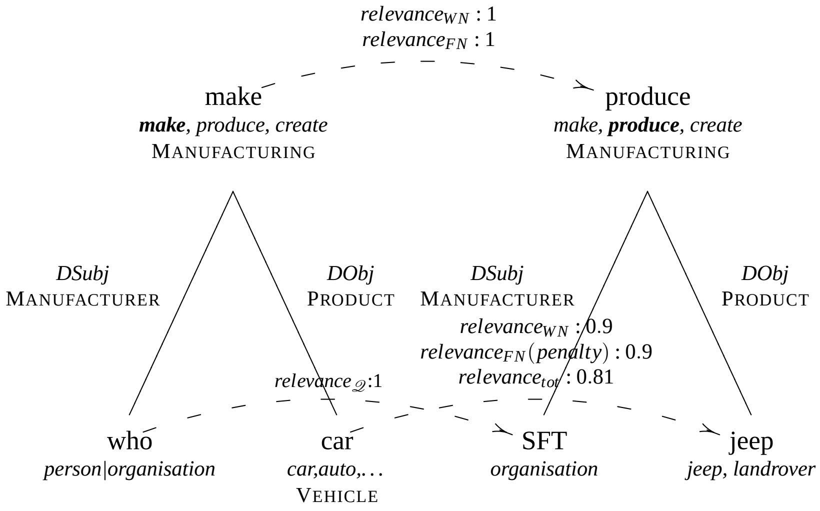


Figure 5.12: Example for Matching Question and Answer: *'Who makes cars?'* is answered by *'SFT has been producing jeeps [for Mercedes for years].'*

We have currently set the penalty factors for missing matches at the GermaNet level to 0.8 and for missing matches at the FrameNet level to 0.9. This reflects the fact that FrameNet information is missing relatively often, so that a missing match is punished relatively lightly, whereas missing GermaNet/dependency information is punished more severely.

Figure 5.12 shows an example for a match that combines different levels. The structures represented are *'Who makes cars?'* (left-hand side) and *'SFT has been producing jeeps [for Mercedes for years].'* (right-hand side, the bracketed part of the sentence is not shown for space reasons).

Let us look at the matches in some more detail:

- In fig. 5.12, lemmata are given in Roman font, WordNet (GermaNet) synsets are indicated by *italics* and FrameNet information is in CAPS.

- The *'who'* node matches the *'SFT'* node via $\mathcal{Q}$ with a relevance value of 1.0: The *'who'* node is a question node; it matches the WordNet synsets *person* or *organisation*. As the organisation name *'SFT'* is labelled as *organisation*, this is a perfect match.

- The *'car'* node matches the *'jeep'* node: As *jeep, landrover* is a direct hyponym of *car, auto, . . .* in WordNet, the relevance value is 0.9. Since *'jeep'* was not assigned a FrameNet frame (car got VEHICLE), the penalty for a missing frame match (0.9) is used. Thus, the overall relevance for matching this elementary tree is 0.81. Note that this tree consists only of a node without children.
- The *'make'* node and the *'produce'* node match also. As both lemmata belong to the same WordNet synset, the relevance of the match is 1.0. The same applies for the FrameNet level: Both are labelled as MANUFACTURING. The outgoing edges are also perfect matches at both levels: *DSub* equals *DSub*, *DObj* equals *DObj*, MANUFACTURER equals MANUFACTURER and PRODUCT equals PRODUCT. Now, the relevance values of the daughter labels (1.0 and 0.81) are multiplied in, yielding an overall relevance value of 0.81.

This example shows how less-than-perfect matches (*'car'*, *'jeep'*) on the one hand and missing information (no FrameNet representation for *'jeep'*) are integrated into a single overall relevance score for a potential answer: This answer is judged to be a relatively good answer for the given question, but it is not perfect. If there was, for example, another potential answer in the document collection that had *'car'* instead of *'jeep'*, that answer would have been ranked higher.

5.2.3 Conclusion

In this section, we have taken a closer look how the linguistic phenomena that we have described above as relevant for indirect answerhood (3.5) can be handled by the the indirect answerhood relation between textual representations as defined in 5.1. We have shown how the linguistic information contained in the dependency structures, on the one hand, and in the lexical knowledge bases GermaNet and FrameNet (cf. chapter 4) can be utilised in this context. One important point was the definition of suitable relevance values for the different underlying textual inference and answerhood relations.

5.3 Searching for Indirect Answers

We have defined the notion of indirect answerhood above (5.1) and shown by selected examples that it can be employed as a useful approximative description of the question answering process (5.2). We will now turn to the question of how an efficient search algorithm can be developed on the basis of that definition.

Actually this algorithm will not only check answerhood for a given question and a given answer candidate, but search the best answer candidates in document collections.

As a starting point, we will relate the search problem at hand to general well-researched graph and tree matching problems and identify tree matching by unordered path inclusion as a useful basis for defining the search algorithm (5.3.1).

To derive a search algorithm from this matching algorithm, we first change indirect answerhood by 'reversing' the direction and integrating sequences of textual inferences into a single inference step. This revised relation of indirect answerhood can then be used for answer searching; the efficiency of the resulting algorithm is ensured by interleaving inference and search.

5.3.1 Linguistic Graph Matching

We regard the document collection representation as a single graph in which a question graph needs to be embedded. In this way, we arrive at a very simple definition in terms of general graph matching (5.3.1.1).

We then relate this graph embedding to the underlying problem of subgraph isomorphism (5.3.1.2). The *general* problem of subgraph isomorphism is computationally intractable. We will shortly explore several restricted versions of subgraph isomorphism but find that none of them provides a suitable solution.

We will therefore go back to the core of the definition of text representations, namely tree structures. For matching trees (more specifically for unordered path inclusion), an efficient general algorithm exists (5.3.1.3). This algorithm forms the basis for the search algorithm that we employ for finding answer representations for a given question representation.

5.3.1.1 Searching Answers as Graph Matching

The linguistic representations that we employ to represent texts in a document collection and questions (def. 5.2) are, in the general case, graphs. We have stated above that the underlying structures are trees (representing single sentences) and forests (representing whole texts). However, proper graph structures are necessary because of the introduction of additional edges expressing coreferences and predicatives.

We have defined the relation of direct answerhood (def. 5.8) to hold when a question representation can be embedded in a text representation; the matching subpart of the text representation is then considered to contain the answer.

The task of answer searching can now be viewed as an extension of this definition: Instead of embedding the question representation in a given answer candidate representation, answer searching is defined as searching an embedding of the question representation in the graph representing the *whole* document collection. We have defined a text representation as a graph above. The union of the representations of all texts that make up the document collection is then again a graph where the sub-graphs that represent single documents are not connected.

Thus, answer search can be defined as finding a suitable embedding for a given question representation in the graph representation of the whole document collection to be searched, made up from the union of all graph representations of its documents.

We will now turn to describe the underlying graph matching problem in more detail from the viewpoint of graph theory.

5.3.1.2 Graph Matching through Subgraph Isomorphism

Graph theory is a well-researched area in mathematics and theoretical computer sciences: A large number of algorithmic problems can be intuitively described in terms of graph manipulations (Diestel, 2005). For these, a great number of algorithms and complexity results have been obtained (cf., e. g., Garey and Johnson, 1979).

The general problem that underlies checking for answerhood (cf. def. 5.8) and also the search for answers in the 'document collection graph' as just described (5.3.1.1) is known as the subgraph isomorphism problem (Ullmann, 1976). The subgraph isomorphism problem is a special case of the graph isomorphism problem (Fortin, 1996).

Subgraph Isomorphism. The graph isomorphism problem can be described as follows: For two given graphs, G and H, find out whether there is an isomorphism that maps G onto H, i. e., whether G and H are structurally equivalent.

For subgraph isomorphism (again for two graphs G and H), it is sufficient that G can be *embedded* by an isomorphism in H, that is, H has a subgraph that is isomorphic with G. This is the core of the definition of direct answerhood, cf. def. 5.8. More exactly, it can be described as finding a subgraph isomorphism for labelled, unordered graphs.

It is well known that the subgraph isomorphism problem is NP-complete in the general case (Garey and Johnson, 1979, 202). Under the assumption that $P \neq NP$, it is computationally intractable[2].

[2]That is, it is assumed to be computationally intractable on current (deterministic) computers under the assumption that $P \neq NP$, i. e., the class of problems that can be computed in polynomial time on deterministic Turing machines is properly included and thus distinct from those that can be computed in polynomial time on non-deterministic Turing machines. This issue is one of the

Therefore, an implementation based on the general problem of subgraph isomorphism is not a useful basis for a working QA system. We will shortly inspect some computationally tractable subclasses of the general subgraph isomorphism problem to see whether our problem of answer searching falls into one of the 'good-natured' subclasses. It turns out, however, that none of the subclasses that we have found in the literature exactly fits our problem.

Tractable Subclasses of Subgraph Isomorphism. While the general problem of subgraph isomorphism has been established to be NP-complete, there are a number of special cases that are known to be solvable in polynomial or even linear time with regard to the size of the input graphs. They mostly depend on certain properties to hold for both input graphs:

- Linear time algorithms exist that decide subgraph isomorphism for planar graphs (Eppstein, 1999). Planar graphs can be intuitively described as graphs that can be drawn on paper (i. e., on a plane, and thus in two dimensions) without any crossing edges (cf. Diestel, 2005, 83–110).

- Polynomial time algorithms are known to exist for graphs where the number of vertices is smaller than some constant *k* (Eppstein, 1999, 2).

- Polynomial time algorithms also exist for graphs with bounded degree or bounded tree-width (Gupta and Nishimura, 1995), that is, for graphs where either the maximum number of outgoing edges for any node in the graph itself or in a (minimal) tree representation is smaller than some constant *k* (cf. Diestel, 2005, 5; 13–16).

Thus, it would be possible, by imposing a suitable restriction on the text representation graphs, to ensure that subgraph isomorphism (and thus answerhood) can be checked in polynomial time. There are, for example, very few natural language sentences for which a node in the corresponding dependency structure has more than ten daughters (complements and adjuncts together). More complex structures may be found in natural language texts and would be excluded from the search.

However, the ensuing algorithm might not be practically useful since the bounding constant usually determines the polynomial of the time complexity: An algorithm with time complexity $O(n^{10})$, that can be obtained for finding subgraph isomorphisms for graphs with a maximum out-degree of ten (where *n*

most-discussed unsolved issues in theoretical computer science. However, the inclusion is generally assumed to be proper. We will take this assumption for granted, and we will not further pursue the issue here. See Garey and Johnson (1979).

is the number of vertices in the graph, cf. Gupta and Nishimura, 1995) may just not be efficient enough for practical purposes, even though it is 'only' polynomial in time complexity.

Pre-Computation. Another approach to this problem is suggested by Messmer and Bunke (1995). They give an algorithm that allows, for a given graph H, to pre-compute a decision tree representation and to check, for a given graph G, whether a subgraph isomorphism can be found for H against this decision representation in quadratic time with regard to the size of G (and not depending on the size of H).

This approach could, in principle, be used for building a question answering system based on indirect answerhood, as the text representations to be searched are known in advance and could thus be transferred into the required format. However, this is not a viable general solution, because the size of these decision tree representations may grow exponentially with the size of the input graphs. This is all the more the case since a QA system should be able to handle large document collections and, consequently, large text representations.

However, we can make use of the fact that the graph representation of the document collection has only to be computed once and does not change. It is therefore possible to pre-compute indices for the structure to make searching more efficient. This is the general idea behind the approach described in Messmer and Bunke (1995), namely of making the matching more efficient by computing structures that enable a more efficient match within the 'static' representation of the graph that is to be searched.

We will make use of the general idea of pre-computing in that we build indices for the labels in the text representations of the document collection to speed up searching.

5.3.1.3 Tree Matching by Path Inclusion

In contrast with subgraph isomorphism, which we have reported to be a computationally intractable problem in the general case, many problems related to tree matching can be solved efficiently (Kilpeläinen, 1992). We will therefore base the answer search algorithm on tree matching.

The problem of answer searching can be couched in terms of the unordered path inclusion problem on labelled trees (Kilpeläinen, 1992, 15–16). The (unordered) path inclusion problem is a specialised subproblem of the (unordered) tree inclusion problem (Kilpeläinen, 1992, 11–13). The tree inclusion problem can be paraphrased as follows: Given two trees P (pattern) and T (target), tree inclusion holds exactly if an embedding of P in T can be found so that labels and ancestorship are preserved, that is, for every node in P a corresponding node in T with the same label exists and the parent of this node in P corresponds to an ancestor of the corresponding node in T. For the path inclusion problem, it is additionally required that parenthood is preserved, that is that for every node in P, its parent corresponds to that of the corresponding node in T. Note that unordered path inclusion imposes no ordering on children in the trees and that it allows proper inclusion, that is, a node in T may have 'additional' children that do not correspond to any nodes in P[3].

Kilpeläinen gives a general algorithm for the unordered path inclusion problem and establishes its time complexity as $O(m^{3/2}n)$, where m is the number of nodes in P and n the number of nodes in T (Kilpeläinen, 1992, 39–42). Thus, the path inclusion problem can be solved efficiently. This is especially the case if P is small. This will generally apply for question representations, since questions will mostly be relatively short.

Algorithm 5.1 Unordered Path Inclusion, adapted from Kilpeläinen (1992, pp. 41–2, his alg. 4.14)

```
Input: Trees $P = (V, E, root(P))$ and $T = (W, F, root(T))$
Output: Nodes $w$ of $T$ such that there is a root preserving path embedding of
    $P$ in $T[w]$
 1: for $w \leftarrow 1 \dots n$ do                    ▷ Go through target nodes bottom-up
 2:     Let $w_1, \dots, w_l, l \geq 0$, be the children of $w$
 3:     for $v \leftarrow 1 \dots m$ do                 ▷ Go through pattern nodes bottom-up
 4:         $a(v, w) \leftarrow 0$
 5:         Let $v_1, \dots, v_k, k \geq 0$, be the children of $v$
 6:         $G \leftarrow (X \cup Y, E)$, where
              $X = \{v_1, \dots, v_k\}, Y = \{w_1, \dots, w_l\}$, and
              $E = \{(x, y) | x \in X, y \in Y, a(x, y) = 1\}$
 7:         if LABEL$(v)$ = LABEL$(w)$ and size of a max. matching of $G$ is $k$
    then
 8:             $a(v, w) \leftarrow 1$
 9:     if $a($ROOT$(P), w) = 1$ then
10:         OUTPUT$(w)$                                  ▷ An occurrence found
```

The algorithm is repeated here as alg. 5.1; it works roughly as follows (for details see Kilpeläinen, 1992, 39–41): Embeddings are computed recursively in a bottom-up fashion by checking, for every node v in P and every node w in

[3]We follow Kilpeläinen's terminology here. He notes that the path inclusion problem is sometimes also referred to as subtree inclusion. He restricts this term to fully isomorphic embeddings (i. e., no additional children may be present).

T, whether the subtree rooted in v is path-included in the subtree rooted in w with v and w corresponding to each other. Intermediate results are stored in a Boolean array a: for every v and w, the corresponding cell is set to 1 iff path inclusion holds for the subtrees rooted in v and w, respectively.

As the algorithm proceeds in a bottom-up fashion, checking new nodes can be done comparatively easily: For all child nodes, it has already been established whether or not they match (these nodes have already been 'visited' and the result has been stored in the Boolean array). All that needs to be done, then, is to check whether the node labels themselves match and whether the outgoing edges from node v can be mapped onto the outgoing edges from w so that the respective corresponding child nodes match. An embedding is found whenever two nodes v and w match and v is the root node of P.

The matching of outgoing edges is defined via maximum matching of bipartite graphs (in the alg. 5.1, ll. 6 and 7), that is by considering edge matching as a separate graph problem: The child nodes of v and w are used to construct a graph, edges are added for those nodes for which a match was found. A maximum matching is then *one* possibility of aligning all child nodes (and thus the edges) in P each exactly to one child node in T. See Kilpeläinen (1992, 40–41) for details and Hopcroft and Karp (1973) for the original algorithm for finding maximum matchings in bipartite graphs; the important point here is that in this way a possible mapping between pattern and target edges can be described and efficiently computed.

By storing the intermediate results, each pair of nodes from P and T only needs to be visited once, the additional complexity factor results from the need to find a mapping of the outgoing edges onto each other for each pair of nodes.

We will use Kilpeläinen's algorithm for path inclusion as a basis to defining a search algorithm. First, we consider how textual inferences can be reintegrated into the search algorithm. We show that this can be done without unduly impairing the overall complexity. We then present the integrated algorithm and describe further efficiency improvements.

5.3.2 Answer Search

We have related the definition of indirect answerhood to graph and tree matching problems and shown that the answer search can be couched in terms of graph matching by regarding the whole document collection representation as a single graph and embedding the question representation in it. We will now work out the details.

First, we show how a 'single-step' relabelling and matching can be derived from the definition of indirect answerhood. We concatenate and reverse the sin-

$$\begin{tikzcd}
T \arrow[r, "\mathcal{K}"'] \arrow[rrr, bend left, "\textit{indirect answerhood}"] & \cdots \arrow[r, "\mathcal{K}"'] & \bar{T} \arrow[r, squiggly, "\mathcal{K}"'] & T^* \\
T & \cdots \arrow[l, "\mathcal{K}"'] & \bar{T} \arrow[l, "\mathcal{K}"'] & T^* \arrow[l, squiggly, "\mathcal{K}"'] \arrow[lll, bend left, "\textit{indirect answerhood}^{-1}"]
\end{tikzcd}$$

Figure 5.13: Indirect Answerhood and Its Inverse Relation

gle steps so that from a given question, all possible (partial) representation that would provide an answer can be generated (5.3.2.1).

From this, we then derive the final search algorithm that uses tree matching based on the unordered path inclusion algorithm described above and integrates relabelling steps into it (5.3.2.2).

We will conclude with a number of pruning techniques that we have used to gain an additional speed-up of the practical implementation; they make use of the relevance values defined above and prune all but the most relevant results to limit the search space (5.3.2.3).

5.3.2.1 Generating Answers from Questions

We will now integrate textual inference into the answer search algorithm.

Integrating Textual Inferences. We have defined the relation of indirect answerhood as a compound of concatenated local textual inferences and direct answerhood. We will now integrate the textual inferences directly into the answer search algorithm.

We will not utilise indirect answerhood, but its inverse relation. This can be seen as a generation algorithm for possible answers: It generates text representations of possible answers from a given a question representation. This is sketched in fig. 5.13

This could be seen as a way of doing query expansion (cf. 2.2.2) while observing the linguistic structure: As every node label (and thus lemma) may be replaced by related labels from the linguistic knowledge base (e. g., synonyms), this is similar to adding the respective terms to a search. However, in contrast to bag-of-word-based query expansion strategies, this algorithm matches only representation that have the same structure.

Complexity. Note that by re-integrating textual inference into the answer search, the task is made more complex: Instead of embedding a single question representation into the representation of the document collection, we now have to first generate all text representations that would provide possible answers for the given question and search for each of these possible answers in turn.

A simple algorithm for this generation process is Algorithm 5.2. It is directly based on the definitions of direct answerhood (def. 5.8) and local inference (def. 5.5), with the difference, that the operations are carried out backwards. It is ‘driven’ by the central queue S, which first holds the question representation Q. Then, all possible local relabellings on Q are carried out and *each* resulting structure is put back onto the queue. Thus, all possible answer representations for Q as licenced by $\mathscr{K}$ are produced.

Note that the number of possible answers that need to be generated will, at least in the worst case, grow exponentially with the size of the question (more exactly, with the number of nodes in the question representation): As local relabellings are independent of each other, all possible relabellings for all nodes must be combined with one another to produce all possible answer representations.

The number of relabellings that need to be carried out for every node is bounded by the number of relabelling rules in the linguistic knowledge base: In the worst case, all relabelling rules in the knowledge base can be carried out for every label (even though this situation is probably linguistically not very interesting). Thus, all possible variations must be computed; the number of variations is then bounded by $|\mathscr{K}|^{|V|}$, where $|\mathscr{K}|$ is the number of (edge relabelling) rules in the knowledge base and $|V|$ the number of nodes in the question representation.

We will now change the simple generate-and-test approach into an efficient search algorithm.

5.3.2.2 The Search Algorithm

The final search algorithm presented in the following integrates the generation of possible answers with a search for partial structures, so that search and generation control each other. It extends the algorithm for tree matching based on unordered path inclusion given above.

Algorithm 5.2 Algorithm for Constructing Indirect Answer Representations for a Given Question Representation

```
procedure CONSTRUCT_POSSIBLE_ANSWERS($Q, \mathscr{K}$)
    $S \leftarrow \emptyset$                                    ▷ Intermediate queue
    QUEUE($S, Q$)
    while $S \neq \emptyset$ do
        $T \leftarrow$ POP(S)                               ▷ For first rep. on queue
        for all $v \in T.V$ do
            if $T.W(v) = 1$ then                             ▷ Wh-Node
                for all $\langle l_N, l'_N \rangle \in \mathscr{Q}$ do
                    if $\langle v, l'_N \rangle \in T.P$ then
                        $T' \leftarrow T$                               ▷ Copy $T$
                        SET_NODE_LABEL($T'.v, l_N$)              ▷ Relabel $T'$
                        QUEUE($S, T'$)              ▷ Re-queue relabelled $T'$…
                        OUTPUT($T'$)               ▷ …and output as one result
            else                                          ▷ ‘Normal’ node
                for all $\langle\langle l_N, l'_N \rangle, \mathscr{R} \rangle \in \mathscr{K}.\mathscr{P}$ do
                    if $\langle v, l'_N \rangle \in T.P$ then
                        $T' \leftarrow T$                               ▷ Copy $T$
                        SET_NODE_LABEL($T'.v, l_N$)       ▷ Relabel node…
                        for all $u, \langle v, u \rangle \in T.E$ do
                            for all $\langle l_N, l_E, l'_N, l'_E \rangle \in \mathscr{R}$ do
                                if $\langle v, u, l'_E \rangle \in T.R$ then
                                    SET_EDGE_LABEL($T'.v, T'.u, l_E$)        ▷
 …and edge
                        QUEUE($S, T'$)                          ▷ Re-queue…
                        OUTPUT($T'$)               ▷ …and output as one result
```

We first integrate the coreference edges, which were so far kept separate from the dependency edges, into the text representations. We do so by adding additional dependency edges, i. e., by multiplying the coreference edges out.

By using a tree matching approach instead of full graph matching, we introduce a trade-off of search speed against overall expressivity. We will therefore start by showing in what respect the algorithm overgenerates (i. e., it may identify text representations as being potential answers that are, in fact, none). We will show that this applies only to a very restricted class of questions and is therefore irrelevant from a practical vie-point; in addition, these false positives can be efficiently filtered.

We pre-compute the transitive closures of the relabelling relations in the linguistic knowledge base as a prerequisite for the application of our search algorithm.

We then describe the algorithm and explore its general complexity.

Multiplying out Coreference Edges. Before storing the text representations (see below, 5.4), we collect, for every node in the text representations, all other nodes with which it is linked through coreference edges. Thus, all references to and predications of the same 'underlying' entity are gathered into one equivalence class.

Additional edges are added in the following way: Each mother node of a node in one equivalence class is connected with all other nodes in this equivalence class with additional edges. These additional edges are marked as 'equivalence edges' so that the original text structures can be reconstructed. During search, however, they behave exactly like normal edges. The original coreference edges are not stored and not used during searching at all.

In this way, coreference links must not explicitly be followed during search (as defined in defs. 5.7 and 5.8). The equivalence edges are explored like other edges. This keeps the search algorithm simpler.

In practice, the number of edges that are added by this additional step is not large. It generally grows with the square of the 'participating' nodes for an equivalence class: As all nodes have at most one incoming edge, re-linking each of the participating n nodes with the (at most) n in-nodes results in n^2 edges. We have rarely observed, for text representations, equivalence classes with more than ten members.

The example in fig. 5.14 shows a representation of a text where *'John F. Kennedy (JFK)'* is introduced first and then taken up by the sentence *'He was murdered by Lee Harvey Oswald (LHO).'* When multiplying out the coreference between *'JFK'* and *'he'*, an additional DOBJ link is added to *'JFK'*. Thus, the

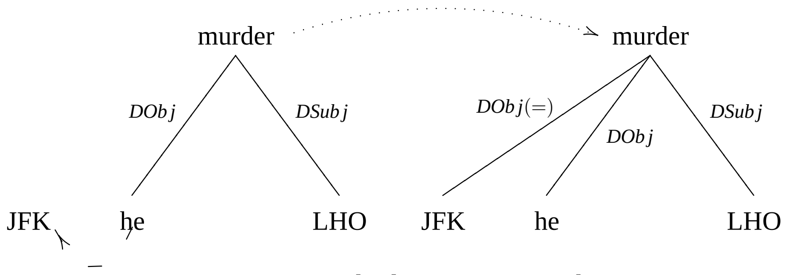


Figure 5.14: ‘Multiplying out’ Equivalences

resulting representation on the right hand side would now directly be found as an answer for *‘Was John F. Kennedy murdered by Lee Harvey Oswald?’*.

Note that if a *wh*-node of a question matches an answer node that forms part of an equivalence class, then, typically, a number of different answer structures will be returned, namely one structure for every member of the equivalence class. During answer generation, these are recognised as equivalent (as they belong to the same equivalence class) and the most suitable candidate to represent the class is chosen (for example, if a named entity, such as a person name, is member of the equivalence class, that name is used).

A similar technique of ‘multiplying out’ is used for every conjunction: Besides the original edges that link the dominant node with the conjunction and the conjunction in turn with its conjuncts, direct edges from the dominant node to each of the conjuncts are added.

This ensures that when matching questions and answers, both the ‘conjoined’ construction can be matched, but also the single conjuncts. Even though the phenomena that are handled are quite dissimilar for coreference links and for conjunctions, the technique of multiplying out that we use for storing and retrieving them is essentially the same.

Tree Matching. We have introduced the algorithm for resolving unordered path inclusion above (5.3.1.3). We will use this algorithm as a basis for defining the search algorithm.

We have defined indirect answerhood as a relation that holds between graph representations of texts and questions. While these representations are based on tree structures (namely the syntactic dependency structures), we have introduced additional edges to express coreferences and predicatives (cf. def. 5.2

and 5.2.2.8). Obviously, by introducing these additional edges, the underlying tree representations are changed into directed graphs in the general case.

To be able to use the path inclusion algorithm, we thus have to make a number of assumptions that we will describe now. The overall consequence is that the search algorithm may ‘overgenerate’ in certain cases, that is, return representations that should not count as answer representations. These cases, however, are rare in practice and can be filtered with an additional filtering step.

First, it should be noted that while the pattern tree P in the algorithm needs to be a tree, the algorithm will work for directed graphs as target T, as long as a strict bottom-up ordering of nodes can be defined on it: The important point is that whenever two nodes v and w in P and T are considered for matching, all their child nodes must be guaranteed to have been matched against each other and the results must be stored in the Boolean array.

Such an ordering can be found for the text representation graphs in the following way: Starting with all leaf nodes (i. e., nodes with no outgoing edges), a bottom-up breadth first search of the whole graph is done. The ordering on the nodes needed for the path inclusion algorithm is exactly the order in which the nodes are visited during this bottom-up breadth first search. Thus, the whole document collection representation can be used as input structure T, even though it is not a tree structure in general.

For the question representation, the case is somewhat more complex: In the algorithm, P must be a tree, otherwise the algorithm will not work properly. By its design, it would, when a graph structure is used, only search the spanning tree of that structure: For each node that can be reached by more than one incoming edge, possible matches against the target structure would be considered independently of each other. Thus, while the outgoing subtrees would still need to match for the different matches, they would *not* be required to be identical.

Therefore, a directed graph may be used as the structure P, as long as it has a single root, but only the underlying spanning tree (Diestel, 2005, 13–16) for that root will be searched.

This has the consequence of possible overgeneration: For example, when in a question like ‘*Who has sold his house?*’, the coreference of *who* and *his* is represented by folding the corresponding nodes into one node, the identity of the corresponding nodes in matching structures is not guaranteed. This has the effect that it would match representations like ‘*Peter has sold Tom’s house.*’; for matching purposes it would be indistinguishable from the representation of ‘*Who sold whose house?*’. This is shown in figure 5.15.

In practice, we replace anaphoric nodes in the question representation with a copy of their antecedent. Inter-sentential coreference, i. e., ‘back references’

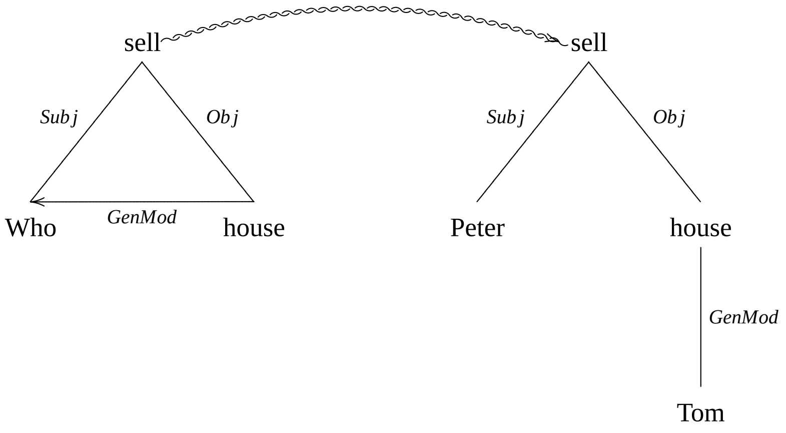


Figure 5.15: Tree Matching: *‘Who sold his House?’* Wrongly Matches *‘Peter sold Tom’s house.’*

to earlier questions or answers, can be simply resolved in this way. This will be explained in more detail in 6.4.4.

In cases of intra-sentential anaphora, the anaphor will also be replaced by the representation of its antecedent. This will in most cases restrict the answer representations enough. For example, the representation for *‘What did Isaac Newton$_i$ think about his$_i$ books?’* would be the same as *‘What did Isaac Newton think about Isaac Newton’s books?’*, allowing a match against an answer representation like *‘Isaac Newton considered his own books unsatisfactory.’*, as the answer representation would be compiled out.

The only problematic case arises if the antecedent cannot be resolved to a referring expression: This is the case in the example above, where the antecedent of *‘his’* is the interrogative pronoun *‘who’*. Thus, the number of cases where the approach actually overgenerates are comparatively few.

Pre-Computing Textual Inferences. We have defined textual inferences (def. 5.6) as a series of local inference steps, corresponding to local relabellings. We will now ‘compile’ all possible sequences of local relabelling steps for one node into one single relabelling step that may produce more than one output.

As the number of relabelling rules is finite by definition (cf. def. 5.4 where both the labelling alphabets, the node relabelling relation $\mathscr{P}$ and the edge relabelling relation $\mathscr{R}$ are defined as finite sets), the finite transitive closure for the relations in the linguistic knowledge base, $\mathscr{P}$ and $\mathscr{R}$, can be computed in finite time, as we can interpret the relations as graphs: Computing the transitive closure of graphs can be done with the classic Warshall algorithm with worst-case complexity $O(n^3)$ with regard to the number of edges in the input graph (Warshall, 1962).

Note that we need to slightly change the algorithm so that the relevance value is properly accounted for. For each $\langle l_N, l'_N, r_1 \rangle \in \mathscr{P}$ and every $\langle l'_N, l''_N, r_2 \rangle \in \mathscr{P}$, a new $\langle l_N, l''_N, r_1 \times r_2 \rangle$ needs to be added; analogously for each $\mathscr{R}$. If for a given pair l_N, l'_N more than one relabelling is possible, only the one with the highest relevance value must be retained.

From a given linguistic knowledge base $\mathscr{K}$, we can thus derive a related one – let us call it $\mathscr{K}^*$ – that differs from $\mathscr{K}$ exactly by the fact that its $\mathscr{P}^*$ and the respective relations $\mathscr{R}^*$ are the transitive closures of $\mathscr{P}$ and its relations $\mathscr{R}$, respectively.

We can compute the textual inferences off-line so that by using $\mathscr{K}^*$ instead of $\mathscr{K}$, only one single 'cycle' is needed to derive each possible answer representation for a given question.

Divide and Conquer. We have shown that the naïve approach, namely, given a question representation, to generate all possible answer representations licenced by the knowledge base and search them in turn will be inefficient: The number of possible answer representations may grow exponentially with the number of nodes in the question representation, possibly necessitating an exponential number of searches as a result.

We accordingly change this generate-and-test approach into a *divide et impera* one by interleaving answer generation and answer searching.

Consider the (somewhat simplified) search algorithm (alg. 5.3), that takes a document collection representation T, one question representation Q and a 'compiled' linguistic knowledge base $\mathscr{K}^*$ as its input and returns mother nodes of matching answer representations. The algorithm is essentially the same as that for unordered path inclusion (5.1), but additionally allows to carry out textual inferences (relabelling) as licenced by the knowledge base[4]. Note that a function MMG is assumed that computes the maximum matching of a bipartite graph as above.

The main changes are the following: When two given nodes in the text representation and in the question representation are matched, we do not check

[4]We originally defined this algorithm without knowing Kilpeläinen (1992). When searching for known complexity results, we found the path inclusion algorithm defined there to be almost identical with ours.

equality of the labels, but rather whether the nodes stand in relation $\mathscr{Q}$ (*wh*-nodes) or in relation $\mathscr{P}$ ('normal' nodes). When checking for possible matches of the outgoing edges, it is additionally checked whether the edge labels stand in relation $\mathscr{R}$.

Algorithm 5.3 Algorithm for Searching Answer Representations for a Given Question Representation

Input: $T, Q, \mathscr{K}^*$, document collection representation, question representation, knowledge base ('compiled')

Output: Nodes $w \in T$, such that a possible answer representation is rooted in w

```
1:  function FIND_ANSWERS($T, Q, \mathscr{K}^*$)
2:      for all $u \in V$, sorted in bottom-up sequence do
3:          for all $u' \in V'$, sorted in bottom-up sequence do
4:              $a(u,u') = 0$
5:              if $W(u') = 1$ then                                  ▷ wh-node
6:                  for all $\langle l_N, l'_N\rangle \in \mathscr{K}^*.\mathscr{Q}$ do
7:                      if $\langle u', l'_N\rangle \in P' \wedge \langle u, l_N\rangle \in P$ then
8:                          $a(u,u') = 1$
9:              else                                                 ▷ 'normal' node
10:                 $Y \leftarrow \{v'_1,\dots,v'_k\}, k \geq 0, \langle u',v'_1\rangle \in E',\dots,\langle u',v'_k\rangle \in E'$   ▷ all children
11:                 for all $\langle\langle l_N, l'_N\rangle, \mathscr{R}\rangle \in \mathscr{K}^*.\mathscr{P}$ do
12:                     $X \leftarrow \{v_1,\dots,v_l\}, l \geq 0, \langle u,v_1\rangle \in E,\dots,\langle u,v_l\rangle \in E$   ▷ all children
13:                      $G \leftarrow \langle X \cup Y, F\rangle$,
                         $F = \{\langle x,y\rangle | x \in X, y \in Y, a(x,y) = 1,$
                        $\langle\langle u,x\rangle, l_E\rangle \in R, \langle\langle u',y\rangle, l'_E\rangle \in R', \langle l_E, l'_E\rangle \in \mathscr{R}\}$
14:                     if $|\mathrm{MMG}(G)| = k \wedge \langle u', l'_N\rangle \in P' \wedge \langle u, l_N\rangle \in P$ then
15:                         $a(u,u') = 1$
16:         if $a(u, \mathrm{ROOT\_NODE}(Q)) = 1$ then
17:             OUTPUT($w$)
```

Complexity. Compared to the original algorithm, the search in the linguistic knowledge base for each node and each edge must be accounted for: For every edge matching, a suitable relabelling in the linguistic knowledge base must be found. Algorithm 5.3 assumes a linear search through all possible relabellings of the knowledge base (ll. 11, 13). We will designate the number of relabellings in the linguistic knowledge base by $|\mathscr{K}^*|$. By inserting this factor for each edge matching into Kilpeläinen's computation of the overall complexity (Kilpeläinen, 1992, 42), we arrive at the overall worst-case complexity of $O((m \times |\mathscr{K}^*|)^{3/2}n)$, with $|\mathscr{K}^*|$ the number of edge relabellings in $\mathscr{K}^*$.

Note how the generation and search are interleaved: The algorithm generates relabellings for elementary trees only. Each relabelling is directly matched against the document collection representation. If no match can be found, this possible relabelling is immediately discarded. Thus, the algorithm no longer generates all possible answer representations and searches them one after the other, but rather constructs a small part of an answer representation and immediately checks whether this is compatible with the answer found so far.

Pre-Computing Indices. As both the document collection representation and the knowledge base are known, the linear searches in the algorithm can be replaced by more efficient ones that use indexing. For example, we can arrive at logarithmic time complexity by using an index based on balanced trees.

Thus, the lookup in the knowledge base can be reduced from $|\mathscr{K}^*|$ to $\log(|\mathscr{K}^*|)$. For every node in the question representation, only the potentially matching nodes in the document collection representation are returned by an index lookup and need to be further considered. Since this set of potentially matching nodes will typically be small, search time is greatly reduced. Note that this latter speed-up holds only for the general, but not for the worst case: For a document collection representation where all nodes match all nodes in a given question representation, the algorithm would still need to go through all nodes. However, this is not a realistic case.

5.3.2.3 Practical Speed-Up

The algorithm as it stands does not yet return full answer representations but only the respective root node. It can be extended to return all possible answer representations. As their number may, at least in principle, be exponential with regard to the number of nodes in the question representation (cf. 5.3.2.1), we will use additional pruning techniques.

Retrieving Answer Representations. The search algorithm (alg. 5.3) returns only nodes in the document collection representation where possible answer representations are rooted. Note that the actual correspondences between nodes and edges in the question and the answer representation are checked, but immediately discarded again.

To retrieve full answer representations, the algorithm would therefore have to be extended to store additional intermediate results: For each pair of nodes from question and answer representation, not only the fact whether they match, but also the ‘alignment’ of the sub-trees would need to be stored. This could be done compactly, as for each pair of nodes only the matching of the outgoing edges needs to be stored; the information about how the child nodes can be matched could then be retrieved from a similar structure for the child nodes.

Note that, in general, there may be more than one way in which two nodes can be matched. Thus, for each pair of nodes, one would need to store *all* possible mappings of the outgoing edges.

Retrieving answer structures could then be done by looping recursively through the intermediate results for each identified answer root node and building up all possible combinations of all possible different matches for the nodes. In the worst case, the number of possible answers may grow exponentially with the size of the question representation (cf. 5.3.2.1).

We therefore use two pruning techniques, described in the following: First, we limit the number of local inference steps in the linguistic knowledge base. Second, we only retrieve and combine the elementary trees with the highest relevance scores.

Limiting the Number of Local Inference Steps As stated above, the number of possible answer representations grows exponentially with the size of the question representation (cf. 5.3.2.1). The exact number is influenced by the size of the underlying linguistic knowledge base – more precisely, by the maximal number of relabelling rules that match a single node (edge) label for each linguistic level.

Therefore, it makes sense to keep the number of applicable relabelling rules small to achieve better performance, both in searching and retrieving answer representations. This can be achieved by limiting the number of local inference steps that are allowed per node/edge: The original definition (def. 5.6) does not restrict the number of local inference steps. Thus, when pre-computing textual inference, the transitive closures of the relations $\mathscr{P}$ and $\mathscr{R}$ are computed. If, instead of computing the transitive closure, for every $\mathscr{P}$ and $\mathscr{R}$ only transitive relations up to a limited depth are compiled, the number of possible answer representations to be searched for can be restricted.

We have currently limited this maximum depth k to three. This means in particular that only hypernyms and hyponyms of third and lower degrees will considered as candidates. As the evaluation shows, this does not affect accuracy (7.2.2.2). This seems natural because limiting k corresponds to a sharp cut-off in local relevance (question and answer representations that contain a node that cannot be matched with a relevance above a local threshold is considered as altogether irrelevant).

Pruning. We have defined above how relevance values can be computed on the basis of information in the knowledge base for textual inferences and answer matching (defs. 5.6 and 5.8). We can make use of the relevance value for pruning during answer retrieval.

The relevance value can be used to select the edge mapping that leads to the highest relevance for that node, for every pair of nodes in question and answer representation. For all pairs of child nodes, the relevance value for the optimal match is known. When computing the possible edge matchings, these relevance values and the relevance values for the different edge relabellings can be taken into account to produce overall relevance values. Then, for each node only the optimal matching with regard to relevance values must be stored and, during retrieval, only one single answer representation, namely the optimal one, needs to be retrieved.

5.3.2.4 Conclusions

In this section, we have described how the relation of indirect answerhood can be integrated into an efficient search algorithm. This algorithm is based on a known algorithm for matching trees via unordered path inclusion.

We have shown that this provides only an approximation and may overgenerate in cases where the question contains certain coreference phenomena, but have argued that these cases are practically not relevant.

The underlying search algorithm exhibits a worst-time complexity of $O(m^{3/2}n)$ with regard to the number of nodes in question representation and document collection representation, respectively. By integrating textual inferences into the algorithm, the size of the linguistic knowledge base becomes an additional complexity factor.

However, indices can be computed for the knowledge base and the document collection representation, since both are known in advance: Search can be changed from linear to efficient, logarithmic lookup in the general case.

We have finally shown that, if only the most relevant answer representations are considered, retrieval can also be made more efficient.

5.4 Storing and Retrieving Structured Representations

We will show how the algorithm for searching answer representations can be implemented quite straightforwardly using a general relational database system. Using such a system has two advantages for a practical implementation of the search algorithm: On the one hand, it allows to store and retrieve large amounts of text representations in an efficient way. On the other hand, searching for elementary representations and checking for constraints can be directly integrated into the database search.

We will first motivate the choice of a generic relational database (5.4.1).

We will then outline the database scheme that we use: As generic relational databases provide no support for storing tree or graph structures, we have defined a 'flattened' representation that is mainly based on storing nodes and edges in separate tables and using the edge table as a (database) relation between nodes (5.4.2).

We then use the search algorithm defined above (5.3.2.2) as starting point for defining a version that systematically builds up database queries from given question representations (5.4.3).

5.4.1 Using a Database System for Storage and Retrieval

Linguistically informed QA as we have defined it is centred around the storage and efficient retrieval of structured text representations of the system's text collection. We use a standard relational database system for this purpose. We will summarise the motivations of this choice in this section.

A QA system should be able to handle quite large amounts of data: The raw texts for shared competitions like TREC or CLEF are currently around the 1 GB mark in size. At the time of writing (2006), most desktop computers feature about 512 MB to 2 GB in main memory.

This means that for a QA system that is to run on standard hardware, it will not be possible to keep all required data in main memory, especially since storing structured data derived from the raw texts in a way that permits efficient access (e. g., balanced tree indices) will require markedly more space than storing the underlying raw text. Therefore, the QA system needs to be able to store and efficiently retrieve data on hard disk.

General, off-the-shelf database systems provide a flexible and efficient possibility of storing and accessing large amounts of data. They are a well-researched area in computer sciences (cf., e. g., Abiteboul et al., 1995). A number of mature general-purpose systems are available today, such as MySQL and PostgreSQL (both open source systems) or the ORACLE database system and IBM DB2 (both commercial system). Currently, most systems make use of the successful relational database paradigm (Abiteboul et al., 1995).

Database Systems for Structured Data. It would be ideal to employ a database system that provides built-in support via native data types and optimised retrieval techniques for tree (graph) structures. The past few years have seen an increasing interest in such systems, especially fuelled through the growing amount of data that is stored in XML format (W3C, 2004a): As XML naturally provides a tree structure and can thus very easily be used to structure data in any tree-based format, there is a growing demand for databases that directly support handling XML-formatted data.

This demand has lead to the definition of generalised query languages for XML structures, especially XQuery (W3C, 2006) and XPath (W3C, 1999) and, quite recently, also to the inclusion of XML modules into commercial databases (Oracle, 2005). Note however, that these XML modules currently do not provide native support but are rather extensions that make XML and XQuery interfaces available to the user, while data is stored in a 'classical' relational database, using a scheme not unlike the (simpler) one we define below.

The theoretical properties of tree-like structures in databases and algorithms for handling them are currently being researched (Gottlob et al., 2004; Gottlob and Koch, 2004). Only experimental systems with native support for tree-like structures and queries exist. They are not yet geared towards actually handling large amounts of 'real' data and provide no general support for generic queries.[5]

Standard Relational Database Systems. At the time when we started this project, using a general off-the-shelf relational database system and hand-coding the required data structures and queries was therefore the obvious choice. This was supported by the following considerations: Database systems as described above are readily available, several systems are even distributed as open source. SQL (Structured Query Language, now an ISO-standard; the currently latest revision of the standard is ISO/IEC 9075:2003, SQL:2003) has evolved as a standard interface language to these systems. As mature systems, these database systems are well documented and supported. Even though the described systems are not optimised for our specific requirements, they are highly optimised for a wide range of generic queries; query optimisation is a well-researched field, and many results have by now be integrated into the available systems.

[5]Christoph Koch, personal communication, July 2005.

In the light of more recent developments, it might be worthwhile to reappraise the situation; using a database with XML support might now be a viable solution.

5.4.1.1 A Note on Complexity of Database Queries

We have investigated algorithmic descriptions of the search algorithm under complexity considerations above. By using a database system, we will let the database engine take over a part of the search. It is therefore interesting to look at the general complexity of database queries. This is, of course, an important area in database theory and a number of general results are known. The following short summary follows Abiteboul et al. (1995, esp. 429 ff.).

In general, queries over databases can be regarded under three different aspects regarding complexity, namely data complexity (i. e., considering the query expression as fixed and only regarding the complexity induced by evaluating it over databases of different sizes), expression complexity (i. e., considering the data in the database as fixed and only regarding the complexity induced by queries of different sizes) and combined complexity (Vardi, 1982). As the combined complexity is generally in the same class as expression complexity it is rarely listed separately.

Queries over relational databases[6] are known to be LOGSPACE-complete[7] regarding data complexity and PSPACE-complete[8] regarding expression complexity (Vardi, 1982; Chandra and Merlin, 1977).

This means that database access can be performed in polynomial time – this explains the huge success of relational databases and query language like SQL, whose core (namely conjunctive queries) thus allows efficient computation.

In practice, current database engines are fast for general queries. A number of techniques is known for optimising queries, for example, by deleting redundancies or by ordering sub-queries in a way that ensures that constraints on the query are computed first and taken into account during the actual search (cf. Abiteboul et al., 1995, 105–40).

[6]More exactly, conjunctive queries over relational databases; this is equivalent with 'basic' queries over relational databases, cf. Abiteboul et al. (1995).

[7]The complexity class LOGSPACE (problems for which algorithms exist that can be computed on deterministic Turing machines using space whose size is logarithmic with regard to the size of the input) is known to be included within PTIME (problems for which algorithms exist that can be computed on a deterministic Turing machine within polynomial time with regard to input size). This means that for each problem in the complexity class LOGSPACE an algorithm exists that is at least as efficient as for those in PTIME. It is generally assumed that this inclusion is proper, i. e., that more efficient algorithms exist for problems in LOGSPACE. See also Garey and Johnson (1979).

[8]The complexity class PSPACE (problems for which algorithms exist that can be computed on deterministic Turing machines using space whose size grows polynomially with regard to the size of the input) is, again, known to be included within PTIME.

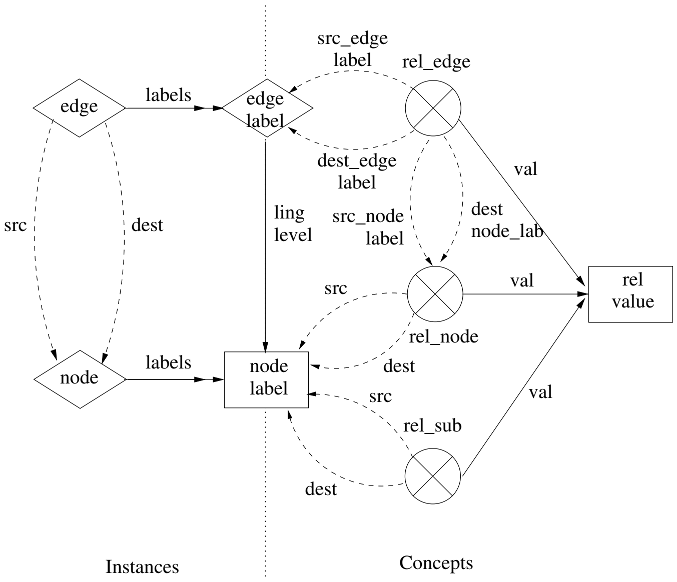


Figure 5.16: Database Scheme for Flattened Text Representations

As the number of disk accesses required to retrieve the queried data is the ‘bottleneck’ for all database queries, they use a variety of heuristics for keeping as much data as possible, especially as many indices as possible, in main memory to avoid disk access as far as possible (cf., e. g., the MySQL database system documentation, `http://dev.mysql.com/doc/refman/4.1/en/`). Thus, one can expect very good performance for general queries. We will come back to the issue of database optimisation below (6.3.5).

5.4.2 Database Scheme

A general, off-the-shelf relational database system is used for storage and efficient retrieval of text representations (5.4.1).

As standard database systems do not directly support storing and retrieving graph structures, we need to define a graph representation format that is suited for use with relational databases. We will describe this flat representation here and show how queries that retrieve graph structures can be built up automatically from given question representations.

The ensuing database scheme is shown in fig. 5.16 (as a Generic Semantic Model, cf. Abiteboul et al., 1995, 240–269). As indicated in the figure, both text representations and a representation of the linguistic knowledge base are stored together in one database. From a linguistic point of view, the information contained in the knowledge base is about concepts, while that contained in the text representations represents actual linguistic instances.

To store the text representation in a relational database, we flatten the graph structures in the following way: Each node is represented by one row in a node table; each edge is represented by a complex class, where one row in an edge table that stores one pointer to its start node and one pointer to its end node. Node and edge labels are represented as (multi-valued) attributes of nodes and edges, respectively. These also link the 'instance side' to the 'concept side'.

The relations between node and edge labels contained in the linguistic knowledge base are again represented as complex classes, with source and destination labels as members and the corresponding relevance values as attribute.

5.4.3 Retrieval

In 5.3, we have defined answer search for a given question (representation) by deriving possible answer representations from the question representation and searching for them in turn within the text representations derived from the QA system's text collection. Based on the database scheme just defined, we can now sketch an algorithm that allows us to integrate the derivation of possible answers and the search into a single database query. By using appropriate indexing for the database, the table joins within the query can, in general, be efficiently executed by the database engine. By using indexed joins the practical efficiency can – in general – be crucially improved, as the number of required disk accesses can be reduced (cf. Abiteboul et al., 1995, 105–140).

To retrieve text representations that are stored in a relational database using the database scheme just described, the database query must be formulated in a way that encodes the desired structure adequately, i. e., by explicitly flattening it out in the same way. We will describe this process here.

The core algorithm for systematically constructing SQL queries from question representations is given as alg. 5.4. The algorithm is very similar to the basic algorithm 5.3. It assumes the database structure given in fig. 5.16. In the description, we will refer to the line numbers in alg. 5.4.

The SQL query for a question representation is built up by looping through the nodes in the question representation bottom-up and constructing and executing, for each node, a single query. Thus, a flattened representation for each

Algorithm 5.4 Constructing SQL Queries from Question Representations

```
 1: function SQL_QUERY($Q$)
 2:     for all $u' \in V'$, sorted in bottom-up sequence do
 3:         $uid \leftarrow$ GET_ID($u'$)
 4:         $S \leftarrow$ MAKE_SQL_QUERY(INSERT INTO TABLE tmp SET (node, node_inst) SELECT $uid$, node_$uid$.key FROM node AS node_$uid$)
 5:         ADD_TO_FROMS($S$, JOIN node_label AS node_label_$uid$ ON node.labels = node_label.key)
 6:         ADD_TO_FROMS($S$, JOIN rel_node AS rel_node_$uid$ ON node_label_$uid$.key = rel_node_$uid$.dest)
 7:         $W \leftarrow \emptyset$
 8:         for all $l'_N, \langle u', l'_N \rangle \in P'$ do
 9:             QUEUE($W$, OR rel_node_$uid$.src = $l'_N$)
10:         ADD_TO_WHERE($S, (W)$)
11:         for all $v', \langle u', v' \rangle \in E'$ do
12:             $uid \leftarrow$ GET_ID($v'$)
13:             ADD_TO_FROMS($S$, JOIN edge AS edge_$uid$_$vid$ ON edge_$uid$_$vid$.src = node_$uid$.key)
14:             ADD_TO_FROMS($S$, JOIN node AS node_$vid$ ON edge_$uid$_$vid$.dest = node_$vid$.key)
15:             ADD_TO_FROMS($S$, JOIN edge_label AS edge_label_$uid$_$vid$ ON edge_$uid$_$vid$.labels = edge_label_$uid$_$vid$.key)
16:             ADD_TO_FROMS($S$, JOIN rel_edge AS rel_edge_$uid$_$vid$ ON edge_label_$uid$_$vid$.key = rel_edge_$uid$_$vid$.dest)
17:             $W \leftarrow \emptyset$
18:             for all $l'_E, \langle \langle u', v' \rangle, l'_E \rangle \in R'$ do
19:                 QUEUE($W$, OR rel_edge_$uid$_$vid$.src = $l'_E$)
20:             ADD_TO_WHERE($S, (W)$)
21:             ADD_TO_WHERE($S$, rel_node_$uid$.src = rel_edge_$uid$_$vid$.src_node AND rel_node_$uid$.dest = rel_edge_$uid$_$vid$.dest_node)
22:             ADD_TO_FROMS($S$, JOIN tmp AS tmp_node_$vid$ ON tmp_node_$vid$.node_inst = node_$vid$.key)
23:             ADD_TO_WHERE($S$, tmp_node_$vid$.node = $vid$)
24:         EXECUTE_SQL($S$)
25:     return EXECUTE_SQL(SELECT node_inst FROM tmp WHERE node = GET_ID(ROOT_NODE(Q)))
```

node with its outgoing edges is retrieved as a single joined table. The core idea is as follows: For each node, the node itself, its outgoing edges and the respective nodes that are the destinations of the outgoing edges are each represented by a distinct node (edge) table. These tables are joined in a way that exactly represents the structure of the node. By executing the join, the table is filled with suitable answer representations. The joined table then contains, for every node and every edge in the question representation the *corresponding* instances (rows) as stored in the document collection representation.

We assume that the SQL query is built up as a structured representation; the procedures ADD_TO_FROMS and ADD_TO_WHERE are assumed to insert additional table joins to the *from* part and additional constraints to the *where* part of the query in *S*.

As above, we give – for ease of exposition – the simplified version of the algorithm that only returns the root nodes of matching structures but not the structures themselves.

Like alg. 5.3, the algorithm relies on intermediate results to be available for all child nodes. The place of the Boolean array that stores intermediate results is taken by a temporary SQL table tmp (ll. 4, 22–23).

We assume a function GET_ID that returns a unique identifier for a given node or edge. These unique node (edge) identifiers are then used to construct column names for the SQL query.

In some more detail, the algorithm works as follows: For every node (for every edge) contained in the question representation, a reference to the node table (to the edge table) is added to the query (ll. 5, 13, 14). For the edges, a join condition is added that ensures that the start node and that of the end node of the edge are identical to those within the query (join conditions on ll. 13, 14).

Now, we need to add a match of node and edge labels to this ‘skeleton query’ that already encodes the graph structure of the question representation. This is done by adding, to every node and every edge, a constraint based on the labels present in the question representation (ll. 6, 15). Labels of question representation and answer representation are not matched directly, but via the rel_node (rel_edge) table of the database representation of the linguistic knowledge base. This ensures that relabelling can be done during matching (ll. 9, 16).

As we have argued above that only one label needs to match per elementary tree, we will encode the labelling match as a local disjunction (ll. 7–10, 17–20).[9]

[9]Note that by introducing disjunction, the space-complexity of the ensuing query may, in general, grow exponentially, cf. Abiteboul et al. (1995, 502–503). As the number of different linguistic levels (and thus that of different labels) is very small and as the disjunction is fully local, i. e., only extends over one table (recall that the domain of disjunction is defined to be the elementary tree), this does not, in practice, lead to intractable queries.

Finally, the matching is constrained by ensuring, for every child node, that it was actually matched against the corresponding node in the question representation. This is done by selecting the suitable rows from the tmp table (ll. 22–23).

Note that some details are not shown in this core algorithm, such as the handling of *wh*-nodes, the additional constraints to ensure that the outgoing edges of each node be distinct and the handling of relevance values and pruning. All of these can be added easily; we have left them out here for ease of readability.

5.5 Conclusions

In this chapter, we have defined the relation of indirect answerhood between text representations. Syntactic dependency structures form the core of the text representations; they are extended with lexical semantic information (here, GermaNet and frame information). The indirect answerhood relation is compounded by the relations of textual inference and direct answerhood; textual inference has been defined as local relabelling steps over text representations and direct answerhood as matching question representations against text representations (5.1).

We have shown how linguistic information, especially information derived from the resources introduced in chapter 4, can be utilised in a modular way to define a linguistic knowledge base that is needed both for textual inferences and direct answerhood (5.2).

Based on the concept of indirect answerhood, we have developed an efficient search algorithm. It is derived from a known algorithm for tree matching based on unordered path inclusion and interleaves generation steps with the search for partial structures to ensure efficient searching (5.3).

Finally, we have described how a database system can be used for storing and retrieving text representations. An adapted version of the search algorithm utilises the database engine for searching suitable representations, thus answering the need for a means of storing large text representations and efficiently searching them (5.4).

In the next chapter, we will describe the prototype implementation of a linguistically informed QA system for German, based on the system design detailed in this chapter.

Chapter 6

SQUIGGLI: A System for Question Answering in German Grounded on Linguistic Information

This chapter describes design and realisation of SQUIGGLI, a prototype German QA system that implements our approach of linguistically informed QA based on the relation of indirect answerhood. So far, we have specified (chapter 3) and defined (chapter 5) the relation. In this chapter we will describe the design of a working system that can automatically derive suitable text representations from German texts and make use of them for answering questions. This prototype system has been built as a proof of concept to show the viability of our approach.

In 6.1, we consider the overall system design that uses structured text representations for matching questions and answers: The most important consequence is that the text representations need to be pre-computed off-line.

In 6.2, we will describe a chain of Natural Language Processing modules that derives text representations as defined in chapter 5 from German texts: From the input, a dependency structure extended with lexical semantic information, namely GermaNet and FrameNet information, is derived (Fliedner, 2004a). We employ a cascaded chain of partial parsing and annotation modules (similar to Abney, 1996), comprising a topological parser (Braun, 2003), a named entity recogniser, a NP/PP chunker (Fliedner, 2002), a coreference res-

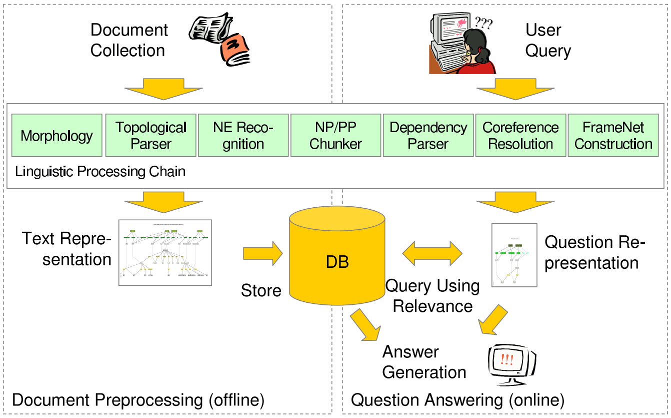


Figure 6.1: Design of the SQUIGGLI System

olution module and GermaNet and frame annotation modules. In the final QA system, this parsing chain is used to process both the text documents and the users' questions.

In 6.3, we describe a number of database issues, especially concerning query optimisations (Abiteboul et al., 1995, 106–8).

In 6.4, a description of our user interface is given, focussing on features that were integrated as a basis for interactive QA (Fliedner, 2006b).

6.1 Overall System Design

Figure 6.1 shows the overall design of the SQUIGGLI system: On the left-hand side, the document preprocessing phase is sketched, on the right-hand side, the on-line question answering phase. The same linguistic processing chain is used for deriving text representations from the document collection and from the users' questions.

This core system can, in principle, be combined with an existing QA system, especially to improve recall (i. e., to find answers in cases where linguistically informed QA cannot due to lacking coverage or robustness). In such a setup, both systems are run in parallel and results are merged during answer generation/answer presentation.

In chapter 5, we have defined the relation of indirect answerhood as a matching of structures text representations. This relation of indirect answerhood forms the core of linguistically informed QA.

To enable a linguistically informed QA system to efficiently search for an answer, text representations of the documents in the document collection therefore need to be precomputed and stored in a way that allows efficient retrieval.

This is an important difference from the general QA system architecture as described above (2.2.2): Most current QA systems use IR engines to retrieve candidate documents or passages that are likely to contain an answer to a given question and only perform linguistic processing on these candidate documents.

In consequence, a system for linguistically informed QA needs to perform a full linguistic analysis of the whole document collection off-line, that is, before any questions can be sent to the system. This can be seen as an extended indexing phase where all possible analysis steps are conducted off-line and no processing apart from the search proper is done on-line (cf. 2.3).

6.1.1 Off-line Processing

Since a full linguistic analysis of the whole document collection must be performed, off-line processing of a large document collection will be time-consuming. It is therefore advisable to parallelise that task, i. e., to distribute it over as large a number of computers as possible. The task of corpus preprocessing is well-suited for distribution: The corpus just needs to be split into sub-corpora whose number correspond to the number of available machines, these sub-corpora are then distributed over the machines, processed independently and the resulting outputs are finally joined again. The different sub-corpora can be processed independently from each other. Given the current cost of computer hardware, setting up a cluster of machines for corpus processing is a realistic option.

Using compute clusters to process linguistic corpora is becoming increasingly popular. oumaBOUMA et al. (2006) describes a QA system that performs a deep HPSG-style analysis over its document collection. In an experiment, the 300 million word Twente newspaper corpus was parsed. The process was run on the 128 machines of the Groningen Beowulf cluster in parallel, with an overall processing time of under a month (Bouma et al., 2006). The 80 million word sub-corpus of the Twente corpus that is used for the Dutch competition of the

QA@CLEF challenge (2.2.1), producing about 2.5 GB of parse data, can be processed in two to three weeks (Jörg Tiedemann, personal communication, Autumn 2006).

For the German QA@CLEF competition, a similar approach is described in Hartrumpf (2004). The described system constructs deep syntacto-semantic structures from all texts in the document collection as a basis for answering questions. The author reports that to reduce the overall running time for parsing the CLEF corpus (approximately 80 million words), namely around 5–6 months on a single CPU, they processed the collection on a small cluster of six machines.

Another well-known example of massively parallelising processing for IR is the Internet search engine run by Google: Google currently runs about 250,000 comparatively inexpensive machines[1] in parallel to both index the whole Internet and to allow efficient parallel search access for millions of users (Barroso et al., 2003).

These examples show that pre-processing corpora on parallel machines is doable and – due to dropping hardware cost – also comparatively affordable. Note that this pre-processing only needs to be done once per document collection. Later additions to the collection (e. g., adding one issue per day or even per week to a newspaper archive) can usually be done on a single machine within hours. Thus, the compute cluster is not needed 'full-time', but only for the one-time effort of generating the first document collection representation. However, it limits the approach to comparatively stable document collections, such as newspaper archives. On the Internet, which shows a rather rapid change of information and is orders of magnitude larger, such massive pre-processing seems currently not a realistic option. This may change with dropping hardware costs in the future.

6.2 Deriving Extended Syntactic Representations from Text

In describing linguistically informed QA, we have taken the parsing process that derives text representations from texts for granted (cf. def. 5.3). In an implementation of a linguistically informed QA system, they need to be automatically derived from input texts by suitable modules.

[1]As Google does not publish exact figures, this is based on an estimate. See `http://en.wikipedia.org/wiki/Google_platform`.

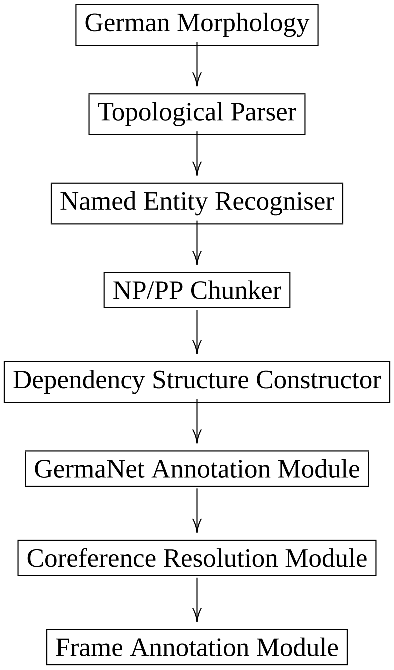


Figure 6.2: System Components in Parsing

Figure 6.2 shows the system components in our prototype system. The modules form a processing chain; every component has access to the results of the earlier modules and builds on their results.

Before describing the different contributing modules, we will first motivate the choice of parsing architecture that we use, viz. a cascaded chain of partial parsers (similar to the ideas described in Abney, 1996).

6.2.1 Overall Parsing Architecture

In this section, we will describe the overall parsing architecture that we have chosen, namely a chain of cascaded parsing modules. This architecture is loosely based upon Abney (1996). We will first motivate the choice by listing a number of requirements derived from the overall system design. Then we will describe the chosen architecture in more detail before turning to other possible alternatives.

6.2.1.1 Requirements

In selecting suitable parsing components for a linguistically informed QA system (cf. chapter 5; 6.1), a number of – at least partially – conflicting requirements, constraints and desiderata can be identified:

Fine-Grained Dependency Information. Linguistically informed QA as described in chapters 3 and 5 is built upon the idea of matching question and answer representations containing a fair amount of linguistic information using controlled inferences, licenced mostly by information on syntactic and lexical semantic variation. It is this pervading use of structured linguistic information that crucially distinguishes linguistically informed QA from other work on QA (2.2.1) that puts its main emphasis on recognising named entities and surface patterns, viewing QA as an application of information extraction techniques (see, e. g., Soubbotin and Soubbotin, 2003; Soubbotin, 2002; Srihari and Li, 2000): Linguistically informed QA relies on using fuller linguistic information.

This means, however, that relatively complete and detailed linguistic structures must be available both for the questions and for the texts in the system's document collection in order for answers to be found. Consequently, the parsing component must correctly identify dependency relations, label them correctly and add lexical semantic information.

Broad Coverage. As full representations are required for linguistically informed QA to work properly, it is especially important that the parsing component reaches a broad coverage. Only for input sentences for which an (at least partial) structured representation is available, inferences can be drawn and answers can be found, so the overall system coverage directly depends on the coverage of the parsing component. Thus, the parsing component needs to be designed in a way that a useful representation can be derived for as many ‘typical’ input texts as possible. This means especially that both the typical vocabulary and the typical grammatical constructions must be covered by the system grammar.

Robustness. Another key requirement for the parsing component is that it should be robust. We note that the term robustness is often used in computational linguistics but hardly ever defined (cf. Fliedner, 2001; Stede, 1992). By robustness we refer here to the system’s ability to respond appropriately to unforeseen input, that is, input that is not covered by the system’s grammar. Note that this definition encompasses two cases, namely ‘absolutely’ ill-formed input, i. e., input that actually contains some error, such as a misspelling or a grammatical error, and ‘relatively’ ill-formed input, i. e., an input that human readers would consider grammatical but that is not covered by the system’s grammar (Stede, 1992). A robust system should, for every input, even input that cannot fully be analysed by the system grammar, not fail completely, but rather return (at least partial) structures as a result and should gracefully (‘monotonically’) degrade (Menzel, 1995). For input that contains, for example, an unknown word or some parenthetical construction, the parsing component should still derive at least a partial representation. This point is directly linked to the preceding one: In addition to a broad coverage, the system needs to gracefully handle all sorts of unexpected linguistic (and also meta-linguistic) material in the input.

Uniform Structures for Different Levels. Structures for different linguistic levels (especially the syntactic and lexical semantic levels) need to be structurally uniform. This was described in more detail in 5.2.2.1, where we have also argued that this is a linguistically plausible assumption.

Therefore, deriving the different description levels either by an integrated process or by a process where different level consecutively build on top of each other (this is the general approach that we have used) will in general be simpler, as no conflicts for the different levels can ensue. If different modules and/or different techniques are used to build structures for the different levels (if, for example, frame structures are derived from text by a process that is completely different from the one that derives the syntactic dependency structures), then the integration of these different structures may turn out to be difficult.[2]

Tokens of the input text could be used as the basis for a stand-off annotation (cf. 6.2.2). We note in passing that this is not trivial: When tokenising texts, different decisions with regard to what *is* a token can be taken; and even then, automatically distinguishing different cases is far from easy, cf. Grefenstette and Tapanainen (1994). Different levels can be related by describing linguistic entities through their token spans in the original text in the stand-off annotation. However, resolving conflicts between the annotations is not simple: How would, for example, cases be handled where two components identify two linguistic entities differently so that the respective text spans overlap but that there is no inclusion? For a similar discussion, see also Frank et al. (2003): In their approach, results from shallow and deep parsers are integrated through a set of manually written rules with confidence scores.

It should be noted, however, that by using completely separate systems for deriving the different parts of the overall representations, additional robustness can be achieved: In cases where one system fails on an input, a completely independent different system may still derive a useful representation. For example, it could be possible to assign a frame representation to an input even though no dependency structure could be found.

High Parse-Speed. We have already noted above that an efficient parsing component is needed to realistically implement linguistically informed QA: As the whole document collection that is to be used to answer questions must be completely parsed, parsing time is an important limiting factor of the approach (cf. 6.1).

In addition, the parser should be fast enough for processing the user's questions in an on-line QA session: To be practically usable, a QA system (especially one that is intended for interactive QA, cf. 6.4.4) must return answers within a few seconds: We would consider that an average answering time of ten seconds will already feel distinctly slow to a user, while answering times of more than twenty seconds are probably unacceptable.

Note that this applies to short questions and answers. For more complex, especially for open questions, longer answering times may be acceptable.

[2]The three approaches outlined here have been discussed in literature on semantic construction (especially in the context of LFG) as integrated description, description by analysis and co-description, respectively, see Halvorsen and Kaplan (1995); Frank and Erk (2004).

> In Sizov et al. (2003), for example, a ‘focused web crawler’ is suggested: The user takes a few minutes to set up an initial query. The system then searches web pages fitting this query and visits web pages in the neighbourhood of these initial pages to gather further information. After several hours (generally on the next working day), the user will receive a detailed, structured summary of information that is relevant to the original query.
>
> As question parsing is only one part of the answer finding process (querying the database will also take time), this again indicates that the parser should preferrably process an input sentence in less than a second.
>
> Note that another interesting time limit that could be called the absolute upper time limit for QA can be established: As described above, Gregor Erbach has shown in an experiment (Erbach, 2004) that untrained users will find a satisfactory answer to the questions of the 2004 QA@CLEF task using a standard Internet search engine in an average time of 77 seconds. Taking the accuracy of actual current systems into account, Erbach found that users do at least as well as a QA system if only answers that they found within 30 to 40 seconds were counted. He concludes that QA systems will only be considered helpful if they find answers faster than users could themselves using standard search engines.

Most of the requirements enumerated above argue for using what has generally been labelled shallow parsing approach: Both broad coverage and robustness are associated with shallow rather than with deep approaches, as deep approaches generally rely more heavily on linguistic knowledge, especially in the form of very detailed system grammars. As such grammars are difficult and costly to compile, there seems to be a certain inherent contradiction between such knowledge-rich approaches, on the one hand, and broad coverage and robustness, on the other hand. On the other hand, too little linguistic knowledge goes against broad coverage: A system whose lexicon covers only a small percentage of the input words will not be able to derive any meaningful structures from the input.

High processing speed is also taken to be a feature of shallow rather than deep approaches, as they use less information and thus generally have to explore smaller search spaces. Note that through better parser implementations and the use of efficient pruning techniques, the gap between deep and shallow parsers is closing fast. There are, e. g., very fast implementations of parsers for HPSG grammars, that – by definition – use very much linguistic knowledge (Callmeier, 2000).

These points seem to recommend the use of shallow techniques for parsing. However, as a relatively detailed and rich output structure is required, we had to find some middle ground. We will now turn to describing this approach in more detail.

6.2.1.2 Cascaded Parsing

We decided to use a relatively shallow approach, namely a system of simple cascaded parsers, where each parsing module is responsible for one linguistic level, each of them using only limited linguistic knowledge. This concept is loosely based on Abney (1996, 1991). These are the key points of the approach:

Separation of Labour. In our system, different levels of linguistic structures directly correspond to one processing module each. Some examples are morphological analysis (6.2.3), topological parser (6.2.4), NP/PP-chunker (6.2.6) and coreference resolution (6.2.9). Each of these modules encodes only knowledge about that particular level, namely how structures discovered by earlier modules are to be combined.

In Abney (1996), a uniform cascaded chain of finite-state transducers is used. We use a range of different techniques for the different modules. However, the underlying idea of separating levels of linguistic structures into different modules is essentially the same.

Sequential Processing. All modules are called sequentially, with each module using the output of the previous module as its input. Each module is called only once per processed sentence, there is no recursion within the processing chain. Data exchange between the modules is done on XML structures; these are consecutively enriched with the output from the different modules (see also 6.2.2).

By sequential processing, partial structures that do not contain information at all levels may be derived.

Easy-First Parsing. The different processing modules cannot always take all possible decisions. For example, subject and object of a verb may be syntactically ambiguous, so that lexical semantic information (e. g., when the head of one phrase is marked as animate and the matrix verb subcategorises for an animate/agentive subject) is needed for disambiguation.[3] In such cases, the different modules may leave their output un-

[3]Recall that German has relatively free word order. Therefore, in cases where a sentence with a transitive matrix verb in active voice contains two NPs that are morpho-syntactically ambiguous between nominative and accusative case, no certain decision on which phrase serves as subject and which as object can be reached at syntax level.

derspecified and mark it accordingly; the following processing modules will then, if possible, resolve the underspecification. This feature will be discussed in greater detail in 6.2.7.1.

In most cases, however, the separate modules will be able to come up with a single, disambiguated solution (parse) for the input. This solution is, in general, locally optimised, so that there are rare cases when the derived structure is not globally optimal (e. g., the NP-chunker lumps two NP chunks into one single chunk erroneously). The overall aim was to reach a good compromise between producing as few ambiguous (and thus, underspecified) output structures as possible, on the one hand, and as few wrong ones as possible, on the other hand.

Island Parsing. With the exception of the topological parser (6.2.4), none of the modules require that the structures it constructs span the whole input. They rather proceed selectively by building structures that can be recognised with a high degree of certainty and leaving other material alone. This leads to a comparatively high robustness with respect to unexpected material (such as unknown words or parenthetical expressions), as it may be skipped during parsing. Approaches along these lines are generally classed under the term of island parsing (Stock et al., 1988). Steven Abney fittingly calls the parts of the input that can be analysed with a high degree of reliability ‘islands of certainty’ (Abney, 1996, 10). With regard to Manfred Stede’s classification of different strategies for achieving robustness, we thus mainly use the fourth general strategy for achieving robustness, namely ‘Represent only those parts of the input that you understand’ (Stede, 1992, 410).

Processing Speed. The different modules only have to construct relatively shallow structures with few levels – if any – of local recursion. There are two levels at which recursion plays a rôle: The topological parser has to deal with embedded subordinate sentences and the NP/PP-chunker with embedded NP/PP structures. While there is no theoretical limit for these types of recursion, their depth in ‘real’ sentences hardly exceeds three or four. Each module only has to explore a comparatively small search space, resulting in an overall processing speed that is sufficient for our purposes.

We will now describe two possible alternatives. The main reason for not choosing one of the alternatives was that no comprehensive, running system for German was available when we started our work (there is still no system that we know of that integrates all the different linguistic levels up to and including GermaNet and FrameNet information for German).

Besides, we considered it an advantage to work with system components that we could relatively easily change and adapt to the purposes at hand.

6.2.1.3 Alternative 1: Statistical Parsing

Statistical parsers need to be trained on a suitable corpus to construct a model from the training data. When used on new input they will return a description of this input that is optimal with regard to the learned model. Quite generally, statistical methods have two important advantages: Rather than making binary decisions for or against one description of the input or another, they assign probabilities to different possible descriptions. This generally leads to an increase in robustness: Even if the training data did not contain examples for a certain combination of features, the system may still be able to make a good guess. The second advantage is that they are generally relatively fast (see below).

Recent interest in statistical methods has lead to a steady increase in system performance: In evaluations on material that is similar to the respective training material, statistical part-of-speech taggers reach about 97 % accuracy (Brants, 2000), NP/PP-chunkers F-scores of about 84 % (Skut and Brants, 1998b) and full stochastic parsers about 89 % for English (Charniak, 2000) and 79 % for German (Dubey, 2004). Statistical methods for (English) Coreference Resolution reach F-scores of about 65 % (Uryupina, 2006). In the complex task of FrameNet annotation, 75.1 % F-score (English) and 60.0 % F-Score (German) have been achieved for frame element recognition (for perfectly assigned frames) and 78.4 % accuracy (English) and 67.3 % accuracy (German) for frame element labelling (Erk and Padó, 2006).

This short summary shows that statistical methods generally reach very good results (mostly at least on a par with symbolic methods); the results from the evaluations also generally reflect the difficulty of different natural language processing tasks quite well: While the first tasks listed seem to be comparatively simple, the latter ones are far more difficult and often also potentially controversive.

As most methods are well suited for the use of efficient pruning techniques that can be used to keep the search space of possible solutions small, statistical methods are generally fast and allow efficient processing: The Viterbi algorithm, for example, that is often used for part-of-speech tagging and also for speech recognition, provably generates optimal results based on local pruning (Viterbi, 1967).

6.2.1.4 Alternative 2: The Hybrid Approach

Recently, combining deep and shallow methods in a single approach (often called hybrid approach) has gained a lot of interest in computational linguistics. One good example is the Heart of Gold (HoG) system (Callmeier et al., 2004)[4].

The central idea is to use a common output formalism (in the case of HoG, Robust Minimal Recursion Semantics, RMRS, Copestake, 2006). All modules may enter their results into a joint output structure. Information flow between the modules is organised by a central communication management module. This may be seen as a (controlled) blackboard architecture (Nii, 1986). A set of hand-crafted rules resolves conflicts between the results from different modules (Frank et al., 2003).

In Callmeier et al. (2004), an example is given where the output from a specialised named entity recogniser for product names is used as input for a deep HPSG parser (Crysmann et al., 2002; Müller and Kasper, 2000). This is a plausible division of labour, as there would be little point in describing scores of product names in the HPSG grammar.

In addition, the communication management module can be used to set timeouts for the different modules. This feature can especially be employed to limit the time alloted to deep processing modules: If no solution is found within a certain amount of time that parsing process is cancelled. In this case, only a partial structure will be derived for the input (containing, e. g., only lemmatisation, part of speech tags and NP/PP-chunk annotation). However, this partial structure may still be sufficient for a number of purposes (cf. also Frank et al., 2003).

The approach described in Daum et al. (2003) is related, but differs slightly. Here, results from two shallow parsers (TnT as part of speech tagger, Brants, 2000 and TreeTagger as NP-chunker, Schmid, 1994) are used to control a ‘deep’ dependency parser through additional weighted constraints. Thus, there arises no need to integrate partial structures from different parsers. Rather, the results from the shallow components are used to select an optimal solution from the possible parses constructed by the deep parser.

The most important reason why we have not used a hybrid approach (and, particularly, not the HoG system) was that no running system was available for German when we started our work. Note that, currently, the Heart of Gold package does not include a GermaNet and frame annotation module; suitable additional modules would have to be integrated.

[4]Actually, ‘Heart of Gold’ rather refers to a framework and an architecture for integrating different language processing tools. We will follow the authors and also use the name to describe a current system incarnation based upon that architecture.

6.2.2 Implementation Issues

Most of the modules described in this chapter were implemented by the author and Christian Braun in the time from 2002 to 2006.

The tool chain comprises a full shallow symbolic processing system from surface text to a syntactic dependency structure with lexical semantic annotation. Evaluations have shown that the modules perform quite well (cf. 7), even though they fall a little short of the best dedicated systems in the different disciplines that have been reported in the literature. Using our own systems had the advantage, however, that we could easily adapt and change them. Besides our own work, the parser is currently being used for annotating corpora of legal documents to facilitate searching for legal definitions (Walter and Pinkal, 2006).

Most of the software has been implemented in the programming language Perl. We started the implementation in Perl as it provides a very powerful and nicely integrated handling for regular expressions which was to form the core of our named entity recogniser.

We used Perl version 5.8, together with a number of modules from the Perl CPAN archive (`http://www.cpan.org/`), especially for XML handling, for handling extended regular expressions, for parsing (context free parser), and a database interface (cf. 6.3).

Data between the modules is passed using XML structures. The modules in the processing chain consecutively extend the XML structures with additional information. The different linguistic levels are related to each other in a stand-off annotation (Thompson and McKelvie, 1997), where the different linguistic levels are kept separate; we use text words as the basic level of reference. In contrast with other approaches, we keep annotations within one multi-level XML file for ease of data exchange.

6.2.3 Morphology

Input to the parser is first split into sentences and then tokenised. Both sentence splitting and tokenisation use a set of heuristic rules with a comprehensive list of German abbreviations.

The tokens of the input are then analysed morphologically using a German morphology. As mentioned above, a morphologic analysis is essential for processing the German language, as it has a rich inflection[5] and uses much compounding; as compounds are generally written in one word, a morphology component is essential for recognising and analysing them. We employ the

[5]Distinguishing, among others, four cases, two numbers, three genders, six tenses, two moods, two *genera verbi* and three declension types for adjectives and certain pronouns.

Gertwol two level morphology that is commercially available from Lingsoft Oy, Helsinki (Haapalainen and Majorin, 1994, 1995).

Gertwol is built upon the two level morphology technique (Koskenniemi, 1983). It provides a full morphology of the German language with inflection, derivation and compounding. Its lexicon comprises some 350 000 stems. This covers all closed class words (apart from very few dated forms) and provides a very good coverage of open class words, including quite a large number of proper names and place names. All analyses returned by Gertwol are retained; disambiguation is done by the following modules.

6.2.3.1 Reformed German Orthography

In August 2000, a number of spelling rules for German were changed. After an unexpected wave of protests, a compromise was introduced in August 2006. As a result, the spelling for a number of words has changed. While the number of *tokens* in a given text that actually change depends on the text sort, in general less than 5 % of the text words actually are different. About 60 % of the changes are occasioned by abolishment of an old exception rule that demanded that an underlying *'ss'* be changed to *'ß'* (a distinctive letter of the German alphabet) in syllable codæ; hence the old spelling, for example, of *'Fluss'* (river) was *'Fluß'*.

By using two different Gertwol lexicons, our system can analyse texts both in the old and in the reformed spelling. Words in old spelling are mapped to the reformed variant, so that the parser output consistently uses the reformed spelling. By this normalising step, it is ensured that, no matter whether old or reformed spelling are used, text representations of questions and answers will always match.[6]

In addition, texts in Swiss German, which does not use the letter *ß* at all, but rather *ss*, is also recognised and normalised accordingly.

6.2.4 Topological Parser

The topological parser is used to identify the sentence structure of the input. It relies on the fact that, while word order is relatively free in German, the topological sentence structure is comparatively rigid.

Topological theory (*Satzfeldtheorie*, Eisenberg, 1999; Höhle, 1983; Engel, 1970) assumes that German sentences consist of a number of different topological fields (*Vorfeld*, left sentence bracket, *Mittelfeld*, right sentence bracket, *Nachfeld*) to explain regularities in German word order and sentence structure.

[6]In fact, not all cases of compound verbs that are to be written as separate words according to the reformed spelling are normalised. We have not yet found that to be a practical problem, as these cases are rather rare in texts.

Using a number of rules, it can be described, for different sentence types, which fields must and must not be filled and where finite, infinite and auxiliary verbs must be positioned. For example, in yes/no-interrogatives, the *Vorfeld* must be empty, the finite verb resides in the left sentence bracket, while infinitive parts of complex verbs reside in the right sentence bracket (cf. 3.2.1.4).

Our topological parser builds upon the results from Braun (1999), but it has been completely re-implemented, refined and extended. It uses a set of some 350 manually written context-free rules describing all common sentence types in German. In case of ambiguities, an additional set of preference rules is used to find optimal solutions. In an evaluation with about 350 sentences from a newspaper and 100 sentences from German Civil Code (Bürgerliches Gesetzbuch, BGB), we found the parser to average around 87 %, both in recall and precision (F-score 87.02 %), with slightly higher values (F-Score 88.36 %) for the newspaper texts (Braun, 2003).

Figures 6.3 and 6.4 show a number of the most important structures recognised by the topological parser together with examples. Note that the analysis of the topological parser not only represents the sentence structure of the input with main clauses, subordinate clauses and other clausal constructions, such as infinitive clauses, but also the structure of complex verbal constructions, including split verbs, complex tense forms and modal constructions.

The topological parser also identifies ellipses in a number of cases of coordination. In these cases, the elided material (for example, the elided subject of coordinated non-finite clauses, cf. fig. 6.4) is copied into the elliptical structures.

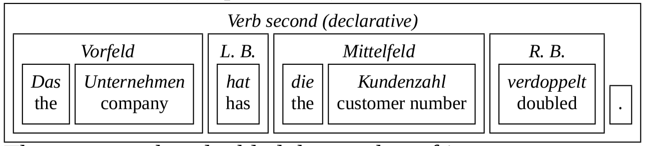
Another possible approach to topological parsing in German is introduced in Becker and Frank (2002). The authors describe a stochastic topological parser, which is based on a probabilistic context-free grammar (PCFG). To train the grammar, topological structures were automatically generated for the syntactically annotated Negra treebank (Brants et al., 1999).

6.2.5 Named Entity Recognition

The Named Entity Recogniser uses a combination of regular expression grammars describing different named entities, information from the morphology and gazetteers (special lexicons for NEs, such as a list of first names). Grammars and gazetteers are based upon a corpus study and annotation (Bering et al., 2003; Callmeier et al., 2003). However, some types of NEs were not used here, while a number of others were added.

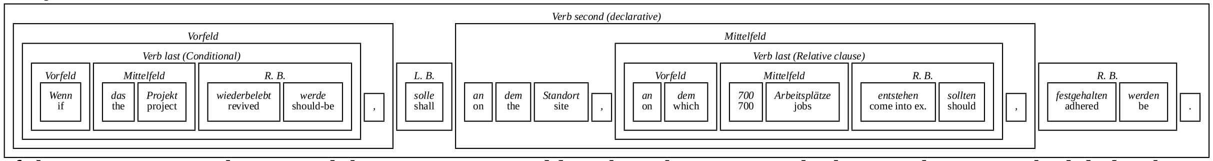
Table 6.1 shows the different types of named entities that are covered by our named entity recogniser (NER) together with examples.

- Main Sentence (complex tense)



The company has doubled the number of its customers.

- Subjunctive clause, relative clause, auxiliaries

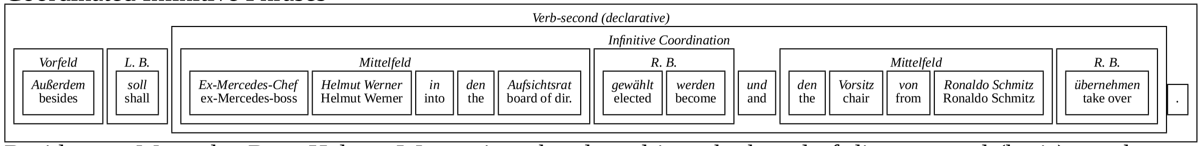


If the project was to be revived, however, one would stick to the site, on which 700 jobs were scheduled to become available. (*Süddeutsche Zeitung*, 2 Jan 1998)

L. B.: Left (verbal) bracket; R. B.: Right (verbal) bracket

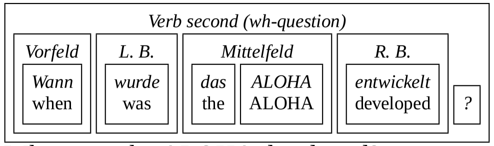
Figure 6.3: Examples for Structures Recognised by the Topological Parser

- Coordinated Infinitive Phrases



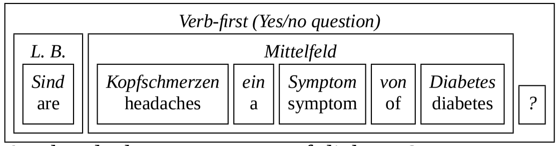
Besides, ex-Mercedes-Boss Helmut Werner is to be elected into the board of directors and (he is) to take over its chair from Ronaldo Schmitz. (*Süddeutsche Zeitung*, 2 Jan 1998)

- *Wh*-Question



When was the ALOHA developed?

- Yes/No-Question



Are headaches a symptom of diabetes?

L. B.: Left (verbal) bracket; R. B.: Right (verbal) bracket

Figure 6.4: Examples for Structures Recognised by the Topological Parser (continued)

Description	Example
Proper Name	Jürgen E. Schrempp, Theodor W. Adorno, H. P. Grice, Deng Xiaoping, Elizabeth II.
Organisation Name	CLT Sprachtechnologie GmbH
Location Name	New York, St. Peter-Ording, Bad Boll
Money Expression	261 Mio. DM
Date Expression	1. Januar 2001, 20.4.1999
Percent Expression	22,7 %
Measure Expression	42,195 km
Legal Citations (German Law)	§ 3 Abs. 1 GG
Generic Named Entity	Abbottabad, Abd ar-Rahman Munif, Zinnchlorür

Table 6.1: Types of Named Entities Recognised by the Named Entity Recogniser

6.2.5.1 Grammar

The NER grammar uses several hundred, often rather specialised rules encoded as regular expressions. They generally rely on matching some ‘safe’ anchor (for example, a first or last name or a company form part such as *‘GmbH’*, *Ltd*, in a company name) and then search for several possible parts of NEs ‘surrounding’ it, accepting less certain matches. For example, when matching the proper name *‘Theodor Wiesengrund Adorno’*, *‘Theodor’* is recognised as a first name, *‘Adorno’* as a last name and *‘Wiesengrund’* is conjectured to be a middle name – correctly, even though *‘Wiesengrund’* (meadow ground) is an extremely uncommon name.

Besides the manually built gazetteers, the NER exploits information provided by the morphology: The several thousand first names and family names recognised by the morphology are used as anchors. In addition, unknown words (especially sequences of unknown words) are tagged as ‘generic NEs’. This simple heuristic has proven to be quite efficient, as in many cases, unknown words are indeed NEs of some description (see the examples in tab. 6.1). In combination with the chunk parser, it also allows building NPs with unknown nouns as heads. This has proven very useful for parsing sentences containing, e. g., technical terms that are not covered by the morphology (cf. also 6.2.6).

6.2.5.2 Learning

Our NER can 'learn' from recognised entities: When some person or organisation is introduced using a 'long form' of their name, the NER module learns a number of possible 'short forms' of the name.

We have already mentioned the fact that for several types of NEs, short forms may be used (cf. 5.2.2.4). In most texts, the first reference to persons or entities will be by their 'full names'. Later references to the same person or entity will then generally use short forms. A text may, for example, first mention *'Indian prime minister Manmohad Singh'* and later refer to him as *'Dr. Singh'*.

Thus, whenever the NER identifies a proper name or an organisation name, it temporarily adds the parts of the name to the gazetteers for the respective type (e. g., a previously unknown last name is added to the last name gazetteer). This allows recognising short forms of that name later in the same text. At the end of each text, the temporarily added names are deleted again from the gazetteers to prevent loss of precision: The NER would, for example, when former German chancellor *'Helmut Kohl'* is mentioned, learn to recognise the last name *'Kohl'* within this text. In other texts, however, that talk about *'Kohl'* (cabbage), this last name reading should not be used. This 'learning' approach is similar to that described in Volk and Clematide (2001).

6.2.5.3 Other Approaches.

As described above, we have chosen a symbolic, rule-based approach to NER, even though most current named entity recognition systems are based on machine learning techniques.

Machine learning approaches to NER can be grouped into two main classes: supervised and unsupervised/semi-supervised approaches (cf. also 2.1.3).

For supervised approaches, a corpus is used that must be annotated for the desired named entities. The system uses the corpus to train some statistical model (generally, Hidden Markov Models, HMM) for recognising named entities based upon the likelihood of a certain part of the input being tagged as a certain NE. A well-known example is the IdentiFinder system (Miller et al., 1998). We decided against this approach, as comparatively large annotated corpora are needed: Appelt and Israel (1999) report that an annotated corpus of about 100 000 words is required as a starting point, and that for training corpora of up to about one million words, there will generally be a marked improvement in system quality: The system IdentiFinder was trained on a corpus of about 800 000 words (Miller et al., 1998) and reached F-Scores of 90.4 % in MUC-7.

For semi-supervised approaches (also called boot-strapping approaches), a small number of seed patterns and seed 'fillers' (e. g., a list of a dozen or so titles, first names and last names) for each type of NE is fed into the system. Starting from this seed, the system uses a corpus to find additional fillers and then additional rules for the different types. Recent examples include the systems described in Quasthoff et al. (2002); Rössler (2002); Stevenson and Gaizauskas (2000); Declerck and Neumann (2000); Borthwick (1999); Collins and Singer (1999).

6.2.6 Noun Phrase Chunking

Noun phrases and prepositional phrases are recognised and structurally analysed by a NP/PP-chunker based on our earlier work on German grammar checking (Fliedner, 2002, 2001).

The chunker module uses the input from the Gertwol morphology and a NP/PP grammar based on an extended finite state automaton with about 350 states and 8 000 edges. As described in more detail in Fliedner (2002, 2001), the chunker can build embedding structures that allow to handle complex German NPs and PPs: In German, pre-nominal adjectives can take arguments to their immediate left, leading to (in principle) arbitrarily deeply embedded NPs/PPs.

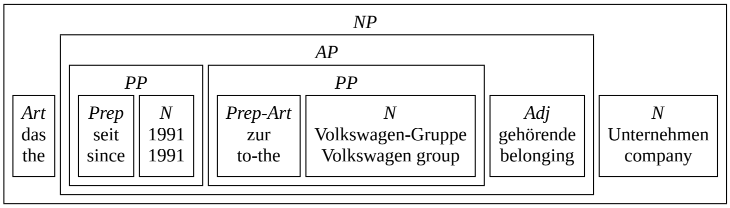
The chunker does not handle post-nominal attachment of PPs or genitive modifiers, this is left to later processing stages. Ambiguities are resolved by an approach inspired by optimality theory (OT, Kager, 1999), so that the chunker module always returns a single solution that is considered optimal. This is described in more detail in Fliedner (2001).

One important feature of our tool chain is that NER and NP/PP-chunker work in a closely integrated fashion: Named entities that are identified by the NER are passed on to the chunker, which handles them like complex lexical nouns. Thus, NEs can form parts of more complex NPs, for example with pre-nominal articles and/or adjectives or in conjunctions. Figure 6.5 shows some examples for more complex NPs/PPs, as recognised by the chunker.

An obvious alternative would have been to use a statistical chunker. One system for German is described in Skut and Brants (1998a,b). It is based on cascaded Hidden Markov Models (HMM) and was trained on the NeGra corpus of German newspaper texts. The authors report F-scores for labelled bracketing of 84 % (precision and recall being virtually equal).

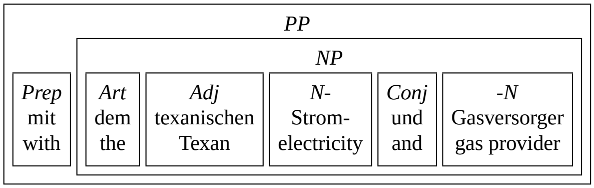
We found that this system does not output agreement information for the detected chunks. Since this information is required for constructing the PReDS, however, we could not use it.

- Embedded adjective phrase with arguments



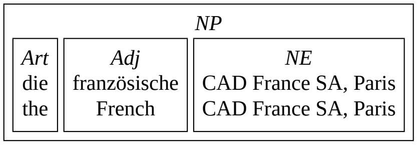
the company, which has been part of the Volkswagen group since 1991

- Split compound



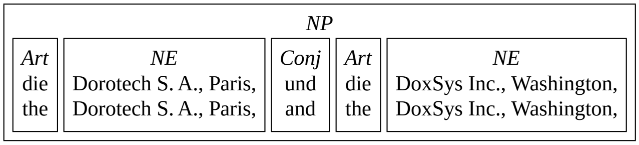
with the Texan provider of electricity and gas

- Company Name (NER) as head of NP



the French CAD France SA, Paris

- Company Names, Coordination


NP
Art die the
NE Dorotech S. A., Paris, Dorotech S. A., Paris,
Conj und and
Art die the
NE DoxSys Inc., Washington, DoxSys Inc., Washington,

the Dorotech S. A., Paris, and the DoxSys Inc., Washington,

Figure 6.5: Examples for Complex NPs/PPs

6.2.7 Constructing Dependency Structures

The results from the different modules are integrated into a dependency structure. This is done by applying a number of rules that attach the structures to each other.

Dependency structures in general and PREDS in particular were introduced in 4.1.1 above. An example for a PREDS structure for the sentence *'Das Unternehmen hat 1997 die Zahl seiner Kunden auf über eine Million verdoppelt.'* (*The company has doubled the number of its customers to more than one million in 1997.*) is shown in fig. 6.6. A number of details regarding the example will be described in this section.

We will explain how attachment ambiguities in parsing are represented as (partially) underspecified dependency structures by our parser, before describing the construction process in more detail.

6.2.7.1 Representing Underspecification

Researchers who have designed parsers have often taken one of the following two possible approaches when dealing with ambiguity: One option is for the parser to only return one single structured representation of the input that is considered optimal based on some local measure of optimality. This approach is often employed by statistical parsers. In this case, important information may be lost by discarding a structure that the parser regarded as less likely than the one it returned. The second option is to enumerate all possible representations for the input. This is again problematic, as the number of fully specified structures tends to explode: In the worst case, all possible combinations of choices for all ambiguities in the sentence must be computed and returned, potentially leading to a number of representations that grows exponentially with the number of local ambiguities.[7]

An intermediate solution is to use underspecified representations: Partial structures are derived together with a set of constraints that specify how the partial representations can be connected. In the case of an input for which ambiguous PP attachment is computed, for example, such an underspecified representation could contain a representation of the PP together with information about possible 'landing sites' in the representation of the rest of the input. This sort of representation has the advantage of being relatively compact and keeping local ambiguities contained. In a cascaded approach like ours, following processing steps may resolve such local ambiguities and 'fasten' the attach-

[7]This is the case when the local ambiguities are fully independent of each other. Often, for example when the attachment of two consecutive PPs is concerned, not all theoretically possible attachments are grammatical.

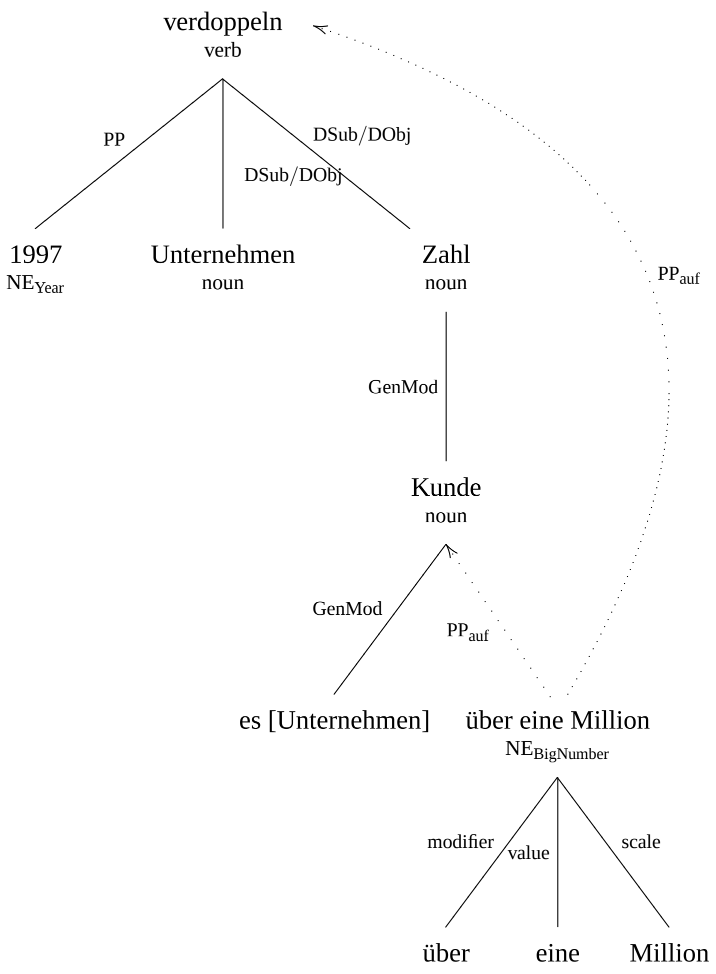


Figure 6.6: Underspecified PReDS Representation for ‘*Das Unternehmen hat 1997 die Zahl seiner Kunden auf über eine Million verdoppelt.*’

ment without influencing other underspecified partial structures. If necessary, it is also possible to enumerate all possible 'global' representations.

Underspecification has been quite extensively used for semantic representations in the last years, for example in the form of Quasi Logical Forms (QLF, Alshawi and Crouch, 1992), Underspecified Discourse Representation Structures (UDRS, Reyle, 1993), Underspecified Semantic Description Language (USDL, Pinkal, 1995), Minimal Recursion Semantics (MRS, Copestake et al., 2005) and Constraint Language for Lambda Structures (CLLS, Egg et al., 2001, 1998). However, these approaches focus on semantic structures, so that we could not use them to represent syntactic underspecifications.

The approach described in Federici et al. (1996) is similar to ours in a number of respects, however, we consider their notation for underspecification too implicit: In their approach, different possibilities for attachment are not directly specified in the structure. Additional, language specific knowledge is required to identify possible attachment sites.

There is obviously some overlap between the idea of underspecified structures and the use of 'packing' ambiguous structures during parsing (Oepen and Carroll, 2000; Billott and Lang, 1989; Tomita, 1987): In many parse algorithms, starting with Earley's algorithm (Earley, 1970), ambiguous intermediate structures are not directly integrated into larger structures, as for structures containing two or more ambiguous structures, all possibilities of combination would have to be spelled out. Rather, lists of pointers to different possible 'fillers' are stored to keep processing overhead small. From these packed representations, fully disambiguated structures can be recovered where necessary. However, this technique is generally used during parsing only; it is (somewhat implicitly) assumed that the structure is later 'exploded'. In our approach, the *resulting* structure may be left underspecified.

We have integrated a relatively simple means of representing syntactic underspecification into the PREDS (hence *Partially Resolved* Dependency Structures). We will now describe these structures in some more detail.

Our shallow parsing approach only has access to a limited amount of linguistic information. As described above, the different modules (namely the topological parser and chunker, cf. 6.2.4 and 6.2.6) generally derive a single, locally optimal solution. However, we are lacking reliable valency information for German (see below, 6.2.7.2). Therefore, a number of decisions concerning attachment cannot reliably be taken, leading to underspecified outputs.

There are four main phenomena for which PREDS uses underspecification. For attachment ambiguities, a default attachment is used, but the resulting link in the dependency structure is marked as a default attachment, so that other possible attachment sites can be identified. For example, PPs are always attached 'low', i. e., to a directly preceding NP/PP and marked accordingly. Such a default-attached PP can afterwards be re-attached to a higher position.

Attachment of PPs. Prepositional phrases always receive low attachment: If an NP or PP directly precedes a PP in a sentence, the PP is attached directly to it. This attachment, however, is marked as a default attachment. Modules later in the processing chain may re-attach such a PP to any ancestor up to and including the lowest verb node. Figure 6.6 shows an example of underspecified PP attachment: Two possible landing sites are identified for the PP *'auf über eine Millionen'*, indicated by dotted arrows.

Clausal Arguments. There are a number of clause types in German whose instances can form clausal arguments of either nouns or verbs, namely subjunctive clauses headed by *'dass'* (*that*), indirect interrogatives and infinitive clauses extended with *'zu'*. For example, an noun like *'Aussage'* (*statement*) can take a subjunctive clause as its argument (headed by *'dass'*, *that*). Like PPs, these are attached low by default (generally, to an immediately preceding NP/PP), and marked accordingly.

Internal Structure of Compounds. As mentioned above, German compound words are generally written in one word (or sometimes with a hyphen). Compounds with more than two parts are, in principle, ambiguous with regard to internal structure. As the PREDS decomposes compounds and assigns a dependency structure to them, the attachment of the compound parts is treated similar to that of PPs: By default, attachment is low, but later processing steps can re-attach default-attached parts to higher components (up to and including the head part of the compound).

Subjects and Objects. In the absence of valency information, NPs are assumed to be arguments of the sentence predicate[8]. Their grammatical function (subject, direct object, dative object, genitive object) is assigned on the basis of case information. However, there are cases when this assignment is ambiguous. In such cases, an underspecified grammatical function label is used and the possible grammatical functions (according to possible cases) are recorded. In fig. 6.6, both *'das Unternehmen'* and *'die Zahl'* are ambiguous between subject and object, if only morpho-syntactic criteria are applied; this is indicated by the DSubj/DObj labels.

[8]Excluding appositions and genitive modifiers, these are attached by separate rules, cf. 6.2.7.2 below.

Using these limited and relatively simple means, we can express all important cases of underspecification that cannot be resolved by our parser. We will now describe the construction process in more detail.

6.2.7.2 Construction Process

The module that constructs the dependency structures assembles all structures built by the previous steps into a single dependency structure, namely the PREDS, based upon a set of rules. In ambiguous cases, parts of the structure are left underspecified; they can be resolved by later processing steps in the chain (especially the frame annotation module, 6.2.10).

Note that our approach is generally similar to that described in Abney (1991): We also use a chunker first and then an 'attacher' as a second step. By using a topological parser before chunking, however, the input to the 'attacher' is generally more reliable: The topological parser recognises sentence structures and verbal brackets with high accuracy.

As we did not have access to a comprehensive electronic valency lexicon of German when we started our work[9], PREDS are built without making use of any valency information. This, of course, leads to a strong increase in numbers of sentences that cannot fully be disambiguated. There are several points during the construction process where this plays a rôle.

The PREDS is built by recursively traversing the topological structure output by the topological parser (6.2.4), i. e., visiting each clausal structure that was identified in turn. First, the main verb lemma is extracted from the verbal brackets of the topological structure, then, the material from the other topological fields is integrated into the emerging structure.

The main verbal node of each clause can be assembled from the information in its verbal brackets. Recall that all parts of the verb are assigned either to the right or to the left verbal bracket (6.2.4), including split verbal prefixes, modal verbs and auxiliary verbs. The main predicate becomes the root node of the PREDS of the clause. For text input without a main verb that does not constitute a sentence, an empty node is used as root. At this point, split verbs are joined into a single structure. Modal and auxiliary verbs do not receive separate nodes in the PREDS: Using a set of rules, their morpho-syntactic function in

[9] Recently, Sabine Schulte im Walde has kindly made the results of her Ph. D. thesis (Schulte im Walde, 2003) available to us. Using a broad coverage context free grammar of German and a large corpus, she has induced rule probabilities enhanced with lexical anchors. We extracted some information from the data that allows us to gauge the relative probability of attaching a PP to a preceding NP or the matrix verb based on the frequency of the corresponding attachment in the training corpus for the respective lemmata. However, these informations are currently only used in very reliable cases.

the clause is analysed and added as a feature of the main verb. For example, the combination of a finite form of *'haben'* (*'have'*) with a past participle is identified as a perfect tense form of the verb in the participle form: That verb becomes the main verb, the PReDS-node is labelled with its lemma and additionally receives a tag for perfect tense.

After building the root node, the remaining topological fields of that clause, viz. *Vorfeld*, *Mittelfeld* and *Nachfeld*, are traversed and their different components attached successively. First, from the NPs and PPs recognised by the chunker modules (6.2.6), dependency structures are recursively derived.

Compound words are assigned an 'inner' dependency structure: Instead of a word node, the head part of the compound is used, the other compound parts are attached to it using a special compound relation. Note that for compounds with more than two parts, the attachment is ambiguous. We therefore use default low attachment as for PP attachment (as discussed above).

Using a set of attachment rules, the construction module then attaches adjacent structures to each other. Here again, the core idea of identifying structures bottom-up one after another is applied, starting with those that are easier to recognise (cf. 6.2.1): First, appositions are attached, then, post-nominal genitive modifiers and PPs.

As described above, the attachment of post-nominal modifiers[10] is syntactically ambiguous: For example, a PP can, in principle, form a post-nominal modifier of all NPs and PPs directly preceding it in a sentence, or, alternatively be a direct dependent of the main verb[11]. Thus, attachment is left underspecified in the PReDS in the following way: Such modifiers are always attached 'low' if possible, i. e., to their immediate left neighbour in the sentence. The attachment is marked as a default attachment. From this underspecified representation, all possible structures can be derived, by 'cutting' the attachment and re-attaching the modifier to any node directly or indirectly dominating it, as long as domination is through post-nominal modification relations, only, or to the dominating verb node. This may, in fact, done when frame annotation is added that resolves ambiguities in the dependency structure (6.2.10).

In a next step, subordinate clauses, namely indirect interrogatives, extended infinitive clauses and certain subjunctive clauses[12], are attached using a similar technique. All of these can, as post-nominal modifiers, be dependents of a pre-

[10]We will use the term modifier here to generally describe a dependency relation without discriminating between arguments and adjuncts: A post-nominal PP may be used to express either. The distinction may not even be clear in all cases, depending on the valency of the head noun.

[11]Always provided, of course that there would be no 'crossing edges' in a corresponding phrase structure description.

[12]Namely potential clausal arguments (mostly headed by *'dass'*, *that*)

ceding NP/PP. As post-nominal modifiers, they are, if possible, attached low to their left neighbour, otherwise to the dominating verb.

Relative clauses are attached next, using the agreement features of the relative pronoun as the guiding factor: A relative clause can only be attached to a preceding NP or PP if the relative pronoun agrees in gender and number with that antecedent.

Finally, NPs and PPs that have not yet been integrated into the structure are attached to the main predicate as dependents: As no valency information is available, this is the only available option, as the required arguments cannot be identified. From the case features of the NPs, the possible grammatical functions are inferred: Nominative case corresponds to subject, accusative case to direct objects, dative case to indirect objects and genitive case to genitive objects. Double accusative objects and nominative case for predicative nouns are also handled, based on a small verb valency lexicon. In cases where an unambiguous assignment of grammatical functions can be found, the dependency relations are labelled accordingly. In all other cases, the labels are left underspecified and tags indicating its possible grammatical roles are assigned to each NP.

6.2.8 Adding GermaNet Information

The syntactic dependency structure is extended with GermaNet information (cf. 4.2, Kunze and Lemnitzer, 2002b). For every node in the structure, the lemma, combined with the part of speech, is looked up in GermaNet and, if it is found, one or more GermaNet indices are added to the node. We use the May 2003 version of GermaNet. All data was extracted from the GermaNet XML (Kunze and Lemnitzer, 2002b) and is stored in a database for efficient access (cf. 6.3).

As mentioned above, no disambiguation is performed in this step. Thus, each node in the dependency structure is annotated with all possible readings that GermaNet provides for the respective lemma-part of speech combination. We have chosen this strategy because preliminary experiments with unsupervised sense disambiguation that we conducted lead to unsatisfactory results. This ties in with similar experiments reported in De Boni (2004, 134). Thus, we decided to conduct no disambiguation at all in order to improve recall. Our assumption was that precision would not be hurt too seriously in QA, as our matching algorithm for questions and answers is based on matching structured representations. We considered it comparatively unlikely that one word in a question structure would be matched onto another word in a assumed answer where no match should be possible, because we would, in that case, also expect the embedding structures to differ. The evaluation shows that precision errors caused by the lack of GermaNet disambiguation are indeed rare (we observed one single error in our evaluation, which is subsumed under ‘other error types’, cf. 7.2.2.3).

All recognised NEs are assigned generic GermaNet identifiers: Person NEs, e. g., are assigned to the GermaNet synset *‘Person’* (*‘person’*). Thus, NEs are ‘anchored’ in the semantic hierarchy. This allows us to handle them correctly in anaphora resolution (6.2.9) and when matching *‘which’*-NPs in questions (5.1.4).

The GermaNet annotation is also used to assemble morpho-syntactically complex expressions into lexico-semantic simplicia. This happens for three sorts of constructions: compound words (e. g., *‘Krankenpflege’*, *‘nursing’*, from *‘Kranke(r)’*+*‘Pflege’*, literally *sick-care*), adjective-noun compounds (e. g., *‘Olympische Spiele’*, *‘Olympic games’*) and certain phrasal verbs (e. g., *‘Ski laufen’*, *‘(to) ski’*, literally *‘run ski’*).

In the PReDS, these constructions receive a structured representation. For example, in *‘Olympische Spiele’*, *‘olympisch’* would be analysed as an adjective modifier of *‘Spiele’*. As the compound expression can be found in GermaNet, the original structure is collapsed into a single, complex noun node labelled *‘Olympische Spiele’*. This allows us to correctly associate *‘Olympische Spiele’* with, e. g., its GermaNet synonym *‘Olympiade’* (*‘Olympics’*) or its hypernym *‘Meisterschaft’* (*‘championship’*).

6.2.9 Anaphora Resolution

We employ an adapted version of the algorithm by Shalom Lappin and Herbert Leass for anaphora resolution (Lappin and Leass, 1994). We have extended the original algorithm so that it also handles coreference for definite descriptions and named entities (Fliedner, 2006b). Before describing the details, we will give a short account of other work on anaphora resolution.

6.2.9.1 Other Approaches

Anaphora resolution has received a lot of attention in (computational) linguistics research. We base our description on Ruslan Mitkov’s overview (Mitkov, 2002) and generally use his terminology. We will not go into the rôle that anaphora play in linguistic theories (cf. Mitkov, 2002, esp. 53–67), but focus on describing practical approaches to anaphora resolution.

Anaphors refer back[13] to an entity or an event mentioned in the text, which is called the antecedent. The process of identifying the correct antecedent of an anaphor is called anaphora resolution. The following approaches to anaphora

[13]Or forward, in the case of cataphora.

resolution can be distinguished (for additional approaches, especially based on Centering theory and probabilistic approaches, see Mitkov, 2002).

Syntax-based Approaches. A number of approaches to anaphora resolution on syntax have been proposed. These approaches use either full or partial syntactic structures derived from input text as basis. Identifying anaphoric expressions and resolving them to their respective antecedents is typically done by sets of syntactic constraints (such as *c*-command, see below) and preference rules (preferring, e. g., coreference between syntactically parallel structures). Examples include Hobbs (1978)[14]; Mitkov (2002); Stuckardt (2001); Kameyama (1997); Kennedy and Boguraev (1996); Lappin and Leass (1994).

Semantics-based Approaches. Anaphora resolution has also been approached as a sub-part of a discourse semantics construction (cf., e. g., Fischer et al., 1996; Quantz, 1992; Pinkal, 1986). If a full semantic representation of a given discourse is constructed, semantic constraints and preferences can be utilised for anaphora resolution. However, these approaches have only been used in toy implementations.

Machine-Learning Approaches. Recently, a number of machine-learning approaches to anaphora resolution have been proposed. Interest in these has grown especially since the integration of coreference resolution into several shared-task competitions namely MUC-6 and MUC-7 (Message Understanding Conferences, cf. 2.1.3, MUC-6: `http://cs.nyu.edu/cs/faculty/grishman/muc6.html`; MUC-7: `http://www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_toc.html`) and ACE (Automatic Content Extraction, `http://www.itl.nist.gov/iaui/894.01/tests/ace/index.htm`) and the ensuing availability of corpora annotated for coreference. Most current systems use a number of different information sources (especially shallow parsing, named entity recognition and often WordNet), which are employed to define a number of features for statistic modelling. Given these features, the approaches use machine learning on annotated corpora to derive a decision tree or similar model to combine the features optimally. A recent system that reaches state of the art F-scores of slightly over 65 % on the MUC-7 data is described in Uryupina (2006). Largely similar approaches for German are described in Strube et al. (2002); Hartrumpf (2001) and in Postolache and Forăscu (2004); Ng and Cardie (2002); Soon et al. (2001) for English.

[14] Actually, Hobbs (1978) describes two algorithms, only the 'naïve approach' is syntax-based.

6.2.9.2 Algorithm

The algorithm we have implemented for anaphora resolution handles three different types of anaphoric expressions, viz. pronominal anaphora, definite descriptions and co-referring named entities.

We use an extended version of the algorithm described in Lappin and Leass (1994): While the original algorithm by Lappin and Leass only handles pronominal anaphora, we have extended it to also handle definite descriptions and named entity coreference. Since we have dependency structures and lexical semantic information at our disposal, we can utilise this rule-based approach and do not need to revert to knowledge-lean approaches: The main argument for the development of the so-called knowledge-lean approaches (such as Mitkov, 2002; Stuckardt, 2001; Kennedy and Boguraev, 1996) was that using full parsing as basis for anaphora resolution was considered undesirable, as a) no full parsing might be available and, if so, was considered b) too slow and c) too fragile.

The algorithm works as follows (for additional details, cf. Lappin and Leass, 1994): First, a list of possible antecedents (basically, all NPs and PPs) in the input text is put together. Whenever a (possibly) anaphoric expression is found, it is matched against each candidate on the list. Candidates that are incompatible with the anaphoric expression under consideration are filtered from the candidate list. The remaining candidates are ranked by a scoring function that is based mostly on the grammatical function of the candidate and its recency in addition to syntactic parallelism with the anaphoric expression. The candidate ranked highest is considered to be the antecedent of the anaphoric expression. In case of ties, the most recent candidate is chosen.

While all NPs including pronouns on the one hand and person and company NEs on the other hand are considered as possible *antecedents*, only the following types of NPs are considered as possible *anaphors*. Different antecedent filters are used for the different types of possible anaphors.

Pronouns. All referring pronouns (i. e., third person personal pronouns, demonstrative pronouns, relative pronouns and possessive pronouns) are considered as possible anaphors. Both expletive *‘es’* (*‘it’*) and reflexive pronouns (forms of *‘sich’*, *‘oneself’*) are generally not expressed as dependents of the verbs in the PREDS, but only as features of the verb and are thus not considered as possible anaphors.

Possible antecedents for pronoun anaphors are filtered on the basis of agreement criteria: Only antecedents that agree in gender and number with the pronoun are retained. As a special case, we also allow the demonstrative pronouns *‘das’* (*‘that’*) and *‘dies’* (*‘this’*) to be considered as anaphorically referring to whole clauses.

In addition, antecedents that *c*-command the pronoun are filtered, as they cannot refer to the same entity, as in **‘The boss$_i$ fired him$_i$.’*, cf. Stuckardt (2001, 482–3).

Definite NPs. Nominal phrases with a definite article or a demonstrative determiner are considered candidates for anaphoric expressions. From these, we filter definites that are likely to refer to discourse new entities, i. e., definites that do not refer back to a discourse referent but rather introduce new discourse referents (cf. Poesio et al., 2004; Uryupina, 2003). As heuristics for recognising discourse new definites, we consider a modification of the NP by pronouns, an apposition, an ordinal number or an attributive adjective of superlative degree.

The remaining definite noun phrases are filtered using a semantic compatibility check: Using the GermaNet annotation (6.2.8), we check whether the head of the definite NP and that of the antecedent stand in synonymy or close hyponymy relation. Our current heuristics is to test whether there is a direct hyponymy path between any GermaNet synset of the two and whether the length of this path is at most half of the length of the path of the concept ‘lower’ in the hierarchy to the root of the hierarchy.

Other heuristics could be used in principle. In Harabagiu et al. (2001a), for example, a method is introduced to generate seed patterns for bootstrapping similarity relations between definite NP and antecedent heads. It uses path distances in WordNet, based on hyponymy, meronymy and used-in-gloss relations (cf. 4.2.1.2), as a measure of semantic similarity between definite NP and possible antecedent. From this seed, new similarities are learned on corpora.

Person, Organisation and Generic NEs. These three types of NEs are considered as possible anaphora in the sense that they may refer to a person, organisation or entity that has already been mentioned. For the other types of NEs that our system recognises (cf. 6.1), we did not observe any cases of coreference. As ‘short forms’ (cf. 6.2.5) are likely to be used, the surface forms of the antecedent NE and the anaphoric NE may differ for person and organisation names. Only other NEs of the same type are considered as possible antecedents for NEs. The candidates are filtered based on the same technique described above for *matching* NEs (cf. 5.2.2.4): Only if a (partial) match of the right ‘component’ is found (say, the last name part of two person names is the same) and no conflicts

Salience factor	Initial weight
Same sentence	100
Subject	80
Nominal predicate (‘*be X*’)	70
Direct object	50
Indirect object	40
Head noun emphasis	80
Non-adverbial emphasis	50
PP	30

Table 6.2: Initial Weights for the Salience Factors for Anaphora Resolution, adapted from Lappin and Leass (1994, 541)

arise (say, the last name matches, but the first name is different) are they considered a potential match.

Ranking the remaining candidates (i. e., those antecedent candidates of a particular anaphor that are not filtered by the specific filtering rules described above) is done in a uniform way; the technique we use is close to that described in Lappin and Leass (1994). We proceed as follows: Each NP that is put on the possible antecedent list is assigned an initial weight. The initial weight is computed by checking a number of syntactic features (called salience factors) and adding constant weights for each salience factor present. If the NP is the subject of a sentence, for example, the associated weight of 80 is added to the respective initial weight. We use the salience weights reported in Lappin and Leass (1994), repeated in in tab. 6.2. These salience factors mirror the observation that the grammatical function which is used in mentioning a discourse entity influences the perceived salience and thus the likelihood of it forming the antecedent of an anaphoric expression.

The initial salience weights assigned to the candidate NPs change dynamically:

Recency. When a new sentence is processed, the current weights of all antecedent candidates are decreased by 50 %. Thus, more recent candidates are more likely to be selected.

Anaphoric Chains. All anaphoric expressions and antecedents referring to the same entity form an anaphoric chain. This group can be seen as constituting an equivalence class: All expressions that refer to the same entity belong to one equivalence class. As the salience of a discourse entity is assumed to increase with each reference to it, the current salience weights for *all* referring expressions in this class are added up to form a class weight instead of an individual weight for a candidate within such a class.

Syntactic Parallelism. A constant factor of 35 is added to the weight of an antecedent candidate when the anaphoric expression and that antecedent candidate have the same grammatical function, for example, when they are both used in subject position.

As described above, the candidate from the filtered candidate list that has the highest final salience weight is returned by the algorithm as the most likely antecedent candidate for the anaphoric expression under consideration.

Note that all salience weights were determined by Lappin and Leass based on a number of experiments. We found the weights that they suggest to work well and thus left them unchanged in our implementation.

6.2.9.3 Limitations

The algorithm as it stands covers most common types of nominal anaphora. For definite descriptions, a few types cannot yet be properly handled: First, our recognition of so-called discourse new descriptions (Poesio et al., 2004; Uryupina, 2003; Vieira and Poesio, 2000) does not work in all cases: We recognise cases where the definite description in question is modified by certain types of modifiers. In addition, cases where no antecedent candidate can be found are considered as discourse new as well. What we lack is systematic recognition of 'larger situation definites', such as *'the United States', 'the pope', 'during the summer'* (cf. Poesio et al., 2004). Handling such cases could probably be done using the method suggested in Uryupina (2003), based on comparing, for every head noun of a possible discourse new description, the ratio of definite and indefinite uses, using the Internet as a corpus. We did not use this approach, as parsing would then require an on-line Internet connection. We considered this too grave a constraint.

Second, we do not handle bridging anaphora at all (Gardent et al., 2003; Vieira and Poesio, 2000)[15]. The term 'bridging' is used for definite descriptions that are not directly coreferent with an antecedent, but whose referents are, nevertheless, related to a previously mentioned entity. For example, in

[15]Note that the definition of bridging anaphora in Vieira and Poesio (2000, esp. 558) is rather restrictive: They consider all definite descriptions whose head noun differs from that of the antecedent phrase as bridging anaphora. We will assume the definition in Gardent et al. (2003), where only non-coreferent definite descriptions are labelled as bridging anaphora.

‘Jones has moved to a large new flat. The living room alone is 50 square metres large.’, the *living room* is understood to be part of Jones’s new flat. Thus, a meronymy relation holds between the so-called anchor and the bridging description (cf. 3.5.2.4).

6.2.10 Constructing Frame Structures

As the last stage in our processing chain, frame structures are added to the text representations. We will give an overview of existing systems for automatically assigning frame structures to texts before describing our own approach in more detail.

6.2.10.1 Systems for Frame Assignment

There are currently several systems that derive frame structures to texts. Most of them use supervised machine learning approaches: the systems need to be trained on annotated corpora. In 2004, semantic role assignment based on FrameNet was used as a task for the Conference ‘Evaluation Exercises for Word Sense Disambiguation’ (SensEval-3, `http://www.clres.com/SensSemRoles.html`).

Note that most systems described here only perform the task of frame element identification and labelling and assume that, in the input, frame evoking elements are annotated and that the proper frame has been assigned to each frame evoking element. An additional word sense disambiguation step based on frame information would thus have to be performed *before* these systems could actually do the role assignment.

Most systems will split the task of frame element annotation further down into two sub-tasks, both viewed as classification problems: In a first step, frame elements are identified in the input (classify text segments into one of the following: frame element, target, none). In a second step, the frame elements are assigned. We will now give a short overview of some of the systems.

- The first system for automatic frame element assignment was presented in Gildea and Jurafsky (2002). The authors evaluated different combinations of features (especially grammatical functions based on the output from Michael Collins’s parser, Collins, 1997, and WordNet information) with learners. For the best combination of features, they achieved 65 % precision and 61 % recall. Frame assignment was not done by this system at all.

- Michael Fleischman, Ed Hovy and Namhee Kwon have presented a similar system, with which they took part in the SensEval-3 competition

(Kwon et al., 2004a,b; Fleischman et al., 2003b). Again, no frame assignment is performed by this system; it relies on a suitably annotated disambiguated input. The authors split the task into frame element identification and frame element assignment. Both subtasks are tackled with an approach based on Maximum Entropy modelling. As input, they use a number of features mostly derived from a syntactic parse tree for the input (obtained using Eugene Charniak's statistical parser, Charniak, 2000), mostly describing grammatical functions and lexical heads of phrases. In an evaluation, the system reaches around 71.1 % precision and 58.5 % recall for the overall task.

- In the context of the SALSA project (cf. 4.3.2), a similar system has been developed by Ulrike Padó (née Baldewein), Katrin Erk, Sebastian Padó and Detlef Prescher (Baldewein et al., 2004). It was also tested in the SensEval-3 competition where it reached 73.6 % precision and 59.4 % recall. Again, the task was split as before and Maximum Entropy modelling was used on input parsed by the LoPAR parser (Schmid, 2000).

- Another approach, also developed in the SALSA project (cf. 4.3.2), uses an integrated architecture that currently combines two modules, one for frame assignment (considered here as word sense disambiguation) and a second one for frame element recognition and labelling (Erk and Padó, 2006)[16]. For deriving syntactic structures as input for the machine learner, different statistical parsers can be selected. The authors report the following F-scores: For frame assignment in English text, the accuracy is 93.2 %, for German text 79.0 %. Combined frame element recognition and assignment (based upon perfect frame assignment) reaches 78.4 % (English) and 67.3 % (German).

- A different, mostly symbolic approach to frame assignment has been described in Frank et al. (2007). The authors use a set of manually written transfer rules to add frame structures to a comparatively rich text representations derived from texts using the modular Heart of Gold system (Callmeier et al., 2004, see also 6.2.1.4). As described above, Heart of Gold uses a number of collaborating shallow and deep parsers (including an HPSG parser and an extended NER). All results are conflated into semantic representations based on Robust Minimal Recursion Semantics (RMRS, Copestake, 2006). The transfer rules identify the frame evoking element, the evoked frame and the associated frame elements all in one

[16]In fact, the frame element recognition system, Rosy, can be said to be a descendant of the system described in Baldewein et al. (2004). Sebastian Padó, personal communication, October 2006

integrated structure. However, the FrameNet transfer module covers only a limited domain (Nobel prizes) and is therefore not directly usable for general purposes.

- Yet another, related symbolic approach, again developed in the Salsa project (cf. 4.3.2), has been suggested in Frank and Erk (2004). The authors describe the construction of a mapping from LFG structures automatically constructed by a large-scale, broad-coverage LFG parser for German (Zinsmeister et al., 2002) and suggest how such a mapping may be learned automatically.

Most of the systems for frame element assignment described here are based on machine learning techniques (such as Maximum Entropy Modelling). The learners mostly rely on syntactic features, such as phrase heads and (approximated) grammatical functions, thus input needs first to be parsed syntactically.

For training, these approaches need a corpus that has been annotated with frame information as input. When we started our work, no such corpus for German was available. We therefore decided to build a simple and easily extensible system based on manually written transfer rules. When adjudicated data from the Salsa corpus (cf. 4.3.2) became available for German, we used information from the corpus to semi-automatically extend the coverage of our grammar. This will be described in more detail below.

6.2.10.2 Our Approach

Our approach for building frame structures uses the PREDS extended with GermaNet information as input. It uses a set of rules that work as follows: The left-hand side (matching) of each rule describes a local tree structure with node and edge labels, the right-hand side (output) describes the corresponding frame. By co-indexing left-hand side and right-hand side, frame elements of the frame[17] are filled with references to the head nodes of the corresponding dependency structures. Rules and parts of rules (see below) are annotated with scores. These scores are used to choose the best analysis in cases where more than one possible analysis is found.

We have described above how we have compiled the rule set, namely by semi-automatic extraction of rules from the Salsa corpus and addition of some manually written rules (4.3.3).

[17]We assume frame elements here to include the 'frame evoking element' or 'target' of the frame, i. e., its 'head word': For a sentence like *'Vodafone bought Mannesmann.'*, e. g., that evokes the frame COMMERCE_BUY, the special frame element or 'role' FRAME_EVOKING_ELEMENT would be assumed to be filled by *'bought'*.

One can either view the resulting structures as an additional level of (shallow) trees anchored in the dependency structures (each leaf node, i. e., each frame element, corresponds to a node in the dependency structure) or, as we have done earlier, as adding FrameNet labels to the existing structures (cf. 5.1). These two views are equivalent under the assumption that the underlying *structures* are equivalent (cf. 5.2.2.1).

An example of the derivation process is presented at the end of this section.

Disambiguation of Syntactic Structures. One important feature of our frame analysis is that whenever it can disambiguate an ambiguous PReDS input, i. e., when a rule matches an underspecified structure, the disambiguated structure replaces its original. As described above, this applies especially for the following cases (cf. 6.2.7.1): when PP attachment is ambiguous, when grammatical functions for verb arguments cannot be assigned on morphological features alone, or when the internal structure of complex compound words cannot be resolved.

Use of GermaNet Information. The matching part of the rules in the frame grammar can make reference either to PReDS labels, to GermaNet concepts or both. By using GermaNet concepts, selectional restrictions or selectional preferences for fillers of a frame element can be expressed. When matching GermaNet concepts, all hyponyms of that concept are considered to match. This allows us to easily specify more general semantic concepts like *Animate* by specifying one or a small number of corresponding GermaNet synsets (here, *'Lebewesen, Kreatur, Wesen'*, *'living thing, creature, being'*). Note that this selectional information was added manually to our frame assignment rules.

Most matches with GermaNet concepts in the rules are marked as optional, but they have a score greater than zero. This ensures that in syntactically ambiguous cases (for example, when subject and direct object cannot be distinguished morphologically), an analysis where the frame element preferences are met (and thus the GermaNet concepts can be matched) is preferred.

In some cases, we also provided for some typical metonymies to improve recall. For example, to represent the *'Animate'* concept, we added the synset corresponding to *'Organisation'*. This allows for conventionalised metonymies that would otherwise cause a sortal mismatch (for example in *'Vodafone bought Mannesmann.'* where *'Vodafone'* is allowed to be a filler of the BUYER frame element, though it is not an *Animate* entity). Beyond these very common cases, we made no attempt to resolve metonymies at all (cf., e. g., the approaches described in Markert and Hahn, 2002; Harabagiu, 1998); as we use GermaNet information as selectional *preferences* only (and not restrictions), more complex cases will still be correctly annotated, but no disambiguation will be performed.

Parsing. The parsing process itself proceeds as follows: For every node in an input structure, all potentially matching rules are determined. This is done by first finding all descriptions of frame evoking elements that match the node and then checking whether all obligatory frame elements are present (using their syntactic description, possibly extended with GermaNet information, as described above).

When the input structure is underspecified, a match is considered possible when one of its resolved forms matches. We have described above (cf. 6.2.7.1) that PPs are always attached low by default. In ambiguous cases, they are marked accordingly. That means that for all 'chains' of PPs in a sentence, the corresponding structures within the resulting overall structure are attached one below the other. Whenever the low attachment was incorrectly chosen, the node corresponding to the wrongly attached phrase must be re-attached. Thus, when matching a rule that includes PP daughters of a node, all PPs in all PP chains below the currently investigated node must be visited and tested for potential matches. If such a match is possible (and if the overall parse containing that structure is globally optimal), the overall structure will eventually be re-structured accordingly. Thus, when matching, a PP daughter in a rule behaves somewhat like a rubber-band in that it can stretch over a chain of default-attached PPs of arbitrary depth.

Finding Optimal Solutions. To find optimal solutions, we compute overall scores for all matching frame structures in a sentence. These scores are stored in a structure that includes information about the consequences of re-attaching parts of the structure: If, for example, a PP that is attached below an NP could be a frame element of either that NP or of the matrix verb, the corresponding scores for both attachment alternatives can be read off directly from this structure, possibly also including the information that without this particular frame element the corresponding rule that matches the matrix verb or the NP cannot be applied, as the PP contains a core frame element of the relevant frame. The optimal parse can then be found by traversing the structure after all alternatives have been entered. In cases where several options receive the same score, they are kept according to the principles described above (6.2.1.2).

As an example, consider the PREDS graphically represented in fig. 6.7 for the sentence '*Das Unternehmen hat 1997 die Zahl seiner Kunden auf über eine Million verdoppelt.*' (*The company has doubled the number of its customers to more than one million in 1997.*). We have used the same example above (as fig. 6.6), however, the example above contains no GermaNet information. The fully resolved structure is shown in fig. 6.8 and further discussed below in 6.2.11. The initial representation derived by the parser is underspecified in two ways: The NPs *'das Unternehmen'* (*the company*) and *'die Zahl'* (*the number*) could both be either subject or object of the sentence. Then, the PP *'auf über eine Million'* (*to over a million*) could (syntactically) be attached both to *'seiner Kunden'* (*its customers*) or to the main verb (indicated by the dotted lines).

In the frame annotation process, *'verdoppeln'* is identified as frame evoking element for the frame CAUSE_CHANGE_ON_SCALAR_POSITION. As the AGENT frame element is marked in the grammar with a selectional preference for an agentive argument, the phrase headed by *'Unternehmen'* is preferred (as organisations are considered as possible agentive arguments by virtue of conventionalised metonymy). Thus, the ambiguity between subject and object is resolved.

As PP_{auf} is listed as a possible way of expressing the VALUE_2 frame element of CAUSE_CHANGE_ON_SCALAR_POSITION and as there is no other frame evoked that is competing for this argument (as the NP *'seiner Kunden'* might), high attachment of the PP is preferred and both ambiguities are thus resolved during the frame annotation. The resulting disambiguated structure is shown in fig. 6.8.

Parsing Complexity. In practice, the problem of matching rules of this form is a (strongly constrained) instance of the unordered tree inclusion problem (Kilpeläinen, 1992). Though the 'full' problem is known to be NP-complete, linear time algorithms exist for fixed patterns (Kilpeläinen, 1992, 36). Besides, as the 'rubber-band' only applies to PP chains, the problem can mostly be simplified to the linear-time solvable problem of unordered child inclusion (Kilpeläinen, 1992, 19–20, 39–42). As, moreover, the 'included' trees (i. e., the left hand sides of the rules) are in general very small (typically of depth 1), parsing will be fast.

6.2.11 Sample Output

In this section, we describe the sample output for the following two sentences, exemplifying a number of points discussed in this section. The actual XML output can be found in Appendix A. A somewhat abridged graphic representation of the final parser output is shown in fig. 6.8. Figure 6.7 shows the intermediate, partly underspecified representation of the second sentence before frame assignment (cf. 6.2.10.2).

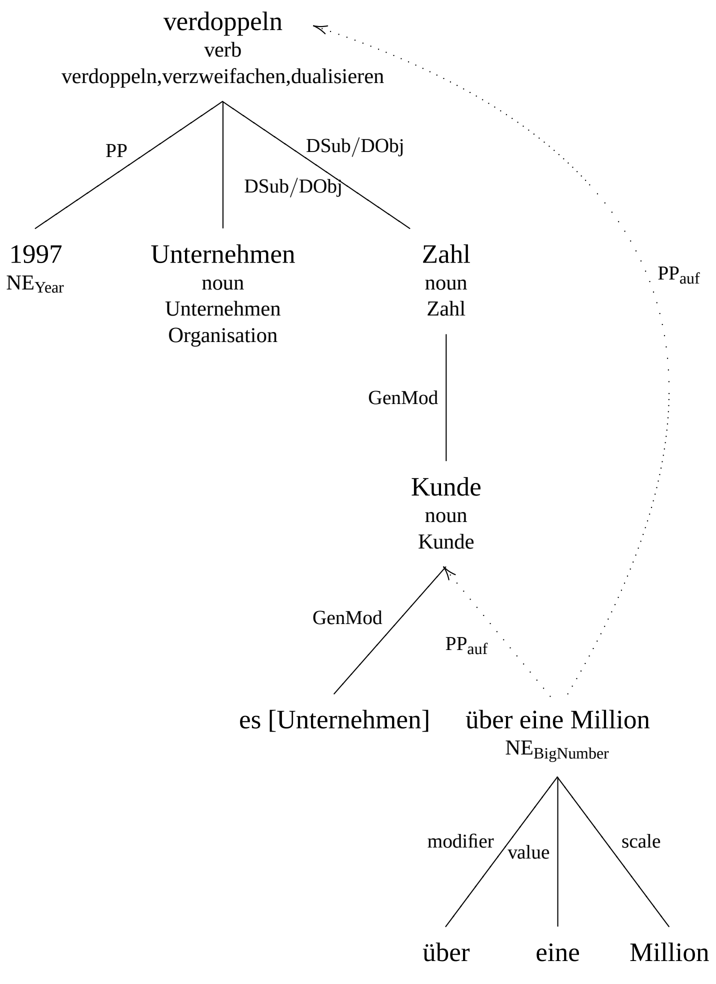


Figure 6.7: Underspecified PReDS Representation for ‘*Das Unternehmen hat 1997 die Zahl seiner Kunden auf über eine Million verdoppelt.*’

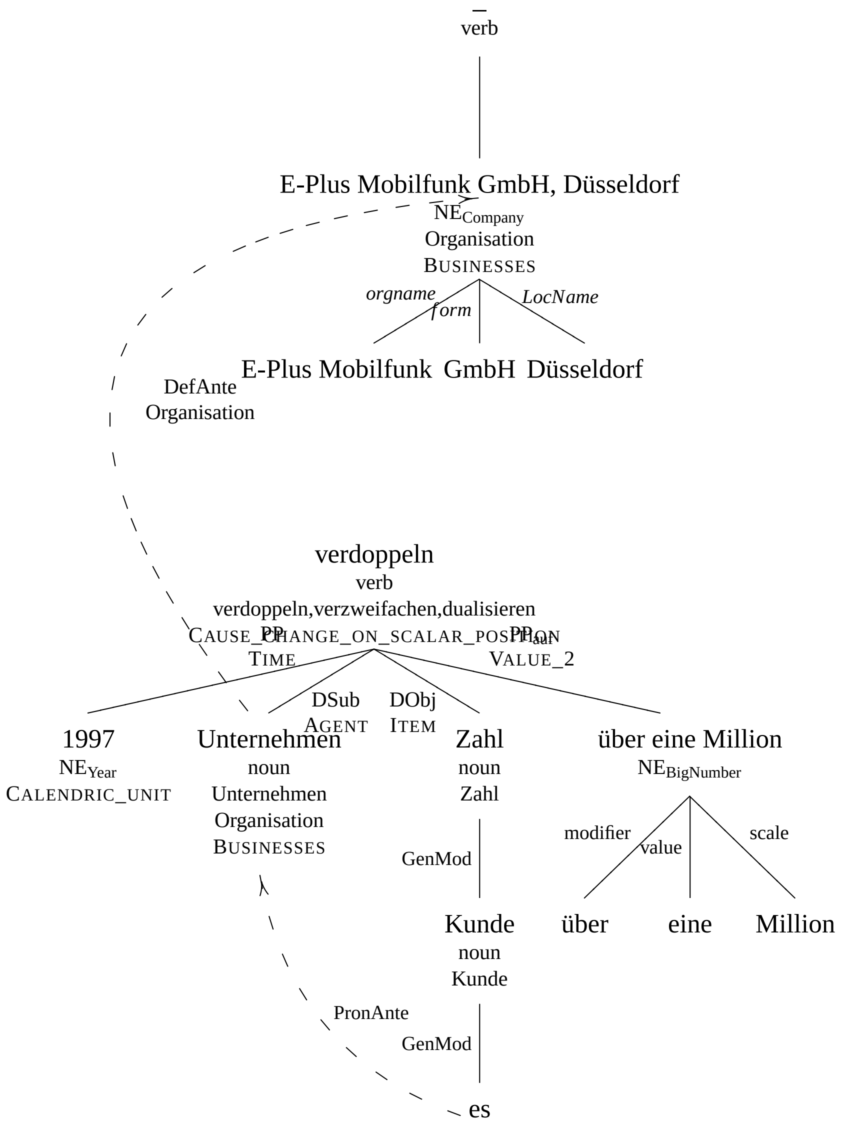


Figure 6.8: PReDS Representation for *'E-Plus Mobilfunk GmbH, Düsseldorf: Das Unternehmen hat 1997 die Zahl seiner Kunden auf über eine Million verdoppelt.'*

(6.1)

E-Plus	*Mobilfunk*	*GmbH,*	*Düsseldorf:*
E-Plus	mobile wireless	GmbH,	Düsseldorf:

Mobile phone provider E-plus GmbH, Düsseldorf:

Das	*Unternehmen*	*hat*	*1997*	*die*	*Zahl*	*seiner*	*Kunden*	*auf*	*über*
The	company	has	1997	the	number	of its	customers	to	over

eine	*Million*	*verdoppelt.*
one	million	doubled.

The company has doubled the number of its customers to more than one million in 1997.

As discussed above, the PP *'auf über eine Million'* (*to more than a million*) receives low default attachment, i. e., it is attached to the NP *'seiner Kunden'* (*of its customers*). Only during the frame structure construction, the PP is recognised as a frame element (namely VALUE_2) of *'verdoppeln'* (*double*, CAUSE_CHANGE_OF_SCALAR_POSITION) and attached directly below it. In addition, the NPs *'das Unternehmen'* (*the company*) and *'die Zahl seiner Kunden'* (*the number of its customers*) are both ambiguous between nominative and accusative case. Thus, subject and object of the sentence cannot unambiguously be assigned. Only additional information during the frame structure construction (namely a preference for an animate filler of the AGENT frame element) allows the correct assignment of the subject and object tags.

Note also especially the links added by the anaphora resolution for *'das Unternehmen'* (*the company*), referring to *'E-plus'*, and the possessive determiner *'seiner (Kunden)'* (*'(of) its (customers)'*), referring to *'das Unternehmen'*.

6.3 Storing and Retrieving Text Representations

In this section, we will describe issues concerning the choice of database and database interface issues. See also the general discussion on the use of a database system in our approach in 5.4.1.

6.3.1 MySQL as Relational Database

As described above (5.4.1), we have decided to use a general-purpose relational database for storing and retrieving text representations: While these systems lack specialised support for tree-like queries, they are highly optimised for efficiently handling general queries.

We chose MySQL after a number of experiments with the two best-known freely available databases MySQL (`http://www.mysql.org/`, distribution 4.0.20, version 12.22) and PostgreSQL (`http://www.postgresql.org/`, we used version 8.0). This decision was mainly based on the observation that queries on the MySQL database system outperformed the same queries on the PostgreSQL system by 30–50 % regarding overall retrieval time (on the same machine and the same experimental data-set (including the same database index structures). We assume that this may be due to some special query optimisation heuristics integrated in MySQL enabled that system to handle our special type of queries (cf. 5.4.3) more efficiently.

The MySQL database is accessed through the standard Perl database interface DBI. As we mostly use standard SQL[18] and only a small number of specialised MySQL constructs and data types, porting the system to use other databases with a standard SQL interface would be relatively easy.

6.3.2 Storing Linguistic Knowledge

To extract and store linguistic knowledge from the lexical semantic databases GermaNet and FrameNet (cf. 5.2), we proceeded as follows:

First, we transferred the data into suitable tables in our database. Both GermaNet and FrameNet data were available to us in the form of XML files. The respective file formats for GermaNet are described in detail in Kunze and Lemnitzer (2002a); Kunze and Wagner (1999), for FrameNet in Baker et al. (2003).

From these databases that hold the original information from the lexical databases, the relevant information for our lexical knowledge base was extracted using SQL-queries. In a first step, numerical indices were generated for all labels to allow efficient processing. Then, inference (relabelling) rules were generated from the relations in the linguistic knowledge bases. For example, for all GermaNet synset, pairwise inference rules were created for all lemmata in the synset (cf. 5.2).

Afterwards, transitive relations were computed (again using a SQL-query) and added to the database. This step was repeated until the maximum depth of inference (here, three) was reached (cf. 5.3.2.3).

6.3.3 Storing Text Representations

For each document that is to be searched for answers by the QA system, first a text representation is derived using our full parse chain (6.2). The XML files

[18] SQL has been standardised by ISO. The latest revision is ISO/IEC 9075:2003 (SQL:2003).

that are output by the parser are then stored in the system database. The database structure has already been described in 5.4.2.

In order to allow efficient processing, all lexical semantic labels are translated into integers using the indices generated when storing the knowledge bases (6.3.2).

In storing the text representations, a number of ‘shortcuts’ are additionally stored for coreferent and for coordinated entities. This has been described above in 5.2.2.8.

Alternative structures are added to the database for underspecified structures that could not be resolved during the parse process (cf. also 5.2.2.7): For PPs that can be identified using the preposition and GermaNet information for the nominal head of the phrase as belonging to a number of ‘semantic’ groups, such as TIME, PLACE, CAUSE and REASON, with a high degree of certainty and that have not been disambiguated as filling some FrameNet frame elements, additional dependency links are added to all possible landing sites (cf. 6.2.7.1). This allows us to store unresolved cases of underspecification in a way that does not lead an ‘explosion’ of the database, as only a limited number of additional dependency relations is added. This strategy can be seen as a sort of fall-back strategy to overcome shortcomings of the parsing process.

The original document is also stored in the database to allow it to be displayed upon request. This feature is described in 6.4.6.

6.3.4 From Question to Database Query

Questions that the user enters (6.4) are first translated into the text representation using out parsing chain (6.2). From the question representation, a database query is then built up recursively. The algorithm for translating structured question representations into ‘flat’ database queries has been described above (5.4.3).

The generated sub-queries are sent to the database and executed. All matching structures are returned from the database in the form of flattened tree structures (5.4.2). These flat structures are translated back into the original PREDS for further processing: The full representation can be easily reconstructed since the database representation contains all information from the original structure.

Coreference information is integrated into the retrieved structures: For all entities that form part of a coreference chain, all other elements of that coreference chain within that text are also retrieved. This information is available for answer generation (cf. 6.4.3), so that pronominal forms can be replaced by definite descriptions or named entities and the most informative referring expression in the chain can be used. For a named entity, e. g., a full form (containing, for example, first name and last name of a person) will be preferred over a short form (containing only the last name, for example).

6.3.5 Query Optimisation

Suitably indexing the data for the expected type of query is of central importance for efficient data access on relational databases (Abiteboul et al., 1995, 106–8): Without indexing, data from the tables relevant for that query must be compared with the query row by row. This requires that the whole table must be loaded from disk into main memory – a comparatively time-consuming operation.

As indices contain only a small portion of the information in the database table, they can be kept in main memory in most cases. Indices are usually organised as binary trees; rows that match the search key can be found in time that is logarithmic with respect to the table size. In addition, optimally only relevant (viz. matching) lines from the tables need to be loaded from disk when indices are used, further reducing overall execution time.

In all queries, we join tables on integer columns only and use indices over these columns, ensuring efficient table joins. Multi-column indices were added to further speed up retrieval after a manual evaluation of sample queries.

The query optimisation module of the database systems automatically produces an optimisation for all queries by deleting redundant sub-queries (if any) and then generating an evaluation plan, i. e., an ordering of all sub-queries through a number of heuristics that is considered to most efficiently examine the search space of the query (Abiteboul et al., 1995, 106–120). Query optimisation will only produce an approximation and may occasionally produce less than optimal results for complex queries since finding the globally optimal search strategy is NP-complete (Abiteboul et al., 1995, 120–2).

The optimisation generated by these database systems may differ according to such factors as the number of identical items within one index and therefore are difficult to foresee. Adding control (via selecting search indices) only solves this problem for some query instances, but not in general.

We still occasionally encountered queries that perform poorly on the MySQL engine, leading to answering times of up to minutes. Manual evaluation of these queries showed that MySQL's query optimisation modules produced a sub-optimal sequence of sub-queries. These less-than-perfect optimisations are due to differences in the data: For example, a large number of hits for one sub-query (say, for one word) may induce the query optimiser to postpone this sub-query, even though without using the constraints from this sub-query early, the *overall* query evaluation becomes much slower. In each observed case, the engine could be forced (by manually selecting a preference for different indices) to use the optimal sequence of sub-queries, leading to optimal overall query evaluation and thus to answering times of a few seconds at most. However, no general strategy for optimising all queries could be found.

Our interim conclusion is that ‘linguistic’ queries using tree-like structures are simply so dissimilar from generic queries, for which database engines like the MySQL engine are optimised, that the occasional suboptimal query may be produced by the engines. In principle, this could be overcome through further experimentation and further manual optimisation for different combinations of query and underlying data. It should be noted that this task is dissimilar from general query optimisation where the *structure* of queries is determined by the application and known in advance. In contrast, the structure of the queries in our application changes heavily, because the queries simulate linguistic structures. As suboptimal queries produced for questions are comparatively rare (cf. 7.2.1.4), we have currently not further pursued this point. In the long run, the use of databases supporting storing and searching rich structures may provide a more promising option, in our opinion.

As answer search is split into sub-queries corresponding to elementary trees, we can find out exactly which part of the query failed. From this information, appropriate user feedback can be generated (cf. 6.4.7).

6.4 User Interface

In this section, we give an overview of the user interface of our system. Several aspects of its design, especially anaphora and ellipsis resolution on questions, have also been described in Fliedner (2006b).

6.4.1 Basics

The user interface has been implemented as a terminal application; we consider this to be the most natural representation of a written language dialogue. User input (questions) and system output (answers) will alternate in a single window, and the last couple of questions and answers will remain visible. Earlier dialogue moves can be accessed by scrolling up in the terminal window. The users can type in their questions via keyboard, the system answers will then be output below the question (using a different font for better readability). This allows an easier dialogue-style interaction than, for example, an HTML interface displayed in an Internet browser that only displays a single question and answer would.

Besides posing questions to the system, the user can also influence the system behaviour in a number of ways using a set of special command keywords.

They can, among other things, be used to choose the level of system verbosity, to limit the number of returned answers and to toggle the answer justification mode. These features will be described in the following sections.

6.4.2 Walk-Through Example

Here is a short example of an interaction with the system. It shows how the different linguistic modules cooperate. Features of the user interface will be described in more detail below.

The example is based upon the following example text extracted from the German Wikipedia by the SmartWeb AnswerGatherer project; Internet users were asked to submit test questions about the article (the procedure is described in greater detail below, 7.2.1.2).

> Die Abtei von Hambye ist die vollständigste mittelalterliche Konventsiedlung nach dem Mont-Saint-Michel und befindet sich in der Nähe der Kreisstadt Percy (Manche). Die Abtei wurde im Jahre 1145 von Guillaume Paynel, dem Herren zu Hambye, gegründet.
>
> Hambye Abbey is, after Mont Saint Michel, the most complete medieval convent settlement and is located in the vicinity of the district town of Percy (Manche). The abbey was founded in the year 1145 by Guillaume Paynel, Lord of Hambye.
>
> (German Wikipedia, article *Abtei von Hambye*, 2006-10-18)

We will now discuss how the questions gathered for this article would be answered by our system.

```
> Wo steht die Abtei von Hambye?
Hier die Antworten (gefunden in 1.029s):
1: In der Nähe der Kreisstadt Percy
(WikiPedia, 19.8.2005, Relevanz: 0.9000)
Keine weiteren Antworten gefunden.
```

> **Where does Hambye Abbey stand?**
> Found the following answers (in 1.029s):
> 1: In the vicinity of the district town of Percy
> (WikiPedia, 2005-08-19, relevance: 0.9000)
> No further answers found.

To arrive at this answer, the first sentence of the original text needs to be correctly analysed. The topological parser (6.2.4) here identifies the coordination and assigns *‘die Abtei von Hambye’* (*Hambye Abbey*) as subject to the verb *‘sich befinden’* (*be located*). Note that a shallow, pattern-matching approach to QA might conclude that *Mont Saint-Michel* is located near Percy, as it is the closest NE.

During matching, *‘die Abtei von Hambye’* (*Hambye Abbey*, literally *Abbey of Hambye*) in question and answer matches, as *‘Hambye’* (*Hambye* is not recognised by the morphology and thus becomes a Generic NE), and *‘Abtei’* (*Abbey*) is identical, and so is the PP_{von} labelling the dependency edge between the two. Then, *‘stehen’* (*stand*) and *‘sich befinden’* (*be located*) must be matched. As they are related in GermaNet, a suitable relation between the predicates exists in the linguistic knowledge base, allowing a good (0.9) match.

Correctly identifying the answering constituent in this case happens through an additional *Place* relation (cf. 5.2.2.7): Both the interrogative adverb *‘wo’* (*where*) and the phrase *‘in der Nähe. . . ’* (*in the vicinity. . .*) are assigned an additional *Place* relation, linking them to their main verb (in addition to the PREDS relations Mod and PP_{in}, respectively). Thus, question and answer structure can be matched with a high relevance score. Consider the following justification that the system produces upon request (we give only the English translation here):

```
Justification:
"Stehen" in the question and "befinden" in the answer
are similar (hyponymy, Relevance: 0.90).
    "Stehen" belongs to the GermaNet synset "stehen"
     and "befinden" to the synset "befinden"
     (Relevance: 0.90).

    The relation "Place" between "stehen" and "wo"
    in the question is identical with the one between
    "befinden" and "Nähe" in the answer
    (Relevance: 1.00).

    The grammatical relation "DSub" between "stehen"
    and "Abtei" in the question is identical with that
    between "befinden" and "Abtei" in the answer
    (Relevance: 1.00).
"Abtei" in the question and "Abtei" in the answer are
identical (Relevance: 1.00).
    "Abtei" belongs to GermaNet synset "Abtei" and
    "Abtei", as well (Relevance: 1.00).

    The grammatical relation "PPvon-dat" between
    "Abtei" and "Hambye" in the question is identical
    with the one between "Abtei" and "Hambye" in the
    answer (Relevance: 1.00).
"Hambye" in the question and "Hambye" in the answer are
identical (Relevance: 1.00).

"Wo" in the question is answered by "Nähe".
```

The answer is then generated from the node matching the node with the interrogative adverb in the question, providing a constituent answer.

```
> Wann wurde die Abtei gegründet?
Hier die Antworten (gefunden in 0.771s):
1: Im Jahre 1145
(WikiPedia, 19.8.2005, Relevanz: 1.0000)
Keine weiteren Antworten gefunden.
```

When was the abbey founded?
Found the following answers (in 0.771s):
1: In the year 1145
(WikiPedia, 2005-08-19, relevance: 1.0000)
No further answers found.

Here, the reference to *'die Abtei'* (*the abbey*) must first be resolved. The preferred candidate is *'die Abtei von Hambye'* (*Hambye Abbey*) from the first question.

To find the correct answer, the coreference in the original text (namely between *'die Abtei von Hambye'* (*Hambye Abbey*) in the first sentence of the text and *'die Abtei'* (*the abbey*) in the second sentence must be resolved. As described above (5.2.2.8), an additional relation is added, so that, again, question and answer representation can be directly matched.

```
> Von wem?
Ich habe die Frage in folgende "Langform" übersetzt:
"Die Abtei von Hambye wurde von wem gegründet?"
Hier die Antworten (gefunden in 1.671s):
1: Von Guillaume Paynel
(WikiPedia, 19.8.2005, Relevanz: 1.0000)
Keine weiteren Antworten gefunden.
```

By whom?
I have translated the question into the following long form: "By

```
whom was Hambye Abbey founded?"
Found the following answers (in 1.671s):
1: By Guillaume Paynel
(WikiPedia, 2005-08-19, relevance: 1.0000)
No further answers found.
```

This example (not in the original questions gathered for the document) shows ellipsis resolution in questions. The new question consists of a single phrase with a question word, therefore the representation of the last question is re-used with the *wh*-phrase replaced by the new one. The new question is presented by letting the generation module produce an output for the derived representation. We added this feature to help avoid misunderstandings through errors during ellipsis resolution.

Again, the answer can be found by matching the representation of the second sentence of the original text with the definite description *'die Abtei'* (*the abbey*) resolved to its antecedent *'die Abtei von Hambye'* (*Hambye Abbey*).

The answer *'von Guillaume Paynel'* (*by Guillaume Paynel*) is generated in neutral answer verbosity mode, i. e., only the person name is used. By switching the generation module to verbose mode, a more informative answer can also be generated, namely *'Von Guillaume Paynel, dem Herren zu Hambye'* (*by Guillaume Paynel, Lord of Hambye*).

Note that the system produces answers so that question-answer congruency holds. If, for example, the question is in the active voice (*'Wer gründete die Abtei von Hambye?, Who founded Hambye Abbey?*), the generated answer will be *'Guillaume Paynel'* or *'Guillaume Paynel, der Herr zu Hambye'*, as the grammatical function of the interrogative pronoun is utilised in answer generation to ensure congruency (6.4.3).

6.4.3 Answer Generation

In 3.2, we have argued that question-answer congruency plays an important rôle when answering questions in natural language. It is, in fact, an important advantage of linguistically informed QA that it provides a basis for generating answers that exhibit answer congruency through the structured representations of both questions and answers. We achieve this by using an answer generation module that employs both the representation of the user's question and the text representation retrieved from the database to generate a suitable answer.

6.4.3.1 Motivation

The answer generation uses the PREDS retrieved from the database that matches the question to construct the answer. In cases where question representation and answer representation differ syntactically, the answer representation will be transformed so that it fits the question. If, for example, the question uses the passive voice while the retrieved answer originally used the active voice, the answer (and especially phrasal answers) will be changed accordingly. Given the representation of *'Mannesmann was bought by Vodafone.'*, the active-voice question *'Who bought Mannesmann?'* would be answered by *'Vodafone'*, while the passive-voice variant *'By whom was Mannesmann bought?'* would be answered by *'By Vodafone'*. We think that by generating answers that structurally agree with the question instead of just presenting the user with a text snippet (cf. 2.2.2) a marked increase in naturalness can be achieved.

Note that this problem is far more important for languages with strong case marking (such as German) than for languages like English: While a bare NP as an answer will sound clumsy at worst in English (i. e., it is generally sufficient to identify an NP snippet that answers a constituent question and return it as an answer), extracting an NP is not sufficient in German, as the text snippet in the original text will often be in the wrong case and thus be ungrammatical as an answer.

6.4.3.2 Generating Answers from PREDS

Our generation module takes a PREDS (i. e., a syntactic dependency structure) representing (part of) a sentence as its input and produces German surface strings from it. The generation module is a sentence and phrase generation module rather than a full text generation module: As our approach matches syntactic structures representing question and answer sentences, only phrases up to single sentences (possibly with subordinate clauses) can be generated. In general, only short constituent answers are generated (see below).

Work on natural language generation is mostly concerned with generating text from semantic representations (cf., e. g., Reiter and Dale, 2000, 1997; Traum and Habash, 2000; Busemann, 1996); the generation process is often divided into the three steps of text planning, sentence planning and realisation (cf., e. g., Reiter and Dale, 2000; Lavoie and Rambow, 1997; Reiter, 1994). Our generation module is limited to the last of these steps, namely realisation, as it takes a lexicalised syntactic dependency structure as its input rather than an abstract, semantic representation. It is thus similar to the work described in Lavoie and Rambow (1997); Elhadad and Robin (1996).

The task of realisation consists of correctly inflecting the content words represented in the input, adding any necessary function words (especially determiners, prepositions and auxiliary and modal verbs) and properly linearising the resulting forms. The goal was to implement a module that can generate an adequate output for every German input sentence that the PREDS parser can handle (and – even more importantly for the default constituent answers – for all its sub-parts). Our approach thus goes beyond template-based generation (as that would not be flexible enough for the task at hand, cf. the discussion in Reiter, 1995), but remains short of a full generation from a semantic input.

Word Form Generation. For inflection, we use the Gertwol morphology (6.2.3) in generation mode. In this mode, the module takes a word stem and an inflection description as its input and generates the corresponding surface form. As the coverage is identical with that of the analysis mode, we can thus generate all surface forms for all the words that can be morphologically analysed. The generation module is configured to always produce output in reformed German orthography (cf. 6.2.3.1).

Generation Rules. Larger structures are built recursively by the generation module: A small set of local rules describe for every node type in the dependency structure how a surface structure for this node type is to be generated and in what order its dependents are to be generated. For example, the rules for generating a text string from a noun node[19] will stipulate that its determiner (if any) be generated first, followed by attributive adjectives, followed by the inflected form of the noun itself, followed by any post-nominal modifiers.

We currently make a number of simplifying assumptions: We generally allow no stylistic variations and we especially use a simple unmarked word order. As far as possible, the order of the constituents used in the original text is preserved. As word order in German is influenced by a great number of factors (among them the relative ‘weight’ of phrases in a sentence and, of course, text structure), we decided to go for simplicity when that is not possible. For example, sentences will be generated using an unmarked word order (subject, indirect object, direct object, complementisers)[20].

Another example is that relative clauses always immediately follow their antecedent, whereas in ‘manually written’ German texts, heavy relative clauses tend to be extraposed. The results may occasionally be less than perfect from

[19]The description is somewhat simplified for ease of exposition.

[20]Using some additional rules to account for changes of word order when, e. g., pronouns are used.

an æsthetic point of view, but they will be grammatically correct and they tend to be un-ambiguous due to their preference of low, direct attachment.

For NPs that refer to entities that are known to be contextually salient because they were mentioned in the question (and are thus present on a stack of discourse entities that our system builds during the interaction, that is also used for anaphora and ellipsis resolution on the user's input, see below, 6.4.4), personal pronouns or possessive pronouns are generated as a short form in the answer.

One important feature of the generation module is that it allows to generate text for partial structures only: In general, it will be called to produce a short answer (i. e., a constituent answer, cf. 3.2). This is achieved in the following way: After matching question and answer representation (cf. 5.1), the node that matched a question node will be identified and only that node with its dependents will be passed to the generation module; only from this partial structure, an answer will be generated. The case to be used for generation of NPs and prepositions for generation of PPs is determined by that of the *wh*-phrase in the *question*, ensuring question-answer-congruency as described above.

Setting the Level of Detail. The generation process can be further controlled by the user by setting the required level of detail for the generated answers. Changing the level of detail can be done through a user interface command. On the one hand, the user can switch the system to produce only a minimal answer. On the other hand, the generation of a full sentential answer can be requested. Consider, for example, the question *'Who was killed by Lee Harvey Oswald?'*. After matching the the representation of *'It was in 1963 that President John F. Kennedy was killed by Lee Harvey Oswald.'*, in neutral mode the answer *'President John F. Kennedy'* would be produced. When switched to minimal answer mode, only the NE *'John F. Kennedy'* would be output. In sentence mode, the whole matching sentence, namely *'President John F. Kennedy was killed by Lee Harvey Oswald in 1963.'*, would be returned.

As mentioned above, the structure retrieved from the database contains representations of all expressions that form part of coreference chains in the sentence underlying the answer structure. More or less additional information for an entity in the answer can be generated using different heuristics for different levels of detail that the user can request. While the heuristics for minimal answer mode prefer single NE answers, in neutral mode an additional describing NP will be generated (if information is available in the text) as a close apposition of an NE. By switching the system to loquacious mode, *all* expressions in the coreference chain can be generated as appositions.

We consider this possibility of easily changing the answer verbosity an additional advantage of the answer generation approach.

Note that the system automatically switches to sentential answer mode in the case of uncertain answers (3.2.3.3, 5.1.5): First, a suitable warning is presented, then, the full answer sentence is generated. This method is used to make it easier for the user to gauge the correctness of the answer.

Generating Answers to Yes/No-Questions. For yes/no-questions, a slightly different strategy is used: As described in 3.5.1.2, our matching algorithm will find both 'positive' and 'negated' answers to a question, the latter including answers that use antonyms of words in the question. By answer checking (5.1.5), we can identify answers with inverse polarity.

We use this information to generate answers to yes/no-questions in the following way: First, we output *'yes,'* if question and answer have the same polarity and *'no,'* if the polarity differs and then the whole answering sentence. We consider this a natural and informative way of answering yes/no-question, as it combines the short answer (yes/no) with the relevant information. So, for example an answer like *'Yes, he killed him.'* may be generated for the question *'Did Lee Harvey Oswald kill John F. Kennedy?'*. Note that pronouns are generated here to refer to the entities in the question, as the entities are present on the current stack of discourse entities.

6.4.4 Towards Interactive Question Answering

Using text representations of both questions and documents (and thus potential answers) allows us to extend the QA user interface in another interesting direction, namely making a move towards a fuller, dialogue-style interaction. Most 'classical' QA systems today are geared towards 'one shot' questions and answers where each question is processed independently from earlier questions and answers. For users, however, it would be easier and also more natural if they could rather lead an information seeking dialogue with a QA system: The users' questions to such an extended system are interpreted taking the dialogue so far into account.

As a first step towards a natural, interactive QA system, our system resolves anaphora and some simple types of ellipsis on the user input. To add to this natural 'flavour', it also employs pronoun anaphora in its generation (cf. 6.4.3.2). These features are discussed in Fliedner (2006b); we will repeat some key points here. Adding these features is relatively easy in a linguistically informed QA system, as structured representations are available for both the user's questions and the system's answers.

6.4.4.1 Anaphora and Coreference in Questions to Qustion Answering Systems

Recently, there has been a growing interest in extending QA systems towards a fuller interaction. This is marked by the integration of tasks that require some anaphora resolution and also ellipsis resolution into the shared competitions, namely in the TREC 2004 QA TREC (Voorhees, 2005) and the NTCIR-5 Question Answering Challenge 3 (Kato et al., 2005). In both competitions, questions concerning one topic are grouped together and questions may make use of anaphoric expressions referring back to entities mentioned earlier.

While in TREC, anaphors generally only refer back to entities mentioned in earlier questions, in NTCIR, they may also refer to *answers* to earlier questions that the systems are expected to find. As in TREC, the topic for each group of questions was explicitly given, most systems simply replaced all potentially anaphoric expressions by that string (Voorhees, 2005).

The Japanese questions and answers in the NTCIR QA challenge often contain zero anaphora and ellipses, as these are very common devices in Japanese. In Matsuda and Fukumoto (2005), an overview of the different types of anaphora is given. We expect that spoken language dialogue systems for German (and English) would also have to handle these more difficult types, however, as long as typed input is used, we expect to see few of them. In the evaluation we have not observed any cases of zero anaphora.

6.4.4.2 Anaphora Resolution

For anaphora resolution on the users' questions, we employ the anaphora resolution module described above (6.2.9). This module is mainly used to resolve anaphoric references when deriving text representations from the texts in the QA system's document collection, that is, on coherent text.

In re-using the coreference module, we make the assumption, that the users' questions and the system's answers will form a coherent discourse (cf. Vitacolonna, 1988), so that the same methods for anaphora resolution can be applied. We found this a useful starting point, even though we expect that additional phenomena will eventually have to be handled in a real dialogue[21].

We thus process both the representations derived from the users' questions and the text representation that are used to generate the system's answers with the anaphora resolution module, turn by turn. When a possible anaphor is found

[21]For example, in an investigation of English *spoken* dialogues, Miriam Eckert and Michael Strube found almost 30 % of pronominal anaphora to have no single NP antecedent (Eckert and Strube, 2000). We assume that the information seeking dialogue using written (or rather typed) interaction will make less use of such 'vague' references. See also 7.2.1.4.

in a user's question and it can be resolved to an antecedent either in an earlier question or an answer returned by the system, the antecedent simply replaces the anaphor in the question representation before it is translated into a database query. An example was shown above (6.4.2).

Note that the expanded form of the question representation is used to query the whole database representation of the text collection, not only the document answering earlier questions, because the answer to the new question could be contained in a different document.

By this relatively simple method, we allow the users to make use of pronominal anaphora, of short forms of named entities and also of definite descriptions to refer back to persons or entities that were mentioned either by the user in an earlier question of by the system in an answer to a question.

In 6.4.2, we have shown several examples for anaphora resolution in users' questions (see also Fliedner, 2006b, for additional examples).

6.4.4.3 Ellipses

In addition to anaphora, we also allow the users to use simple ellipses in their questions to the system. Currently, two sub-types of what might be regarded as VP ellipsis are supported (see below).

Whenever an input does not consist of a full sentence, but rather a single phrase, it will be considered a candidate for a repetition of the last question where most of the question is elided. (Users can employ such elliptical questions to save a lot of typing.) We use a comparatively simple approach based on copying the representation of the phrase that makes up the elided form of the question into its 'proper' place in a copy of the original question representation. Thus, a syntactically and/or semantically parallel element is replaced. This is similar to the approach to ellipsis resolution described in Kehler (1993). The following two types of phrases are currently considered:

***Wh*-phrase.** When a follow-up question consists of a *wh*-phrase and the last question was a *wh*-question, the *wh*-phrase of that question is replaced by the 'new' *wh*-phrase. An example has been shown above (6.4.2).

NPs/PPs. This case is somewhat more complex, since the question may contain more than one phrase that could be parallel. Thus, we first check for syntactic parallelism: If the NP/PP is unambiguously parallel to a phrase in the last question that phrase is replaced (for example, if the question contains an accusative object and the NP is unambiguously in accusative case).

If this syntactic check does not unambiguously identify a candidate, then the possible candidates are checked for semantic compatibility. We use the same compatibility check as for definite NP antecedents in anaphora resolution (cf. 6.2.9).

These heuristics will, in most cases, return a full question representation assembled from the last question by replacing the parallel element with the new phrase. This ‘long form’ of the question is then translated into a database query as described above and evaluated.

In general, the comparatively simple strategies used for anaphora and ellipsis resolution in users’ questions work quite well.

6.4.5 Making Use of Answer Relevance

We have described above that a question may have more than one answer. While different forms of questions seem to be associated with differing preferences concerning the number of answers (e. g., *wh*-phrases in singular seem to prefer single answers), these are difficult to determine automatically and also not very ‘stable’ (the user may be mistaken in expecting only a single answer, for example).

As described above (3.2.4), our approach cannot reliably distinguish whether different descriptions that are found as answers refer to the same or to different entities. Within one text, anaphora resolution will generally allow us to identify co-reference of expressions. Across documents, currently no attempt is made at identifying co-referent expressions except for conflating answers that produce *identical* answer strings. This could be further improved by using cross document co-reference resolution.

We have therefore currently implemented the following methods for determining the number of answers to output: The user can manually select either the number of answers or a minimum threshold for the relevance score of the single answers (cf. 5.2). The system always searches all possible answers in the database. The answers are then output in decreasing order of relevance until either the maximum number of answers is reached or until the answer relevance drops below the set threshold.

In addition, the user is informed when the number of found answers is below the requested maximum number of answers or when answers are cut off, either because the answer limit is reached or because their relevance is below the minimum threshold.

This combination of methods for setting a requested number of answers provides the user with a balanced possibility of ensuring that the preferred number of answers be returned for a question. As a default, the number of answers is set to three. Thus, at most the three best answers are shown to the user who can then request additional answers to be displayed. If the user explicitly wants to see all answers, (s)he can also directly switch to the ‘all answer’ mode.

As the answers are always sorted for relevance, the answers best fitting the question will generally be returned first, facilitating answer browsing.

6.4.6 Answer Support and Answer Justification

We have described above that users will generally consider it helpful if a QA system can, upon request, furnish additional information together with an answer, allowing the user to assess the answer and clarifying why this answer was chosen (3.2.4). Our user interface provides two general ways for the user to have the answer explained, namely through answer support and answer justification.

6.4.6.1 Answer Support: Displaying the Original Text

The system can output (part of) the original text in which the answer was found. Two different levels of output are supported, namely sentence level or document level.

When answer support at sentence level is requested, the system will, together with every answer, output one or more sentences from the original document containing the answer, formatted similar to a citation: When the answer is found through coreference resolution, sentences containing antecedents to anaphoric references in the sentence matching the question will be displayed together with that sentence. Intervening sentences will not be shown and replaced by ellipses (...). This allows the user to efficiently verify the answer, since all relevant sentences are shown – and no others.

The user can also request that the whole document containing the answer be output together with each answer. If this option is chosen, all sentences that are part of the full answer (as described above) are highlighted, i. e., displayed in a bold font. Again, this feature helps the user to quickly find the relevant places in the document. While displaying the whole document makes, of course, for a lengthier output, it also allows the user to gather more additional information.

We think that both answer support modes provide an interesting extension of the possibilities discussed by Jimmy Lin and his colleagues in their study on how much context should be displayed to users in a QA system to support an answer (Lin et al., 2003): As described above, they explicitly discuss the problem of pronouns for ‘exact’ (constituent) and sentential answers (this argument can be extended to other types of anaphora as well) and note that by displaying only the sentence containing the answer is not sufficient in cases where the sentence contains pronouns. They have identified this as an important reason why users tend to prefer paragraph-length answer support[22]. We think that the inclusion of sentences containing antecedents of anaphora in sentence level answer support and/or the highlighting of such sentences in document level answer support provides users with specific and focussed additional information. Of course, it would be necessary to test this assumption in controlled user experiments (8.3.3).

6.4.6.2 Answer Justification: Describing Textual Inference

In addition to answer support, our system can also justify answers by describing the matching process between question and answer (cf. chapter 5). This allows the user to follow the 'reasoning' of the system that lead to generating a certain answer.

When the answer justification mode is switched on, the system will generate, for every answer, a textual description (using some additional formatting for clarity's sake) that summarises the match found between question and answer. An example output was shown above (6.4.2).

This is done as follows: For each node in the question representation, the node and the corresponding node in the answer are identified (by their lemmata).

Then, for every linguistic level for which a match for the elementary tree rooted in that node is found, the used relabelling rules are described with reference to the knowledge source that provided them. If, for example, a hyponymy relation between the node lemma in the question and the answer was found in GermaNet, this is reported, together with corresponding relevance score (5.2). Then, for every daughter of the node in the question, the edge and the daughter are identified (by the edge label and the lemma of the daughter node, respectively), and the employed re-labelling rule is described as above.

In cases where coreferences had to be resolved, the corresponding coreference relation (or coreference chain) is also described, mentioning the sort of anaphoric relation (i. e., pronominal anaphor, definite description or Named Entity coreference).

Answer justification especially allows finding unlikely or wrong inferences. These may be due to the system assuming a wrong word sense relative to the linguistic knowledge base (remember that no word sense disambiguation is performed, cf. 6.2.8) or to mistakes in coreference resolution (cf. also 7.2.2.3). In fact, the method was originally implemented for debugging purposes, but we

[22]Paragraph length answer support was preferred by 53.3 % of the users, while 20.0 % preferred sentence length answer support, 23.3 % documents as answer support and only 3.3 % 'exact answers', cf. Lin et al. (2003, their fig. 5).

found it potentially helpful for system users and extended it accordingly to generate textual descriptions instead of pure debugging information.

6.4.7 From Answers to Responses: Providing Additional Information

We have described above that we consider it important for a QA system that it not only gives answers, but rather responses in the sense of Bonnie Webber (3.2.3.3, Webber, 1986). We have already mentioned several points where responses, that is additional feedback to the user beyond the answer (or instead of an answer, if no answer can be found) plays a rôle.

The first point in question is the importance of notifying the user if either fewer or more answers than requested are available (6.4.5).

Then, generating a meaningful response is especially relevant in cases where the system cannot provide an answer: In that case the reason should be reported as transparently as possible to the users so that they can try to avoid the problem. In Androutsopoulos et al. (1995), supporting the user through detailed analyses in cases of failure is cited as a central (but often missing) requirement in natural language user interfaces.

The system needs to provide a meaningful output in cases where no parse for the user's question can be found. In our system, this case relatively rarely arises, especially since unknown words are considered as generic NEs (6.2.5) and thus do not lead to a parse failure. When no topological structure can be found, that is reported to the user.

By using separate database queries for elementary subtrees in the question representation, we can provide the user with detailed feed-back when no result can be found: By executing partial queries one after another, in case of a failing query, the sub-graph of the question representation can be identified for which no match can be found. This information is used to provide detailed feedback to the user.

For example, the system can, when faced with a question like ‘*Who murdered John F. Kennedy?*’ instead of returning an error message like ‘*Could not find an answer.*’ a more meaningful message such as ‘*I could not find any information on “John F. Kennedy”.*’, saving the user a lot of trouble in re-formulating the question.

6.5 Conclusions

In this chapter we have described SQUIGGLI, a prototype implementation of a German QA system based upon linguistically informed QA.

We have shown how a comparatively detailed text representation including anaphora resolution and information from GermaNet and frame semantic information can be derived by a tool chain of shallow parsing modules. One important goal in system design was to achieve a compromise between the level of detail required from the resulting structures on the one hand and broad coverage, robustness and processing speed on the other hand.

We have then summarised the interface to the database system that is used for storing the structured text representations derived from the QA system's document collection and for efficiently retrieving answer structures for the user's questions.

At the close of the chapter, we have presented user interface issues, especially how answers are generated from the linguistic structures retrieved from the database so that they very naturally fit the user's question, both in terms of answer congruency and also with regard to forming a coherent discourse through the use of anaphoric expressions.

Chapter 7

Evaluation

In this chapter, we describe the evaluation that we carried out to assess the performance of the SQUIGGLI system and its most important components.

In 7.1, we give an overview of evaluation of natural language processing systems in general and of QA systems in particular. Evaluation of QA systems is currently almost exclusively being done in the context of competitions like TREC and CLEF. Beyond this, there are currently no worked-out frameworks for evaluating QA systems, even though there are suggestions that more user-oriented types of evaluation should be employed.

In 7.2, we describe an end-to-end evaluation of the whole system. We have used a corpus of German questions and answers, collected in the SmartWeb AnswerGatherer experiment, where Internet users were asked to enter questions pertaining to articles in the German version of the Wiki-based on-line encyclopedia Wikipedia. Besides standard measures such as accuracy and NIL recall and precision, we also report a number of more detailed measures, such as answer recall and precision. This is complemented by an evaluation of some of the user interface features that we have implemented, namely anaphora resolution in questions, answer justification and uncertain answers and warnings.

This end-to-end performance evaluation is complemented by a diagnostic evaluation. We first evaluate which resources were actually made use of for correctly answering questions. We then report the results of an error analysis, in which we have investigated the reasons for recall errors (the system did not find an answer contained in the document collection) and precision errors (the system gave a wrong answer).

In 7.3, we report the set-up and the results of a component evaluation: Both the PREDS parser and the anaphora resolution module were evaluated against manually annotated ‘gold standard’ corpora. The section is concluded by an evaluation of parse times.

7.1 Evaluating Question Answering Systems

We will give a short overview of evaluation of Natural Language Processing (NLP) systems in general and QA systems in particular.

We will then describe QA competitions, like the TREC QA track, as the currently most important type of evaluation for QA systems (cf. also Voorhees, 2006).

7.1.1 Evaluating Natural Language Processing Systems

With the increasing availability and maturity of natural language processing (NLP) systems, evaluation plays a more and more important rôle (EAGLES, 1999b; Hirschman and Thompson, 1997; Sparck Jones and Galliers, 1996; EAGLES, 1995). This work mostly refines and extends general methods for software evaluation for natural language processing systems.

The suggested way to go about the evaluation of an NLP system is to use a top-down approach and, starting from a definition of the goals of the evaluation, to break the evaluation down to arrive at useful and – given the goal of the evaluation – meaningful measures that can be established through valid and reliable test procedures (cf. Sparck Jones and Galliers, 1996; EAGLES, 1999a, 193–196).

One premise of this approach is that the goal of the evaluation and the so-called setup (i. e., the context in which the system is run) are carefully determined (Sparck Jones and Galliers, 1996, 11–20).

Designing user-oriented evaluations has proven to involve large efforts in time and labour; these efforts have to be repeated for every new scenario and type of evaluated system. There has been work at specifying and (partly) standardising evaluation procedures for different types of NLP systems (e. g., for proofing tools and translator’s aids, EAGLES 1999b, 1995, and machine translation systems, Hovy et al. 2002b; King 2004).

Currently, no comparable worked-out evaluation framework exists for QA systems. Some of the special challenges of evaluating systems that work on large document collections are described in King (2004); some ideas on usability evaluation of QA systems are described in Fliedner (2004b).

Evaluation of QA systems is currently almost exclusively being done in competitions like TREC and CLEF (2.2.1; cf. also Voorhees, 2006).

7.1.2 Question Answering Competitions

The different QA competitions (2.2.1), namely the TREC QA track (Voorhees and Dang, 2006; Voorhees, 2001), the QA@CLEF (Magnini et al., 2006) and the NTCIR QA challenge (Kato et al., 2005), share the following common features:

Open-Domain QA. In the competitions, open-domain QA systems are evaluated: Questions used as system input may be from any domain (somewhat limited, however, by the use of newspaper archives as the source for the document collections, see below).

Focus on Factoid Questions. Questions are currently mostly factoid questions, i. e., the expected answer must consist of a single phrase (see 2.2.1.2 above)[1]. Systems are not expected to return complex answers that would be appropriate for the open question interpretation (cf. 3.2.2.5).

Newspaper Texts as Corpus. The competitions use large corpora of newspaper or newswire texts as corpus. The corpora have remained the same in the yearly instalments of the competitions for some time.

NIL Answers. As an additional challenge, NIL answers have been introduced by the competitions: For some questions, no answer is contained in the document collection. To answer such a question correctly, systems must return 'NIL' as an answer to indicate that they believe that no answer is present.

Post-hoc Manual Scoring. Answers returned by the evaluated systems often differ in their surface form more or less from 'gold' answers. In addition, sometimes more than one correct answer can be found in the text corpora. Therefore, answers from all systems must be manually scored after the competition.[2]

Scoring. Following the TREC-standard, individual answers are labelled as right, wrong, inexact (answer contains irrelevant extra material) or unsupported (answer is correct, but is not contained in the document with the returned document id). From these assessments, the evaluation scores are computed. The competitions are mostly interested in what is generally

[1]The (marked) list questions used in TREC may be seen as a sub-type of factoid questions in so far as they only differ from these in the requested number of answers.

[2]In Breck et al. (2000), a system is described that can be used for the automatic scoring of QA systems in the TREC setting. However, as it is based on manually written answer keys expressed as regular expressions, its major use is for regression testing of systems based on past competitions.

called accuracy for factoid questions. Accuracy is defined as the proportion of all questions that the system correctly answers of all questions.

For NIL answers, recall (i. e., for which ratio of questions that did not have an answer in the document collection did the system actually return 'NIL') and precision (i. e., what was the ratio of correct NIL answers to all NIL answers that a system returned) are reported.

While the TREC QA task is purely mono-lingual (questions, document collections and answers are all in English), its European and Japanese counterparts QA@CLEF and NTCIR provide both mono-lingual and cross-lingual tasks: The QA@CLEF competition has both mono-lingual and cross-lingual runs with currently nine different source and seven different European target languages, allowing most combinations of source and target languages, resulting in both mono-lingual and cross-lingual tasks. The NTCIR has the Japanese mono-lingual QA challenge and the cross-lingual QA task with Japanese, Chinese and English as source and target languages (Sasaki et al., 2005).

The different competitions are now well-established. Every year, they attract several dozens of participating groups (Voorhees and Dang, 2006; Magnini et al., 2006; Kato et al., 2005). Results are somewhat difficult to compare, as both the task difficulty and the maturity of the participating systems differ (with systems for English typically having had most development effort spent on them). Results in the most recent competitions show that the top-performing systems for English now reach accuracy scores of between 60 % and 80 %. It should be noted, however, that this top group is very small with only three systems, with the next best performing systems reaching around 30 % to 40 % accuracy (Voorhees and Dang, 2006; Voorhees, 2005). An accuracy of 30 % to 40 % is also a typical accuracy level for the best-performing systems for most languages other than English (Magnini et al., 2006; Kato et al., 2005).

7.1.3 Discussion

The current state of the art in evaluating QA systems, viz. mainly through QA competitions, has repeatedly been challenged. Alternative or at least complementary, more user-centred types of evaluation have been called for (De Boni, 2004; Hirschman and Gaizauskas, 2001; Breck et al., 2000). So far, however, no serious, worked-out alternatives have been presented, at least not to our knowledge.

The focus on accuracy as central measure in the competitions has been contested and additional criteria have been suggested, as a possible first step towards more user-oriented evaluation. We repeat one list of possible criteria for assessing answers here:

- Relevance: the answer should be a response to the question.
- Correctness: the answer should be factually correct.
- Conciseness: the answer should not contain extraneous or irrelevant information.
- Completeness: the answer should be complete, i.e. a partial answer should not get full credit.
- Coherence: an answer should be coherent, so that the questioner can read it easily.
- Justification: the answer should be supplied with sufficient context to allow a reader to determine why this was chosen as an answer to the question.

Hirschman and Gaizauskas (2001, 294)

Note that it is far from clear how these different criteria (especially the more 'subjective' ones like conciseness and coherence of answers) could be translated into valid and reliable quantitative measures. So, simply extending the current evaluations with a number of additional measures does not seem a viable option. This is discussed in some detail in Breck et al. (2000). At least some criteria can probably only be measured using qualitative methods, e. g., by user interviews in a scenario test.

There seem to be few concrete results beyond the statement that users need to be taken into account and the tentative suggestion of additional evaluation criteria. We believe that this is – at least partly – due quite simply to the lack of typical, real scenarios for QA systems in use.

In our end-to-end evaluation (7.2), we will establish and report a number of additional measures (such as answer recall and precision and the proportion of questions that the system did not answer at all).

We will also report additional measures that are concerned with user interaction, such as search times, usefulness of answer justifications and of warnings in connection with uncertain answers.

We think that such a detailed evaluation gives a clearer picture of the evaluated system and improves the comparability of results.

7.2 End-to-End Evaluation

We break the end-to-end evaluation down into two parts, namely a performance evaluation and a diagnostic evaluation. The former evaluates the system from the viewpoint of a potential system user and provides insights into the usefulness of the system as it currently stands (conducted as a black-box performance evaluation, Hirschman and Thompson, 1997, 410). The latter takes a different view and looks at the system from the perspective of a computational linguist. The results from the performance evaluation are further analysed to find out more systematically, on the one hand, how positive results could be achieved and, on the other hand, what lead to negative results (glass-box diagnostic evaluation, Hirschman and Thompson, 1997, 410).

7.2.1 Performance Evaluation

We carried out a performance evaluation of the SQUIGGLI system using a test set of 854 question-answer pairs collected in the SmartWeb project based on German Wikipedia articles as a starting point. We will first describe the evaluation design and the measures that we report, then turn to a description of the test data, that is, the SmartWeb AnswerGatherer corpus, manual clean-up on the data, the collection of a suitable document collection, the test procedure itself and then describe the results of the evaluation in detail.

7.2.1.1 Evaluation Design

Setup. We will use a fairly general setup, determined mainly by the availability of suitable test data. We have chosen to use test data, namely question-answer pairs, that were collected in the AnswerGatherer Internet experiment within the SmartWeb project. Internet users were asked to formulate questions pertaining to articles of the German version of the Wikipedia. The experiment will be described in more detail below (7.2.1.2).

We assumed the following setup: The SQUIGGLI system is used as an interactive interface that allows a user to access information contained in the German version of the on-line encyclopedia Wikipedia. The user can enter questions in German to retrieve factual information. The information will be provided as a short answer. The answer can be accompanied by an extract from the original article that minimally contains the answer (typically one or two sentences), or the full text of the document from which the answer was retrieved.

Using a QA system as an interface instead of Wikipedia's keyword-based search engine can provide advantages especially in three cases: First, the searched information may reside in an article that is not retrieved by the search engine or receives a low rank. Second, the information may not catch the user's eye, that is, it is embedded somewhere in the middle of an article so that by extracting the information the user is saved the effort to peruse the whole document. Third, information access via question and answer would be especially useful when the user cannot use a (large) screen to access the Wikipedia itself. This would be the case, for example, if the user employs a mobile platform such as a Personal Digital Assistant or a mobile phone with Internet access.

Criteria and Measures. For the evaluation setup described above, a number of general evaluation criteria can be derived. We will describe the criteria and relate them to measures that will be used to report evaluation results. Several measures are used in the QA competitions like TREC and CLEF; however, we break a number of these measures down more finely and report additional results.

Accuracy. An important criterion in evaluating a QA system is the proportion of questions that it answers correctly. This is measured by (average) answer accuracy over a test set: It reports the proportion of correctly answered questions of all questions. Accuracy is used as the main measure in QA competitions like TREC and CLEF.

Answer Precision. Another important criterion is which proportion of the questions that the system answers at all it answers correctly. This is reported as answer precision. Note that answer precision is bounded by accuracy, but answer precision does not count *'I don't know'* answers (NIL answers). If the system returns an answer to all questions, then answer precision and accuracy will be identical. However, answer precision is useful as it tells the user how reliable the system's answers typically are.

Answer Recall. How good the system actually is at finding answers that are contained in its document collection can be measured as answer recall. Answer recall is the proportion of correctly answered questions of all questions that are known to have an answer in the document collection. As always with evaluation of Information Access systems, this figure is somewhat difficult to define: In general, it will not be possible to detect, with certainty, whether or not an answer to a given question can be found within a large document collection. As, however, the questions in the AnswerGatherer test set were constructed from a given document collection, it is possible to say for which questions at least one answer exists in the document collection (see also below). Note that answer recall is also bounded by accuracy: A perfect accuracy of 100 % also means that the system achieves an answer recall of 100 %.

NIL Precision and Recall. We measure precision and recall for NIL questions as described above (7.1.2). Note that we differentiate between NIL questions and non-NIL questions when reporting all measures; this is generally not done for shared task evaluations, thus especially lumping correct NIL answers with other correct answers for determining accuracy.

Answer Ranking. The method and current instantiation of the answer ranking method is evaluated by assessing how often the results were not optimal: We list in how many cases correct answers are found, but ranked lower than at first position.

Justification. Users can request the system to display sentences from the original text document to support an answer. We report an evaluation of the justifications returned by the system together with its answers, separated for correct and wrong answers. We report the proportion of justifications that are correct and useful (i. e., the user can see with one glance whether the answer is correct or wrong). We also report whether the system displayed any sentences that are not relevant for this decision. These measures show whether the system actually provides a useful justification, i. e., one that is sufficient on the one hand, but as short as possible, on the other hand.

Anaphora and Ellipses in Questions. Questions in the SmartWeb AnswerGatherer corpus contain inter-sentential anaphoric references and ellipses. We report the proportion of correctly resolved anaphora and ellipses.

Uncertain Answers and Warnings. We aimed at improving recall by returning answers that are uncertain, but likely to be relevant. We have evaluated the answers that were marked as uncertain by the system for how helpful the warning actually was.

Answering Time. Answering time is hardly ever reported explicitly for QA systems. However, it forms an important criterion for the evaluation of a system for actual use. Especially if the system is to be run interactively, it is crucial that the system provides answers to questions quickly. We report minimum, maximum and average response times.

Note, that most of these measures report the average performance of the system over a given test set. It should be kept in mind that these results are only as representative for the system performance as the test set is for a certain user's information needs. The AnswerGatherer collection consists of questions stemming from about 100 different users (see below), so that it can be taken to represent a useful average of user interests and ways of formulating questions in the given setup.

7.2.1.2 Test Data

In this section, we take a closer look at the test data (question-answer pairs) used in the system evaluation. We first describe how the data was collected in the SmartWeb AnswerGatherer project. We then summarise post-processing steps and give a number of statistics about the data. Finally, we explain how, starting from the question-answer pairs, we collected a suitable document collection holding correct answers and distractors.

Data Collection. This evaluation was based on test data gathered by the AnswerGatherer experiment in the SmartWeb project at Saarland University (Cramer et al., 2006).[3] The authors conducted an Internet experiment to collect pairs of German questions and answers for texts in the German edition of the on-line encyclopedia Wikipedia (`http://de.wikipedia.org`).

A web site was set up that presents randomly chosen documents from the German version of the Wikipedia collection and allows the users to enter up to three questions and mark portions of the text containing the answer to the question. The interface forced users to mark an answer in the text using the mouse before they could enter a question in order to ensure that questions would actually have an answer and to prevent abuse of the site.

Users were invited to participate via personal email and announcements on a number of appropriate mailing lists. Approximately 100 persons took part in the experiment, entering about 1 400 question-answer pairs. The number of unusable question-answer pairs (malicious ‘e-vandalism’ and ungrammatical questions obviously entered by non-native speakers) are reported to be comparatively low.

The authors compared the collected data with other, similar collections of German questions and answers (namely a collection of questions and answers done within the project by domain experts and with the data from the German monolingual track of the QA@CLEF 2004 competition, Magnini et al., 2004). They found the data collected on the Internet to be similar in a number of respects.

One main aim of the experiment was to show that it is possible to collect question-answer pairs for testing, tuning and evaluating QA systems fairly quickly and at low cost through the Internet. This is important, as data collection for QA ‘manually’ is very time-consuming and costly (cf., e. g., Magnini et al., 2003; Voorhees, 2001).

[3]We would like to thank Jochen Leidner, Irene Cramer and Dietrich Klakow, Signal Processing Unit, Saarland University, who made the data available to us.

Question Type	Number	Percentage
Answer in document	595	69.7 %
No answer in document	66	7.7 %
Answer in table or similar	134	15.7 %
‘Summary style’ question and answer	59	6.9 %
Σ	854	100.0 %

Table 7.1: Question-Answer Pairs in the SmartWeb AnswerGatherer Corpus

Data Preparation. We manually removed some examples of e-vandalism and empty questions from the raw data. The manually cleaned-up collection consisted of 854 question-answer pairs. Table 7.3 shows some examples for questions.

In a further clean-up step, we corrected questions for spelling errors and similar problems (encoding problems, especially for German umlauts).

We manually annotated all question-answer pairs to show whether the given question was actually answered by the answer string marked by the user. A number of cases were marked (cf. tab. 7.1).

We distinguished three cases in addition to ‘normal’ question-answer pairs (i. e., an answer for the question was actually contained in the portion of the Wikipedia article marked by the user). First, there were a number of question-answer pairs where the question was not answered within the document. This was the case for 66 pairs (7.7 %). Here is an example case:

(7.1)

Wie	*weit*	*liegt*	*Alsleben*	*von*	*der*	*Mündung*	*der*	*Elbe*	*entfernt?*
How	far	lies	Alsleben	from	the	mouth	of the	Elbe	removed?

How far away from the mouth of the Elbe river is the city of Alsleben located?

Die	*Stadt*	*[Alsleben]*	*liegt*	*am*	*Westufer*	*der*	*unteren*
The	city	[of Alsleben]	lies	at the	western bank	of the	lower

Saale,	*ca.*	*30 km*	*vor*	*deren*	*Mündung*	*in*	*die*	*Elbe.*
Saale,	approx.	30 km	before	its	confluency	into	the	Elbe.

The city [of Alsleben] sits on the western bank of the lower Saale river, about 30 km from its confluency with the Elbe river.

The supposed answer sentence does not contain any information about the mouth[4] of the Elbe river at all. Presumably, the user who entered the question did not read the text very attentively.

We also annotated a second class of pairs, where the answer was not given in a textual form, but rather implicitly, especially in the form of tables. For example, many Wikipedia articles about people contain a table that gives a quick summary of the biographic facts. A number of questions asked for information that was only given in tabular form or epentheses. Of the 854 question-answer pairs, 134 (15.7 %) were considered to fall into this category.

Finally, a small group of 59 pairs (6.9 %) were marked as 'summarisation style' and thus similar to open questions (cf. 3.2.2.5). Here is example.

(7.2) *Wie wird Cidre hergestellt?*
How is Cidre produced?

Für die Herstellung werden Apfelsorten mit hohem Tanningehalt
For the production are apple sorts with high tannin content
verwendet. Die Fermentierung findet bei relativ niedrigen
used. The fermentation takes at relatively low
Temperaturen von 4 °C – 15 °C statt, was einen wesentlichen
temperatures of 4 °C to 15 °C place, which a decisive
Einfluss auf die Dauer der Fermentierung und somit auf das
influence on the duration of fermentation and thus on the
Aroma hat. Kurz bevor der Zucker vollständig durch die Hefen
taste has. Shortly before the sugar fully by the yeasts
umgesetzt (fermentiert) wurde, wird der Cidre in neue Fässer
transformed (fermented) is, is the cider in new barrels
umgefüllt. Die meisten Hefen und Schwebstoffe verbleiben
filled. The most yeasts and suspended matters remain
im alten Fass. Das neue Fass wird ohne Lufteinschluss
in the old barrel. The new barrel is without inclusion of air
gefüllt und dicht verschlossen. Durch Fermentation des
filled and tightly sealed. Through fermentation of the
restlichen Zuckers entsteht dann die Kohlensäure und macht
remaining sugar is produced then the carbonic acid and makes

[4]The German word *'Mündung'* can stand for both a river mouth (i. e., a river flowing into the sea) and a confluency (i. e., two rivers joining).

den	*Cidre*	*somit*	*auch*	*haltbar.*	*Durch*	*Zugabe*	*von*	*Zucker*	*kann*	*der*
the	cider	thus	also	keepable.	By	addition	of	sugar	can	the

Alkoholgehalt	*noch*	*gesteigert*	*werden.*
alcohol content	further	increased	be.

For [its] production, varieties of apples with a high content of tannins are used. Fermentation takes place at relatively low temperatures of 4–15 °C. This decisively influences the duration of the fermentation and thus the taste. Shortly before the sugar has been fully transformed (fermented) by the yeasts, the cider is transferred into new barrels. Most yeasts and suspended matters remain in the old barrel. The new barrel is filled without inclusion of air and tightly sealed. Through the fermentation of the remaining sugar, carbonic acid is produced, which helps to make the cider keep well. By adding sugar, the alcohol content can be further increased.

Note that this type of question-answer pair can only be distinguished by looking at both question and answer: In most cases, the questions might be answered by a single sentence. The example question might be answered by a sentence like *‘Cider is produced by fermenting apples.’* However, in the text above, the core facts reside in different sentences; thus, the text would need to be summarised – or displayed as a whole.

All question-answer pairs were used for evaluation. However, we considered the special cases as NIL questions. We think that questions of these types actually provide a very natural source of NIL questions, especially the ‘near-misses’, where a lot of word overlap exists between question and answer as in the example above. Providing suitable NIL questions poses a considerable challenge for data collection (cf., e. g., Magnini et al., 2004). Note that the proportion of NIL questions (30 %) is considerably higher than used for the TREC and CLEF competitions, where it has constantly been 10 % of the questions for the last years.

Tables 7.2 and 7.3 show the distribution of question words (question constructions) and question/answer types for the corpus, respectively. The former indicates the syntactic form of the question; the latter gives a more fine-grained grouping of the questions with respect to the expected answer.

Distractors. The documents, from which the users constructed questions during the Internet experiment, formed the core of the document collection that we used for evaluation. The 854 questions in the corpus pertained to 383 different Wikipedia articles, bringing the average number of questions per article to 2.2.

	All		Non-NIL	
Question Word/Question Construction	#	%	#	%
‘was’ (*what*)	230	26.9 %	183	30.8 %
‘wer’ (*who*)	129	15.1 %	105	17.6 %
‘wo’ (*where*)	107	12.5 %	72	12.6 %
‘wann’ (*when*)	106	12.4 %	67	11.2 %
‘welch’ (*which*)	104	12.2 %	57	9.6 %
‘wie’ (*how*)	69	8.1 %	39	6.5 %
‘wie’+Adj (*how*+Adj)	30	3.5 %	21	3.5 %
Yes/No	27	3.2 %	18	3.0 %
‘wieviel’/*‘wie viele’* (*how much/many*)	27	3.2 %	18	3.0 %
‘woher’ (*whence*)	11	1.3 %	8	1.3 %
‘warum’/*‘weshalb’* (*why*)	10	1.1 %	5	0.8 %
Other	4	0.5 %	2	0.3 %
Σ	854	100.0 %	595	100.0 %

Table 7.2: SmartWeb AnswerGatherer Corpus: Distribution of Question Words and Question Constructions

To this core set, we added a larger set of distractors, i. e., documents that contained related information, to test the system’s answer precision. We extracted distractors from the Wikipedia itself in the following way: We used the question topics of the questions in the collection as search terms for the search facility integrated into the Wikipedia itself to find related documents. This was done automatically, and the top-ranking documents returned by the search were then automatically downloaded.

The Wikipedia web-site uses the open source search engine Lucene (`http://lucene.apache.org/`, `http://en.wikipedia.org/wiki/Lucene`). The resulting set of relevant documents was thus similar to those used by ‘standard’ QA systems that use a search engine to retrieve documents from the overall document collection first and conduct post-processing and answer extraction over this limited set (cf. 2.2.2).

In a manual inspection of an example set, we found that this method retrieved interesting distractor documents. For example, for questions about a certain person, the retrieved distractors generally contained a number of documents that mentioned that person (for example, documents describing that person’s discoveries or exploits or documents describing famous spouses or

	All		Non-NIL	
Question/Answer-Type	#	%	#	%
Definition (*What is Acarbose?*)	137	16.0	122	20.6
Place (*Where is Wapno located?*)	98	11.5	77	12.9
Date (*When did Cole Porter die?*)	94	11.0	58	9.7
Name (*What is breaking in Taekwondo called?*)	75	8.8	43	7.2
Person (*How was James Rennell's wife called?*)	59	6.9	37	6.2
Person Definition (*Who is Jim Kerr?*)	59	6.9	58	9.7
Complex (*How was the Kithara played?*)	44	5.2	3	0.5
Other (*What is Joseph LeDoux's area of research?*)	39	4.6	34	5.7
Measure (*How big is a square mile?*)	32	3.7	19	3.2
Yes/No (*Do louse flies have eyes?*)	25	2.9	17	2.9
Number (*How many inhabitants does Ancona have?*)	24	2.8	15	2.5
Reason (*Why was 'The Trouble with Harry' unavailable for a long time?*)	17	2.0	10	1.7
Special (*What is special about the Bresenham algorithm?*)	16	1.9	12	2.0
Purpose (*What are folding marks used for?*)	16	1.9	10	1.7
Abbreviation Def. (*What does WGPSN mean?*)	15	1.8	8	1.3
How (*How does an elementary analysis work?*)	14	1.6	8	1.3
Etymology (*Where does the name 'mapalé' come from?*)	10	1.2	7	1.2
Material (*What is a bivouac box made of?*)	9	1.1	8	1.3
WWW (*Where is homepage of Verchen?*)	8	0.9	0	0.0
Time-start (*Since when are axioms used?*)	8	0.9	6	1.0
Period (*How long was Félix Malloum exiled?*)	8	0.9	7	1.2
Vocation (*What was Manfred Bochmann's vocation?*)	7	0.8	5	0.8
Circumst. (*When can a marriage be annulled?*)	6	0.7	3	0.5
Appearance (*What do Jersey cows look like?*)	5	0.6	4	0.7
Currency (*How much did the development of EE-T2 Osório cost?*)	5	0.6	4	0.7
Source (*Where does BZN come from?*)	4	0.5	3	0.5
Organisation (*With which team was Roland Meier most successful?*)	4	0.5	2	0.3
Ratio (*What is the proportion of premature births?*)	3	0.4	3	0.5
Document (*Where can regulations on classified files be found?*)	3	0.4	3	0.5
Assessment (*How did audiences like 'Wonder Boys'?*)	3	0.4	2	0.3
Abbreviation (*How is Amateur Athletic Union abbreviated?*)	3	0.4	3	0.5
Employer (*Where does Oliver Tolmein work?*)	2	0.2	2	0.3
Alternative (*Is Niobe a male or female first name?*)	2	0.2	1	0.2
Σ	854	100.0	595	100.0

Table 7.3: SmartWeb AnswerGatherer Corpus: Distribution of Question-Answer-Type

co-workers), but also similar-named persons (for example, for questions about Francis Bacon, the painter, documents about the philosopher of the same name, but also about the philosopher Roger Bacon and the US actor Kevin Bacon were returned). We thus found the type of returned documents to be similar to those that are found by IR engines when processing questions on document collections used in shared tasks such as TREC or CLEF.

For each question, up to 20 distractors were retrieved. We then complemented the collection with different, unrelated texts, bringing the total number of documents in the collection to about 5 000. These were all processed off-line with our whole linguistic processing chain (chapter 6) and stored in the database (5.4). Processing all documents took about a week on a single machine.

7.2.1.3 Test Procedure

After storing structured representations for all documents in the database, all questions in the collection were sent to the SQUIGGLI system in batch mode; care was taken that the questions remained in the original order in which they were entered. This was especially important, as the question collection contained questions using anaphoric references to entities mentioned earlier and elliptical questions (cf. 7.2.1.4). We ran the system in constituent answer mode, that is, for *wh*-questions, the system would display exactly the constituent matching the *wh* phrase (cf. 6.4.3.2).

The system's answers were recorded and evaluated against the gold standard answers from the question-answer pair collection. In competitions like TREC and CLEF, it has turned out that a fully automated comparison of answers returned by a QA system with a set of 'gold standard' answers is not possible (Magnini et al., 2004; Voorhees, 2001). Automated evaluation methods using answer patterns and word overlap have been suggested (e. g., Breck et al., 2000), but are difficult to set up and fail for even slightly complex answers. This is, on the one hand, due to the fact that gold standard answers and answers returned by systems may differ in granularity. On the other hand, systems will often return different answers that can be found in the document collection and should thus also be acknowledged as correct. So, in the shared task evaluations answers are manually annotated for correctness.

We found that only about 10 % of the answers returned by the system were string-identical with the answers in the collection of question-answer pairs. Upon inspection, we found the answers in the collection, which were marked by the users in the original Wikipedia article, to differ strongly with regard to granularity: While some users consistently marked sentences containing the answer, others marked only constituent answers or sometimes only bare nouns instead of PPs.

	Correct		Wrong		IneX.		Unsp.		Missing		Σ
Group	#	%	#	%	#	%	#	%	#	%	
Non-NIL	200	33.6	81	13.6	0	0.0	1	0.2	313	52.6	595
NIL	240	92.7	19	7.3	0	0.0	0	0.0	0	0.0	259
Σ	440	51.5	100	11.7	0	0.0	1	0.1	313	36.7	854

Table 7.4: Results for All Questions, First Answer

Therefore, answers were manually annotated for correctness. We employed the system used in TREC and CLEF, marking each answer as Right, Wrong, Inexact or Unsupported.

7.2.1.4 Results

The basic results are summarised in tab. 7.4. For the measures in this table, as for most others in this section, only the highest-ranked answer that the system returns is considered (but cf. the evaluation of answer ranking below).

In tab. 7.4, we give results for non-NIL questions, for NIL questions and combined results. We consider this separation important for a better understanding of what the user can expect from the system. However, these detailed figures are hardly ever reported in evaluations of QA systems.

Accuracy. From tab. 7.4, the system accuracy (i. e., the ratio of correctly answered questions) can be directly read off: It is 33.6 % for non-NIL questions only, and 51.5 % for combined NIL and non-NIL questions. Note that for the combined score (the one that is generally reported as the system accuracy score for shared competitions), both correctly answering a question and outputting NIL for a NIL question is counted as a correct answer. As we consider the two tasks to be different in nature, we also report the ‘split’ scores.

What happens with the rest of the questions? Table 7.4 shows that the number of wrong answers is relatively small: Only for 14 % of non-NIL questions and 7 % of NIL questions (about 12 % combined) the system returned an answer that is actually wrong.

For the rest of the questions, the system does not return an answer (almost 37 % of all question-answer pairs).

Note that we did not observe any inexact and only a single unsupported answer in the evaluation. This is as expected, since using linguistic structures

Recall	Precision	F-Score
200 / 595 = 33.6 %	200 / 301 = 66.4 %	44.6

Table 7.5: Non-NIL Question Answer Recall, Precision and F-Score, First Answer

for matching and for answer generation makes the occurrence of both kinds of wrong answers more unlikely: As *wh*-questions are answered by generating answers based on whole constituents established during parsing, the system is far less likely to display unwanted material than, say, a system based on pattern matching only. Unsupported answers (that is, correct answer that cannot be inferred from the document in question) are also unlikely, because question and answer structure must fully match, making a chance match unlikely.

Note also that the accuracy score should be seen in connection with the low answer redundancy of the corpus. Marc Light and his colleagues have shown that QA systems typically perform better if the answer to a question is contained in the document collection several times and in several formulations (Light et al., 2001). For TREC 8 (2000), they established a mean number of answers of 7.04 answers per question in the document collection. Answer redundancy is also methodically exploited by QA systems that use ‘answer projection’, i. e., that search for answers in external sources and ‘project’ them onto the given document collection (2.2.2.3). In contrast, the corpus that we used for evaluation contained little redundancy (manual inspection for a handful of questions showed typically one single correct answer per question), indicating that mean accuracy may be higher on larger document collections. However, as the number of answers in the corpus was not systematically established, this assumption would have to be separately verified.

Answer Recall and Precision. Table 7.5 shows both recall, precision and F-Score for non-NIL questions. Question recall gives the proportion of questions that have (at least) one answer for which a correct answer is returned (and ranked highest). Note that this figure is the same as the accuracy when counting only non-NIL questions: 33.6 %.

The answer precision measures the proportion of correct (first) answers of all answers that the system gives. We found 200 correctly answered question out of 301 questions, for which the system returned some answer (namely 200

$$F\text{-}score_{\beta} = \frac{(1+\beta) \times Recall \times Precision}{\beta \times Precision + Recall}$$

Figure 7.1: F-Score with Weighting Factor β

NIL Recall	NIL Precision	NIL F-Score
240 / 259 = 92.7 %	240 / 553 = 43.4 %	59.1

Table 7.6: NIL Recall, Precision and F-Score

correct non-NIL, 81 wrong non-NIL, 1 unsupported non-NIL and 19 wrong NIL questions, cf. tab. 7.4).

Answer precision is an important figure for users (however, it is not reported in the QA competitions). In our case, it means that if the system gives an answer *at all*, this answer will be correct in almost two thirds (66.4 %) of all cases. Note that this figure also takes the spurious answers to NIL questions into account; for non-NIL questions only, precision would be 70.1 %.

The F-score combines recall and precision (cf. 7.1). We use $\beta = 1$ for computing all F-scores, i. e., we assume the same weight for both recall and precision. The F-score can be used to gauge the answer performance when the user considers both answer recall and answer precision as equally important.

NIL Recall and Precision. In table 7.6, the corresponding values are given for NIL questions: The recall, i. e., the proportion of questions for which the system correctly recognises that no answer is present in the document collection, is quite high at almost 93 %. The precision, that is, the proportion of NIL answers that is actually correct, as there is no answer to be found in the document collection, is about 43 %. We also give the NIL F-score, which is 59.1.

Position of 1st Correct Answer	1	2	3	4	≥ 5	None
Number of Questions	200	18	8	3	4	68
Per Cent	66.4	6.0	2.7	1.0	1.3	22.6
Per Cent of non-NIL	85.8	7.7	3.4	1.3	1.7	–

Table 7.7: Evaluating Answer Ranking: Position of First Correct Answer

	Mean Reciprocal Rank
All	0.530
Only Non-NIL	0.358

Table 7.8: Mean Reciprocal Rank

Answer Ranking. We evaluated the answer ranking as it is currently implemented and instantiated (cf. 5.2) in the following way: Instead of considering only the first answer that the system returns, we also marked, for all questions for which a wrong first answer was returned, at what rank the first correct answer (if any) was given. The results are summarised in table 7.7.

In addition to the 200 correct answers to non-NIL questions at first rank, a correct answer is found by the system for another 33 questions, leaving 68 questions for which no correct answer is found at all. As tab. 7.7 shows, the answer ranking function as it is currently employed works quite well. If a correct answer is found at all, it is ranked highest in 200/233=85.8 % of all cases.

If a user is ready to consider the first three answers and look at the justification, which the system can provide upon request and which allows to quickly gauge the correctness of an answer (see the next section), they can discover additional correct answers. 'Lenient answer recall' (i. e., correct answer within the first three) is 38.0 % and 'lenient answer precision' (i. e., the user cooperatively selects only correct answers, if available) 75.1 %.

The Mean Reciprocal Rank (MRR) has been defined for the first QA track at TREC 8 (Voorhees, 2000). It is used to evaluate the quality of the ranking function of a QA system if more than one answer is scored and is computed as follows: For each question, the reciprocal rank is either the reciprocal of the rank for which the first correct answer is returned or zero if no correct answer is returned at all (corresponding to an infinite rank). For NIL questions, either 1 (NIL answer) or 0 (spurious answer) is used. These scores are then averaged for all questions.

A MRR of 1 would mean that a system correctly answers all questions and that the correct answers always receive first rank. If all questions are answered correctly, but the correct answers are consistently on second rank, the MRR would become 0.5. The same would hold for a system that returns perfect answers (correct and ranked highest) for half of the questions and NIL for the other half. It is thus somewhat similar to the accuracy, but additionally evaluates the ranking function of the system.

The MRR (both including and excluding NIL questions) is given in tab. 7.8.

Justification. A user-friendly QA system should not only give answers to the users' questions but also be able to back up these answers with justification. The simplest way to do this is by providing a reference to the original document. This kind of justification is generally used in shared competitions like TREC and CLEF. For a user, however, this means that they have to scan the whole document themselves to find the sentence or passage that allows them to ascertain that the system's answer is correct.

We have implemented a more focussed justification mechanism that displays – upon request – exactly those sentences from the original text that the system used in finding and generating its answer. Alternatively, the system can also show the whole document, but with the sentences it deems relevant highlighted. This has been described in 6.4.6.

We were interested to see how good the system actually is at pinpointing exactly those sentences in the original text that allow the user to gauge whether the answer can be inferred from the original text: It should display as few sentences as possible, so that the user can quickly scan the justification.

This task is complex, as the extracted sentences should be self-contained in the following way: If a sentence contains an anaphoric references that has to do with the question topic itself, the justification should include a sentence that contains a uniquely referring expression identifying the entity in question, so that the user can correctly resolve the reference. Especially, no 'dangling' pronouns should be present in the justification, unless they are not important for gauging the correctness of the answer. With the information from the matching process combined with information from the coreference resolution module, we can identify the contributing references and, ultimately, the sentences that need to be displayed.

The following example shows (the English translation of) a sample output of an answer and a justification. Note that the two sentences given as justification are not adjacent in the original text, rather, the system selected them as their representations contributed to finding the answer and as they are thus likely to provide a self-contained justification.

(7.3) For what was Edna Ferber awarded the Pulitzer Prize?
For the novel 'So Big'
Edna Ferber was an American writer of Hungarian descent. [...] She was awarded the Pulitzer prize in 1924 for her novel 'So Big'.

To evaluate answer justification, we looked at correct and incorrect answers separately. For correct answers, we annotated whether the answer could truly

Assessment	Yes, OK				Yes, Inexact			No	Σ
# Sents.	1	2	3	Σ	2	3	Σ		
#	152	32	1	185	7	1	8	7	200
%	76.0	16.0	0.5	92.5	3.5	0.5	4.0	3.5	100.0

Table 7.9: Justification: Can Correct Answers be Recognised by Looking at Justification Sentence(s) Alone?

Assessment	Correct Answer	Certain	Uncertain	Wrong	Σ
#	42	55	3	1	101
%	41.6	54.5	3.0	1.0	100.0

Table 7.10: Justification: Can Wrong Answers be Disregarded when Looking at Justification Sentence(s) Alone?

be justified from the displayed sentences and whether the justification contained additional sentences that were not needed for the justification (rendering it correct, but inexact). Table 7.9 shows that justification works very well for correct answers: Only for 7 of the 200 correct answers (3.5 %), the selected sentences would not allow the user to evaluate the answer. For an additional eight answers (4.0 %), the justification contained a superfluous sentence. When comparing only cases where the system output more than one sentence in justification, we find that for 33/41=80.5 %, the system chose the correct minimal set of sentences.

For the wrong answers (81 wrong answers, 1 unsupported answer, 19 spurious answers for NIL questions), we first found that the justification often (42 of 101 cases, 41.6 %) contained the correct answer. These were generally cases where parsing or coreference resolution had gone wrong, so that the wrong part of a sentence would be returned as an answer (cf. also 7.2.2). Then, for another 55 (54.6 %), the justification immediately showed that the answer was wrong. For three cases, the output sentences alone were not sufficient to safely disregard the answer. And lastly, there was one case where coreference resolution had gone wrong and where the sentences presented as justification would allow the user to (wrongly) conclude that the answer was a correct one. For wrong answers, we did not find any inexact justifications.

These results show that the justification mode that displays exactly those sentences necessary to evaluate the system's answer works well and thus pro-

	Right		Wrong		Not Resolved		
Type	#	%	#	%	#	%	Σ
Pronoun	33	84.6	2	5.1	4	10.3	39
Pronoun (Answer)	0	0.0	1	100.0	0	0.0	1
Possessive Pronoun	6	85.7	1	14.3	0	0.0	7
Definite	10	33.3	2	6.7	18	60.0	30
Named Entity	8	66.7	0	0	4	33.3	12
Pronadv	0	0.0	1	100.0	0	0.0	1
Σ	57	62.2	7	7.8	26	28.9	90

Table 7.11: Inter-sentential Anaphora in Questions

vides an interesting option for user-friendly QA systems. By requesting this justification, the user can very quickly check whether or not the answer returned by the system is correct. We think that this option opens up an interesting new way for increasing the acceptance of QA systems.

We have reported results from a study on user preferences in QA above (cf. Lin et al., 2003, 6.4.6): Results in that study indicate that users prefer paragraph-length output from QA systems. We have suggested above that we consider a combination of short answers and minimal justifications (that is, not paragraph length, but exactly those one or two sentences sufficiently, but minimally justifying the answer) provides an interesting new way of presenting answers in a QA system. The figures reported here indicate that it is indeed possible to set up a QA system in a way that allows displaying just this kind of information. It remains to be seen, however, if users actually find this information useful and whether they would prefer it over paragraph-length answers (cf. 8.3.3)

Anaphora and Ellipses in Questions. Even though the instructions for the participants of the SmartWeb AnswerGatherer data collection did not contain any explicit instructions regarding the use of anaphoric references in questions, users made frequent use of them: In 90 of 469 follow-up questions (19.2 %), some anaphoric reference was employed. This shows that it is very natural for users to use anaphora in follow-up questions.

Table 7.11 shows the distribution of different types of anaphora and how well the system did at resolving them. Note that any such resolution is currently reported to the user, so that errors during anaphora resolution could be corrected by asking a further follow-up question.

The table shows that pronoun resolution worked well (over 80 % correctly resolved). For definites, the largest problem is that 60 % are not resolved. Note that, while pronouns can be simply identified as anaphoric references, this is not straightforward for definites, as they may not be anaphoric references at all.

For named entities, only cases are reported where a short form of the named entity is used (especially ‘leaving out’ of a person’s first name in follow-up questions).

In principle, unresolved definites can lead to overgeneration. For example, a follow-up question about the movie *‘Ich kämpfe um dich’* (English original title: *Spellbound*) reads *‘Upon which novel is the movie based?’*. As the system failed to resolve the definite *the movie*, the question representation is insufficiently constrained and will match any movie. However, the number of precision errors in the evaluation was small, as the questions were generally constrained enough not to match widely (cf. 7.2.2.3).[5]

All in all, the number of unresolved definites in questions is currently still too high for a full dialogue-style interaction. This figure could probably be lowered by using a version of the anaphora resolution algorithm that is further optimised for recall (cf. 7.3.2): For example, matching definites and antecedent candidates without any GermaNet similarity (at least in cases where no other match can be found) would increase recall and – presumably – not hurt precision to much, as the number of possible antecedent candidates is typically smaller for follow-up questions than for longer texts.

Note that only one single question makes reference to the (expected) answer to the previous question: The question *‘Who is the author of “Polyglotta Africana”?’* is followed by *‘Who was he?’*. The user obviously expects the name (Sigismund Wilhelm Koelle) as the answer to the first question and a person definition as answer to the second one.

While anaphora are frequently used in follow-up questions, questions in the SmartWeb AnswerGatherer corpus contain only two cases of elliptical questions. Both are cases where the follow-up question consists of a single *wh*-phrase, interestingly in both cases ‘signalled’ by *‘und’* (*and*). For example, the question *‘When was the University Notre Dame in Indiana founded?’* is followed by the elliptical *‘And by whom?’*. Both cases were correctly resolved by the ellipsis resolution module. However, the overall number is too small to allow any significant conclusions.

The evaluation of the resolution of anaphora and ellipses in questions highlights a number of points. First, it shows that using anaphora comes very natural to people asking questions and should thus be supported by an interactive QA

[5]Note that a number of anaphora resolution errors occurred for NIL questions (true negatives) and thus do not figure in the error analysis in 7.2.2.3.

Answer Warning Type	Corr. Ans. Superfl. #	%	Wrong Answer Corr. W. #	%	Diff. Reason #	%	Σ
Hypernym	14	50.0	9	32.1	5	17.9	28
Wrong Person Name	0	0.0	3	100.0	0	0.0	3
Opaque Context	3	100.0	0	0.0	0	0.0	3
Modal Verbs	1	50.0	1	50.0	0	0.0	2
Σ	18	50.0	13	36.1	5	13.9	36

Table 7.12: Uncertain Answers and Warnings: Was the Answer/Warning Correct?

system. Using structured representations of questions and answers provides a basis, upon which anaphora resolution can be comparatively easily be implemented. Interestingly, both 'answer-anaphora' and ellipses play a far less important rôle than we had originally expected. While this may be partly due to the way that data was collected, it seems to indicate that support for these phenomena may be less important for a dialogue-style QA system. These points bear out the assumptions summarised in (Fliedner, 2006b), namely that coreference and ellipsis resolution can be implemented in the framework of linguistically informed QA and that both are needed for interactive QA.

Uncertain Answers and Warnings. Different types of uncertain answers are returned by the system. We have argued that uncertain answers are often relevant for the question (3.5.1.3). We have described above that by using an answer checking step, we can identify uncertain answers, lower their relevance score accordingly and display them together with a warning message (5.1.5). Remember that in these cases not only a constituent answer, but a whole answering sentence is displayed to make gauging the relevance of the answer easier.

We will now take a look at the question whether this strategy works and how well it does. Table 7.12 summarises the results. The leading questions were, on the one hand, whether recall was improved, on the one hand, whether warnings were correct.

The first column lists correct answers found through uncertain inferences (gain in answer recall). In these cases, the warning was superfluous: Even though the answer contains potentially 'dangerous' material, it is fully correct. This is especially often the case for closely related words that are listed as hyper-

Minimum	Maximum	Average
0.23s	510.19s	16.59s

Table 7.13: Database Search Times for all Questions: Overview

nyms in GermaNet. For example *'liegen'* (*lie*) is a hypernym of *'sein'* (*be*) in GermaNet. Many location questions are phrased like the following one *'Wo liegt Assab?'* (*Where lies Assab?, Where is Assab located?*), leading to a hypernym match and a warning for answers like *'Assab is a city in Eritrea'*.

The second column lists cases in which the warning was correct, that is, the associated answer does not answer the question, even though it provided information that is typically related to the question in some way.

The third column lists a number of cases where the answer is wrong for reasons not directly connected with the warning: Thus, the user is alerted, but for the warning message may not be immediately helpful.

On inspecting the precision errors (wrong or spurious answers, cf. also 7.2.2.3), we found no case of uncertain answers that was not flagged.

Thus, the general method of answer checking and flagging uncertain answers works well. Using uncertain inferences has improved recall, while the user is alerted to potentially wrong answers. When counting wrong answers with a warning not as wrong, but rather as NIL answers (as the user can easily see that they are wrong and disregard them), modified answer precision goes up to $200/282 = 70.9\,\%$.

We think that it would be possible to 'fine-tune' the lists of dangerous inferences, so that the current number of false alarms (almost 50 %) can be reduced. For example, GermaNet hypernyms that are very similar and thus act more like synonyms might be excluded from the lists.

Search Times. In addition to the measures reported so far, users will also be interested in the time it takes the system to return an answer (or a *'I don't know'*). Table 7.13 reports the minimum, maximum and average time used for answer searching.

The timings were taken on a standard laptop PC with a Intel® Mobile Pentium® 4-M processor at 2.2 GHz clock rate with 1 GB of RAM, running under Windows® XP Professional™. We used Cygwin Perl version 5.8.5 and MySQL version 4.0.20.

Time *n*	1s	2s	5s	10s	20s	50s	100s	200s	500s
% Ans.	15.4	39.7	66.1	77.9	87.5	93.1	95.1	98.24	99.9

Table 7.14: Database Search Times: Percentage of Answers Found within *n* Seconds

For the evaluation, we set a timeout for individual database queries of 300 seconds (5 minutes). This was done to prevent queries for which the MySQL database produced sub-optimal query strategies (cf. 6.3.5) to unduly hold up the process. Note that to search an answer, a number of individual queries that is proportional to the number of nodes in the question representation is actually carried out (cf. 6.3.5).

Table 7.14 shows the (cumulated) percentages of answers found within a certain amount of time. Within ten seconds – we would consider this to be about the average time a user would be willing to wait for a response of an interactive QA system – more than 75 % of the questions could be answered.

These results show that the proposed search algorithm for linguistic structures, which is based on unordered path inclusion and makes use of a relational database for storage and retrieval (cf. 5.3, 5.4), actually provides a reasonable performance and is suitable for tackling the task at hand. As the search itself makes use of highly optimised database queries, we expect it to scale up well also for larger amounts of data. We believe that it would be possible to additionally increase search performance by further optimising the data base queries.

7.2.1.5 Conclusions and Comparison

The measures reported so far indicate that the SQUIGGLI system as it stands might be used advantageously for the task at hand: It will correctly answer about a third of the users' questions if an answer can be found in the document collection at all. Answer precision is quite high at 66 %. Answers will be found within an average of 16.6 seconds, with almost 78 % of all answers found within 10 seconds at most.

However, the results also show that using the system in a combined setup with a system aiming at high recall (especially an IR system) would be useful. Both systems could search for an answer independently and answers could then be combined: If the SQUIGGLI system finds an answer at all, this could be presented separately at the top of the list. Then, the results from the other system (or systems) could be presented.

Results from this evaluation are not directly comparable to the results of QA competitions like TREC and CLEF, as the evaluation data (questions and input texts) are quite different. Comparison with past results from TREC and CLEF indicates that differences between questions and the text passages answering them are typically greater for TREC and CLEF than in the SmartWeb Answer-Gatherer corpus. Nevertheless, it is interesting to take a look at results for these standard evaluations.

The most recent QA tracks at the TREC competitions have shown that English open-domain factoid QA systems have reached a very high standard (Voorhees and Dang, 2006). The best-performing systems reach accuracy scores of 66 % and 71 % and typical NIL F-scores of up to 60 %. Note that there is a marked drop then: the third-best performing system reached 'only' 33 % accuracy.

The best-performing monolingual QA systems for German in the recent CLEF competitions still achieve 32 % to 42 % accuracy and NIL F-scores of 25 % to 35 % (Magnini et al., 2006).

At first glance, the results of this evaluation, namely 33.6 % accuracy (51.5 % including NIL questions) and 59.1 NIL F-score (cf. tabs. 7.4 and 7.6) tie well with these scores. However, as the evaluation data is quite different, we do not think they can be directly compared. In addition to the differences in text and question types, for example, comparing accuracy scores directly is not simple, as our data contained about 30 % NIL questions, compared with 10 % in TREC and CLEF. We think that by reporting more detailed evaluation scores, the comparability of results across evaluations could be improved in the future.

7.2.2 Diagnostic Evaluation

In the previous section, we have reported the results of a system performance evaluation. We will now turn to a diagnostic evaluation. We will be interested in where and why the system worked well and where and why it failed. We will first investigate the contributions of different components and resources before turning to an analysis of errors and failures and a more detailed view of the results for different question-answer types.

7.2.2.1 Contribution of Different Components and Resources

We will first investigate which sorts of information played how big a rôle in finding the answers for correctly answered questions. Table 7.15 summarises the results for the 200 correct answers that the system returned in the evaluation.

The table should be read as follows: It shows the sources of linguistic information that we used and lists the number (and percentage) of correctly an-

	Used		Needed	
Resource	#	%	#	%
Question Node Matching	**197**	**98.5 %**	**197**	**98.5 %**
Node Match	188	94.0 %	188	94.0 %
Hyponymy Match (*which*+CN)	6	3.0 %	6	3.0 %
Number Match (*how many*)	3	1.5 %	3	1.5 %
Named Entity Matching	**173**	**86.5 %**	**173**	**86.5 %**
Grammatical Function Matching				
Identity Match	184	92.0 %	130	65.5 %
Uncertain Match	155	77.5 %	26	12.7 %
Additional Semantic Role Matching	**61**	**30.5 %**	**59**	**29.5 %**
Place	34	17.0 %	32	16.0 %
Time	22	11.0 %	22	11.0 %
Source	3	1.5 %	3	1.5 %
Cause	1	0.5 %	1	0.5 %
Time-start	1	0.5 %	1	0.5 %
Frame Matching	**127**	**63.5 %**	**29**	**14.5 %**
Same Frame	127	63.5 %	29	14.5 %
GermaNet Matching	**32**	**16.0 %**	**19**	**9.5 %**
Synonymy	16	8.0 %	15	7.5 %
Hypernymy	16	8.0 %	14	7.0 %
Anaphora Resolution	**47**	**23.5 %**	**47**	**23.5 %**
Pronominal	17	8.5 %	17	8.5 %
Relative	2	1.0 %	2	1.0 %
Definites	16	8 %	16	8 %
NE	14	7.0 %	14	7.0 %
Equivalence/*is* Relation	**18**	**9.0 %**	**18**	**9.0 %**
Apposition	13	6.5 %	13	6.5 %
Predicatives (*be X*)	5	2.5 %	5	2.5 %

Table 7.15: Resources Used for Answering Questions: Results for 200 Correctly Answered Non-NIL Questions

swered questions to which it contributed. As we allow, during answer matching, that structures match at different linguistic levels, we distinguish between cases where the information was *needed* (i. e., without it, no match would have been possible) and where it was *used* (i. e., some other linguistic level matched as well). In the latter case, the information would not have been required to establish the match, however, it contributed to the relevance score of the match (5.1.1) and additionally provided redundancy. Note that, obviously, information from different sources have to interact in answering any single question.

The table shows that all different information sources did indeed contribute to the overall answer recall. This result confirms the conclusion drawn in chapter 3, namely that finding answers for questions is a demanding task for a Natural Language Processing system, requiring that information of different types and from different sources must be joined.

We will comment upon the results for the different information sources in turn.

Question Node Matching. Question node matching is needed for all *wh*-questions and consequently virtually all questions. In most cases, a simple node match was used; less than 5 % of the questions altogether involved a more complex match of the *which*+COMMON NOUN or *how many* type (cf. 5.1.4).

Named Entity Matching. For 86.5 % of the correctly answered questions, at least one named entity had to be matched in question and answer representation (5.2.2.4). Note that this figure includes words that are unknown to the morphology and are handled as Generic Named Entities (such as *Naphtholes* or *chlamys*, cf. 6.2.5).

Grammatical Function Matching. As described above (5.2.2.3), grammatical functions are used to label and match the edges in question and answer representations. In most cases, the label will be identical for question and answer representation, but there are also cases (especially in conjunction with a node relabelling on the basis of GermaNet information, cf. 5.2.2.5) when an uncertain match is needed, as the labels in question representation and answer representation differ.

Additional Semantic Role Matching. The additional semantic role relations (5.2.2.7) have proven to be quite important for finding answers, especially ones that ask for an answer of the corresponding type: When comparing the figures in this table with the ones in tab. 7.19 below, it is obvious that the semantic role relations were used in matching almost all correct answers of the corresponding type.

Frame Matching. With the current limited coverage of the frame resource that we used (4.3.3), only the same-frame relation is currently of any practical importance (5.2.2.6). The most-used frame is NAME_BEARING/BEING_NAMED that is used to represent definition-like descriptions like *X is called Y*.

GermaNet Matching. Of the GermaNet relations, only the synonymy and hyponymy/hypernymy relations were actually used in question matching. This is due to the limited coverage of the other relations (cf. 4.2 and 5.2.2.5).

Anaphora Resolution. Information from the anaphora resolution module is needed when matching question and answer involves anaphoric references. The figures show that all types of anaphoric reference are needed, the number of cases being almost identical for the different types.

Equivalence/*is* Relation. As described above (3.5.2.5), both appositions and predicatives form an important source of definitional information. Note that these figures refer to the use in a equivalence relation as described above (5.2.2.8), not a direct match: While the question *'What is X?'* would match the answer *'X is Y.'* directly, the additional relation is used for 'transitive' cases such as matching *'Where is Auswil?'* against *'Auswil is a parish that is located in Switzerland.'*.

On a more abstract level, one can draw the conclusion that linguistically informed QA based on matching structured linguistic representations of questions and answers provides a useful means of finding answers to questions. As a basis, linguistic dependency structures are required. Named entity recognition plays an especially important rôle, at least for 'encyclopedic knowledge'. Anaphora resolution is essential to bring together information in the document. Then, information on paraphrases, i. e., synonymous or near-synonymous ways of expressing a fact are needed in order to match related, but differing representations. The combination of information about similar frames, GermaNet concepts and syntactically similar structures works, but needs to be complemented with additional information (see the failure analysis below).

7.2.2.2 Error Analysis: Recall Errors

We will now summarise an analysis of errors and failures found in the evaluation. First, we will take a look at non-NIL questions, for which the system did not find any answer (recall errors), and the reasons of these failures. We will

Reason of Failure	Number	Per Cent
Missing Paraphrase	126	40.3 %
Parser Error	38	12.1 %
Anaphora Resolution	26	8.3 %
Missing Inference	22	7.0 %
Error in Named Entity Recogniser	14	4.5 %
Query Timed Out	12	3.8 %
Adjective/Adverb Structure	11	3.5 %
Error in Original Text (Unparsable)	10	3.2 %
Bridging Anaphora not Resolved	9	2.9 %
Period of Time not Handled	8	2.6 %
Missing Local Inference Rule	6	2.0 %
Missing Fusion of Information	6	2.0 %
Compound	5	1.6 %
Anaphora Resolution in Question	5	1.6 %
Morphology Errors	2	0.6 %
Other	13	4.2 %
Σ	313	100.0 %

Table 7.16: Reasons of Failure: No Answer Found

then turn to wrongly answered questions (for both NIL and non-NIL questions; precision errors).

Table 7.16 shows a summary of the reasons why the system failed to find an answer for non-NIL questions. Note that there were cases that involved multiple errors. However, we only marked the most important error type in these cases. We will describe the categories in some more detail.

Missing Paraphrase. The most frequent reason of failure turned out to be missing information on paraphrases: Question and answer are worded differently, and the information that the two formulations are similar is not contained in the linguistic knowledge base of local inferences (5.2). Many of them are comparatively simple (for example, nominal/verbal paraphrases like *N. N. donates X* vs. *X is a donation of* N. N.). In principle, such information might come from electronic dictionaries, morphologies or other lexicographic resources like a future version of German FrameNet. However, we are not aware of any current resource that systematically provides such information for German (cf. 8.3.1.2).

Others are harder to capture systematically (for example, *Domenico Campagnola belongs to the Venice school (of art)* vs. *Domenico Campagnola shows himself as a representative of the Venice school*; *premature infants are prone to Attention Deficit Disorders* vs. *premature infants have a high risk of developing Attention Deficit Disorders*; *Paul Landers used to be a stoker in a library* vs. *Paul Landers made his first money as a stoker in a library*).

Work on automatic paraphrase acquisition from large corpora, especially from the Internet (such as Lin and Pantel, 2001a,b), has produced examples similar to these. However, it is unclear how systematic a coverage can be achieved.

Parser Error. This group of failures is due to some parser error, leading to different dependency structures for question and answer (or a parse failure for the potential answering text, we did not observe any parse failure for any of the 854 questions). One relatively frequent single reason of failure is that the dependency parser currently cannot handle 'sentences within sentences', that is, epentheses in arbitrary positions containing whole sentences. We will further analyse parser errors in the diagnostic evaluation (7.3.1).

Anaphora Resolution. Here, some anaphoric reference that is crucial for matching question and answer has not correctly been resolved. The diagnostic evaluation of the anaphora resolution module gives an overview of types of errors (7.3.2).

Missing Inference. In this group, we have put together cases where a more complex inference would need to be drawn to match question and answer. There is, arguably, an overlap with the missing paraphrases (according to our definition of textual inference, paraphrases just give rise to local inferences, cf. 5.1). We have (somewhat arbitrarily) labelled as missing inferences the harder cases where it is difficult to see how a suitable paraphrase could be acquired. Here is one example: To correctly answer the question *'Who sang "Dajes Mi Krila"?'*, one would have to use the information that *'Ivan Mikulić is a singer'* and that *'he qualified for the Eurovision Song Contest with the song "Dajes Mi Krila"'* and infer that he (very probably) sang the song.

Error in Named Entity Recogniser. Failures in the Named Entity Recognition module (especially in recognising uncommon person and organisation names) lead to mismatches between question and answer representation. This is often connected to subsequent errors in anaphora resolution or parse errors (wrong dependency structures).

Query Timed Out. For the evaluation, we set a time-out for individual database queries. Twelve queries were actually timed out (1.4 %). These were due to the MySQL engine producing sub-optimal query optimisations as described above (6.3.5).

Adjective/Adverb Structure. Currently, no provision is made for 'extracting' information provided by the arguments of attributive adjectives and matching it with their predicative use (and similar for adverbial use). For example, currently the representations of *'How high is the Seekofel?'* and *'Seekofel is a 2 810 m high mountain.'* cannot be matched, as the dependency structures differ. A possible solution would be to systematically use frame structures (e. g., the frame DIMENSION) to 'normalise' these structures.

Error in Original Text. In these cases, the original Wikipedia text contains some grammatical error that makes it unparsable. Even though the PReDS parser is fairly robust, some grammatical errors may lead to a parse error or parse failure. Missing or spurious commas for certain constructions, for example, may lead the parser to 'overlook' a subjunctive clause – or introduce a spurious one. Unfortunately, grammatical errors in the text are relatively frequent in Wikipedia articles (partly probably due to the nature of text genesis, where different users change parts of sentences). As the parser was originally designed to parse newspaper texts and legal documents with a far lower rate of grammatical errors and therefore was not designed for this kind of error, it might be worthwhile to add methods for handling the most common kinds of errors.

Bridging Anaphora not Resolved. As described above, bridging anaphora are not handled by our anaphora resolution module (6.2.9). However, bridging anaphora are quite frequently used in Wikipedia articles; without resolving them, question and answer representation cannot be matched. For example, in an article about the Nigerian federal state Kaduna the sentence *'The capital and largest city is Kaduna.'* is to be found. Matching the representation against that of the question *'What is the capital of Kaduna?'* is not possible without resolving *capital* to *capital of Kaduna*.

Period of Time not Handled. Currently, matching periods of time against *how long* questions is not implemented.

Missing Local Inference Rule. Here, a match between question and answer representation cannot be made, as some local inference rule is missing. For example, *'when'* questions are currently matched against the additional semantic role relation of TIME, but not of CIRCUMSTANCES, loosing answers that describe the circumstances and not the time of an event.

Missing Fusion of Information. So far, information from predicatives and appositions is not fully fused into text representations. Thus, the system can currently not match the representation of *'What is the Germanist von der Hagen known for?'* against *'Friedrich Heinrich von der Hagen was a Germanist. He is known for his editions of old German poetry.'*, as the apposition *'Germanist von der Hagen'* does not match directly.

Compound. A bug in the current system version prevented the matching of certain compound words with three or more components.

Anaphora Resolution in Question. An error occurred during anaphora resolution for the question (cf. 7.2.1.4): some anaphor was either not resolved or wrongly resolved, so that the final question representation is wrong and thus does not match the answer representation.

Morphology Errors. Here, word in question and answer could not be matched, as the morphology (6.2.3) produced no or only spurious analyses for inflected forms.

Other. This group comprises 'singleton' errors, i. e., errors that only occurred once in the evaluation and cannot be systematically grouped with any others.

The overview shows that failures at all levels lead to recall errors, that is, missing answers for questions. By far the largest single group is made up by missing information on paraphrases, viz. variant wordings that are equivalent. The other failures are associated with more unsystematic errors in the different processing modules, especially the dependency parser, the anaphora resolution module and the NER module. Then there is a group of systematic gaps in the system's coverage (for example, missing normalisation for certain structures or the lacking resolution of bridging anaphora). Except for the first group (paraphrases), none of the other reasons of failures account for substantially more than ten per cent of the overall recall errors each. This indicates that to markedly increase recall beyond the current level, a large number of small improvements would be required.

We will take a closer look at the performance of the PREDS parser and the anaphora resolution module in the component evaluation below.

Type of Error	Number	Per Cent
Irrelevant Answer	9	47.4 %
Wrong Person (Warning Output)	3	17.8 %
Anaphora Resolution Error in Question	2	10.5 %
Other	6	31.6 %
Σ	19	100.0 %

Table 7.17: Types of Errors: Spurious Answers for NIL Questions

Type of Error	Number	Per Cent
Irrelevant Answer	18	22.0 %
Parser Error	18	22.0 %
Answer is Question Topic	7	8.5 %
Error in Database Query	6	7.3 %
Error in Original Text	5	6.1 %
Anaphora Resolution Error	4	4.9 %
Wrong GermaNet Hypernymy	4	4.9 %
Anaphora Resolution Error in Question	3	3.7 %
Other	18	22.0 %
Σ	82	100.0 %

Table 7.18: Types of Errors: Wrong Answers for Non-NIL Questions

7.2.2.3 Error Analysis: Precision Errors

Why does the system return wrong or spurious answers to questions? Tables 7.17 and 7.18 list the types of errors for NIL questions (spurious answers) and for non-NIL questions, respectively.

We will describe the categories of error types in more detail in the following.

Irrelevant Answer. Interestingly, the most common type of error is caused by irrelevant answers: These answers are factually correct, they are about the question subject, and yet they are not satisfactory. Thus, the issue is mainly a pragmatic one (3.2.3) and hence difficult to systematically resolve. Here are a few selected examples:

(7.4) How is Cider produced?
Commercially

Cider was first commercially produced by William Magner in Clonmel in 1935. (German Wikipedia, ‘Bulmers Original Irish Cider’, 2006-07-24)

(7.5) When did he [Felix von Luschan] live?
1878/79
1878/79, he [Felix von Luschan] was in Bosnia as a military doctor. (German Wikipedia, Felix von Luschan, 2005-06-08)

(7.6) What is ordination?
A German word
The German word ordination is used almost exclusively in Protestant church. (German Wikipedia, Ordination, 2005-09-08)

Wrong Person. As described above, the Named Entity match allows matching person names if last names match, even though first names differ, to improve recall (5.2.2.4). In cases where no better match is found, such an answer is returned. For three NIL questions, no correct match is found and therefore the (wrong) answer is generated. However, it is shown together with a warning, stating that the answer is probably about a different person and asking the user to check.

Anaphora Resolution Error in Question. As described above, errors occur during the resolution of anaphora in follow-up questions. In most of the cases, a definite description was not resolved, leading to overgeneration. For example, the ‘starter question’ *‘What does “Larmoyance” mean?’* is followed by *‘Where does the term come from?’*. The anaphora resolution module fails to resolve the definite *the term*, thus the retrieved answer is *‘[The term “circle” is derived] from the Latin word circulus.’*

Parser Error. Parse errors may result in the production of a wrong dependency structure (cf. 7.2.2.2, 7.3.1). Especially attachment errors, for example the wrong attachment of an apposition, will lead to wrong answers.

Answer is Question Topic. For definition questions, the system will sometimes display the question topic as an answer. For example, the system’s (first) answer to the question *‘Who is Lisa Bonet?’* was simply *‘Lisa Bonet’*. A filter routine usually discards such tautological answers, however, it had a bug so that under certain conditions a tautological answer could slip through.

Error in Database Query. Due to an error in the function that translates question structures into actual database queries, for a small number of queries, one constraint is not added to the database query, leading to overgeneration and thus unpredictable results.

Error in Original Text. Here, an error in the original text lead to a wrong dependency structure, allowing a spurious match of question and answer representation.

Anaphora Resolution Error. An error in anaphora resolution during the processing of the document collection produced a dependency structure that erroneously matches the question representation (cf. also 7.3.2)

Wrong GermaNet Hypernymy. This small group comprises errors where the use of the GermaNet hypernymy relation as a source of (uncertain) inferences actually lead to a wrong inference. Note that the answers are issued together with a warning message to draw the users attention to the fact. For example, the question *'When was the audio CD developed?'* matched the sentence *'In 1988, more than 100 million audio CDs were produced for the first time.'*, using a hypernym relation between *'entwickeln'* (*develop*) and *'produzieren'* (*produce*).[6] As the answer checking module (5.1.5) recognised the uncertain inference, it is flagged and the whole sentence is displayed for evaluation.

Other. As above, errors that occurred only once and cannot be grouped with other errors are not further examined here.

The analysis of the error types for precision errors lists some types that are related to relatively local system errors that should be easily fixable. Another group of errors is connected to uncertain matches (uncertain person name matches, uncertain inferences based on hypernyms) and were accompanied by appropriate warning messages. These uncertain matches could be switched off (hurting recall, however), but we consider the present solution, i. e., to display these uncertain answers together with a warning message, as quite satisfactory. In the future the exact behaviour of the system could also be controlled by user preference settings.

As for recall errors, there is an additional group of errors related to natural language processing problems (parser errors, anaphora resolution errors) that we do not expect can be systematically solved. While proportion of wrong answers caused by parser errors of all wrong answers is relatively high at 22 %, their overall proportion (in relation to all non-NIL questions) is small, at 3 %.

[6]Note that we are not convinced that this hypernym relation should actually exist at all.

The last, and possibly one of the most interesting groups, is the one we have labelled as Irrelevant Answers: Here, it is especially difficult to see how the errors can methodically be solved.

One possibility may be the use of a more fine-grained question typology: If the system would, e. g., identify the question in (7.5) as a question of type, say, DATE-BORN-AND-DIED, it could discard the wrong answer given here (cf. 2.2.2).

7.2.2.4 Results for Different Question-Answer-Types.

Let us take a look at the detailed results for different question-answer types. Table 7.19 summarises all results for the different question-answer types (cf. also tab. 7.3). On the left hand side of the table, results for NIL questions are shown, on the right hand side those for non-NIL questions.

Percentages that differ from the corresponding mean percentages in a statistically significant way ($p < 0.01$) are marked in tab. 7.19 with $^+$ (significantly higher value) and $^-$ (significantly lower value). Especially interesting differences are marked with **bold font**. We will discuss a number of them below. In some of these cases, percentages differed from mean percentages only marginally ($p < 0.05$) or not significantly due to the small number of examples. We will indicate these cases accordingly.

The first point that attracts attention is that the system produces answers for definition questions (definition and person definition) more often than average (hence low values in the ‘no answer’ column).

The system performs especially well for person definitions (*Who is N. N.?*), with an non-NIL answer accuracy of 79.3 % and an answer precision of $46/(46+9) = 83.6\%$ (marginally significant deviation from mean answer precision). Many of the Wikipedia articles about a person start by a defining sentence of the form *N. N. is/was X.*, which can be matched with high accuracy.

For other definitions, answer accuracy is also markedly higher than average at 52.8 %, however, answer precision is almost the same as mean answer precision at $65/(3+65+36) = 62.5\%$ (due to marginally significantly more spurious and significantly more wrong answers).

The system was especially successful at finding answers for definition questions for two reasons: On the one hand, the Wikipedia articles often contain explicit definitions. On the other hand, using different constructions such as predicative constructions, appositives and definite anaphora as sources for definitional information also provides more possibilities of matching than for other question-answer types.

Secondly, the scores for abbreviations and abbreviation definitions are significantly lower (with more spurious and more wrong answers). This is due

Type	NIL Questions					Non-NIL Questions							
	Corr.		Spur.			Corr.		Wrong		No Ans.			
	#	%	#	%	Σ	#	%	#	%	#	%	Σ	$\Sigma\Sigma$
Definition	11	78.6	**3**	**21.4**	14	**65**	**52.8**$^{+}$	**36**	**29.3**$^{+}$	**22**	**17.9**$^{-}$	123	137
Place	20	95.2	1	4.8	21	**34**	**44.2**	9	11.7	34	44.2	77	98
Date	33	91.7	3	8.3	36	21	36.2	4	**6.9**$^{-}$	33	56.9	58	94
Name	31	96.9	1	3.1	32	2	4.7^{-}	7	16.3	**34**	**79.1**$^{+}$	43	75
Pers. Def.	1	100.0	0	0.0	1	**46**	**79.3**$^{+}$	9	15.5	**3**	**5.2**$^{-}$	58	59
Person	22	100.0	0	0.0	22	6	16.2	3	8.1^{-}	**28**	**75.7**$^{+}$	37	59
Complex	37	90.2	4	9.8	41	1	33.3	1	33.3^{+}	1	33.3	3	44
Other	4	80.0	1	20.0	5	4	11.8^{-}	2	5.9^{-}	**28**	**82.4**$^{+}$	34	39
Measure	13	100.0	0	0.0	13	1	5.3^{-}	0	0.0^{-}	**18**	**94.7**$^{+}$	19	32
Yes/No	8	100.0	0	0.0	8	2	11.8	0	0.0^{-}	**15**	**88.2**$^{+}$	17	25
Number	9	100.0	0	0.0	9	3	20.0	0	0.0^{-}	**12**	**80.0**	15	24
Reason	7	100.0	0	0.0	7	3	30.0	0	0.0^{-}	7	70.0	10	17
Special	4	100.0	0	0.0	4	2	16.7	0	0.0^{-}	10	83.3	12	16
Purpose	5	83.3	1	16.7	6	1	10.0	0	0.0^{-}	**9**	**90.0**	10	16
Abbrevdef	3	42.9^{-}	**4**	**57.1**$^{+}$	7	2	25.0	**3**	**37.5**$^{+}$	3	37.5	8	15
How	6	100.0	0	0.0	6	0	0.0	1	12.5	**7**	**87.5**	8	14
Etymology	3	100.0	0	0.0	3	0	0.0	1	14.3	**6**	**85.7**	7	10
Material	1	100.0	0	0.0	1	1	12.5	2	25.0^{+}	5	62.5	8	9
Period	0	0.0^{-}	1	100.0^{+}	1	1	14.3	0	0.0^{-}	**6**	**85.7**	7	8
WWW	8	100.0	0	0.0	8	0	–	0	–	0	–	0	8
Time-start	2	100.0	0	0.0	2	1	16.7	0	0.0^{-}	5	83.3	6	8
Vocation	2	100.0	0	0.0	2	1	20.0	1	20.0	3	60.0	5	7
Circums.	3	100.0	0	0.0	3	0	0.0	0	0.0	3	100.0	3	6
Looks	1	100.0	0	0.0	1	0	0.0	1	25.0	3	75.0	4	5
Currency	1	100.0	0	0.0	1	0	0.0	0	0.0	**4**	**100.0**	4	5
Organ.	2	100.0	0	0.0	2	0	0.0	0	0.0	2	100.0	2	4
Source	1	100.0	0	0.0	1	3	100.0	0	0.0	0	0.0	3	4
Abbrev.	0	–	0	–	0	0	0.0	**1**	**33.3**$^{+}$	2	66.7	3	3
Doc.	0	–	0	–	0	0	0.0	0	0.0	3	100.0	3	3
Assessm.	1	100.0	0	0.0	1	0	0.0	1	50.0^{-}	1	50.0	2	3
Ratio	0	–	0	–	0	0	0.0	0	0.0	**3**	**100.0**	3	3
Employer	0	–	0	–	0	0	0.0	0	0.0	2	100.0	2	2
Altern.	1	100.0	0	0.0	1	0	0.0	0	0.0	1	100.0	1	2
Σ	240	92.7	19	7.3	259	200	33.6	82	13.8	313	52.6	595	854
Overall		28.1		2.2			23.4		9.6		36.7		100.0

Table 7.19: Evaluation Results, Broken Down by Question-Answer-Type

to the fact that the system currently has no means of distinguishing abbreviations from other linguistic material and will, for example, when asked to define *‘ASTER’* not return the correct *‘Advanced Spaceborne Thermal Emission and Reflection Radiometer’*, but rather give a (correct!) definition of the plant aster.

A third interesting point is that the system performs above average for place and date questions, with higher accuracy (more correct answers for place questions, marginally significant, and significantly fewer wrong answers for date questions) and higher answer precision for both (place: $34/(1+34+9) = 77.3\,\%$, date: $21/(3+21+4) = 75.0\,\%$, both differences not significant). This is mainly due to the successful assignment of additional semantic relations during parsing as described above (cf. also 5.2.2.7).

Then there are a number of question types that seem to be especially hard for the system. These can be separated into three different groups.

The first group comprises the question types Name, Person and also Yes/No, with significantly more NIL answers than average. On inspection, we found these questions to be especially difficult, as users especially often used paraphrases when formulating these questions. We have identified the lack of information on paraphrases as the largest single group of failure to find answers above.

The second group is made up of the types Measure, Number, Currency, Ratio and also Period (of time)[7]. While our Named Entity Recognition module can recognise the respective entities forming the answer to such questions, the answer matching module can currently often not handle the different structures used to relate entity and measure to each other.

The last group consists of question types where the answer is often difficult to spot, namely Purpose, How, Etymology and Others[8]. This sort of information can be given in many different ways that are often difficult to distinguish from each other.

7.2.2.5 Conclusions

In this section we have looked at the results from the system evaluation from a diagnostic standpoint to find out more about where the system performs well and why – and what leads to errors.

It turns out that the ‘positive cases’ are not due to one single resource or module, rather, they are achieved through an interplay of different contributing

[7]More NIL answers than average. Results significant for Measure; marginally significant for Number; not significant for Currency, Ratio and Period.

[8]More NIL answers than average. Results significant for Others, marginally significance for Purpose and How, not significant for Etymology

resources. This confirms the intuition that QA is a demanding application that still holds potential for improvement.

The analysis of positive cases shows that the relation of indirect answerhood as developed in chapter 5 can be practically implemented: Syntactic dependency structures, lexical semantic information and information about coreference can be put together to find answers to questions in documents.

The breakdown of types of recall errors has shown that there is one central missing resource, namely a lack of systematic linguistic databases of paraphrases, from simple noun-verb paraphrases up to far more complex cases. While frame semantics can, in principle, provide exactly the sort of knowledge required (especially including information on semantic roles associated with frames in the form of frame elements), the fragment that we were using does not yet provide sufficient coverage. In addition, for the more complex types of paraphrases we would consider automatic paraphrase acquisition an interesting (complementary) source of information (8.3.1.2).

As regards the system, we have identified a number of systematic errors that can be fixed relatively easily. However, the evaluation has also shown that there are a number of errors associated with coverage, especially errors of the different NLP modules (parser errors, anaphora resolution errors and named entity recognition errors). In the following section, we will report the results of the component evaluation of the PREDS parser and the anaphora resolution module.

7.3 Component Evaluation

The diagnostic part of the end-to-end evaluation of the SQUIGGLI system has shown that errors during parsing and anaphora resolution have lead to both recall and precision errors of the overall system. The relative number of errors of these types was not high: For both recall and precision errors, parser errors accounted for about 20 %, anaphora resolution errors for about 10 %. We were still interested in analysing the performance of the respective system components more closely. We mainly aimed to identify errors in the modules that can systematically be fixed. We also wanted to find out whether the modules that we used could be replaced by other, better performing ones.

We therefore carried out a component evaluations of the PREDS parser (7.3.1) and the anaphora resolution module (7.3.2). We employed manually annotated ‘gold standard’ corpora as yard-sticks. We will conclude the section with an investigation of the overall system’s processing speed (7.3.3).

7.3.1 Evaluation of the Dependency Parser

The SQUIGGLI system makes use of a dependency parser that derives Partially Resolved Dependency Structures (PREDS) from German texts (cf. 6.2). To evaluate the parsing component, we are mainly interested in the proportion of text sentences for which it will typically derive the correct dependency structure. This can best be established through the use of a manually annotated corpus (gold standard) and comparing the parser output for the sentences in the corpus with the annotation (see Carroll et al., 1999 for an overview of different parser evaluation methods).

We manually annotated two small corpora of German texts (business news from a newspaper and excerpts from legal documents, about 750 sentences) with dependency structures, as a gold standard.

We found that we could not use either the Negra corpus (Brants et al., 1999) or the Tiger corpus (Brants et al., 2002) for this evaluation. These corpora have become the *de facto* standard for training and evaluating German parsers. Both are manually annotated with syntactic structures, using similar annotation schemes that combine features of phrase structure grammars and Dependency Grammar (Brants et al., 2002). They consist of 20 000 (Negra) and 50 000 (Tiger) annotated sentences.

To use Negra or Tiger, a considerable number of mapping rules would have to be defined to allow a direct comparison between the Tiger/Negra structures and our dependency structures. For example, heads of NPs/PPs are not explicitly annotated and would have to be inferred from the part of speech information combined with the position in the phrase. In addition, in our structures verb complexes (constructions resulting from complex tenses, use of modal verbs and split verbs) are generally collapsed into one single node, while Tiger/Negra keeps them separate.

These observations seem to be related to results from Amit Dubey's Ph. D. thesis (Dubey, 2004): He used the Negra and Tiger corpus to train a statistical parser for German and found that he could improve overall results by 'unflattening' the Tiger annotation, but that the required information was not explicitly present in the annotation and could only be reconstructed with a number of specific rules (cf., e. g. Dubey, 2004, 91–92).

We therefore decided to annotate a gold corpus as a basis for evaluation. We will first describe the annotation scheme and annotation process before turning to the evaluation itself.

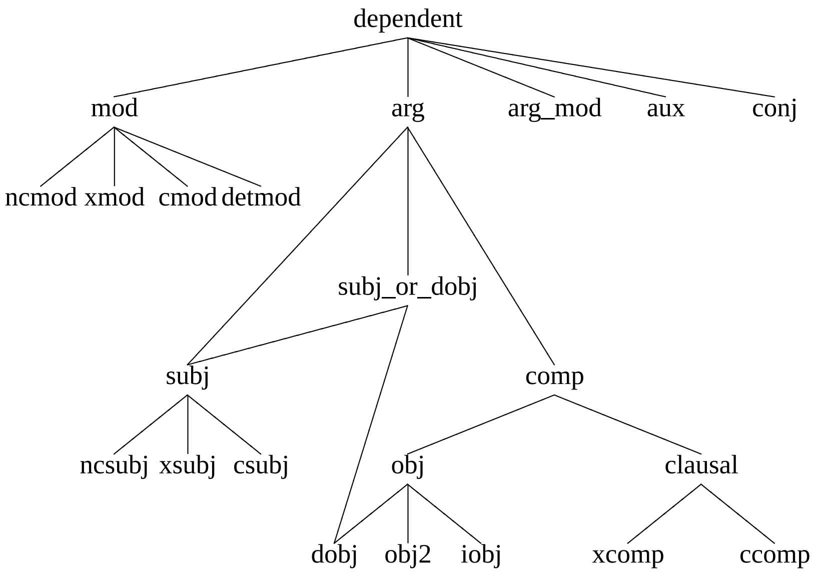


Figure 7.2: Hierarchy of Grammatical Relations (from Briscoe et al., 2002, 5)

7.3.1.1 Corpus Annotation

As annotation format for our corpus, we chose the grammatical relation (GR) annotation scheme described in Carroll et al. (2003); Briscoe et al. (2002); Briscoe and Carroll (2000); Carroll et al. (1999). This annotation scheme is based on named grammatical relations between lemmatised lexical heads, and is thus similar to the PREDS dependency relations. We consider using this scheme suitable for corpus annotation, as it provides a good chance for both the comparability of the evaluation and the re-usability of the annotated corpus. We chose the GR scheme rather than the somewhat similar scheme suggested by Dekang Lin (cf. Lin, 2003, 1998a), because Lin's scheme uses only unlabelled dependency relations.

Annotation in the GR scheme uses a subsumption hierarchy of grammatical relations (cf. fig. 7.2). An annotated example is shown in fig. 7.3. The hierarchy is motivated by experiences made in a sample corpus annotation of the authors (Briscoe et al., 2002). It can especially be used to express underspecified relations, which in turn allows to set up meaningful, 'lenient' scoring methods.

Source Sentence:

Rund	*600*	*neue*	*Mitarbeiter*	*will*	*das*	*Datenverarbeitungsunternehmen*	*in*
about	600	new	employee	want_to	the	data processing company	in

diesem	*Jahr*	*in*	*Deutschland*	*einstellen.*
this	year	in	Germany	hire.

The IT company plans to hire about 600 new staff in Germany this year.

Grammatical Relations:

```
ncsubj(einstellen, Datenverarbeitungsunternehmen, _)
dobj(einstellen, Mitarbeiter, _)
ncmod(_, Mitarbeiter, neu)
ncmod(_, Mitarbeiter, 600)
ncmod(_, 600, rund)
ncmod(in, einstellen, Jahr)
ncmod(_, Jahr, dieses)
ncmod(in, einstellen, Deutschland)
```

Figure 7.3: Example: Grammatical Relation Annotation

Most of the (abbreviated) relations are self-explanatory; here are some additional keys: c stands for clausal (for example, for subjunctive clauses, `cmod` is used), nc for non-clausal (for example, for modifying PPs, `ncmod` is used), x for clausal-control (for example, for infinitive clauses ‘inheriting’ the subject of the matrix verb, `xmod` is used).

Some additional features are worth noting: Prepositions are ‘integrated’ as an additional argument into a `ncmod` (non-clausal modifier) or `iobj` (indirect object) relation. Much like prepositions, subordinating conjunctions become an argument of a single grammatical relation (`xmod`, `cmod`, `xcomp` or `ccomp`). In passive sentences, ‘deep’ subject and object are additionally marked. We have not annotated definite/indefinite articles.

An annotation example from the corpus is shown in fig. 7.3. It shows the general idea of using the lemmata of the content word in the sentence as ‘anchors’, combined with the integration of certain function words (such as prepositions) into the grammatical relations as additional arguments.

	Business News	Legal
Source	Business News section of daily paper *Süddeutsche Zeitung*, January 1998	Two Judgments from German Legal Courts
Size (sentences)	655 sents.	100 sents.
Size (tokens)	10 215 words	3 341 words
Avg. sentence length	15.59 words	33.41 words
Median sentence length	16 words	33 words
Min. sentence length	4 words	5 words
Max. sentence length	45 words	88 words

Table 7.20: Details of the Manually Annotated Evaluation Corpus

As corpus, we have randomly selected 655 sentences from the business news section of the German daily newspaper *Süddeutsche Zeitung* from 1998 (sub-corpus Business News) and 100 sentences from two judgments from German legal courts (sub-corpus Legal). Table 7.20 shows additional details of the corpus.

The corpus was annotated independently by two annotators, both undergraduate students of computational linguistics who were not involved in the development of the system[9]. An annotation guideline was compiled and subsequently refined (Braun and Fliedner, 2005). Annotation was conducted with a specially developed annotation tool that allows annotation with grammatical relations by ‘point and click’.[10] It presents the sentences of the corpus to the annotator together with a list of (not disambiguated) lemmata of the content words in the sentence and the list of grammatical relations (fig. 7.2). Grammatical relations can be added by clicking on the relation name and then the ‘participating’ lemmata. This allows an efficient annotation.

The annotations from the two annotators were merged and corrected in a two-step process using a merging tool that takes two annotations as input and lists all differences. First, the annotators compared their annotations to fix obvious mistakes. All remaining differences were discussed in group meetings to reach a final agreement.

[9]We would like to thank Jana Besser and Niko Felger, who carried out the annotation.

[10]Tools for the evaluation were originally developed by our former colleague Christian Braun at our Department.

We tested inter-annotator agreement during the annotation process. For a first trial annotation, which was conducted without training and with a first, incomplete version of the annotation guidelines, the annotators reached an inter-annotator agreement of 62 %. Agreement was measured as the number of grammatical roles, on which the annotators agreed perfectly (head, dependent, role, additional information) of all grammatical roles that any of the annotators annotated.

For the pre-final version of the corpus (that is, before the final adjudication and merging), inter-annotator agreement (perfect agreement, based on grammatical relations) was 88 % for the Business News sub-corpus and 94 % for the Legal subcorpus. This is slightly below the scores reported for the annotation of an English corpus, for which an agreement level of 95 % was reported (Carroll et al., 2003). Upon inspection, we found that the only substantial source of disagreement was the classification of PPs as arguments or modifiers (i. e., label `ncmod` vs. `iobj`), where it is often indeed not possible to make clear-cut decisions.

We consider the annotated corpus an interesting for evaluating German parsers.[11]

7.3.1.2 Evaluation Results

The sentences of the evaluation corpus were run through the parser. The obtained PREDS were then converted into the GR format with a simple conversion tool[12]: The dependency structures of the PREDS format can be easily transformed into the grammatical relations of the GR annotation format. Thus, both the gold corpus and the test corpus (parser results) were in the same format and could be compared using the comparison and evaluation tool.

As evaluation measures, we computed recall, precision and F-score over grammatical relations with regard to the gold corpus. Recall measures the proportion of grammatical relations that the parser correctly identified in relation to all relations in the gold corpus; precision is the proportion of correctly identified grammatical relations in relation to all relations that the parser returned. F-score were computed according to fig. 7.1, with weighting factor $\beta = 1$.

We computed three sets of scores, namely strict, relaxed and lenient. This is motivated as follows: As described above, our parser returns underspecified representations; it especially does not resolve PP attachment ambiguities (cf. 6.2.7.1). It produces an underspecified representation from which the dif-

[11]The corpus is available from the author upon request.

[12]This tool was also developed by Christian Braun.

Evaluation method	Precision	Recall	F-Score
Strict	51.91	51.52	51.72
Relaxed	61.97	61.51	61.74
Lenient	77.15	78.66	77.89

Table 7.21: Parser Evaluation Overview: Precision, Recall and F-Score for Sub-Corpus ‘Business News’

ferent specific alternatives can be easily derived. We assume that subsequent processing steps will be able to make the correct choice.

The strict scoring method regards only relations that are identical in gold corpus and test corpus as correct. In the case of attachment, this means that only those relations are considered correct in which indeed the default low attachment was the correct choice. Thus, the strict score gives a baseline for an overall system where no further correction of attachment takes place.

Arguments and modifiers are marked with different grammatical relations in the GR annotation format. For example, a PP that functions as an argument will be marked as `iobj`, while a modifier PP will be marked as `ncmod`. In relaxed scoring, arguments and modifiers are not distinguished. This especially applies to PPs: Here, the distinction between argument and modifier is often quite subtle and thus hard to make. As mentioned above, this was the most common case for disagreement between annotators. For most applications, this distinction does not play a rôle.

The lenient scoring method can be seen as a top-line. It resolves all underspecifications in the test corpus optimally with regard to the gold corpus: If the correct analysis in the gold corpus is compatible with the underspecified representation in the test corpus, then it is awarded full credit.

Depending on the actual application, the results from different scoring methods may be most relevant. In our system, the lenient score is most directly applicable, as our approach to matching questions and answers does not distinguish between arguments and modifiers. In addition, we allow different possibilities of modifier attachment, thus correcting a number of attachment errors.

Precision, recall and F-score for both sub-corpora, are shown in tab. 7.21 (business news) and 7.22 (legal). Tables 7.23 and 7.24 show detailed scores for lenient scoring, broken down by grammatical relations.

Evaluation method	Precision	Recall	F-Score
Strict	46.15	40.46	43.12
Relaxed	52.48	45.95	49.00
Lenient	61.58	56.12	58.72

Table 7.22: Parser Evaluation Overview: Precision, Recall and F-Score for Sub-Corpus ‘Legal’

Gramm. rel.	# corr.	# wrong	# miss.	Prec.	Recall	F-score
detmod	63	3	4	95.45	94.03	94.74
ncmod/iobj	2 376	481	628	83.16	79.09	81.08
ncsubj	666	136	148	83.04	81.82	82.43
arg_mod	12	44	1	21.43	92.31	34.78
xmod/xcomp	20	10	37	66.67	35.09	45.98
dobj	375	133	49	73.82	88.44	80.47
cmod/ccomp	68	29	38	70.10	64.15	67.00
conj	84	87	89	49.12	48.55	48.84
subj_or_obj	0	162	0	0.00	–	–
Σ	3 664	1 085	994	77.15	78.66	77.89

Table 7.23: Parser Evaluation Results, Classified according to Grammatical Relation, Business News (Lenient Scoring)

Gramm. rel.	# corr.	# wrong	# miss.	Prec.	Recall	F-score
detmod	15	1	4	93.75	78.95	85.71
ncmod/iobj	720	337	473	68.12	60.35	64.00
ncsubj	127	58	93	68.65	57.73	62.72
xmod/xcomp	28	5	60	84.85	31.82	46.28
dobj	65	60	42	52.00	60.75	56.03
cmod/ccomp	48	70	76	40.68	38.71	39.67
conj	29	31	57	48.33	33.72	39.73
arg_mod	0	4	2	0.00	0.00	0.00
subj_or_obj	0	78	0	0.00	–	–
Σ	1 032	644	807	61.58	56.12	58.72

Table 7.24: Parser Evaluation Results, Classified according to Grammatical Relation, Legal (Lenient Scoring)

7.3.1.3 Error Analysis

We will now discuss a number of systematic errors that a post-hoc analysis of the evaluation revealed. For a small number, corrections or changes to the parser were obvious and have been carried out. Often, however, the problem lies deeper and will not easily be corrected.

No Parse. Three sentences out of 655 in the Business News corpus (0.5 %) and six out of 100 in the Legal corpus (6 %) did not receive any analysis. None of these were caused by failures of the parser itself (program errors). The majority of parse failures (seven out of nine) were due to missing coverage of coordination phenomena in the verb cluster combined with epentheses that the topological parser could not handle. The remaining two failures were due to actual mistakes in the source text that lead to incompatible parts of complex verbs (for example, a verb infinitive was used instead of the past participle in constructing a sentence in perfect tense). As cases like this are notoriously difficult to handle (cf. the discussion on handling ungrammatical phenomena in Fliedner, 2001), our parser currently does not attempt to resolve them.

Complex Conjunctions. Complex conjunctions of NPs including post-nominal PPs and/or appositions are not correctly analysed in all cases. This error is more common in the legal documents, where these structures reach a level of complexity that often makes them unintelligible even for human readers. This is reflected by the comparatively low scores for the grammatical relation `conj` (conjunction).

Missing Valencies. As described above, the PREDS parser can make use only of very limited valency information. Information about required arguments could, for example, help to control decisions such as whether a bare NP is an apposition or an argument of a main verb. Such cases are decided heuristically, and errors may occur. Many of the errors in the assignment of the non-clausal relations (`ncsubj`, `dobj`, `subj_or_obj`, `iobj`, `arg_mod`) are associated with this error type.

Verb/Adjective Reading of Participles. For a number of sentences, two alternative analyses are theoretically possible, namely a complex passive verb form or a copula/adjective construction, similar to the (gradual) difference between English *'The door is closed.'* and *'The case is closed.'* This also includes a difference of active/passive. Our parser prefers the verb/passive reading, occasionally leading to mistakes, where both the head lemma and the involved roles are wrong. This is mirrored by the errors involving the `xmod`/`xcomp` relations.

Errors in Original Text. The newspaper texts contain an unexpectedly high number of different grammatical errors. While our parser only completely fails in few cases, these errors still often lead to wrong structures (for example, NPs containing agreement errors are not properly put together); as annotators were asked to ignore such errors (i. e., annotate the sentences as if they were grammatically correct), such errors in the original text generally lead to mismatches in the evaluation. We have experimented with automatically correcting errors in text[13]; we found, however, that this too often lead to precision errors.

Clerical Errors. Last but not least, a number of clerical errors can be identified: Some unexpected combination of features leads to the wrong overall analysis. These errors are typically quickly fixed, but, together, they lower the system's coverage.

All in all, we have not found any fundamental errors whose corrections would boost system performance, but rather a range of comparatively small errors that can typically be fixed easily, but together limit the coverage.

7.3.1.4 Discussion

The results of of the performance evaluation of the PREDS parser must be put into perspective, of course.

When considering parse errors as a source of errors in the overall system, we found that they do not form a large percentage of all errors (12 % of recall errors, cf. 7.2.2.2, and 22 % of precision errors, 7.2.2.3, respectively).

Could a better overall system performance be achieved by replacing the PREDS parser with a different parser?

There are relatively few full parsers for German. Of these, the reportedly best-performing system is Amit Dubey's statistical parser Sleepy (Dubey, 2004). It was trained and evaluated on the Negra and Tiger corpora.

When we started work on our current project, this parser was not yet available. Using the PREDS parser also had the advantage that we could very simply adapt it to the tasks at hand, such as the correct analysis of questions (cf. also 6.2.1.2).

As one part of the evaluation, Dubey used dependency structures. The reported figures from this evaluation are roughly comparable to ours, as both evaluations are based on dependency structures. Besides, the domain is similar, at least for the business news sub-corpus (the Negra/Tiger corpus consists of newspaper texts like ours, even though it is not limited to business texts).

[13]Recall that our NP/PP chunker was originally built for use in a grammar checking system, cf. Fliedner (2001).

Corpus	Extended unlabelled dep.	Labelled dep.
Negra	85.5 (Smooth+GF)	79.7 (Baseline+GF)
Tiger	90.5 (Sister-head)	82.8 (Baseline+GF)

Table 7.25: Evaluation Results Using Dependency Structures for Amit Dubey's Statistical German Parser Sleepy (Dubey, 2004, 83–87)

Dubey reports a number of F-scores for the evaluation of his statistical German parser using dependency structures for the different corpora and for different models. Table 7.25 shows the best results (F-scores) for both Negra and Tiger corpus; the respective model that achieved the score is given. For details of the different evaluation methods, viz. unlabelled, extended and labelled dependencies, see Dubey (2004, 83–84).

Even though these figures are not directly comparable, they indicate that the PREDS parser does reasonably well: When comparing Dubey's unlabelled dependency scores with our lenient scores (i. e., assuming that the underspecified attachment choices will be resolved by a later processing step), the difference in F-scores is 7 % to 12 %.

We would therefore expect that replacing the PREDS parser with a different parser component could improve overall performance, but only moderately so. An interesting option might be the combination of different parsing approaches in a hybrid architecture (6.2.1.4).

7.3.2 Anaphora Resolution

The second component evaluation is concerned with our anaphora resolution module (6.2.9).

We will first shortly summarise the evaluation schemes that we used (7.3.2.1).

We will then turn to the evaluation of our anaphora resolution module against a corpus of German texts that was manually annotated for anaphora, namely the Heidelberg Text Corpus (7.3.2.2).

7.3.2.1 Measures

The most commonly used scoring scheme for coreference resolution is the Model-Theoretic Scoring Scheme (Vilain et al., 1995). It uses pre-annotated corpora as gold standard.

$$Recall_{Testset} = \frac{\Sigma((|S_i|-1)-(|p(S_i)|-1))}{\Sigma(|S_i|-1)} = \frac{\Sigma(|S_i|-|p(S_i)|)}{\Sigma(|S_i|-1)}$$
$$Precision_{Testset} = \frac{\Sigma(|S'_i|-|p'(S'_i)|)}{\Sigma(|S'_i|-1)}$$

Figure 7.4: Model-Theoretic Scoring Scheme for Coreference Resolution: Formulæ for Recall and Precision according to Vilain et al. (1995)

Coreference relations between referring expressions in the corpus are viewed as equivalence relations, establishing equivalence classes of referring expressions. Recall is defined as the relation of the (minimal) number of missing links in the system output to the overall number of links in the gold corpus. Note that, as equivalence classes are considered, only one link between *any* two members of two different equivalence classes is required to establish the equivalence of all members of these two classes. Precision is computed by inverting the rôle of the gold and the test corpus: It is defined as the proportion of links that would have to added to the *gold* corpus to yield all equivalences in the *test* corpus. Alternatively, this can be interpreted as the proportion of links that would have to be *deleted* from the test corpus.

Figure 7.4 gives the formulæ for both recall and precision from Vilain et al. (1995). Here is a short overview of the different used symbols:

S_i Equivalence class *i* in gold corpus

$p(S_i)$ Partition of the test corpus relative to S_i: set of equivalence classes that share members with S_i; ideally, the singleton set $\{S_i\}$.

$S'_i, p(S'_i)$ Equivalence classes and partitions with gold corpus and text corpus reversed.

This captures the description above, for details see Vilain et al. (1995).

Roland Stuckardt has suggested additional measures to evaluate coreference resolution systems (Stuckardt, 2001): He specifies a second set of recall and precision scores based on single coreference links. These measures based on links allow to report separate scores for different types of anaphora.

We will report recall, precision and F-score (cf. 7.1) according to the MTSS. We also report two measures based on labelling each anaphor-antecedent *pair* for correctness. This procedure is similar to the one suggested in Stuckardt (2001); however, we use a somewhat modified set of labels. We distinguish the following cases:

$$Recall_{Correct\,Chain} := \frac{\#Correct + \#PartiallyCorrect}{\#Correct + \#PartiallyCorrect + \#Wrong + \#Missing}$$

$$Recall_{Correct\,First\,Antecedent} := \frac{\#Correct}{\#Correct + \#PartiallyCorrect + \#Wrong + \#Missing}$$

$$Precision_{Correct\,Chain} := \frac{\#Correct + \#PartiallyCorrect}{\#Correct + \#PartiallyCorrect + \#Wrong + \#Spurious}$$

$$Precision_{Correct\,First\,Antecedent} := \frac{\#Correct}{\#Correct + \#PartiallyCorrect + \#Wrong + \#Spurious}$$

Figure 7.5: Recall, Precision and F-Score for ‘Correct Chain’ and ‘Correct First Antecedent’ Scheme

Correct. The anaphor is marked both in the gold and the test corpus. To be marked as correct, it must be *resolvable* to the correct first anchor: Only if an anaphoric expression can be linked to the ‘globally’ correct antecedent the anaphoric reference it is marked as correct.

Partially Correct. In this case, the anaphor is marked correctly. However, while its antecedent forms part of the correct equivalence class, it cannot be resolved to the correct ‘absolute’ anchor; that is, there is some break in the coreference chain in the test corpus.

Wrong. The anaphor is correctly identified, but the assigned antecedent is wrong.

Missing. This anaphor was marked in the gold corpus, but is missing from the test corpus.

Spurious. The test corpus contains a spurious anaphor, viz. one that is not annotated as an anaphoric expression in the gold corpus.

We compute two different sets of recall and precision values. The first set (‘correct chain’) considers both correct and partially correct anaphora-antecedent pairs as true positives, while the second set (‘correct first antecedent’) counts only correct ones. The corresponding formulæ are given in fig. 7.5.[14]

[14]Note that our definition of the derived measure of precision differs considerably from Stuckardt’s (Stuckardt, 2001, 497): Our definition gives the proportion of correct decisions of all anaphor-antecedent pairs suggested by the system (i. e., in the test corpus). This is in keeping with the general definition for precision and recall (cf., e. g., Hartrumpf, 2001). We would consider

7.3.2.2 Evaluation and Results

As gold standard corpus, we used the Heidelberg Text Corpus for evaluation.

We also considered the Potsdam Commentary Corpus (PCC) as an alternative gold standard for evaluation. The PCC is a corpus of German newspaper texts taken from the daily paper *Märkische Allgemeine Zeitung*, published in Potsdam. Roughly 200 articles (2 000 sentences with 30 000 tokens) were manually annotated for coreference in Manfred Stede's group at the Institut für Linguistik (Department of Linguistics), University of Potsdam.[15]

However, we found that the used annotation scheme was to different from ours (for example, bridging anaphora are not consistently marked as such in the PCC), so that we could not use it in evaluation.

Corpus Description. The Heidelberg Text Corpus (HTC) consists of German texts related to the city of Heidelberg. It has been compiled and annotated at European Media Laboratory GmbH, Heidelberg (Strube et al., 2002; Müller and Strube, 2001a)[16]. The corpus consists of over 200 short articles with 2 000 sentences (roughly 40 000 tokens). Annotation was done with the MMAX annotation tool (Müller and Strube, 2001b) using a stand-off annotation scheme based on words as basic unit. The annotated corpus is stored in an XML format, with markables (here, referring expressions) being anchored to spans of words and related among each other by assigning them to different groups (equivalence classes), additionally marking different types of anaphoric links.

Evaluation Method. Evaluation was done as follows: The PREDS parser with its coreference resolution module was run on the texts in the corpus. From the parser results, all anaphoric links were extracted. As basis for comparison, we reconstructed word spans in the text from the dependency structures of the parser output; this is possible, as the PREDS contains pointers to the tokens in the input text. Gold standard and test corpus were then compared using a small tool that computes recall and precision values for MTSS (Vilain et al., 1995) and for the additional measures described above (7.3.2.1).

For the evaluation, we left out certain types of anaphoric expressions that were not marked either in the gold corpus or by our anaphora resolution module: As our module does not resolve bridging anaphora, we did not consider them, even though a number of bridging anaphora are annotated in the HTC. On the

Stuckardt's definition of precision as a variant *recall* measure rather than a true precision measure, as it does not take false positives into account.

[15]We would like to thank Manfred Stede for permission to use the PCC in evaluation.

[16]We would like to thank Michael Strube at EML for permission to use the HTC for evaluation.

Precision	51.38 %
Recall	50.10 %
F-Score	50.73 %

Table 7.26: Evaluation against the Heidelberg Text Corpus: Recall, Precision and F-Scores according to the Model-Theoretic Scoring Scheme (Vilain et al., 1995)

Type	Correct	Partly correct	Wrong	Missing	Spurious
defnp	223	136	85	286	434
indefnp	1	0	1	17	0
ne	283	93	21	216	65
other	3	1	0	3	0
padv	0	0	0	0	18
pds	7	2	5	6	145
pper	169	125	41	40	48
ppos	118	113	50	25	17
Σ	804	470	203	593	727

Table 7.27: Evaluation against the Heidelberg Text Corpus: Detailed Analysis by Types of Anaphoric Expressions

other hand, we discarded all anaphoric links with sentential antecedents (those are considered as possible antecedents by our coreference resolution module, but such cases are not annotated in the HTC) and relative pronouns (also not annotated in the HTC).

Evaluation Results. The results from the evaluation against the HTC are shown in tab. 7.26 (MTSS measures), tab. 7.27 (Detailed Analysis) and tab. 7.28 (Detailed Percentages).

7.3.2.3 Discussion

Quite generally, the evaluation results exhibit some interesting features: For all different categories and most different types of anaphora, recall and precision figures are relatively close to each other. This suggests that our approach seems to achieve quite a good balance between the two.

	Correct Chain			Correct First Antecedent		
Type	Precision	Recall	F-Score	Precision	Recall	F-Score
defnp	40.89 %	49.18 %	44.65 %	25.40 %	30.55 %	27.74 %
indefnp	50.00 %	5.26 %	9.52 %	50.00 %	5.26 %	9.52 %
ne	81.39 %	61.34 %	69.95 %	61.26 %	46.17 %	52.65 %
other	100.00 %	57.14 %	72.73 %	75.00 %	42.86 %	54.55 %
padv	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
pds	5.66 %	45.00 %	10.06 %	4.40 %	35.00 %	7.82 %
pper	76.76 %	78.40 %	77.57 %	44.13 %	45.07 %	44.59 %
ppos	77.52 %	75.49 %	76.49 %	39.60 %	38.56 %	39.07 %
Σ	57.80 %	61.55 %	59.62 %	36.48 %	38.84 %	37.62 %

Table 7.28: Evaluation against the Heidelberg Text Corpus: Recall, Precision and F-scores for Different Types of Anaphoric Expressions

What can also be observed is that there are marked differences in performance for the different types of anaphora; this is in keeping with other results reported in the literature: Resolving definite descriptions, for example, seems to be quite a hard problem, while recognising and resolving named entities seems to be considerably easier.

We also note that results for the correct first antecedent group are comparatively low. One must keep in mind, however, that to achieve good results under this scoring scheme is a very difficult task for a coreference resolution system, as errors for resolving single anaphoric links accumulate for anaphoric chains.

When comparing the results of our evaluation with that of other coreference resolution systems, a somewhat heterogeneous picture emerges. We will take a closer look.

There are only few coreference resolution systems for German and even fewer detailed evaluation results. We will look at two system evaluations in turn.

The first system is described in Hartrumpf (2001). It builds upon a combination of hand-written rules that use syntactic and semantic information and corpus statistics that provide weights for the rules. The approach was evaluated against a manually annotated corpus of German newspaper texts from *Süddeutsche Zeitung* which contained altogether 502 anaphora. Evaluation was done with the model-theoretic scoring scheme (MTSS, Vilain et al., 1995, cf. 7.3.2.1) using twelve-fold cross-validation using different portions of the cor-

	P	R	F
defNP	69.26 %	22.47 %	33.94 %
NE	90.77 %	65.68 %	76.22 %
PDS	25.00 %	11.11 %	15.38 %
PPER	85.81 %	77.78 %	81.60 %
PPOS	82.11 %	87.31 %	84.63 %
all	84.96 %	56.65 %	67.98 %

Table 7.29: Detailed Evaluation Results for Strube et al.'s German Coreference Resolution System (Strube et al., 2002, their tab. 7)

pus for training and evaluation. Unfortunately, neither is the type of texts further specified nor are the results further differentiated by type of anaphor. The reported overall figures are 55 % recall, 82 % precision and 66 % F-score.

A system that is based on decision trees is described in Strube et al. (2002). The described system was trained and evaluated against the Heidelberg Text Corpus using a ten-fold cross-validation. Thus, the evaluation results reported in this paper are directly comparable with our evaluation results on the HTC, as the same corpus was used. The authors report two sets of results, namely 'standard' MTSS scores and a more detailed set of scores, broken down for different types of anaphora. The reported MTSS scores are: recall 55.14 %, precision 80.17 % and F-score 65.34 %. The detailed scores are re-printed in tab. 7.29.

Interestingly, the reported precision figures (and, hence, F-scores) for the two German systems are considerably higher than those reported for English systems and the difference between recall and precision figures is greater: In Uryupina (2006), the currently best-performing system for English (evaluated on MUC-7 data) is described. The system uses a rich set of linguistic features for machine learning and reaches 63.9 % recall, 67.0 % precision and 65.4 % F-score. Thus, compared with the original MUC-7 competition in 1997, systems for English have improved, but only moderately so: In Soon et al. (2001), the results for all participants of the MUC-7 are summarised that show that no system then exceeded a precision of 80 % and a recall of 60 % on the MUC-7 corpus, and that the best F-scores were then around 60 %.

Neither Hartrumpf nor Strube and his colleagues comment on the difference in recall and precision values. We can therefore only speculate that it may have to do with the richer morphology of German when compared with English that may possibly reduce the number of ambiguous cases. Or it may simply be due to

Minimum		Maximum		Average	
.063 s	(1 word)	92.92 s	(238 words)	2.944 s	(9.51 words)

Table 7.30: Evaluation of Parse Times: Minimum, Maximum and Average Parse Time per Sentence

differences in the used corpora that result in a greater number of comparatively simple cases.

We will now compare the results of our system evaluation with those of the other systems. When comparing recall, precision and F-score computed according to the MTSS, our module performs somewhat below the two other reported systems. The difference is comparatively small for recall (about 5 %), but marked for the precision values (about 30 %), leading to a 15 % difference in F-score: While our results are comparable to other anaphora resolution systems reported in the literature, there is room for improvement.

It is therefore interesting to investigate the more detailed analyses (cf. tabs. 7.27 and 7.28). These show that there are marked differences for the different types of anaphoric expressions. Our results tie in with those of other systems for coreference resolution in that resolving different types of anaphora seems to have very different degrees of difficulty: As for most other approaches, Named Entities receive best results, personal pronouns and possessive pronouns do fairly well, while the resolution of definite NPs and, even more so, demonstrative pronouns turns out to be very hard.

We conclude that an increase in overall system performance could be achieved by further improving the anaphora resolution module or by replacing it altogether with another system. Comparison with the currently best-performing anaphora resolution systems for German shows, however, that only a modest increase can be expected.

7.3.3 Parsing Speed

In this section we report the results of an evaluation of system parse speed. The data was gathered by taking timings during the parsing of the document collection used in the end-to-end evaluation (cf. 7.2). The timings were taken on the same machine as those employed in the evaluation of search times, see 7.2.1.4 above.

Component	Parse Time per Sentence s	%
Topological Parser	0.589	20.0
Named Entity Recogniser	0.839	28.5
NP/PP Chunker	0.869	29.5
PReDS Construction	0.086	2.9
GermaNet Lookup	0.032	1.1
Frame Assignment	0.321	10.9
Anaphora Resolution	0.208	7.1
Σ	2.944	100.0

Table 7.31: Evaluation of Parse Times: Minimum, Maximum and Average Parse Time per Sentence

Table 7.30 shows minimum, maximum and average parse time per sentence in the document collection, together with the corresponding sentence length. The average parse time per sentence was 2.944s, with an average sentence length of 9.51 words.

We were also interested in the question how much of the parse time was spent for the different tasks. The corresponding figures are shown in tab. 7.31. They are averaged to seconds per sentence. The table shows that three modules, namely the topological parser, the named entity recognition module and the NP/PP-chunker each take up roughly similar time (together about 80 % of the overall parse time), with all other modules (PReDS construction, frame assignment, GermaNet lookup and anaphora resolution) using little time in comparison.

The reported parse times are reasonable: On the one hand, small document collections can be processed on a single machine within weeks. For mid-size collections, processing can be distributed over several machines. For example, processing the CLEF collection for German (roughly 80 million words), would take roughly a month on a cluster with ten machines.

Even more interestingly, it allows to parse questions on-line, that is, during a QA session, generally within 1–2 seconds.

7.4 Conclusions

In this chapter, we have reported several evaluation results for the SQUIGGLI system.

We started with some introductory remarks on the evaluation of natural language processing systems in general and QA systems in particular.

We have described an end-to-end performance evaluation of the overall system, consisting of a performance evaluation accompanied by a diagnostic evaluation. As evaluation corpus we used the SmartWeb AnswerGatherer corpus of question-answer pairs that were collected from about 100 users in the form of an Internet experiment, based on texts from the German edition of the Wikipedia.

The results obtained for this corpus were an overall accuracy of 51.5 % (33.6 % for non-NIL questions only), with an answer precision of 66.4 %. These figures show that a good compromise between answer precision and recall can be reached by a system based on the notion of indirect answerhood (5.1).

Individual evaluation of three features of the system related to the user interface, namely answer justification, anaphora and ellipsis resolution on questions and the inclusion of warning messages for uncertain answers, has shown that the proposed methods can be practically realised. The evaluation shows that the use of linguistic structures throughout the question answering process provides a basis to implement additional functionality that can improve the usability of the system in a comparatively easy way. We have not yet tested the user acceptance of the proposed extensions to the user interface, this remains the subject of future research (cf. 8.3.3).

The diagnostic evaluation has shown that all linguistic resources were in fact needed, suggesting that QA systems can profit from an interplay of varied linguistic resources.

The diagnostic evaluation further shows that the most important source of recall errors, i. e, questions for which the system cannot find an answer, is the lack of linguistic resources listing paraphrases that would allow matching the respective question and answer representations using the relation of indirect answerhood (5.1).

Other types of errors are more heterogeneous. We evaluated both the PREDS parser and the anaphora resolution module in a separate component evaluation. The aim was to find out how these components could be improved or replaced and how that would affect overall system performance. We found that for both components that they perform reasonably well. There is room for improvement, but the systems fall not much short of the currently best-performing systems for German. Therefore, only moderate improvements can be expected by replacing them.

Chapter 8

Conclusions

In this chapter, we will summarise the thesis and its results and describe directions for further work.

In 8.1, we will highlight upon the central points of the thesis. In 8.2, we give an overview of the most important results.

In 8.3, we list several directions for further work, which we think can be fruitfully explored in the future. We have identified lack of answer recall as a main challenge of linguistically informed QA (7.2.2.2). For practical use, answer recall needs to be further improved. We have identified missing paraphrases, i. e., systematic information about different ways of expressing one fact and about how they are similar, as the most important single source of recall errors. There are several possible sources from which such information may be derived: Growing lexical resources, especially FrameNet and GermaNet, additional morphological resources and automatic paraphrase acquisition. We will also discuss possible modifications of the search algorithm.

Different features of the user interface, especially answer presentation and answer justification, have so far not been evaluated with respect to their user acceptance. This could be done in a suitable usability evaluation.

Another interesting direction that we have not explored in this thesis is the move from factoid questions to more complex, summary-style questions and answers. We will list some first thoughts for extending linguistically informed QA into this promising direction.

8.1 Summary

In this thesis, we have developed linguistically informed QA as a novel approach to QA. The central concept of the approach is that of indirect answerhood. Indirect answers have been described in the literature as answers to questions that require additional inferences by the questioner (Higginbotham and May, 1981).

We have derived a relation of indirect answerhood in QA. Indirect answerhood holds between a given question and a text in the system's document collection if a direct answer to the question can be inferred from this text. We have described this relation as a compound relation of textual inference and matching questions and direct answers: From a text, a direct answer to a given question can be derived by a set of inference steps and this direct answer can then be matched against the question.

We have specified an architecture for a linguistically informed QA system based upon this concept of indirect answerhood.

To model the relation of indirect answerhood between texts, we have formally specified an approximation, namely a relation of indirect answerhood between structured text representations, in turn composed of the relations of textual inference and direct answerhood between text representations.

Linguistic information plays a central rôle for both compound relations. The relations are based on tree structures, namely syntactic dependency structures extended with lexical semantic information. Textual inference is defined in terms of a relabelling operation on these trees; direct answerhood as embedding question representations in text representations by partial tree matching. On the one hand, linguistic information is used to abstract over variants in the corresponding texts (such as active/passive or lexical semantic variations). On the other hand, it is used as a source of inference rules that allow to match structures that differ from each other.

This definition of indirect answerhood permits to modularly integrate additional linguistic information from different sources: As long as it can be represented in terms of relabelling and matching of trees, it can directly be added to the linguistic knowledge base. We have shown how information from GermaNet and a German FrameNet can be integrated into the linguistic knowledge base.

We have developed an efficient search algorithm from this relation of indirect answerhood, based on a known algorithm for unordered path inclusion (Kilpeläinen, 1992). We further transformed this search algorithm to use a relational database for storage and retrieval.

We have implemented the approach in a prototype QA system for German. The system was tested in a performance evaluation based on question-answer pairs derived in an Internet experiment from the German Wikipedia. We also evaluated a number of additional features that can be used to implement enhanced user interfaces for a QA system.

The evaluation has shown that the system reaches 66 % answer precision and 33 % answer accuracy, which is a useful combined level of performance. It has also shown that answering times are suitable for an interactive usage of the system.

8.2 Results

Let us now quickly recapitulate and summarise what we consider the most important results of this work and the central points of linguistically informed QA.

- We have identified **indirect answerhood** as a central concept for QA. Indirect answerhood was derived from work on questions and answers in linguistics and applied to the context of QA. We have further specified the relation of indirect answerhood for QA, citing examples from a corpus study of questions and answering texts from past QA competitions. We have more formally modelled indirect answerhood as a relation between structured linguistic representations of texts and questions. This relation was further developed into an efficient search algorithm using a relational database. We think that this working out of indirect answerhood contributes to both theoretical and practical research in QA.

- We have shown that indirect answerhood can be used as a basis to address the problems of low answer precision and dependence on answer redundancy of knowledge-lean approaches to QA. We use inferences derived from different sources of linguistic information and structured matching of question and text representations. This combination leads to a solution that provides an interesting level of both **answer recall** and **answer precision** and allows **efficient processing**.

- Linguistically informed QA provides a framework for the modular integration of linguistic resources. Different resources can be straightforwardly integrated into the linguistic knowledge base used in indirect answerhood: If the information contained in a linguistic resource can be expressed in a suitable format (i. e., as local relabelling rules or matching rules), it can be directly incorporated into the knowledge base. This provides a good basis for **scalability** of the approach and also for its transfer to other languages.

- **Efficient on-line answer search** is possible through a search algorithm that is based on the concept of indirect answerhood. Structured linguistic representations for the texts in a document collection are derived off-line and stored in a relational database. Answer search is carried out through matching of structured representations. No additional on-line linguistic processing of the documents is required.
- The optional combination of short answers with the relevant passages from the original document displayed as justification is proposed as an **advanced answer presentation mode** for QA. In combination with additional features, such as anaphora and ellipsis resolution on the users' questions and warning messages for uncertain answers, it provides a perspective for interfaces to QA systems that can be adapted to the requirements of individual users.

8.3 Directions for Future Work

We have shown that linguistically informed QA can be used as the basis for the successful implementation of QA systems. We will now point out some ideas for further improvements of the approach.

We will discuss additional knowledge sources, extensions to the indirect answerhood relation, evaluation of the user interface and the handling of open questions as summarisation.

8.3.1 Additional Knowledge Sources

We have identified missing paraphrases as the most important single source of recall errors in the diagnostic evaluation (7.2.2.2). One possibility of improving system recall would therefore be to systematically integrate information on additional paraphrases.

This solution has the advantage that relatively little needs to be changed in the overall system: Extending existing knowledge sources (for example, increasing the coverage of the GermaNet relations) can be done by simply adding rows to an existing database table. To add a wholly new knowledge source (for example, automatically acquired paraphrase information, see 8.3.1.2), some few lines of code would additionally have to be changed. However, the overall effort required is very low.

It should be noted that this requires that the paraphrase information can be couched in terms of local inferences, that is, relabelling rules that work on elementary trees.

We have described in chapter 4 that we have already integrated the linguistic information from the most important currently available lexical resources for the German language that we are aware of. We currently see two main possible new sources of paraphrase knowledge: On the one hand, more knowledge becomes available with new versions of the resources that we have already used, namely GermaNet and German FrameNet. On the other hand, we expect new lexical resources, probably at least partly based or related to the existing ones, to emerge.

We will first consider the extension of GermaNet and German FrameNet before turning to new resources.

8.3.1.1 Extending Linguistic Resources

In this section we will take a look at the extension of the lexical semantic resources that we have already integrated into the linguistic knowledge base.

GermaNet Coverage. Work on GermaNet continues. At the time of writing, a new release was advertised with an extended coverage. By integrating this and future new releases into the linguistic knowledge base, we expect a noticeably broader coverage.

A simple extension of the covered concepts (synsets and lemmata) would be helpful. However, the more important point would be to systematically increase the coverage of the semantic relations: While the coverage of the synonymy and hyponymy relations is already quite good, an improved coverage for the other relations would provide an interesting extended source for inferences in linguistically informed QA, especially the derivation relations, including pertainymy and participle-of.

Extended GermaNet. We have described Extended WordNet above (4.2.1.2). It was derived semi-automatically from WordNet and especially makes information that was ‘hidden’ in the examples and glosses available to NLP systems.

In principle, it is possible to derive a comparable extended version from GermaNet. To our knowledge there are currently no plans to construct such an ‘Extended GermaNet’ in the close future.

FrameNet Coverage. We have found in the evaluation that currently only the ‘same frame’ relation and a small number of frames could be used as sources for local inferences (7.2.2.1). We believe that this could substantially change with a growing coverage of German FrameNet. There are a number of issues involved.

First, updating to the current version 1.3 of the FrameNet lexical database may already provide a marked increase in coverage, as the frame-to-frame relations have been revised and considerably extended in coverage for this version. As described above (6.2.10), we did not yet carry out this step to ensure compatibility with the annotation data from the Salsa project, which uses version 1.2.

Second, by integrating all information from the first release of the Salsa corpus annotation (we currently included only a part of the data that was available in the form of a pre-release) and then further importing additions from future updates will also increase coverage.

Third, the frame lexicon derived from the Salsa data should be extended with systematic annotation for deverbal nouns.

8.3.1.2 Integrating Additional Resources

Extended Information on Morphological Stems. Relating words with (essentially) identical meanings but different parts of speech, especially deverbal nouns, provides an important source for inferences.

Therefore, by integrating information from an extended morphological resource that links words with similar meanings across parts of speech as paraphrases would already take care of a number of recall errors.

We currently know of no such resource for German. However, it might be possible to construct such a resource semi-automatically by linking words that have similar stems, for example *'kaufen'* (*buy.v*) and *'Kauf'* (*purchase.n*).

Automatically Acquired Paraphrases. We have already mentioned in passing work on the automatic acquisition of paraphrases from large corpora, especially from the Internet (cf. 3.5.2.6). These approaches allow to automatically find possible paraphrases for seed expressions. The examples listed in, e. g., Lin and Pantel (2001a,b) look very similar to the sort of paraphrases that we found to be missing in the evaluation (7.2.2.3).

We are currently not aware of any automatically acquired list of paraphrases for German. We would expect the acquisition to be somewhat more difficult for a language that is morphologically richer than English and has fewer constraints on word order.

We would currently still consider using this approach the probably most promising one for finding new and useful paraphrases, as the automatic acquisition seems a useful way of finding a sort of paraphrases that are not included in manually constructed lexical resources but that are very useful for QA: This is certainly true for examples such as *'N. N. wrote X.'* vs. *'N. N. is co-author of X.'*

8.3.2 Extension of the Indirect Answerhood Relation

In addition to, and possibly complementary with, the extension of the linguistic knowledge sources, it might also be interesting to investigate modifications of the answer matching.

The current definition is based on the notion of unordered path inclusion. That means that every node of the question must be matched and also that the structure must be the same: Two nodes connected by an edge in the question representation must correspond to two nodes in the answer representation, also directly connected by a corresponding edge.

Both these constraints might, in principle, be relaxed. We will discuss these possibilities here.

8.3.2.1 Relaxed Answer Matching: Tree Inclusion

A first possibility for extending the matching process would be to use a less constrained approach to tree matching. Instead of path inclusion, unordered tree inclusion could be used (Kilpeläinen, 1992). Unfortunately, this problem is, at least in general, NP complete and thus probably not suitable as a basis for an efficient search algorithm.

Thus, one would have to add other constraints (for example, limiting the maximum length of paths that may match a single edge might be a starting point) to arrive at a computationally tractable algorithm.

We observed only a small number of examples in the evaluation data where such a 'relaxed' approach would lead to a match, especially cases of semantically more or less transparent noun constructions such as *'Where does X come from?'* that currently does not match the potential answer 'The term X *comes from Y.*'

8.3.2.2 Leaving Out Modifiers

One also could systematically leave out modifiers in the questions that cannot be matched. For example, it is currently not possible to match a question talking about *'Germanist von der Hagen'* against *'von der Hagen'*. While we have argued above that this may be considered the correct behaviour, leaving out the representation of *'Germanist'* might lead to a – possibly even unambiguous – match.

However, this 'loose matching' needs to be controlled in a number of ways. First, it should only be used in cases where no other answer can be found. Second, such a match must be considered uncertain and thus be accompanied with a proper warning message. Third, if more than one modifier is present in the question, it becomes difficult to decide which one(s) to leave out – and in what order.

8.3.2.3 Balancing Recall, Precision and Search Time

While it would be relatively easy to implement these possible modifications of the matching process and thus of the search algorithm, the trade-off must be carefully evaluated.

First, both methods carry the danger of introducing false positives (precision errors). It was through the use of the constrained method of path inclusion that our approach achieved 66 % answer precision.

Then, the relatively constrained matching method allowed us to implement a fast search algorithm. If the search algorithm becomes more complex, search times will considerably increase. As mentioned above, the general unordered tree inclusion problem, for example, is known to be NP complete (Kilpeläinen, 1992).

We are therefore uncertain whether it will be possible to find a usefully constrained way of matching trees that is expressive enough to account for the cases mentioned above on the one hand and constrained enough to ensure precision and efficient matching.

8.3.3 Evaluation of the User Interface

We have described a number of features intended at making the system more user-friendly, namely focussed justifications, anaphora resolution for questions and the combination of uncertain answers with warnings. While the evaluation has shown that these features work well in the prototype system, we have not tested them for user acceptance. Usability tests would have to be designed and carried out to find out whether users would actually prefer these advance features. We have summarised some thoughts on the evaluation of interactive QA systems in Fliedner (2004b).

8.3.4 Answering Open Questions Through Summarisation

None of the current linguistic accounts of questions and answers have much to say about the issue of open questions (3.2.2.5). We have pointed out that ‘openness’ is less a quality of a question as such, but is rather a combination of question, answer and situation, i. e., pragmatic considerations.

We think that it would be interesting to investigate questions that require a more complex, ‘summary-style’ answer. Different issues are involved here that should be clearly separated.

The first problem is that the automatic recognition of such questions is a demanding task (cf. 3.2.2.5). However, it might be an interesting option that a user can explicitly set a preferred answer length and thus switch between ‘constituent’ and ‘summary’ answers.[1]

The second problem is the generation of summary-style answers. Current text summarisation systems mostly combine snippets from the original text based on information retrieval measures (Dang, 2005). This general approach is probably to coarse-grained to answer complex questions.

We think that linguistically informed QA may provide an interesting new approach in the following way: Many questions requiring summary-style answers may be systematically broken down into simpler, factoid questions. For example, a summary of a person’s biography can typically be constructed from answers to the following questions:

(8.1)
a. When and where was N. N. born?
b. What was N. N.’s education?
c. In what job did N. N. work and where?
d. What was N. N.’s most famous work?
e. What famous invention did N. N. make?
f. . . .

Note that these questions will often have no answer for one particular person or will depend upon each other (for example, if we find out that N. N. was a composer, it makes sense to ask about famous pieces of music, whereas, if N. N. was an inventor, asking for famous inventions is probably more appropriate).

A linguistically informed QA system could try to answer the different questions one by one. Using a generation module in sentence generation mode, these answers could then be more or less directly concatenated to make up a – possibly somewhat boring, but factually correct – biography. The availability of linguistic structures for the answers is especially advantageous, as it can potentially help with tackling a number of problems of summarisation, such as avoiding redundancy or generating anaphoric references.

The third problem is whether it is possible to define or (semi-) automatically acquire such a breakdown of summary-style questions into underlying sets of simpler questions. In principle, FrameNet, could provide a source of such information. Fillmore and Baker (2001) gives an example of text that is analysed in terms of the CRIMINAL_PROCESS frame with a number of subframes. Once

[1]The use of a QA system for summarisation which uses a long task description instead of a short, single question is described in Mollá and Wan (2006).

this is done, a summary of the criminal process might be produced by generating a description of the subframes in the correct order.

The general idea has been discussed in Narayanan and Harabagiu (2004a,b); the authors report that first results were promising. However, as currently there is no resource that provides a broad coverage of the required sort of information. FrameNet currently contains only a small number of worked-out ‘scenario style’ frames with instantiated relations to all subframes.

Appendices

Appendix A

Sample-XML Output of the PREDS-Parser

This appendix contains an abridged XML structure as output by the PREDS-parser. For an explanation, see 6.2.11.

```
<sent ID='1'>
  <string> E-Plus Mobilfunk GmbH, Düsseldorf: </string>
  <Preds>
    <word stem='_' ref='1' pos='verb' start='1' end='5'>
      <attrs> <noSurface/> </attrs>
      <DSub>
        <word ref='2' type='Company' pos='NE'
string='E-Plus Mobilfunk GmbH , Düsseldorf' start='1'
end='5'>
          <NEContent>
            <Company>
              <string> E-Plus Mobilfunk GmbH ,
Düsseldorf</string>
              <organization>
                <orgname> E-Plus Mobilfunk</orgname>
                <form> GmbH</form>
                <LocName> Düsseldorf</LocName>
              </organization>
            </Company>
          </NEContent>
          <attrs> <fem_sg/> <sg/> </attrs>
          <GermaNet gn_pos='noun'>
            <sense id='1'>
```

```
                <Animate/>
                <Organisation/>
              </sense>
            </GermaNet>
          </word>
        </DSub>
      </word>
    </Preds>
</sent>
<sent ID='2'>
  <string> Das Unternehmen hat 1997 die Zahl seiner Kunden
auf über eine Million verdoppelt.</string>
  <Preds>
    <word stem='verdoppeln' ref='3' end='13' pos='verb'
start='1'>
      <attrs> <sg/> <ind/> <past/> </attrs>
      <PPMod>
        <word stem='_' ref='11' end='4' pos='praep'
start='4'>
          <arg>
            <word ref='5' string='1997' end='4'
type='Year' pos='NE' start='4'>
              <attrs> <neutr_sg/> <sg/> </attrs>
              <NEContent>
                <Year>
                  <string> 1997</string>
                  <year> 1997</year>
                </Year>
              </NEContent>
              <GermaNet gn_pos='noun'>
                <sense id='1'>
                  <Time></Time>
                </sense>
              </GermaNet>
            </word>
          </arg>
        </word>
      </PPMod>
      <DSub>
        <word stem='Unternehmen' ref='4' end='2'
pos='noun' start='1'>
          <attrs> <defArt/> <neutr_sg/> <sg/> </attrs>
          <GermaNet gn_pos='n'>
            <sense id='1'>
            <Event/>
          </sense>
          <sense id='2'>
            <Place/> <Organisation/> <Animate/>
<Inanimate/> <Object/>
          </sense>
        </GermaNet>
        <DefAnte value='105'>
          <word ref='2' string='E-Plus Mobilfunk GmbH ,
Düsseldorf' end='5' type='Company' pos='NE' start='1'/>
        </DefAnte>
      </word>
    </DSub>
    <DObj>
      <word stem='Zahl' ref='6' end='6' pos='noun'
start='5'>
     <attrs> <defArt/> <fem_sg/> <sg/> </attrs>
        <GenMod>
          <word stem='Kunde' ref='7' end='8' pos='noun'
start='7'>
            <attrs> <detArt/> <defArt/> <mask_pl/>
<fem_pl/> <pl/> </attrs>
            <GenMod>
              <word stem='er' ref='8' end='7'
pos='noun' start='7'>
                <attrs> <det/> <pron/> <mask_sg/>
<neutr_sg/> </attrs>
                <PronAnte value='308.333333333333'>
                  <word stem='Unternehmen' ref='4'
end='2' pos='noun' start='1'/>
                </PronAnte>
              </word>
            </GenMod>
            <GermaNet gn_pos='n'>
              <sense id='1'/>
              <sense id='2'/>
              <sense id='3'/>
            </GermaNet>
          </word>
        </GenMod>
        <GermaNet gn_pos='n'>
          <sense id='1'></sense>
          <sense id='2'></sense>
          <sense id='3'></sense>
          </GermaNet>
        </word>
      </DObj>
      <PP>
        <word stem='auf' ref='9' end='12' pos='praep'
start='9'>
          <attrs> <akk/> </attrs>
          <arg>
            <word ref='10' string='über eine Million'
end='12' type='bigNumber' pos='NE' start='10'>
              <attrs> <pl/> </attrs>
              <NEContent>
                <bigNumber>
                  <string> über eine Million</string>
                  <amount>
                    <modifier> über</modifier>
                    <value> eine</value>
                    <scale> Million</scale>
                  </amount>
                </bigNumber>
              </NEContent>
            </word>
          </arg>
        </word>
      </PP>
    </word>
  </Preds>
   <FrameNet>
    <FNReading preds='1' id='1'>
      <FrameInst value='40' id='1'>
        <Frame>Calendric_unit</Frame>
        <FEE>
          <word ref='5' string='1997' end='4'
start='4'></word>
        </FEE>
        <FEs>
          <FE name='Name'>
            <word ref='5' string='1997' end='4'
start='4'></word>
          </FE>
        </FEs>
      </FrameInst>
      <FrameInst value='40' id='2'>
        <Frame>Businesses</Frame>
```

```
          <FEE>
            <word stem='Unternehmen' ref='4' end='2'
start='1'></word>
          </FEE>
          <FEs>
            <FE name='Business'>
              <word stem='Unternehmen' ref='4' end='2'
start='1'></word>
            </FE>
          </FEs>
        </FrameInst>
        <FrameInst value='81' id='11'>
          <Frame>Cause_change_of_scalar_position</Frame>
          <FEE>
            <word stem='verdoppeln' ref='3' end='13'
start='1'></word>
          </FEE>
          <FEs>
            <FE name='Time'>
              <word ref='5' string='1997' end='4'
start='4'></word>
            </FE>
            <FE name='Item'>
              <word stem='Zahl' ref='6' end='6'
start='5'></word>
            </FE>
            <FE name='Value_2'>
              <word ref='10' string='über eine Million'
end='12' start='10'></word>
            </FE>
            <FE name='Agent'>
              <word stem='Unternehmen' ref='4' end='2'
start='1'></word>
            </FE>
          </FEs>
        </FrameInst>
      </FNReading>
    </FrameNet>
    <Relations>
      <Reading preds='1' id='1'>
        <RelInst value='1' id='12'>
          <Relation>TIME</Relation>
          <Mother>
          <word stem='verdoppeln' ref='3' end='13'
start='1'></word>
        </Mother>
        <Daughters>
          <Daughter name='Time'>
            <word ref='5' string='1997' end='4'
start='4'></word>
          </Daughter>
        </Daughters>
      </RelInst>
    </Reading>
  </Relations>
</sent>
```

Index

accuracy, 20, 35, 114, 118, 195, 196, 235, 253, 256, 271, 281, 309, 312, 315, 316, 324, 325, 327, 335, 346, 348, 368, 371
Advanced Question Answering for Intelligence, 20, 116, 126
alternative question, 44–46, 86, 93
anaphor, 126, 133, 134, 231, 274, 276, 278, 301, 302, 342, 360, 361, 365
  pronominal, 305
anaphora, 3–5, 16, 21, 90, 126, 132–135, 138, 143, 231, 274–277, 279, 288, 292, 299–305, 307, 309, 316, 330–332, 338–346, 349, 359–368, 372, 376
  bridging, 133–135, 279, 339, 341, 342, 362
  definite noun phrase, 133
answer, 1–5, 8, 13, 14, 18–21, 23–44, 46–74, 76, 78–98, 100–105, 107, 108, 110, 113, 115, 116, 119–126, 129–141, 143–145, 151, 157, 166, 171, 178–180, 182, 186, 188–193, 195, 197, 200, 203, 205, 207, 210, 213, 215, 216, 218–222, 224–226, 228–237, 241, 243, 245, 247, 250–253, 259, 273, 289, 292–297, 299–307, 309, 311–320, 322–335, 337–346, 348, 349, 355, 368–372, 375–377
  constituent, 26, 43, 46, 54, 57–61, 88, 93, 123, 136, 295, 297–299, 323, 332
  direct, 33, 38, 51, 52, 66, 70, 80, 88, 89, 113, 126, 136, 213, 370
  indirect, 33, 34, 63–66, 68, 83, 88, 93, 101, 103, 105, 113, 124, 144, 370
answer congruency, 46, 93, 296, 307
answer extraction, 23–28, 31, 32, 200, 321
answer generation, 27, 93, 94, 150, 229, 232, 247, 290, 296, 297, 300, 325
answer justification, 3, 4, 37, 92, 103, 104, 293, 294, 304, 305, 309, 313, 316, 327–330, 368, 369, 372, 376
answer precision, 1, 3, 30, 31, 315, 321, 325–327, 333, 334, 346, 348, 368, 371, 376
answer projection, 25, 325

answer recall, 25, 28, 31, 309, 313, 315, 326, 327, 332, 337, 369
AnswerGatherer, 293, 309, 314–317, 330, 335
AnswerGatherer corpus, 314, 316, 331, 335, 368
answerhood, 1, 2, 4, 41, 48, 49, 55, 56, 61–63, 65, 66, 68, 88, 96, 101, 105, 121, 122, 143, 144, 177–180, 189, 191, 200, 216, 218–221, 370
  indirect, 2–4, 32–34, 36, 42, 65, 89, 114, 115, 117–122, 124, 134, 135, 140, 143–145, 147, 150, 151, 163, 176–180, 183, 193, 201–203, 205, 215, 218, 219, 222, 224, 225, 229, 236, 244, 245, 247, 349, 368, 370–372
antonymy, 128, 154, 155, 160, 204, 210
AQUAINT, *see* Advanced Question Answering for Intelligence
ASR, *see* Automatic Speech Recognition
Automatic Speech Recognition, 21
axiom, 27, 66, 98, 100, 107, 112, 119, 148, 157, 173–175

Boolean model, 9
bootstrapping, 16–18
bridging anaphora, 133–135, 279, 339, 341, 342, 362

candidate document analysis, 23, 26, 31
candidate document selection, 23, 25, 27
Categorial Approach, 48, 53, 57, 60, 61
CLEF, *see* Cross-Linguistic Evaluation Forum
COGEX, 118
competition, 1, 3, 4, 12, 13, 16, 19, 20, 24, 29, 34, 113, 118, 120, 126, 145, 237, 247, 248, 275, 280, 281, 301, 309–312, 315, 317, 320, 323, 324, 326, 328, 335, 365, 371
complexity, 99, 106, 195, 200, 220–224, 228, 232, 234, 236, 239, 243, 357
component evaluation, 5, 309, 342, 349, 359, 368
compound, 109, 127, 128, 143, 206, 207, 225, 258, 266, 270, 272, 274, 283, 339, 342, 370
conceptual cover, 68
Conceptual Dependency, 77, 97, 98
conjoined question, 57
consistency, 108, 110
constituent answer, 26, 43, 46, 54, 57–61, 88, 93, 123, 136, 295, 297–299, 323, 332
corpus, 3–5, 16–18, 26, 32, 34, 116, 118–120, 125–127, 129, 130, 132, 137, 139, 143, 145, 148, 149, 161, 164, 167–171, 177, 203, 205, 247, 248, 256, 260, 264, 265, 271, 279, 282, 309, 311, 320, 325, 350–362, 364, 365, 368, 371, 374
coverage, 2, 14, 15, 49, 97, 98, 104, 105, 109, 111, 112, 119, 167, 169–171, 207, 213, 215, 247, 250, 251, 253, 259, 271, 282, 298, 307,

338, 340, 342, 349, 357, 358, 372–374, 378
Cross-Linguistic Evaluation Forum, 12, 115, 120, 125, 126, 128, 143, 145, 237, 248, 309, 310, 315, 320, 323, 324, 328, 335, 367
Cyc, 100, 111, 112, 148, 172–175

database, 2, 3, 5, 8, 11, 13–16, 18, 23, 25, 28, 31, 32, 50, 52, 94, 112, 141, 147, 151, 153, 158, 161, 164, 167, 170, 172, 175, 178, 212, 213, 215, 237–241, 243, 244, 246, 253, 258, 273, 288–292, 296, 297, 299, 302, 303, 306, 307, 323, 334, 341, 343–345, 349, 370–372, 374
declarative, 40, 41, 44, 45, 48, 61, 75, 93, 115, 122, 135
definite noun phrase, 277
definite noun phrase anaphora, 133
Dependency Grammar, 147–150, 175, 350
dependent, 11, 12, 26, 40, 44, 56, 74, 81, 148–150, 163, 174, 202, 272, 273, 276, 298, 299, 354
derivation, 11, 54, 62, 87, 154, 155, 198, 206, 210–212, 241, 259, 283, 373
Description Logic, 100, 107, 108
diagnostic evaluation, 5, 309, 313, 314, 335, 340, 368, 372
diathesis, 127, 143, 199
Direct Answerhood, 120–124, 135–137, 143, 144, 177, 178, 184, 188–191, 201, 202, 219, 220, 225, 226, 244, 370
distinctness, 50–53, 90, 92
DL, *see* Description Logic
document, 1–4, 8–16, 18, 19, 23–29, 31, 32, 36–39, 41, 49, 88–90, 92–96, 106, 108, 110, 111, 113, 119, 126, 144, 170, 180, 194–196, 199, 200, 218–220, 222, 224, 226, 230, 232–234, 236, 243, 246–248, 250, 252, 258, 289, 290, 296, 300–305, 307, 309–312, 314–318, 320–323, 325, 326, 328, 334, 338, 341, 345, 349, 350, 357, 366, 367, 370, 372
document retrieval, 9, 20

echo question, 57
edge, 32, 118, 149–151, 169, 180–186, 189, 190, 194–198, 200, 203, 206, 208–212, 215, 216, 218, 219, 221, 224, 226, 228–237, 241–244, 265, 272, 282, 294, 305, 337, 375
efficiency, 108, 113, 197, 200, 219, 224, 241
embedded question, 62, 69
English, 12, 41, 43–45, 47, 59, 86, 120, 127, 148, 158, 159, 161, 167–170, 203, 205, 207, 256, 275, 281, 294, 297, 301, 312, 328, 331, 335, 354, 357, 365, 374
Erotetic Logic, 48–50, 52
evaluation, 3–5, 12, 13, 17, 19, 24, 34, 35, 38, 44, 77, 111, 112, 114, 116–118, 167, 170, 171, 193, 197, 204, 216, 235, 256, 258, 260, 273,

274, 280, 281, 291, 292, 301, 309–317, 320, 323–325, 331, 334, 335, 338, 341, 342, 345, 348–351, 353–359, 362–366, 368, 369, 371–376
exhaustiveness, 69, 82, 92
Extended WordNet, 27, 119, 153, 157, 158, 373

F-score, 256, 260, 264, 265, 275, 281, 325, 326, 335, 354–356, 359, 360, 363–366
factoid question, 20, 21, 29, 125, 136, 311, 312, 369, 377
First-Order Predicate Logic, 50, 51, 100, 101, 103, 106–109, 173, 175
FOPL, *see* First-Order Predicate Logic
frame, 42, 147, 156, 160–172, 175, 181, 186, 187, 198, 201–203, 205, 212–214, 216, 218, 244, 246, 251, 252, 256, 257, 271, 272, 280–285, 288, 307, 336, 338, 341, 349, 367, 373, 374, 377, 378
frame element, 162–165, 169, 171, 172, 186, 198, 209, 212, 213, 256, 280–285, 288, 290, 349
frame-to-frame relation, 161, 164, 166, 167, 170, 206, 212–215, 374
FrameNet, 2–5, 112, 147, 161–164, 166–171, 173, 175–177, 179, 180, 186, 187, 193, 195, 198, 199, 205, 209, 212, 213, 215, 217, 218, 245, 255, 256, 280, 282, 283, 289, 290, 339, 369, 370, 373, 374, 377, 378

German, 2, 4, 5, 37, 40, 41, 44–47, 58, 59, 66, 86, 93, 126–128, 134, 147–150, 153, 158, 161, 167–171, 173, 176, 178, 180, 194, 201, 203, 205, 207, 215, 244, 245, 248, 255–260, 263–265, 269–271, 275, 281, 282, 293, 297, 298, 301, 306, 309, 314, 317, 318, 335, 339, 342, 344, 350, 353, 354, 358, 359, 362, 364–368, 370, 371, 373, 374
GermaNet, 2–5, 129, 132, 147, 153, 158–161, 167, 171, 175–177, 179, 180, 193, 195, 201, 205, 207–212, 215–218, 244–246, 255, 257, 273, 274, 277, 282–285, 289, 290, 294, 305, 307, 331, 333, 337, 338, 343, 345, 367, 369, 370, 372, 373
Gertwol, 259, 265, 298
Google, 9, 12, 20, 25, 248
GR, *see* Grammatical Relation annotation scheme
Grammatical Relation annotation scheme, 205, 294, 351–357
graph, 37, 98, 194, 196–200, 219–222, 224, 229, 230, 232, 237, 238, 240, 241, 243, 306
graph matching, 118, 197, 200, 219, 220, 224, 228
Gricean maxim, 66, 74, 79, 90, 92

HALO, 103, 108, 110

head, 26, 27, 41, 134, 137, 148, 150, 188, 194, 202, 206, 211, 254, 263, 266, 270, 272, 277–279, 281, 282, 290, 350, 351, 354, 357, 359
Head-driven Phrase Structure Grammar, 41, 247, 253, 257, 281
Higher-Order Logic, 106, 107
holonymy, 155, 156, 160
HPSG, *see* Head-driven Phrase Structure Grammar
hypernymy, 128, 154, 156, 160, 161, 196, 209, 336, 338, 343, 345
hyponymy, 119, 137, 154–157, 159, 164, 166, 204, 208, 210, 211, 213, 277, 294, 305, 336, 338, 373

IA, *see* Information Access
IE, *see* Information Extraction
index
  inverted file, 9
indexing, 9, 11, 12, 23, 108, 234, 241, 247, 282, 291
inference, 2–5, 33, 34, 65, 66, 84, 88, 89, 92, 96, 99–101, 105, 111, 112, 114–118, 120–126, 128–130, 139–145, 160, 166, 167, 175, 178, 179, 183–185, 187, 192, 193, 196, 197, 199–201, 203, 205, 206, 210, 212–216, 219, 226, 231, 235, 244, 250, 251, 289, 305, 332, 333, 339, 340, 342, 345, 370–374
inference rule, 2, 4, 66, 89, 96, 100, 101, 145, 157, 175, 193–195, 206, 207, 209, 210, 213, 289, 339, 342, 370
Information Access, 4, 7, 8, 32, 117, 193, 194, 314, 315
Information Extraction, 4, 7, 8, 13, 16, 18, 23, 25, 27, 32, 250
Information Retrieval, 1, 3, 4, 7–13, 18–20, 23, 24, 27–29, 31, 32, 35, 36, 96, 157, 179, 195, 198, 200, 247, 248, 323, 334, 377
Internet, 8, 9, 12, 20, 25, 28, 118, 140–142, 164, 193, 248, 253, 279, 292, 293, 309, 314, 315, 317, 320, 340, 368, 371, 374
interrogative, 40, 41, 44, 47, 48, 60, 62, 74–77, 82, 84–86, 93, 115, 122, 135, 231, 260, 270, 272, 294–296
Intranet, 8, 20
inverse relation, 131, 132, 156, 186, 214, 225
IR, *see* Information Retrieval

knowledge representation, 100, 103–105, 107, 142
Knowledge Representation and Reasoning, 100, 101, 105, 113, 116
KR&R, *see* Knowledge Representation and Reasoning

lemma, 150, 156, 157, 159–161, 167, 168, 171, 195, 197, 198, 205, 209, 212, 217, 218, 225, 271–273, 289, 305, 352, 353, 357, 373
Lexical Functional Grammar, 198, 252, 282
lexical semantics, 36, 96, 97, 99, 120, 128, 153, 165, 167, 175

LFG, *see* Lexical Functional Grammar
Lingsoft, 259
Linguistically Informed Question Answering, 2, 3, 5, 28, 30, 32–36, 71, 85, 86, 91, 115, 119, 124, 143–145, 147, 148, 167, 172, 173, 177, 178, 237, 244, 245, 247, 248, 250, 252, 296, 300, 306, 332, 338, 369–373, 377
LUNAR, 13, 19

matching
  graph, 118, 197, 200, 219, 220, 224, 228
  tree, 2, 4, 177, 195, 196, 200, 219, 222, 224–226, 228, 244, 370, 375
matching question, 3, 26, 42, 56, 57, 72, 123, 129, 151, 181, 216, 229, 244, 245, 250, 299, 338, 340, 355, 370
meronymy, 156, 160, 204, 210, 277, 280
message understanding, 16, 275
Message Understanding Conference, 16–18, 264, 275, 365
model generator, 102
Model-Theoretic Scoring Scheme, 359, 360, 362–366
modifier, 42, 131, 136, 149, 150, 206, 207, 211, 265, 272–274, 279, 298, 352, 354, 355, 375
morphology, 11, 258–260, 263, 265, 294, 298, 337, 339, 342, 365
MTSS, *see* Model-Theoretic Scoring Scheme
MUC, *see* Message Understanding Conference
Multi Word Expression, 129
MURAX, 19
MySQL, 237, 240, 289, 291, 292, 333, 334, 341

Named Entity, 5, 8, 17, 23, 24, 26, 27, 29, 118, 134, 201, 208, 229, 245, 250, 257, 258, 260, 263–265, 274, 276, 277, 290, 294, 299, 302, 305, 306, 330, 331, 336, 337, 339, 340, 344, 363–367
Named Entity Recognition, 12, 16–18, 25, 28, 31, 36, 260, 263–266, 275, 281, 338, 340, 342, 348, 349, 367
Natural Language Database Frontends, *see* Natural Language Interfaces to Databases
Natural Language Interfaces to Databases, 7, 8, 13–16, 19, 32, 74, 79
NE, *see* Named Entity
NER, *see* Named Entity Recognition
NII-NACSIS Test Collection for Information Retrieval, 12, 301, 311, 312
NLIDB, *see* Natural Language Interfaces to Databases
node, 149–151, 169, 180–186, 188–190, 192, 194–198, 203, 206, 209, 210, 212, 216–218, 221, 223–226, 228–237, 241–244, 270–274, 282–284, 295, 298, 299, 305, 334, 336, 337, 350, 375

NTCIR, *see* NII-NACSIS Test Collection for Information Retrieval

ontology, 100, 104–106, 111, 112, 142, 148, 161, 172–176
open question, 70, 71, 86, 87, 144, 252, 311, 319, 372, 376
orthography, 298

paragraph retrieval, 20
paraphrase, 43, 118, 119, 140, 142, 196, 199, 201, 206, 338–340, 342, 348, 349, 368, 369, 372–374
paraphrase acquisition, 140, 142, 340, 349, 369, 374
parser, 5, 27, 46, 47, 86, 111, 118, 119, 168, 183, 190, 207, 245, 250, 252–260, 263, 267, 269, 271, 280–282, 285, 290, 293, 298, 309, 339–345, 349, 350, 354, 357–359, 362, 367, 368
parsing, 5, 17, 26, 27, 31, 86, 111, 150, 166, 182, 186, 187, 190, 245, 246, 248, 250–255, 257, 258, 260, 263, 267, 269, 275, 276, 279, 284, 285, 290, 307, 325, 329, 348–350, 359, 366
Partially Resolved Dependency Structures, ix, 46, 147–152, 171, 175, 180, 186, 206, 207, 265, 267, 269–272, 274, 276, 282–284, 290, 294, 297, 298, 309, 341, 342, 349–351, 354, 357–359, 362, 367, 368, 381
participle-of relation, 210, 373
Partition Approach, 33, 48, 52, 63–65, 67–70, 81, 84, 90, 96
passage retrieval, 9, 25
path inclusion, 222–224, 230, 232, 375, 376
  unordered, 2, 3, 5, 177, 200, 219, 222, 223, 225, 226, 229, 232, 236, 244, 334, 370, 375
pattern, 17, 23, 25, 26, 28, 29, 31, 55, 96, 113, 126, 138, 140, 142, 200, 215, 222, 224, 230, 250, 264, 277, 285, 294, 323, 325
performance evaluation, 5, 309, 313, 314, 335, 358, 368, 370
Perl, 258, 289, 333
pertainymy, 132, 155, 160, 210, 373
possessive, 119, 127, 132, 133, 199, 276, 288, 299, 330, 366
pragmatics, 39, 40, 52, 69, 70, 74, 78, 80, 88, 144
precision, 4, 23, 25, 28, 29, 32, 113, 143, 170, 200, 208, 260, 264, 265, 273, 280, 281, 309, 312, 313, 315, 325, 326, 331, 333, 339, 345, 349, 354–356, 358, 360–366, 376
PReDS, *see* Partially Resolved Dependency Structures
presupposition, 69, 70, 73, 74
Propositional Approach, 48, 55, 61, 63

QA, *see* Question Answering
QA@CLEF, 19, 20, 126, 194, 248, 253, 311, 312, 317
query, 8–15, 23–25, 32, 36, 37, 52, 178, 180, 237–241, 243, 246, 253, 288–292, 302,

303, 306, 334, 339, 341, 343–345
query expansion, 11, 12, 24, 96, 113, 195, 225, 226
question
  alternative, 44–46, 86, 93
  conjoined, 57
  echo, 57
  factoid, 20, 21, 29, 125, 136, 311, 312, 369, 377
  matching, 3, 26, 42, 56, 57, 72, 123, 129, 151, 181, 216, 229, 244, 245, 250, 299, 338, 340, 355, 370
  open, 70, 71, 86, 87, 144, 252, 311, 319, 372, 376
  quiz, 57
  wh, 41–46, 49, 53, 54, 56–61, 64, 84–87, 90–93, 95, 102, 187, 262, 302, 323, 325, 337
  yes/no, 43–46, 48, 53, 60, 64, 85, 93, 101, 262, 300
question analysis, 23, 24, 26, 28, 32, 39, 47, 85
Question Answering, 1–5, 7, 8, 13, 15, 18–21, 23–25, 27–41, 43, 46, 47, 49, 53, 57, 60, 62, 66, 77, 79, 85, 87–106, 108, 112, 113, 115, 117, 120, 122, 123, 125–127, 139, 142, 144, 157, 159, 166, 167, 173, 175–177, 180, 193–195, 200, 207, 208, 212, 215, 216, 218, 221, 222, 237, 241, 245–247, 250, 252, 253, 273, 289, 294, 300, 301, 304, 306, 307, 309–317, 321, 323–328, 330–332, 334, 335, 349, 367, 368, 370–372, 374, 376
question type, 20, 24–26, 40, 41, 43, 45, 46, 54–56, 60, 61, 63, 77, 85, 124, 136, 318, 335, 348
quiz question, 57

reasoning, 2, 4, 14, 21, 27, 77, 95–97, 100, 101, 103, 105, 106, 108, 110, 112, 113, 119, 140, 144, 172, 174, 305
recall, 4, 12, 31, 80, 82, 108, 113, 133, 191, 200, 208, 243, 247, 260, 265, 271, 273, 280, 281, 283, 309, 312, 315, 316, 325, 326, 331–334, 338, 342, 344, 345, 349, 354–356, 358, 360–366, 368, 369, 372, 374
Recognising Textual Entailment, 34, 113–119, 122, 193, 194, 196, 197, 199
relevance, 9, 10, 12, 96, 115, 124, 178–180, 184–187, 189, 190, 196, 201, 203, 205, 207, 208, 210, 211, 213, 214, 216–218, 225, 232, 235, 236, 241, 244, 293–296, 303–305, 313, 332, 337
response, 36, 37, 40, 74, 79, 80, 92–94, 306, 313, 316, 334
retrieval, 1, 2, 5, 8, 11, 12, 35, 236–238, 240, 247, 289, 291, 334, 370
retrieval model, 9, 10, 29, 180
RTE, *see* Recognising Textual Entailment

SAarbrücken Lexical Semantics Annotation and analysis, 161, 167–171, 281, 282, 374
SALSA, *see* SAarbrücken Lexical Semantics Annotation and analysis
scalability, 15, 105
search term, 9, 11, 96, 321
search time, 234, 313, 366, 376
semantics, 39, 40, 48, 60, 63, 67, 69–71, 75, 78, 80, 81, 83, 88, 96, 99, 107, 110, 111, 150, 162, 164, 213, 257, 269, 275, 281, 349
SmartWeb, 293, 309, 314, 316, 317, 330, 335, 368
speech act, 40, 44, 74–76, 84, 85
SQL, *see* Structured Query Language
SQUIGGLI, v, vii, 2, 5, 47, 161, 170, 178, 179, 183, 193, 245, 246, 306, 309, 314, 323, 334, 349, 350, 368
stem, 11, 181, 186, 200, 259, 298, 374
stemming, 11, 23, 28, 96, 316
stop-word, 11, 23, 28
summary, 32, 87, 239, 253, 256, 318, 319, 339, 369, 376–378
SUMO, 112, 148, 161, 172–175, 198
synonymy, 119, 128, 143, 147, 153–156, 161, 183, 199, 209–211, 213, 277, 336, 338, 373
syntax, 29, 35, 39, 40, 90, 92, 111, 118, 143, 150, 169, 195, 198, 254, 275

term, *see* search term
text, 1, 2, 4, 8, 12, 14, 16, 17, 27, 29, 32–34, 36, 37, 42, 49, 55, 85, 86, 88, 90, 91, 96–98, 103–107, 109–115, 117–122, 124–126, 128–134, 136, 139, 141, 143–145, 147–151, 158, 161, 166–170, 175, 177, 178, 180–187, 190, 194–200, 202, 206–208, 212, 216, 219–222, 228, 229, 232, 237, 241, 244–248, 250–252, 258–260, 264, 265, 271, 274–276, 280, 281, 288, 290, 293, 295–299, 301–304, 311, 314, 316, 317, 319, 320, 323, 328, 331, 335, 339–343, 345, 350, 357–360, 362, 364, 365, 368, 370–372, 377
text representation, 2, 34, 98, 145, 148, 151, 168, 170, 175, 177–193, 196, 201, 203, 206, 207, 216, 219–222, 225, 226, 228, 230, 232, 237, 240, 241, 244–248, 259, 280, 281, 288–290, 296, 300, 301, 307, 342, 370, 371
Text Retrieval Conference, 12, 19–21, 23, 29, 30, 34–36, 91, 93, 115, 120, 125–127, 129–143, 145, 195, 216, 237, 301, 309–312, 315, 320, 323–325, 327, 328, 335
text summarisation, 8, 377
Textual Entailment, 34, 113–115, 118, 122, 196
Textual Inference, 115–121, 123–125, 128–136, 139, 143–145, 151, 170, 173, 177–179, 183, 184, 186, 187, 193, 194, 200–202, 208, 215, 216, 218, 219, 224–226,

231, 232, 235, 236, 244, 340, 370
TF/IDF, 10
theorem prover, 27, 101–103, 108, 110, 113, 118, 119, 157
TREC, *see* Text Retrieval Conference
tree, 2–4, 17, 137, 149–151, 155, 169, 177, 178, 180–185, 191, 192, 195, 196, 198–200, 202, 203, 206, 208, 209, 216, 218, 219, 221–223, 225, 229, 230, 234–238, 243, 275, 281–283, 285, 288, 290–292, 305, 365, 370, 372, 376
tree inclusion, 222
    unordered, 285, 375, 376
tree matching, 2, 4, 177, 195, 196, 200, 219, 222, 224–226, 228, 244, 370, 375
troponymy, 154, 155

uncertain answers, 121, 123, 124, 139, 178, 190, 191, 193, 300, 309, 313, 316, 332, 333, 345, 368, 372, 376
unordered path inclusion, 2, 3, 5, 177, 200, 219, 222, 223, 225, 226, 229, 232, 236, 244, 334, 370, 375
unordered tree inclusion, 285, 375, 376
user, 1–5, 7–16, 18, 20, 21, 23, 24, 27, 33, 36–39, 42, 49, 52, 79, 84–86, 88–90, 92–95, 98, 103, 113, 117, 123, 124, 179, 180, 191, 210, 238, 246, 248, 252, 253, 290, 292, 293, 296, 297, 299–307, 309, 310, 312–320, 323, 324, 326–331, 333, 334, 341, 344, 345, 348, 368, 369, 371, 372, 376, 377
user model, 21, 36, 84, 94, 95

vector model, 9, 10

wh-question, 41–46, 49, 53, 54, 56–61, 64, 84–87, 90–93, 95, 102, 187, 262, 302, 323, 325, 337
Wikipedia, 5, 293, 295, 296, 309, 314, 317–321, 323, 341, 344, 346, 368, 371
WordNet, 2, 4, 11, 24, 27, 108, 112, 118, 119, 125, 147, 153, 154, 156–161, 167, 170, 172, 173, 175, 180, 181, 196–199, 203–205, 217, 218, 275, 277, 280, 373
world knowledge, 90, 106, 112, 116, 117, 139–141

XML, 14, 238, 239, 254, 258, 273, 285, 289, 362
XWN, *see* Extended WordNet

yes/no-question, 43–46, 48, 53, 60, 64, 85, 93, 101, 262, 300

Name Index

Abelson, Robert P. 97, 98
Aberdeen, John xxiii, xxvi, 359, 360, 362–364
Abiteboul, Serge 237–239, 241, 243, 246, 291
Abney, Steven P. 26, 27, 245, 250, 254, 255, 271
Agichtein, Eugene 17, 24
op den Akker, Rieks 21
Alexandrov-Kabadjov, Mijail 142, 277, 279
Allen, James 49, 106, 111
Allen, Paul G. 97, 100, 103–105
Aloni, Maria 63, 67, 68
Alshawi, Hiyan 269
Androutsopoulos, Ion 14, 15, 306
Angele, Jurgen 103, 105
Annotation Group, the 17, 264
Appelt, Douglas E. 14, 16–18, 264
Atkins, Sue 162, 164
Attardi, Giuseppe 122
Ayache, Christelle iii, 1, 311, 312, 317, 320, 323, 335

Badulescu, Adriana 27, 96
Baeza-Yates, Ricardo 198
Baker, Collin F. 161, 162, 164, 165, 167, 289, 377
Bakshi, Karun 20, 304, 305, 330
Baldewein, Ulrike 168, 281
Balogh, Jennifer 39
Banerjee, Satanjeev 204
Bar-Haim, Roy 114, 115, 118
Barker, Ken 97, 100, 103–105
Barroso, Luiz André 248
Baxter, David 97, 100, 103, 104
Bayer, Samuel 116
Bear, John 17
Becker, Markus 252, 257, 260
Becker, Tilman 25, 28
Belnap, Nuel D., Jr. 48–51, 53, 73, 74, 424
Bensley, Jeremy 118
van Benthem, Johan 107
Bering, Christian 260
Berners-Lee, Tim 107, 193
Biemann, Christian 265
Billott, Sylvie 269
Bilotti, Matthew 25, 28
Biwer, Michael 253
Blackburn, Patrick 102, 106–108, 111
Blair-Goldensohn, Sasha 20, 138
Bobrow, Daniel G. 107, 192
Boguraev, Branimir 275, 276
Böhmová, Alena 147, 149
Bolohan, Orest 27, 96
Bond, Francis 129
Boniforti, Marco Aldo Piccolino 122
Borthwick, Andrew 265

Bos, Johan 102, 106–109, 111, 112, 116, 119
Bosma, Wauter 199
Bouma, Gosse 23, 24, 26, 32, 247
Bowden, Mitchell 20, 27
Brachman, Ronald J. 100
Brahm, Uwe 101
Brants, Sabine 149, 168, 350
Brants, Thorsten 149, 169, 256, 257, 260, 265, 350
Braun, Christian 147, 245, 260, 353
Breck, Eric J. 25, 26, 311–313, 323, 325
Bresnan, Joan 198
Brin, Sergey 12
Briscoe, Ted xxiii, 350, 351, 354
Bristot, Antonella 122
Brown, Eric W. 23
Buch, Arnim 101
Buchholz, Sabine 25
Budanitsky, Alexander 204
Buitelaar, Paul 153, 159
Bunescu, Răzvan C. 26, 27, 277
Bunke, Horst 222
Bunt, Harry 21
Burchardt, Aljoscha 161, 166, 168–170, 192, 197, 199, 200
Burger, John D. xxiii, xxvi, 20, 21, 27, 71, 116, 311–313, 323, 359, 360, 362–364
Burhans, Debra Thomas 34, 35, 95, 97, 101, 103
Buscha, Joachim 44
Busemann, Stephan 297

Cabral, John 100, 111, 148, 172–174
Callmeier, Ulrich 111, 253, 257, 260, 281
Cardie, Claire 20, 21, 71, 275
Carroll, John xxiii, 269, 350, 351, 354
Carthy, Joe 116
ten Cate, Balder 62
Chandra, Ashok K. 239
Chang, Nancy 167
Chanod, Jean-Pierre 109
Charniak, Eugene 256, 281
Chaudhri, Vinay 20, 21, 71, 97, 100, 103–105
Chaw, Shaw Yi 97, 100, 103, 104
Chen, Hsin-Hsi 312
Chen, Kuang-hua 312
Chodorow, Martin 204
Chomsky, Noam 41
Chu-Carroll, Jennifer 23
Chua, Tat-Seng 20, 23, 27
Cisternino, Antonio 122
Clark, Christine 20, 27, 111, 118, 157
Clark, Peter 97, 100, 103–105
Clark, Stephen 109, 111, 119
Clarke, Charles L. A. 25, 27
CLEF 126
Clematide, Simon 264
Clifton, Terence 20, 23
Cohen, Michael H. 39
Collins, Michael 26, 27, 265, 280
Condoravdi, Cleo 107, 124, 192
Connolly, Dennis xxiii, xxvi, 359, 360, 362–364
Conrath, David W. 204
Copestake, Ann xxiii, 8, 13, 110, 129, 257, 269, 281, 351
Cormack, Gordon V. 25, 27
Cowie, Jim 16
Cramer, Irene 317
Cronin, Beau 289
Crouch, Richard Dick 107, 114–117, 123, 131, 138, 139, 192, 269
Cruse, David Alan 129, 131, 137, 153, 154, 162
Crysmann, Berthold 14, 252, 257, 281
Crystal, Michael 17, 264

Cui, Hang 20, 23, 27
Curran, James R. 109, 111, 119
Curtis, Jon 97, 100, 103, 104
Cycorp 174
Czuba, Krzysztof 23

Daelemans, Walter 25
Dagan, Ido 34, 114, 115, 117, 118
Dale, Robert 297
DAML 107
D'Andrade, Roy G. 76, 77
Dang, Hoa Trang iii, 1, 311, 312, 335, 377
Daum, Michael 257
Davidson, Donald 157
De Boni, Marco 33, 35, 95, 180, 273, 312
de Paiva, Valeria 107, 192
de Salvo Braz, Rodrigo 196, 197, 199
Dean, Jeffrey 248
Debusmann, Ralph 149, 150
Declerck, Thierry 265
Del Bimbo, Alberto 7
Delmonte, Rodolfo 122
DeOliveira, John 100, 111, 148, 172–174
Diestel, Reinhard 200, 220, 221, 230
Dipper, Stefanie 149, 168, 282, 350
Doets, Kees 107
Dolan, Bill 114, 115, 118
Drożdżyński, Witold 260
Duchier, Denys 150
Dubey, Amit xxvi, 256, 350, 358, 359
Duden 44
Dunnion, John 116
Van Durme, Benjamin 26

EAGLES 310, 418
Earley, Jay 269
Echihabi, Abdessamad 23, 26
Eckert, Miriam 301
Egg, Markus 41, 63, 110, 269
Egli, Urs 50
Eisele, Andreas 111, 257, 281
Eisenberg, Peter 44, 45, 259
Eisinger, Norbert 101, 107
Elhadad, Michael 297
Ellsworth, Michael 161, 162, 164–166, 169
Elworthy, David 26
Engdahl, Elisabet 72
Engel, Ulrich 45, 259
Eppstein, David 221
Erbach, Gregor 20, 253, 260, 317, 320, 323
Erk, Katrin 161, 168–170, 252, 256, 281, 282
van Erk, Sebastiaan 95
Eskin, Eleazar 17
Etzioni, Oren 14

Fahmi, Ismail 23, 24, 26, 32, 247
Falcone, Rino 255
Fan, James 97, 100, 103, 104
Farrar, Scott 175
Federici, Stefano 269
Feldweg, Helmut 158
Fellbaum, Christiane 97, 154, 156, 158, 175
Felshin, Sue 25, 28
Fernandes, Aaron 25, 28
Ferro, Lisa 114–116, 118, 311–313, 323
Ferrucci, David 20, 27
Fikes, Richard E. 175
Fillmore, Charles J. 112, 147, 161, 162, 164–167, 289, 377
Fischer, Ingrid 275
Fleischman, Michael 23, 281
Flickinger, Dan 110, 129, 269

Fliedner, Gerhard 147, 179, 204, 245, 246, 251, 265, 274, 292, 300, 302, 310, 332, 353, 357, 358, 376
Forăscu, Corina 275
Formica, Francesco 122
Forner, Pamela iii, 1, 311, 312, 335
Fortin, Scott 220
Foth, Kilian A. 257
Fowler, Abraham 116, 123
Fox, Heidi 17, 264
Frank, Anette 14, 161, 166, 168–170, 192, 197, 199, 200, 252, 257, 260, 281, 282
Frei, Hans Peter 11
Friedland, Noah S. 97, 100, 103–105
Fukumoto, Jun'ichi 301, 311, 312

Gaizauskas, Robert iii, xxi, 1, 8, 16, 19–22, 35, 71, 97, 265, 312, 313
Galliers, Julia R. 310
Gardent, Claire 97, 110, 134, 279
Garey, Michael R. 198, 200, 220, 221, 239
Geistert, Bernd 275
Gelbukh, Alexander 198
Genesereth, Michael R. 175
Gerber, Laurie 26, 29, 86
Ghani, Rayid 17
Giampiccolo, Danilo iii, 1, 114, 115, 118, 311, 312, 335
Giangola, James P. 39
Gildea, Daniel 147, 196, 197, 280
Ginzburg, Jonathan 41, 72–74, 80–84
Girju, Roxana 27, 96, 196, 197, 199
Glickman, Oren 34, 114, 115, 117, 118
Gnörlich, Carsten 194
Gödel, Kurt 107, 431
Gogelgans, Irina 260
Goolsbey, Keith 173
Görz, Günther 275
Gottlob, Georg 238
Goulart, Rodrigo 142, 277, 279
Graham, Jonathan xxiii, 351
Graupmann, Jens 253
Gravano, Luis 17, 24
Green, Cordell 100, 101
Grefenstette, Gregory 252
Grice, (Herbert) Paul 74, 78, 79, 115
Grishman, Ralph 17
Groenendijk, Jeroen xxi, 33, 40, 41, 48, 52, 63, 64, 66, 68–70, 73–76, 80, 82, 85, 102, 120, 121, 123
Grosz, Barbara J. 14
Guarino, Nicola 172
Guasch, Clara 260
Guha, Ramanathan V. 174
Gulikers, Léon 159
Gupta, Arvind 221, 222

Harabagiu, Sanda M. xxi, 17, 20, 21, 26, 27, 71, 96, 111, 118, 122, 157, 158, 167, 277, 283, 378
Haapalainen, Mariikka 259
Haarslev, Volker 101, 108
Habash, Nizar 297
Haghighi, Aria D. 192, 197, 199, 200
Hahn, Udo 283
Hajič, Jan 147, 149
Hajičová, Eva 147, 149
Halvorsen, Per-Kristian 252
Hamblin, Charles L. 48, 58, 61, 62
Hamp, Birgit 158
Hansen, Silvia 149, 168, 260, 350
Hardy, Hilda 21
Harrah, David 50
Hartrumpf, Sven 194, 200, 248, 275, 361, 364
Hausser, Roland 48, 53, 54
Helbig, Gerhard 44
Helbig, Hermann 194
Henderson, John 116

Hendler, James 107, 193
Hendrix, Gary G. 19
Hermjakob, Ulf 24, 26, 29, 86
Herrera, Jesús 317
Hickl, Andrew 20, 21, 27, 118
Higginbotham, James iv, 2, 33, 40, 41, 48, 62–64, 67, 71–74, 123, 370
Hildebrandt, Wesley 25, 28
Hillenbrand, Thomas 101, 108
Hirschman, Lynette iii, xxi, xxiii, xxvi, 1, 8, 19, 21, 22, 35, 97, 310–314, 323, 359, 360, 362–364
Hirst, Graeme 204
Hladká, Barbora 147, 149
Hobbs, Jerry R. 17, 275
Hockenmaier, Julia 109, 111, 119
Höhle, Tilman 259
Hölzle, Urs 248
Homola, Petr 260
Hopcroft, John E. 224
Hori, Chiori 21
Hori, Takaaki 21
House, David 311–313, 323
Hovy, Eduard 23, 24, 26, 29, 86, 281, 310
Hristea, Florentina 154
Huang, Yifen 26
Hull, Richard 237–239, 241, 243, 246, 291
Hull, Robert D. 48, 53, 54, 73, 74
Huynh, David 20, 304, 305, 330

Iles, Brandon 119
Insinnamo, Patrizia 255
Isozaki, Hideki 21
Israel, David J. 16–18, 20, 21, 71, 97, 100, 103–105, 264

Jackendoff, Ray S. 162
Jacquemin, Christian 20, 21, 71
Jacquey, Evelyne 97
Jiang, Jay J. 204
Jiang, Jing 20
Jijkoun, Valentin iii, 1, 311, 312, 335
Johnson, Christopher R. 147, 161, 162, 164–166
Johnson, David S. 198, 200, 220, 221, 239
Jones, Rosie 17
Jörg, Brigitte 14, 281
Jurafsky, Daniel 280

Kager, René 265
Kaisser, Michael 25, 28
Kaji, Nobuhiro 26
Kameyama, Megumi 17, 275
Kamp, Hans 119, 138, 139
Kan, Min-Yen 20, 23, 27
Kantor, Paul 21
Kaplan, Ronald M. 192, 252
Karger, David R. 20, 304, 305, 330
Karp, Richard M. 224
Karttunen, Lauri 48, 60–62, 71, 72, 114–117, 123, 124, 131, 138, 139, 192, 437
Kasper, Walter 257
Kato, Tsuneaki 301, 311, 312
Katz, Boris 20, 25, 26, 28, 29, 31, 127, 143, 304, 305, 330
Katzir, Roni 25, 28
Kautz, Henry 14
Kawahara, Daisuke 26
Kay, Martin 112, 139
Keen, Enno 101
Keenan, Edward L. 48, 53, 54, 73, 74
Kehler, Andrew 302
Keizer, Simon 21
Kennedy, Christopher 275, 276
Kiefer, Bernd 252, 257
Kilgarriff, Adam 153

Kilpeläinen, Pekka iv, xxvii, 2, 177, 200, 222–224, 232, 234, 285, 370, 375, 376
King, Margaret Maghi 310
King, Tracy Holloway 192
Kingsbury, Paul 147, 169, 196, 197
Kisman, Derek I. E. 27
Klakow, Dietrich 317
Koch, Christoph 238
Koller, Alexander 110, 150, 269
Koskenniemi, Kimmo 11, 259
Kouylekov, Milen 196
Kow, Eric 134, 279
Kowalski, Andrea 161, 168, 169
Krahmer, Emiel 95, 199
Krieger, Hans-Ulrich 14, 257, 260, 281
Krifka, Manfred 48, 53, 55–57, 61, 68, 73
Kuhlmann, Marco 150
Kuhn, Jonas 282
Kunz, Kerstin 260
Kunze, Claudia 147, 158–161, 273, 289
Kupiec, Julian 19
Kupść, Anna 26
Kurohasi, Sadao 26
Kwon, Namhee 281

Lacatusu, Finley 27, 96, 122
Lang, Bernard 269
Langendoen, D. Terence 175
Lapata, Mirella 168
Lappin, Shalom xxv, 274–276, 278
Lassila, Ora 107, 193
Lavoie, Benoit 297
Lawrence, Steve 24
Leacock, Claudia 204
Leass, Herbert J. xxv, 274–276, 278
Lehmann, John 21
Lehmann, Sabine 260
Lehnert, Wendy G. 16, 19, 35, 38, 77, 83, 96, 97
Leidner, Jochen L. 317
Lemnitzer, Lothar 147, 158, 160, 273, 289
Lenat, Douglas B. 174
Levin, Beth 127
Lewis, William D. 175
Lezius, Wolfgang 149, 168, 350
Li, Hong 260
Li, Keya 20, 23, 27
Li, Wei 25, 29, 250
Li, Xiaoge 25, 29
Light, Marc 25, 26, 311–313, 323, 325
Lim, Daniel Chung Yong 275, 365
Lin, Chin-Yew 20, 21, 26, 29, 71, 86
Lin, Chuan-Jie 312
Lin, Dekang 140, 196, 204, 340, 351, 374
Lin, Jimmy 20, 25, 26, 28, 29, 31, 143, 304, 305, 330
López-López, Aurelio 198
Loreto, Daniel 25, 28
Lowe, John B. 161, 162
Luckham, David 101
Lynam, Thomas R. 25

Maeda, Eisaku 21
Magnini, Bernardo iii, 1, 34, 114, 115, 117, 118, 196, 311, 312, 317, 320, 323, 335
Maiorano, Steven J. 20, 21, 27, 71, 111, 118, 157, 277
Majorin, Ari 259
Mani, Inderjeet 311–313, 323
Mann, Gideon S. 25, 26, 325
Manning, Christopher D. 116, 117, 192, 197, 199, 200
Manuélian, Hélène 134, 279
Marcu, Daniel 26, 34, 109, 110

Mardis, Scott 20, 27
Markert, Katja 102, 116, 119, 283
Marsi, Erwin 199
Martin, Paul 14
Marton, Gregory 25, 28
Masui, Fumito 301, 311, 312
Matsuda, Megumi 301
Matthews, Gavin 97, 100, 103–105
Matuszek, Cynthia 100, 111, 148, 172–174
May, Robert iv, 2, 33, 63, 73, 74, 370
Maybury, Mark T. iii, 1, 20, 36, 71
McCune, William 101, 119
McKelvie, David 258
McKeown, Kathleen R. 20, 138
Mel’čuk, Igor A. 128, 148, 149
Melz, Eric 26
Menzel, Wolfgang 251, 257
Merlin, Philip M. 239
Messmer, Bruno T. 222
Michelizzi, Jason 205
Mihalcea, Rada 26, 27, 157
Miller, George A. xxi, 20, 21, 71, 97, 99, 153, 154, 156–158, 175
Miller, Katherine J. 154
Miller, Scott 17, 264
Min, Wu 21
Minnen, Guido 351, 354
Miraglia, Pierluigi 97, 100, 103–105
Mitkov, Ruslan 133, 138, 274–276
Moench, Eddie 97, 100, 103, 104
Moldovan, Dan I. xxi, 20, 21, 26, 27, 71, 96, 111, 118, 119, 153, 157, 158
Mollá, Diego 377
Möller, Ralf 101, 108
Montague, Richard 54, 62, 106, 446
Montemagni, Simonetta 109, 269
Montes-y-Gómez, Manuel 198
Monz, Christof 24, 32, 109, 110
Mora, Federico 25, 28
Morarescu, Paul 27, 96
Morărescu, Paul 17, 26, 27
Moschitti, Alessandro 17
Motta, Enrico 27
Müller, Christoph xxvi, 275, 362, 365
Müller, Stefan 257
Mur, Jori 23, 24, 26, 32, 247

Nairn, Rowan 124, 192
Narayanan, Srini 21, 165, 167, 378
Neumann, Günter 23, 25, 257, 265
Newman, Eamonn 116
Ng, Andrew Y. 192, 197, 199, 200
Ng, Hwee Tou 275, 365
Ng, Vincent 275
Niehren, Joachim 110, 269
Nii, H. Penny 257
Niles, Ian 148, 172–175
Nilsson, Nils J. 101
Nirenburg, Sergei 172
Nishimura, Naomi 221, 222
de Nivelle, Hans 101
van Noord, Gertjan 23, 24, 26, 32, 247
Norvig, Peter 100, 172
Novischi, Adrian 27, 96, 119, 153, 157
Nunberg, Geoffrey 129
Nyberg, Eric 20, 26, 27

Oepen, Stephan 269
Ogden, Bill 20, 21, 71
Ohlbach, Hans Jürgen 101, 107
Oppermann, Henrik 97, 100, 103, 104
Oracle 238
Osenova, Petya iii, 1, 311, 312, 335

Padó, Sebastian 161, 168, 169, 256, 281
Page, Lawrence 12

Palmer, Christopher R. 27
Palmer, Martha 147, 196, 197, 204
Pantel, Patrick 140, 196, 340, 374
Paşca, Marius 26, 27
Patwardhan, Siddharth 204, 205
Pease, Adam 148, 172–175
Pedersen, Ted 204, 205
Peinado, Víctor 317
Peñas, Anselmo iii, 1, 311, 312, 317, 320, 323, 335
Pereira, Fernando 14
Peters, Stanley 48, 61, 71, 72, 115, 138
Petruck, Miriam R. L. 147, 161, 162, 164–167
Pianta, Emanuele 122
Piepenbrock, Richard 159
Pinkal, Manfred 110, 161, 166, 168, 169, 258, 269, 275
Pirrelli, Vito 269
Piskorski, Jakub 257, 260
van der Plas, Lonneke 23, 24, 26, 32, 247
Poesio, Massimo 134, 135, 142, 277, 279
Pollard, Carl 41, 110, 269
Popescu, Ana-Maria 14, 34, 109, 110
Popescu-Belis, Andrei 310
Porter, Bruce 97, 100, 103–105
Porter, Martin F. 11, 451
Postolache, Oana 275
Prager, John M. 20, 21, 23, 71
Prescher, Detlef 168, 281
Priss, Uta E. 156
Punyakanok, Vasin 195–197, 199

Qiu, Yonggang 11
Quan, Dennis 20, 304, 305, 330
Quantz, J. Joachim 275
Quasthoff, Uwe 265

Ramachandran, Deepak 173
Rambow, Owen 297
Ramshaw, Lance 17, 264
Rapp, Stefan xxvi, 275, 362, 365
Raskin, Victor 172
Rattnik, Gilbert 159
Ravichandran, Deepak 26
Reagan, Pace 173
Reich, Ingo 55, 58–60, 70
Reiter, Ehud 297, 298
Resnik, Philip 204
Reyle, Uwe 110, 119, 138, 139, 269
Ribieiro-Neto, Berthier 8–10, 35, 179
Riloff, Ellen 11, 17, 20, 21, 25, 26, 71, 325
Rink, Bryan 118
Ritchie, Graeme D. 14, 15, 306
de Rijke, Marten 109, 110, 317, 320, 323
Roberts, Kirk 118
Robin, Jacques 297
Rocha, Paulo iii, 1, 311, 312, 317, 320, 323, 335
Romagnoli, Simone 317
Rössler, Marc 265
Roth, Dan 195–197, 199
Ruhrberg, Peter 110, 269
Rundell, Michael 162
Ruppenhofer, Josef 161, 162, 164–166
Rus, Vasile 26, 27
Russell, Stuart J. 100, 172
Ryan, Sean 21

Sacaleanu, Bogdan iii, 1, 311, 312, 335
Sacerdoti, Earl D. 19
Sag, Ivan A. 41, 72, 73, 110, 129, 269
Sagalowicz, Daniel 19
Salay, Nancy 103, 105
Sammons, Mark 196, 197, 199

Sanfilippo, Antonio 350, 351
Sasaki, Yutaka 21, 312
Sato, Hiroaki 162, 166
Saurí, Roser 116, 123
Schäfer, Ulrich 14, 111, 252, 257, 260, 281
Schank, Roger C. 97, 98
Scheffczyk, Jan 161, 162, 164–166
Schlaikjer, Andrew Hazen 20, 138
Schleichert, Hubert 50
Schmid, Helmut 257, 281
Schmolze, James G. 100
van Schooten, Boris 21
Schulte im Walde, Sabine 271
Schulz, Klaus U. 238
Schwartz, Richard 17, 264
Searle, John R. 74–76
Segond, Frédérique 109
Shan, Chung-chieh 62
Shasha, Dennis 195
Shepard, Blake 97, 100, 103, 104
Shi, Ying 118
Shimada, Atsuko 260
Shimizu, Nobuyuki 21
Shrihari, Rohini 20, 21, 71
Siegel, Melanie 111, 257, 260, 281
Siersdorfer, Stefan 253
Simi, Maria 122
Simov, Kiril 317, 320, 323
Singer, Yoram 265
Singhal, Amit 20, 21, 26, 27, 71
Sinha, Vineet 20, 304, 305, 330
Sizov, Sergej 253
Skut, Wojciech 149, 169, 256, 260, 265, 350
Slavik, John 119
Slocum, Jonathan 19
Small, Sharon 21
Smith, George 149, 168, 350
Smolka, Gert 150
Soderland, Stephen 17
Soon, Wee Meng 275, 365
Soubbotin, Martin M. 26, 250
Soubbotin, Sergei M. 26, 250
Sparck Jones, Karen 8, 13, 310
Srihari, Rohini K. 25, 29, 250
Staab, Steffen 97, 100, 103–105
von Stechow, Arnim 55
Stede, Manfred 251, 255
Steedman, Mark 109, 111, 119
Steel, Thomas B., Jr. 48–51, 53, 73, 74, 424
Stevenson, Mark 265
Stock, Oliviero 255
Stokes, Nicola 116
Stokhof, Martin xxi, 33, 40, 41, 48, 52, 63, 64, 66, 68–70, 73–76, 80, 82, 85, 102, 120, 121, 123
Stolle, Reinhard 107
Stone, Rebecca 17, 264
Strube, Michael xxvi, 275, 301, 362, 365
Strzalkowski, Tomek 20, 21, 71
Stuckardt, Roland 275–277, 360, 361
Suchanek, Fabian M. 173
Sun, Renxu 20, 23, 27
Surdeanu, Mihai 26, 27
Sutcliffe, Richard iii, 1, 311, 312, 317, 320, 323, 335
Szpektor, Idan 114, 115, 118

Tan, Yee Fan 20
Tapanainen, Pasi 252
Tatu, Marta 119
Teahan, William 20, 23
Tecuci, Dan 97, 100, 103, 104
Terra, Egidio L. 25
Tesnière, Lucien 147–151, 202, 457
Thanisch, Peter 14, 15, 306
Thater, Stefan 150
Theobald, Christian 101

Theobald, Martin 253
Theune, Mariët 199
Thompson, Bozena Henisz 13
Thompson, Frederick B. 13
Thompson, Henry S. 258, 310, 314
Tiedemann, Jörg 23, 24, 26, 32, 247
Ting, Liu 21
Tomita, Masaru 269
Tommasi, Alessandro 122
Tonelli, Sara 122
Topic, Dalibor 101
Traum, David 297
TREC 29, 125, 126
Tsukada, Hajime 21
Turing, Alan M. 107
Tyson, Mabry 17

Ulicny, Brian 20
Ullmann, Julian R. 220
Uryupina, Olga 142, 256, 275, 277, 279, 365
Uszkoreit, Hans 14, 149, 169, 257, 260, 281, 350
Uzuner, Ozlem 25, 28

Vallin, Alessandro 317, 320, 323
Vardi, Moshe Y. 239
Vargas-Vera, Maria 27
Verdejo, Felisa 317
Verleg, André 95
Vianu, Victor 237–239, 241, 243, 246, 291
Vieira, Renata 134, 135, 142, 277, 279
Vilain, Marc xxiii, xxvi, 359, 360, 362–364
Villada Moirón, María Begoña 129
Vitacolonna, Luciano 301
Viterbi, Andrew J. 256
Volk, Martin 264
Voorhees, Ellen M. iii, 1, 19–21, 34, 71, 126, 301, 310–312, 317, 323, 327, 335
Vossen, Piek 161

W3C 107, 238
Wacholder, Nina 21
Wagner, Andreas 158, 159, 161, 289
Walter, Stephan 258
Wan, Stephen 377
Wang, Patrick 20, 27
Warshall, Stephen 232
Wasow, Thomas 129
Webber, Bonnie Lynn 74, 79, 80, 110, 306
Weidenbach, Christoph 101
Weikum, Gerhard 253
Weischedel, Ralph 17, 264
Wenke, Dirk 97, 100, 103, 104
Wilks, Yorick 8, 16
Williams, John 118
Wish, Myron 76, 77
Witbrock, Michael 97, 100, 103–105, 111, 148, 172–174
Wolff, Christian 265
Woods, William A. 13
Wright, Abby 164
Wu, Zhibiao 204

Xu, Feiyu 14, 23, 25, 110, 257, 260, 269, 281

Yamron, Boris 21
Yangarber, Roman 17
Yeh, Alexander 116
Yeh, Peter 97, 100, 103, 104
Yih, Wen-tau 195, 199

Zaefferer, Dietmar 48, 53, 54
Zaenen, Annie 114–117, 123, 131, 138, 139, 192

Zhang, Kaizhong 195
Zholkovsky, Alexander 128
Ziegler-Eisele, Dorothee 260
Zimmer, Patrick 253
Zinsmeister, Heike 282

Bibliography

Anne Abeillé, editor. *Building and Using Parsed Corpora*. Kluwer Academic Publisher, Dordrecht, 2003.

Serge Abiteboul, Richard Hull and Victor Vianu. *Foundations of Databases*. Addison-Wesley, Reading, MA, 1995.

Steven P. Abney. Parsing by chunks. In Robert C. Berwick, Steven P. Abney and Carol Tenny, editors, *Principle-Based Parsing: Computation and Psycholinguistics*. Kluwer, Dordrecht, 1991. URL `http://www.vinartus.net/spa/90e.ps`; 2007/02/04.

Steven [P.] Abney. Partial parsing via finite-state cascades. In *Proceedings of the Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information (ESLLI)*, pages 8–15, Praha, 1996. URL `http://www.sfs.nphil.uni-tuebingen.de/Staff-Old/abney/96h.ps.gz`; 2007/02/04.

Steven [P.] Abney, Michael Collins and Amit Singhal. Answer extraction. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP)*, Seattle, WA, 2000. URL `http://www.aclweb.org/anthology/A/A00/A00-1041`; 2007/02/04.

Eugene Agichtein, Eleazar Eskin and Luis Gravano. Combining strategies for extracting relations from text collections. In *Proceedings of the 2000 ACM SIGMOD Workshop on Data Mining and Knowledge Discovery (DMKD)*, Dallas, TX, 2000. URL `http://www.cs.columbia.edu/~gravano/Papers/2000/dmkd00.pdf`; 2007/02/04.

Eugene Agichtein and Luis Gravano. Extracting relations from large plaintext collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries (DL)*, San Antonio, TX, 2000. URL `http://www.cs.columbia.edu/~gravano/Papers/2000/dl00.pdf`; 2007/02/04.

Eugene Agichtein, Steve Lawrence and Luis Gravano. Learning search engine specific query transformations for question answering. In *Proceedings of the Tenth International World Wide Web Conference (WWW 10)*, pages 169–178, Hong Kong, 2001. URL `http://www.cs.columbia.edu/~gravano/Papers/2001/www10.pdf`; 2007/02/04.

Rieks op den Akker, Harry Bunt, Simon Keizer and Boris van Schooten. From question answering to spoken dialogue: Towards an information search assistant for interactive multimodal information extraction. In *Proceedings of the ninth European Conference on Speech Communication and Technology (INTERSPEECH 2005 – EUROSPEECH)*, Lisboa, 2005. URL `http://wwwhome.cs.utwente.nl/~schooten/vidiam/papers-indexed/akker05qasd.pdf`; 2007/02/07.

James Allen. *Natural Language Understanding*. Benjamin/Cummings, Redwood City, CA, second edition, 1995.

Maria Aloni. *Quantification under Conceptual Covers*. PhD thesis, Institute for Logic, Language and Computation, Universiteit van Amsterdam, 2001. URL `http://staff.science.uva.nl/~maloni/tesi/tesi.ps`; 2007/02/04.

Hiyan Alshawi and Richard [Dick] Crouch. Monotonic semantic interpretation. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, Newark, DE, 1992. URL `http://www.aclweb.org/anthology/P92-1005`; 2007/02/04.

I[on] Androutsopoulos, G[raeme] D. Ritchie and P[eter] Thanisch. Natural language interfaces to databases – an introduction. *Natural Language Engineering*, 1(1):29–81, 1995. URL `http://arxiv.org/pdf/cmp-lg/9503016`; 2007/02/04.

Douglas E. Appelt, Jerry R. Hobbs, John Bear, David [J.] Israel, Megumi Kameyama and Mabry Tyson. SRI International FASTUS system. MUC-6 results and analysis. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 237–248, Columbia, MD, 1995. URL `http://www.isi.edu/~hobbs/muc6.pdf`; 2007/02/04.

Douglas E. Appelt and David J. Israel. Introduction to information extraction technology. `http://www.ai.sri.com/~appelt/ie-tutorial/IJCAI99.pdf`; 2007/02/04, August 1999. Notes from the IJCAI-99 Tutorial.

Sue Atkins, Charles J. Fillmore and Christopher R. Johnson. Lexicographic relevance: Selecting information from corpus evidence. *International Journal of Lexicography*, 16(Special Issue 3):251–280, 2003a. URL `http://ijl.oxfordjournals.org/cgi/reprint/16/3/251.pdf`; 2006/02/16.

Sue Atkins, Michael Rundell and Hiroaki Sato. The contribution of framenet to practical lexicography. *International Journal of Lexicography*, 16(Special Issue 3):333–357, 2003b. URL `http://ijl.oxfordjournals.org/cgi/reprint/16/3/333.pdf`; 2006/02/16.

Giuseppe Attardi, Antonio Cisternino, Francesco Formica, Maria Simi and Alessandro Tommasi. PiQASso: Pisa question answering system. In *Proceedings of The Tenth Text REtrieval Conference (TREC 2001)*, Gaithersburg, MD, 2002. URL `http://trec.nist.gov/pubs/trec10/papers/piqasso.pdf`; 2005/12/12.

Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi and Peter Pantel-Schneider, editors. *The Description Logic Handbook. Theory, Implementation, and Applications*. Cambridge University Press, Cambridge, 2003.

Ricardo Baeza-Yates and Berthier Ribieiro-Neto. *Modern Information Retrieval*. ACM Press, New York, NY, 1999.

Collin F. Baker, Charles J. Fillmore and Beau Cronin. The structure of the FrameNet database. *International Journal of Lexicography*, 16(Special Issue 3):281–296, 2003. URL `http://ijl.oxfordjournals.org/cgi/reprint/16/3/281.pdf`; 2006/02/16.

Collin F. Baker, Charles J. Fillmore and John B. Lowe. The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (Coling-ACL)*, Montreal, Canada, 1998. URL `http://framenet.icsi.berkeley.edu/~framenet/papers/acl98.pdf`; 2004/02/04.

Ulrike Baldewein, Katrin Erk, Sebastian Padó and Detlef Prescher. Semantic role labelling with similarity-based generalization using EM-based clustering. In *Proceedings of Senseval'04*, Barcelona, 2004. URL `http://www.coli.uni-sb.de/~erk/OnlinePapers/senseval.ps`; 2004/07/01.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini and Idan Szpektor. The second pascal recognising textual entailment challenge. In Magnini and Dagan (2006). URL `http://ir-srv.cs.biu.ac.il:64080/RTE2/proceedings/01.pdf`; 2006/10/25.

Luiz André Barroso, Jeffrey Dean and Urs Hölzle. Web search for a planet: The Google cluster architecture. *IEEE Micro*, 23 (2):22–28, 2003. URL `http://labs.google.com/papers/googlecluster-ieee.pdf`; 2006/06/14.

Samuel Bayer, John [D.] Burger, Lisa Ferro, John Henderson and Alexander Yeh. MITRE's submissions to the EU Pascal RTE Challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment (RTE 1)*, Southampton, 2005. URL `http://www.cs.biu.ac.il/~glikmao/rte05/bayer_et_al.pdf`; 2006/01/05.

Markus Becker and Anette Frank. A stochastic topological parser for german. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, 2002. URL urlhttp://www.aclweb.org/anthology/C02-1093; 2006/08/24.

Nuel D. Belnap, Jr. and Thomas B. Steel, Jr. *The Logic of Questions and Answers*. Yale University Press, New Haven and London, 1976.

Johan van Benthem and Kees Doets. Higher-order logic. In D[ov M.] Gabbay and F[ranz] Guenthner, editors, *Handbook of Philosophical Logic, Volume I*, volume 164 of *Studies in Epistemology, Logic, Methodology, and Philosophy of Science*, pages 275–329. Reidel, Dordrecht, 1983.

Christian Bering, Witold Drożdżyński, Gregor Erbach, Clara Guasch, Petr Homola, Sabine Lehmann, Hong Li, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, Atsuko Shimada, Melanie Siegel, Feiyu Xu and Dorothee Ziegler-Eisele. Corpora and evaluation tools for multilingual named entity grammar development. In *Proceedings of Multilingual Corpora Workshop at Corpus Linguistics 2003*, pages 42–52, Lancaster, 2003. URL `http://www.dfki.de/~feiyu/multi-corpus.pdf`; 2004/04/15.

Tim Berners-Lee, James Hendler and Ora Lassila. The semantic web. *Scientific American*, May 2001. URL `http://www.sciam.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21`;2006/02/16.

Sylvie Billott and Bernard Lang. The structure of shared forests in ambiguous parsing. In *Proceedings of the 27th Meeting of the Association for Compu-*

tational Linguistics (ACL'89), pages 143–151, Vancouver, BC, 1989. URL `http://www.aclweb.org/anthology/P89-1018`; 2006/10/13.

Patrick Blackburn and Johan Bos. Computational semantics. *Theoria*, 18(1): 27–45, 2003. URL `http://www.iccs.informatics.ed.ac.uk/~jbos/pubs/theoria.pdf`; 2006/01/19.

Patrick Blackburn and Johan Bos. *Prepresentation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI Studies in Computational Linguistics. CSLI Publications, Stanford, CA, 2006.

Sasha Blair-Goldensohn, Kathleen R. McKeown and Andrew Hazen Schlaikjer. A hybrid approach for answering definitional questions. Technical Report CUCS-006-03, Columbia University, New York, NY, 2003. URL `http://www1.cs.columbia.edu/~sashabg/docs/defscriber_long.pdf`; 2004/03/30.

Sasha Blair-Goldensohn, Kathleen R. McKeown and Andrew Hazen Schlaikjer. Answering definitional questions. A hybrid approach. In Maybury (2004a). URL `http://www1.cs.columbia.edu/~sashabg/docs/defscriber_chapter.doc`; 2004/03/30.

K. H. Bläsius and H.-J. Bürckert, editors. *Deduktionssysteme. Automatisierung des logischen Denkens*. Oldenbourg, München, Wien, 2., völlig überarbeitete und erweiterte Auflage, 1992.

D[aniel G.] Bobrow, C[leo] Condoravdi, R[ichard [Dick]] Crouch, R[onald M.] Kaplan, L[auri] Karttunen, T[racy Holloway] King, V[aleria] de Paiva and A[nnie] Zaenen. A basic logic for textual inference. In *Proceedings of the AAAI Workshop on Inference for Textual Question Answering*, Pittsburgh, PA, 2005. URL `http://www2.parc.com/istl/groups/nltt/papers/textual-inference.pdf`; 2006/11/08.

Alena Böhmová, Jan Hajič, Eva Hajičová and Barbora Hladká. The Prague Dependency Treebank. A three-level annotation scenario. In Abeillé (2003), chapter 7, pages 103–127. URL `http://ufal.mff.cuni.cz/pdt2.0/publications/HajicHajicovaAl2000.pdf`; 2006/02/15.

Andrew Borthwick. *A Maximum Entropy Approach to Named Entity Recognition*. PhD thesis, New York University, New York, NY, 1999. URL `http://cs.nyu.edu/cs/projects/proteus/publication/papers/borthwick_thesis.ps`; 2004/05/07.

Johan Bos. Towards wide-coverage semantic interpretation. In *Proceedings of Sixth International Workshop on Computational Semantics (IWCS-6)*, pages 42–53, Tilburg, 2006. URL `http://www.ltg.ed.ac.uk/~jbos/pubs/bos-iwcs-6.pdf`; 2006/10/25.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran and Julia Hockenmaier. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th International Conference on Computational Linguistics*, Gevève, 2004. URL `http://www.iccs.informatics.ed.ac.uk/~stephenc/papers/bos_etal.pdf`; 2006/08/24.

Johan Bos and Katja Markert. Combining shallow and deep NLP methods for recognizing textual entailment. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment (RTE 1)*, Southampton, 2005. URL `http://www.cs.biu.ac.il/~glikmao/rte05/bos_and_markert.pdf`; 2006/01/05.

Johan Bos and Katja Markert. When logical inference helps determining textual entailment (and when it doesn't). In Magnini and Dagan (2006). URL `http://www.ltg.ed.ac.uk/~jbos/pubs/bosmarkertRTE2.pdf`; 2006/10/25.

Gosse Bouma, Ismail Fahmi, Jori Mur, Gertjan van Noord, Lonneke van der Plas and Jörg Tiedemann. Linguistic knowledge and question answering. *Traitement Automatique des Langues*, 2006. Special Issue on Question-Answering Systems, to Appear.

Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9:171–216, 1985. URL `http://www.cogsci.rpi.edu/CSJarchive/1985v09/i02/p0171p0216/MAIN.PDF`; 2006/01/19.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius and George Smith. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol, Bulgaria, 2002. URL `http://www.ims.uni-stuttgart.de/projekte/TIGER/paper/treeling2002.pdf`; 2006/08/16.

Thorsten Brants. Tnt – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP 2000)*, Seattle, WA, 2000. URL `http://www.aclweb.org/anthology/A00-1031`; 2006/08/09.

Thorsten Brants, Wojciech Skut and Hans Uszkoreit. Syntactic annotation of a German newspaper corpus. In *Proceedings of the Association pour le Traitement Automatique des Langues (ATALA) Treebank Workshop*, pages 69–76, Paris, 1999. URL `http://www.coli.uni-sb.de/~thorsten/publications/Brants-ea-ATALA99.pdf`; 2004/04/19.

Christian Braun. Flaches und robustes Parsen deutscher Satzgefüge. Diploma thesis, Fachbereich Computerlinguistik, Universität des Saarlandes, Saarbrücken, 1999.

Christian Braun. Parsing German text for syntacto-semantic structures. In *Proceedings of the Lorraine-Saarland Workshop Series "Prospects and Advances in the Syntax/Semantics Interface"*, pages 99–102, Nancy, 2003. URL `http://www.loria.fr/~duchier/Lorraine-Saarland/braun.ps`; 2006/08/08.

Christian Braun and Gerhard Fliedner. *Richtlinien für die Annotation mittels Grammatischer Relationen. Technische Dokumentation*. Computerlinguistik, Universität Saarbrücken, Februar 2005.

Eric J. Breck, John D. Burger, Lisa Ferro, Lynette Hirschman, David House, Marc Light and Inderjeet Mani. How to evaluate your question answering system every day… and still get real work done. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC 2000)*, Athens, 2000. URL `http://arXiv.org/abs/cs/0004008`; 2004/04/02.

Joan Bresnan. *Lexical-Functional Syntax*, volume 16 of *Blackwell Textbooks in Linguistics*. Blackwell, Malden, MA, Oxford, 2001.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World Wide Web Conference (WWW-7)*, Brisbane, 1998. URL `http://www7.scu.edu.au/1921/com1921.htm`; 2006/01/04. Extended electronic version.

Ted Briscoe and John Carroll. Grammatical relation annotation. `http://www.cogs.susx.ac.uk/lab/nlp/carroll/grdescription/`; 2005/03/23, March 2000.

Ted Briscoe, John Carroll, Jonathan Graham and Ann Copestake. Relational evaluation schemes. In *Proceedings of "Beyond PARSEVAL - Towards Improved Evaluation Measures for Parsing Systems": Workshop at the 3rd International Conference on Language Resources and Evaluation*, pages 4–

8, Las Palmas, 2002. URL `http://www.cogs.susx.ac.uk/lab/nlp/carroll/papers/beyond-proceedings.pdf`; 2004/04/05.

Michael L. Brodie and John Mylopoulos, editors. *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*. Springer, Berlin, 1986.

Arnim Buch and Thomas Hillenbrand. Waldmeister: Development of a high performance completion-based theorem prover. SEKI-Report SR-96-01, Universität Kaiserslautern, 1996. URL `http://www-avenhaus.informatik.uni-kl.de/seki/1996/Buch.SR-96-01.ps.gz`; 2006/08/09.

S[abine] Buchholz and W[alter] Daelemans. Complex answers: A case study using a WWW question answering system. *Natural Language Engineering*, 7(4):301–323, 2001. URL `http://journals.cambridge.org/action/displayFulltext?type=1&fid=96164&jid=&volumeId=&issueId=04&aid=96163`; 2006/08/09.

Alexander Budanitsky and Graeme Hirst. Evaluating WordNet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32 (1):13–47, 2006. URL `http://ftp.cs.toronto.edu/pub/gh/Budanitsky+Hirst-2006.pdf`; 2006/04/12.

Paul Buitelaar. *CoreLex: Systematic Polysemy and Underspecification*. PhD thesis, Department of Computer Science, Brandeis University, Waltham, MA, 1998. URL `http://www.cs.brandeis.edu/~paulb/thesis.ps.gz`; 2006/08/08.

Aljoscha Burchardt, Katrin Erk and Anette Frank. A WordNet detour to FrameNet. In *Proceedings of the GLDV 2005 Workshop “GermaNet II”*, Bonn, 2005a. URL `http://www.coli.uni-saarland.de/~albu/papers/gnws05_burchardt_erk_frank-final.pdf`; 2006/08/09.

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski and Sebastian Padó. Salto – a versatile multi-level annotation tool. In *Proceedings of the 5th Edition of the International Conference on Language Resources and Evaluation (LREC 2006)*, Genova, 2006a. URL `http://www.coli.uni-saarland.de/projects/salsa/papers/lrec06-tool.pdf`; 2006/08/09.

Aljoscha Burchardt, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó and Manfred Pinkal. The SALSA corpus: A german corpus resource for lexical semantics. In *Proceedings of the 5th Edition of the International Conference on Language Resources and Evaluation (LREC 2006)*, Genova, 2006b. URL `http://www.coli.uni-saarland.de/~pado/pub/papers/lrec06_burchardt1.pdf`; 2006/08/09.

Aljoscha Burchardt and Anette Frank. Approximating textual entailment with LFG and FrameNet frames. In Magnini and Dagan (2006). URL `http://www.coli.uni-saarland.de/~albu/papers/rte-05.pdf`; 2006/10/27.

Aljoscha Burchardt, Anette Frank and Manfred Pinkal. Building text meaning representations from contextually related frames – a case study. In *Proceedings of the 6th International Workshop on Computational Semantics (IWCS-6)*, Tilburg, 2005b. URL `http://www.coli.uni-saarland.de/~albu/papers/iwcs05.ps`; 2005/09/16.

John [D.] Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, Sanda [M.] Harabagiu, David [J.] Israel, Christian Jacquemin, Chin-Yew Lin, Steve[n J.] Maiorano, George [A.] Miller, Dan [I.] Moldovan, Bill Ogden, John [M.] Prager, Ellen Riloff, Amit Singhal, Rohini Shrihari, Tomek Strzalkowski, Ellen [M.] Voorhees and Ralph Weis(c)hedel. Issues, tasks and program structures to roadmap research in question & answering (Q&A). Document Understanding Conferences Roadmapping Documents, 2001. URL `http://www-nlpir.nist.gov/projects/duc/papers/qa.Roadmap-paper_v2.doc`; 2004/02/24.

Debra Thomas Burhans. *A Question Answering Interpretation of Resolution Refutation*. PhD thesis, Computer Science and Engineering, State University of New York, Buffalo, NY, 2002. URL `http://www.cse.buffalo.edu/~burhans/debra.ps`; 2005/10/05.

Stephan Busemann. Best-first surface realization. In *Proceedings of the 8th International Workshop on Natural Language Generation*, Herstmonceux, UK, 1996. URL `http://www.aclweb.org/anthology/W96-0411`;2001/05/03.

Ulrich Callmeier. Pet – a platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6(1):99–107, 2000. URL `http://dx.doi.org/10.1017/S1351324900002369`; 2007/02/07.

Ulrich Callmeier, Andreas Eisele, Ulrich Schäfer and Melanie Siegel. The DeepThought core architecture framework. In *Proceedings of the*

Fourth International Conference on Language Resources and Evaluation (LREC 2004), Lisboa, 2004. URL `http://sprout.dfki.de/publications/applications/deep_thought.pdf`; 2006/07/07.

Ulrich Callmeier, Gregor Erbach, Irina Gogelgans, Silvia Hansen, Kerstin Kunz and Dorothee Ziegler-Eisele. *COLLATE Annotationsschema*. Computational Linguistics, Saarland University, July 2003. URL `http://www.mcgreg.net/pub/collate/AnnotationScheme.pdf`; 2006/08/08.

John Carroll, Ted Briscoe and Antonio Sanfilippo. Parser evaluation: Current practice. In *EAGLES. Evaluation Working Group Final Report* EAGLES (1999b), pages 140–150. URL `http://www.issco.unige.ch/projects/eagles/ewg99/ewg99draft.ps.gz`; 2006/02/28.

John Carroll, Guido Minnen and Ted Briscoe. Parser evaluation. Using a grammatical relation annotation scheme. In Abeillé (2003), chapter 17, pages 299–316.

Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing (STOC '77)*, pages 77–90, Boulder, CO, 1977. URL `http://doi.acm.org/10.1145/800105.803397`; 2006/04/25.

Nancy Chang, Srini Narayanan and Miriam R. L. Petruck. From frames to inference. In *Proceedings of the First International Workshop on Scalable Natural Language Understanding*, Heidelberg, 2002a. URL `http://framenet.icsi.berkeley.edu/~framenet/papers/changscan.pdf`; 2006/08/08.

Nancy Chang, Srini Narayanan and Miriam R. L. Petruck. Putting frames in perspective. In *Proceedings of the Nineteenth International Conference on Computational Linguistics*, Taipei, Taiwan, 2002b. URL `http://framenet.icsi.berkeley.edu/~framenet/papers/chang_narayan_petruck.pdf`; 2006/08/08.

Jean-Pierre Chanod, Simonetta Montemagni and Frédérique Segond. Dynamic relaxation: Measuring the distance from text to grammar. In Carlos Martín-Vide, editor, *Current Issues in Mathematical Linguistics*, pages 373–379. Elsevier, Amsterdam, 1994.

Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of First Meeting of the North American Chapter of the Association for Com-*

putational Linguistics, Seattle, WA, 2000. URL `http://www.aclweb.org/anthology/A00-2018`; 2006/08/09.

Noam Chomsky. *Lectures on Government and Binding. The Pisa Lectures*. Studies in Generative Grammar. Mouton de Gruyter, Berlin, New York, NY, 7th edition, 1993.

Charles L. A. Clarke, Gordon V. Cormack, Thomas R. Lynam and Egidio L. Terra. Question answering by passage selection. In Strzalkowski and Harabagiu (2006), pages 349–382.

CLEF. Test sets at CLEF-2003. Cross-language tasks: German. `http://clef-qa.itc.it/2005/down/data/clef03/CLEF03_QA_cl_GER_ENG.txt`; 2005/03/08, 2003.

Terence Clifton and William Teahan. Bangor at TREC 2004: Question answering track. In *Proceedings of The Thirteenth Text REtrieval Conference (TREC 2004)*, Gaithersburg, MD, 2005. URL `http://trec.nist.gov/pubs/trec13/papers/uwales-bangor.qa.pdf`; 2005/07/01.

Michael H. Cohen, James P. Giangola and Jennifer Balogh. *Voice User Interface Design*. Addison-Wesley, Boston, 2004.

Ronald Cole, Joseph Mariani, Hans Uszkoreit, Giovanni Battista Varile, Annie Zaenen, Antonio Zampolli and Victor Zue, editors. *Survey of the State of the Art in Human Language Technology*. Cambridge University Press and Giardini Editori, Cambridge, 1997.

Michael Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the ACL (ACL'97)*, pages 16–23, Madrid, 1997. URL `http://www.aclweb.org/anthology/P97-1003`; 2006/08/24.

Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, College Park, MD, 1999. URL `http://www.aclweb.org/anthology/W/W99/W99-0613`; 2004/02/24.

Cleo Condoravdi, [Richard] Dick Crouch, Valeria de Paiva, Reinhard Stolle and Daniel G. Bobrow. Entailment, intensionality and text understanding. In *Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning*, Edmonton, 2003. URL `http://www.aclweb.org/anthology/W03-0906`; 2006/01/05.

Ann Copestake. Robust minimal recursion semantics. `http://www.cl.cam.ac.uk/~aac10/papers/rmrsdraft.pdf`; 2007/02/02, 2006. Unpublished Draft.

Ann Copestake, Dan Flickinger, Carl Pollard and Ivan A. Sag. Minimal recursion semantics: An introduction. *Research on Language & Computation*, 3(2–3):281–332, 2005. URL `http://www.springerlink.com/openurl.asp?genre=article&issn=1570-7075&volume=3&issue=2&spage=281`; 2006/07/07.

Ann Copestake and Karen Sparck Jones. Natural language interfaces to databases. *Knowledge Engineering Review*, 4:225–249, 1990.

G[ordon] V. Cormack, C[harles] L. A. Clarke, D[erek] I. E. Kisman and C[hristopher] R. Palmer. Fast automatic passage ranking (multitext experiments for trec-8). In *Proceedings of The Eighth Text REtrieval Conference (TREC 8)*, Gaithersburg, MD, 2000. URL `http://trec.nist.gov/pubs/trec8/papers/waterloo.pdf`; 2007/01/16.

Jim Cowie and Wendy [G.] Lehnert. Information extraction. *Communications of the ACM*, 39(1):80–91, 1996. URL `http://portal.acm.org/ft_gateway.cfm?id=234209&type=pdf&coll=GUIDE&dl=GUIDE&CFID=68846513&CFTOKEN=49798978`; 2004/08/20.

Irene Cramer, Jochen L. Leidner and Dietrich Klakow. Building an evaluation corpus for German question answering by harvesting Wikipedia. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-2006)*, Genova, 2006.

Richard [Dick] Crouch, Lauri Karttunen and Annie Zaenen. Circumscribing is not excluding: A reply to Manning. `http://www2.parc.com/istl/members/karttune/publications/reply-to-manning.pdf`; 2006/10/25, 2006.

[Richard] Dick Crouch, Roser Saurí and Abraham Fowler. Aquaint pilot knowledge-based evaluation: Annotation guidelines. `http://www2.parc.com/istl/groups/nltt/papers/aquaint_kb_pilot_evaluation_guide.pdf`; 2006/10/31, May 2005.

D[avid] A[lan] Cruse. *Lexical Semantics*. Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge, 1986.

Berthold Crysmann, Anette Frank, Bernd Kiefer, Stefan Müller, Günter Neumann, Jakub Piskorski, Ulrich Schäfer, Melanie Siegel, Hans Uszkoreit, Feiyu Xu, Markus Becker and Hans-Ulrich Krieger. An integrated archictecture for shallow and deep processing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, PA, 2002. URL `http://www.aclweb.org/anthology/P02-1056`; 2006/08/24.

Hang Cui, Keya Li, Renxu Sun, Tat-Seng Chua and Min-Yen Kan. National University of Singapore at the TREC 13 question answering main task. In *Proceedings of The Thirteenth Text REtrieval Conference (TREC 2004)*, Gaithersburg, MD, 2005. URL `http://trec.nist.gov/pubs/trec13/papers/nus.chua.qa.pdf`; 2005/04/15.

Cycorp. *Contexts in Cyc®*, 2002. URL `http://www.cyc.com/cycdoc/course/contexts-basic-module.html`; 2006/03/23.

Ido Dagan, Oren Glickman and Bernardo Magnini. The PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment (RTE 1)*, Southampton, 2005. URL `http://www.cs.biu.ac.il/~glikmao/rte05/dagan_et_al.pdf`; 2006/01/04.

DAML. Reference description of the DAML+OIL (march 2001) ontology markup language, 2001. URL `http://www.daml.org/2001/03/reference.html`.

R[oy] G. D'Andrade and M[yron] Wish. Speech act theory in quantitative research on interpersonal behavior. *Discourse Processes*, 8(2):229–258, 1985.

Hoa Trang Dang. Overview of DUC 2005 (draft). In *Proceedings of the Document Understanding Workshop (DUC 2005)*, Vancouver, 2005. URL `http://www-nlpir.nist.gov/projects/duc/pubs/2005papers/OVERVIEW05.pdf`; 2006/03/08.

Michael Daum, Kilian A. Foth and Wolfgang Menzel. Constraint based integration of deep and shallow parsing techniques. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003)*, Budapest, 2003. URL `http://www.aclweb.org/anthology/E03-1052`; 2006/08/24.

Donald Davidson. *Essays on Actions and Events*. Clarendon Press, Oxford, 1980.

Marco De Boni. *Relevance in Open Domain Question Answering: Theoretical Framework and Applications*. PhD thesis, Department of Computer Science, University of York, 2004. URL `http://www.businessfuture.co.uk/library/papers/thesis.pdf`; 2005/12/12.

Rodrigo de Salvo Braz, Roxana Girju, Vasin Punyakanok, Dan Roth and Mark Sammons. An inference model for semantic entailment in natural language. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment (RTE 1)*, Southampton, 2005. URL `http://www.cs.biu.ac.il/~glikmao/rte05/de_salvo_braz_et_al.pdf`; 2006/01/05.

Ralph Debusmann. *Extensible Dependency Grammar: A Modular Grammar Formalism Based on Multigraph Description*. PhD thesis, Computer Sciences, Saarland University, 2006. URL `http://www.ps.uni-sb.de/Papers/abstracts/radediss.pdf`; 2007/01/23.

Ralph Debusmann, Denys Duchier, Alexander Koller, Marco Kuhlmann, Gert Smolka and Stefan Thater. A relational syntax-semantics interface based on dependency grammar. In *Proceedings of the 20th International Conference on Computational Linguistics*, Genève, 2004. URL `http://acl.eldoc.ub.rug.nl/mirror/coling2004/MAIN/pdf/26-753.pdf`; 2007/01/23.

Thierry Declerck and Günter Neumann. Using a parameterizable and domain-adaptive information extraction system for annotating large-scale corpora? In *Proceedings of the LREC Workshop "Information Extraction Meets Corpus Linguistics"*, pages 4–8, Athens, 2000. URL `http://www.coli.uni-saarland.de/publikationen/softcopies/Declerck:2000:UPD.pdf`; 2006/08/09.

Alberto Del Bimbo. *Visual Information Retrieval*. Morgan Kaufmann, San Francisco, CA, 1999.

Rodolfo Delmonte, Sara Tonelli, Marco Aldo Piccolino Boniforti, Antonella Bristot and Emanuele Pianta. VENSES – a linguistically-based system for semantic evaluation. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment (RTE 1)*, Southampton, 2005. URL `http://www.cs.biu.ac.il/~glikmao/rte05/delmonte_et_al.pdf`; 2006/01/05.

Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Heidelberg, third edition, 2005. URL `http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/GraphTheoryIII.pdf`; 2006/04/25.

Denys Duchier and Ralph Debusmann. Topological dependency trees: A constraint-based account of linear precedence. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, 2001. URL `http://www.aclweb.org/anthology/P01-1024`; 2007/01/23.

Amit Dubey. *Statistical Parsing for German: Modeling syntactic properties and annotation differences*. PhD thesis, Institut für Computerlinguistik, Universität des Saarlandes, 2004. URL `http://www.coli.uni-sb.de/~adubey/research/dubey-thesis.ps`; 2006/08/09.

Duden. *Duden. Die Grammatik. Unentbehrlich für richtiges Deutsch*, volume 4 of *Duden*. Dudenverlag, Mannheim, Leipzig, Wien, Zürich, 7., völlig neu erarbeitete und erweiterte Auflage, 2005.

Benjamin Van Durme, Yifen Huang, Anna Kupść and Eric Nyberg. Towards light semantic processing for question answering. In *Proceedings of the HLT-NAACL Workshop on Text Meaning*, pages 54–61, Edmonton, Canada, 2003. URL `http://www.lti.cs.cmu.edu/Research/JAVELIN/vandurme.pdf`; 2004/01/29.

EAGLES. EAGLES. Evaluation of natural language processing systems. Technical Report EAG-EWG-PR.2, Expert Advisory Group on Language Engineering Standards, 1995.

EAGLES. The EAGLES 7-step recipe. `http://www.issco.unige.ch/projects/eagles/ewg99/7steps.html`; 2004/04/02, 1999a.

EAGLES. EAGLES. Evaluation working group final report. Technical Report EAG-II-EWG-PR.1, Expert Advisory Group on Language Engineering Standards, 1999b. URL `http://www.issco.unige.ch/projects/eagles/ewg99/ewg99draft.ps.gz`; 2006/02/28.

Jay Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

Abdessamad Echihabi, Ulf Hermjakob, Eduard Hovy, Daniel Marcu, Eric Melz and Deepak Ravichandran. How to select an answer string? In Strzalkowski and Harabagiu (2006), pages 349–382.

Miriam Eckert and Michael Strube. Dialogue acts, synchronising units and anaphora resolution. *Journal of Semantics*, 17(1):51–89, 2000. URL `http://www.eml-r.org/english/homes/strube/downloads/jos01.ps.gz`; 2006/03/02.

Markus Egg. *Wh*-questions in underspecified minimal recursion semantics. *Journal of Semantics*, 15(1):37–82, 1998. URL `http://odur.let.rug.nl/~egg/Papiere/jos.ps.gz`; 2006/01/04.

Markus Egg, Alexander Koller and Joachim Niehren. The constraint language for lambda structures. *Journal of Logic, Language, and Information*, 10 (4):457–485, 2001. URL `http://www.coli.uni-saarland.de/~koller/papers/clls.ps.gz`; 2006/11/28.

Markus Egg, Joachim Niehren, Peter Ruhrberg and Feiyu Xu. Constraints over lambda-structures in semantic underspecification. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, Montreal, 1998. URL `http://www.aclweb.org/anthology/P98-1058`; 2006/10/25.

Urs Egli and Hubert Schleichert. Bibliography of the theory of questions and answers, by Urs Egli and Hubert Schleichert. In *The Logic of Questions and Answers* Belnap and Steel (1976), pages 155–200.

Peter Eisenberg. *Grundriß der deutschen Grammatik. Band 2: Der Satz*. J. B. Metzler, Stuttgart, Weimar, 1999.

Norbert Eisinger and Hans Jürgen Ohlbach. Grundlagen und Beispiele. In Bläsius and Bürckert (1992), pages 25–90.

Michael Elhadad and Jacques Robin. An overview of SURGE: A reusable comprehensive syntactic realization component. In *Proceedings of the Eighth International Natural Language Generation Workshop, Demonstrations Volume (INLG '96)*, Herstmonceux Castle, Sussex, 1996. URL `http://www.di.ufpe.br/~jr/publi/inlg96.ps.gz`; 2006/08/30.

Michael Ellsworth, Katrin Erk, Paul Kingsbury and Sebastian Padó. Propbank, SALSA, and FrameNet: How design determines product. In *Proceedings of the LREC 2004 Workshop on Building Lexical Resources from Semantically Annotated Corpora*, Lisboa, 2004. URL `http://www.coli.uni-sb.de/~erk/OnlinePapers/pb-fn.ps`; 2006/08/09.

David Elworthy. Question answering using a large NLP system. In *Proceedings of The Ninth Text REtrieval Conference (TREC 9)*, Gaithersburg, MD, 2001. URL `http://trec.nist.gov/pubs/trec9/papers/msrc-qa.pdf`; 2004/02/13.

Elisabet Engdahl. *Constituent Questions. The Syntax and Semantics of Questions with Special Reference to Swedish*, volume 27 of *Studies in Linguistics and Philosophy*. D. Reidel, Dordrecht, Boston, Lancaster, Tōkyō, 1986.

Ulrich Engel. Regeln zur Wortstellung. In Ulrich Engel, editor, *Forschungsberichte des Instituts für deutsche Sprache*, volume 5, pages 7–148. Institut für deutsche Sprache, Mannheim, 1970.

David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3):1–27, 1999. URL `http://www.cs.brown.edu/publications/jgaa/accepted/99/Eppstein99.3.3.pdf`; 2006/04/25.

Gregor Erbach. Evaluating human question answering performance under time constraints. `http://www.mcgreg.net/pub/human-qa/EvaluatingHumanQA.pdf`; 2005/02/07, 2004.

Katrin Erk, Andrea Kowalski, Sebastian Padó and Manfred Pinkal. Towards a resource for lexical semantics: A large german corpus with extensive semantic annotation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, 2003a. URL `http://www.aclweb.org/anthology/P03-1068`; 2006/08/09.

Katrin Erk, Andrea Kowalski and Manfred Pinkal. A corpus resource for lexical semantics. In *Proceedings of the Fifth International Workshop on Computational Semantics (IWCS)*, Tilburg, 2003b. URL `http://www.coli.uni-sb.de/~erk/OnlinePapers/LexProj.ps`; 2004/02/04.

Katrin Erk and Sebastian Padó. Shalmaneser – a flexible toolbox for semantic role assignment. In *Proceedings of the 5th Edition of the International Conference on Language Resources and Evaluation (LREC 2006)*, Genova, 2006. URL `http://www.coli.uni-saarland.de/~pado/pub/papers/lrec06_erk.pdf`; 2006/08/09.

Scott Farrar, William D. Lewis and D. Terence Langendoen. A common ontology for linguistic concepts. In *Proceedings of the Knowledge Technologies Conference*, Seattle, WA, 2002. URL `http://www.emeld.org/documents/knowtech_paper.pdf`; 2006/08/09.

Stefano Federici, Simonetta Montemagni and Vito Pirrelli. Shallow parsing and text chunking: a view on underspecification in syntax. In *Procceedings of the Eight European Summer School In Logic, Language and Information (ESSLLI)*, Praha, 1996. URL `http://foxdrake.ilc.cnr.it/webtools/documenti/Federicietal-1996.pdf`; 2006/08/10.

Christiane Fellbaum. Introduction. In *WordNet: An Electronic Lexical Database and Some of its Applications* Fellbaum (1998c), pages 1–19.

Christiane Fellbaum. A semantic network of English verbs. In *WordNet: An Electronic Lexical Database and Some of its Applications* Fellbaum (1998c), pages 69–104.

Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database and Some of its Applications*. The MIT Press, Cambridge, MA, 1998c.

Charles J. Fillmore. The case for case. In Emmon Bach and Robert T. Harms, editors, *Universals in Linguistic Theory*, pages 1–88. Holt, Rinehart and Winston, New York, NY, 1968.

Charles J. Fillmore. Frame semantics and the nature of language. In Steven Harnard, editor, *Origin and Development of Language and Speech*, volume 280 of *Annals of the New York Academy of Sciences*, pages 20–32, 1976.

Charles J. Fillmore. Scenes-and-frames semantics. In Zampolli (1977), pages 55–81.

Charles J. Fillmore. Frame semantics. In *Linguistics in the morning calm: Selected papers from SICOL-1981*, pages 111–137, Seoul, 1982.

Charles J. Fillmore. Frames and the semantics of understanding. *Quaderni di Semantica*, VI(2):222–254, 1985.

Charles J. Fillmore and Collin F. Baker. Frame semantics for text understanding. In *Proceedings of the Workshop "WordNet and Other Lexical Resources"*, Pittsburgh, PA, 2001. URL `http://framenet.icsi.berkeley.edu/~framenet/papers/FNcrime.pdf`; 2006/08/30.

Charles J. Fillmore, Christopher R. Johnson and Miriam R. L. Petruck. Background to FrameNet. *International Journal of Lexicography*, 16(Special Issue 3):235–250, 2003a. URL `http://ijl.oxfordjournals.org/cgi/reprint/16/3/235.pdf`; 2004/07/14.

Charles J. Fillmore, Miriam R. L. Petruck, Josef Ruppenhofer and Abby Wright. Framenet in action: The case of attaching. *International Journal of Lexicography*, 16(Special Issue 3):297–332, 2003b. URL `http://ijl.oxfordjournals.org/cgi/reprint/16/3/297.pdf`; 2004/07/14.

Charles J. Fillmore and Hiroaki Sato. Transparency and building lexical dependency graphs. In *Proceedings of the 28th Annual Meeting of the Berkeley Linguistics Society*, pages 87–99, Berkeley, CA, 2002. URL `http://framenet.icsi.berkeley.edu/~framenet/papers/cjf_sato_bls02.pdf`; 2006/08/08.

Ingrid Fischer, Bernd Geistert and Günther Görz. Incremental anaphora resolution in a chart-based semantics construction framework using λ-DRT. In *Proceedings of the International Colloquium on Discourse Anaphora and Anaphora Resolution (DAARC)*, pages 235–244, Lancaster, 1996. URL `http://faui80.informatik.uni-erlangen.de/IMMD8/Services/lt/doc/daarc96.ps`; 2006/08/24.

Michael Fleischman, Eduard Hovy and Abdessamad Echihabi. Offline strategies for online question answering: Answering questions before they are asked. In *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics*, Sapporo, 2003a. URL `http://www.mit.edu/~mbf/ACL_03.pdf`; 2004/10/14.

Michael Fleischman, Namhee Kwon and Eduard Hovy. Maximum entropy models for FrameNet classification. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNSP'03)*, Sapporo, 2003b. URL `http://www.mit.edu/~mbf/EMNLP_03.pdf`; 2004/07/04.

Gerhard Fliedner. Überprüfung und Korrektur von Nominalkongruenz im Deutschen. Diploma thesis, Universität des Saarlandes, Computerlinguistik, Saarbrücken, 2001. URL `http://www.coli.uni-saarland.de/~fliedner/publications/Fliedner01_Ueberpruefung.pdf`; 2006/08/08. In German.

Gerhard Fliedner. A system for checking NP agreement in German texts. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02) Student Workshop*, Philadelphia, PA, 2002.

Gerhard Fliedner. Deriving FrameNet representations: Towards meaning-oriented question answering. In *Natural Language Processing and Information Systems: 9th International Conference on Applications* *of Natural Language to Information Systems (NLDB 2004)*, volume 3136 of *LNCS*, pages 64–75, Salford, 2004a. Springer. URL `http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=3136&spage=64`; 2004/08/19.

Gerhard Fliedner. Issues in evaluating a question answering system. In *Proceedings of LREC Workshop "User-Oriented Evaluation of Knowledge Discovery Systems"*, pages 8–12, Lisboa, 2004b.

Gerhard Fliedner. Korrekturprogramme. In K.-U. Carstensen, Ch. Ebert, C. Endriss, S. Jekat, R. Klabunde and H. Langer, editors, *Computerlinguistik und Sprachtechnologie. Eine Einführung*, pages 463–470. Spektrum Akademischer Verlag, Heidelberg, 2. überarbeitete und erweiterte Auflage, 2004c.

Gerhard Fliedner. A generalised similarity measure for question answering. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB 2005)*, volume 3513 of *LNCS*, pages 380–383, Alicante, 2005. URL `http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=3513&spage=380`; 2005/05/11.

Gerhard Fliedner. Korrekturprogramme. In Irene Cramer and Sabine Schulte im Walde, editors, *Studienbibliographie Computerlinguistik und Sprachtechnologie*. Stauffenburg Verlag Brigitte Narr, Tübingen, 2006a.

Gerhard Fliedner. Towards natural interactive question answering. In *Proceedings of the 5th Edition of the International Conference on Language Resources and Evaluation (LREC 2006)*, Genova, 2006b.

Scott Fortin. The graph isomorphism problem. Technical Report TR 96-20, Department of Computing Science, University of Alberta, Edmonton, Alberta, 1996. URL `http://www.cs.ualberta.ca/TechReports/1996/TR96-20/TR96-20.ps.gz`; 2006/06/23.

Anette Frank, Markus Becker, Berthold Crysmann, Bernd Kiefer and Ulrich Schäfer. Integrated shallow and deep parsing: TopP meets HPSG. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, 2003. URL `http://www.aclweb.org/anthology/P03-1014`; 2006/08/08.

Anette Frank and Katrin Erk. Towards an LFG syntax-semantics interface for frame semantics annotation. In *Fifth International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2004)*, 2004. URL `http://www.dfki.de/~frank/papers/CICLing04-frank-erk.ps`; 2006/07/07.

Anette Frank, Hans-Ulrich Krieger, Feiyu Xu, Hans Uszkoreit, Berthold Crysmann, Brigitte Jörg and Ulrich Schäfer. Querying structured knowledge sources. In *Proceedings of AAAI-05 Workshop on Question Answering in Restricted Domains*, Pittsburgh, PA, 2005. URL `http://www.dfki.de/dfkibib/publications/docs/ws1305FrankA.pdf`; 2005/11/29.

Anette Frank, Hans-Ulrich Krieger, Feiyu Xu, Hans Uszkoreit, Berthold Crysmann, Brigitte Jörg and Ulrich Schäfer. Question answering from structured knowledge sources. *Journal of Applied Logic, Special Issue on Questions and Answers: Theoretical and Applied Perspectives*, 5(1), 2007. URL `http://www.dfki.de/~frank/papers/qajal.pdf`; 2005/11/30. To Appear.

Noah S. Friedland, Paul G. Allen, Gavin Matthews, Michael Witbrock, David Baxter, Jon Curtis, Blake Shepard, Pierluigi Miraglia, Jürgen Angele, Steffen Staab, Eddie Moench, Henrik Oppermann, Dirk Wenke, David [J.] Israel, Vinay Chaudhri, Bruce Porter, Ken Barker, James Fan, Shaw Yi Chaw, Peter Yeh, Dan Tecuci and Peter Clark. Project Halo: Towards a digital aristotle. *AI Magazine*, 25(4):29–48, 2004a. URL `http://www.aifb.uni-karlsruhe.de/~sst/Research/Publications/2004/ai-mag-preprint04.pdf`; 2005/12/12.

Noah S. Friedland, Paul G. Allen, Michael Witbrock, Gavin Matthews, Nancy Salay, Pierluigi Miraglia, Jurgen Angele, Steffen Staab, David [J.] Israel, Vinay Chaudhri, Bruce Porter, Ken Barker and Peter Clark. Towards a quantitative, platform-independent analysis of knowledge systems. In *Proceedings of the Ninth International Conference on the Principles of Knowledge Representation and Reasoning (KR2004)*, Whistler, BC, Canada, 2004b. URL `http://citeseer.ist.psu.edu/702642.html`; 2005/12/21.

Robert Gaizauskas and Yorick Wilks. Information extraction: Beyond document retrieval. *Journal of Documentation*, 54(1):70–105, 1998.

Claire Gardent and Evelyne Jacquey. Lexical reasoning. In *Proceedings of International Conference on Natural Language Processing (ICON'03)*, Mysore, 2003. URL `http://www.loria.fr/~gardent/publis/icon03.pdf`; 2006/02/28.

Claire Gardent, Hélène Manuélian and Eric Kow. Which bridges for bridging definite descriptions? In *Proceedings of the ACL/EACL Workshop "4th International Workshop on Linguistically Interpreted Corpora (LINC-03)"*, Budapest, 2003. URL `http://www.loria.fr/~gardent/publis/linc03.pdf`; 2006/08/24.

Claire Gardent and Bonnie [Lynn] Webber. Towards the use of automated reasoning in discourse disambiguation. *Journal of Logic, Language and Information*, 10:487–509, 2001. URL `ftp://ftp.coli.uni-sb.de/pub/people/claire/gardentwebber01.ps`; 2006/02/28.

Michael R. Garey and David S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, Oxford, 22nd printing (2000), 1979.

Michael R. Genesereth and Richard E. Fikes. *Knowledge Interchange Format. Version 3.0 Reference Manual*. Stanford, CA, 1992. URL `ftp://ksl.stanford.edu/pub/knowledge-sharing/papers/kif.ps`; 2006/08/24.

Rayid Ghani and Rosie Jones. A comparison of efficacy and assumptions of bootstrapping algorithms for training information extraction systems. In *Proceedings of the LREC Workshop "Linguistic Knowledge Acquisition and Representation – Bootstrapping Annotated Language Data*, pages 87–94, Las Palmas, 2002. URL `http://www-2.cs.cmu.edu/afs/cs/user/rosie/www/papers/ghanijoneslrec2002.pdf`; 2006/08/09.

Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002. URL `http://www.aclweb.org/anthology/J02-3001`; 2006/08/09.

Jonathan Ginzburg. Resolving questions, i. *Linguistics and Philosophy*, 18 (5):459–527, 1995a. URL `http://www.dcs.kcl.ac.uk/staff/ginzburg/rq1.ps`; 2006/01/11.

Jonathan Ginzburg. Resolving questions, ii. *Linguistics and Philosophy*, 18 (6):567–609, 1995b. URL `http://www.dcs.kcl.ac.uk/staff/ginzburg/rq2.ps`; 2006/01/11.

Jonathan Ginzburg. Interrogatives: Questions, facts and dialogue. In Lappin (1996), chapter 15, pages 385–422.

Jonathan Ginzburg and Ivan A. Sag. *Interrogative Investigations. The Form, Meaning and Use of English Interrogatives*. Number 123 in CSLI Lecture Notes. CSLI Publications, Stanford, CA, 2000.

Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik*, 38(1): 173–198, 1931. URL `http://www.springerlink.com/content/p03501kn35215860/`; 2006/10/25.

Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik*, 149(1): 1–29, 2006. URL `http://www.springerlink.com/content/48407101x70u20tj/`; 2006/10/25. Reprint of Gödel (1931).

Georg Gottlob and Christoph Koch. Monadic datalog and the expressive power of languages for web information extraction. *Journal of the ACM*, 51 (1):74–113, 2004. URL `http://doi.acm.org/10.1145/962446.962450`; 2006/04/25.

Georg Gottlob, Christoph Koch and Klaus U. Schulz. Conjunctive queries over trees. In *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2004)*, Paris, 2004. URL `http://www-db.cs.uni-sb.de/~koch/download/pods2004_159.pdf`; 2005/05/13.

Cordell Green. Theorem-proving by resolution as a basis for question-answering systems. In *Proceedings of the Fourth Annual Machine Intelligence Workshop (Machine Intelligence 4)*, pages 183–205, Edinburgh, 1969.

Gregory Grefenstette and Pasi Tapanainen. What is a word, what is a sentence? Problems of tokenization. In *Proceedings of the third International Conference on Computational Lexicography (COMPLEX'94)*, Budapest, 1994. URL `http://www.info.unicaen.fr/~giguet/pricai96/GrefenTapa94.ps`; 2006/08/08.

(Herbert) Paul Grice. *Studies in the Way of Words*. Harvard University Press, Cambridge, MA, London, 1989.

Jeroen Groenendijk. The logic of interrogation. Classical version. In *Proceedings of the Ninth Conference on Semantic and Linguistic Theory*, Santa Cruz, CA, 1999. URL `http://dare.uva.nl/document/1223`; 2006/03/22.

Jeroen Groenendijk and Martin Stokhof. On the semantics of questions and the pragmatics of answers. In *Varieties of Formal Semantics. Proceedings of the Fourth Amsterdam Colloquium, September 1982*, number 3 in Groningen-Amsterdam Studies in Semantics (GRASS), pages 143–170. Foris, Dordrecht, Cinnaminson, NJ, 1984.

Jeroen Groenendijk and Martin Stokhof. Questions. In J[ohan] van Benthem and A[lice] ter Meulen, editors, *Handbook of Logic and Language*, pages 1055–1124. Elsevier Science, Amsterdam, 1997. URL `http://dare.uva.nl/document/3703`; 2006/03/22.

Nicola Guarino. Formal ontology and information systems. In *Proceedings of the Formal Ontology in Information Systems (FOIS'98)*, pages 3–15, Trento, 1998. URL `http://citeseer.ist.psu.edu/guarino98formal.html`;2003/01/29.

R[amanathan] V. Guha and Douglas B. Lenat. Enabling agents to work together. *Communications of the ACM*, 37(7):126–142, 1994. URL `http://doi.acm.org/10.1145/176789.176804`; 2006/08/30.

Léon Gulikers, Gilbert Rattnik and Richard Piepenbrock. *CELEX German Linguistic User Guide*. Centre for Lexical Information, University of Nijmegen, Nijmegen, 1990. URL `http://www.ru.nl/celex/subsecs/celex_gug.pdf`; 2006/08/30.

Arvind Gupta and Naomi Nishimura. The complexity of subgraph isomorphism: Duality results for graphs of bounded path-and tree-width. Technical Report CS-95-14, David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada, 1995. URL `http://www.cs.uwaterloo.ca/research/tr/1995/14/95-14.pdf`; 2006/05/05.

Sanda M. Harabagiu. Deriving metonymic coercions from WordNet. In *Proceedings of the COLING-ACL Workshop "Usage of WordNet in Natural Language Processing Systems"*, Montreal, 1998. URL `http://www.aclweb.org/anthology/W98-0720`; 2006/08/30.

Sanda M. Harabagiu. Questions and intentions. In Strzalkowski and Harabagiu (2006), pages 99–147.

Sanda M. Harabagiu, Răzvan C. Bunescu and Steven J. Maiorano. Text and knowledge mining for coreference resolution. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, Pittsburg, PA, 2001a. URL `http://www.aclweb.org/anthology/N01-1008`; 2006/08/09.

Sanda [M.] Harabagiu, Andrew Hickl, John Lehmann and Dan [I.] Moldovan. Experiments with interactive question-answering. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 205–214, Ann Arbor, MI, 2005. URL `http://www.aclweb.org/anthology/P05-1026`; 2006/03/12.

Sanda [M.] Harabagiu and Finley Lacatusu. Strategies for advanced question answering. In *Proceedings of the Workshop on Pragmatics of Question Answering at HLT-NAACL 2004*, pages 1–9, Boston, MA, 2004. URL `http://acl.ldc.upenn.edu/hlt-naacl2004/qa/pdf/harabagiu-strategies.pdf`; 2006/01/19.

Sanda M. Harabagiu, George A. Miller and Dan I. Moldovan. WordNet 2 – A morphologically and semantically enhanced resource. In *Proceedings of the SIGLEX Workshop*, College Park, MD, 1999. URL `http://citeseer.ist.psu.edu/harabagiu99wordnet.html`; 2004/02/23.

Sanda [M.] Harabagiu and Dan I. Moldovan. Knowledge processing on extended WordNet. In Fellbaum (1998c), chapter 17, pages 379–406. URL `http://www.utdallas.edu/~sanda/papers/wnb1.ps.gz`; 2005/12/22.

Sanda [M.] Harabagiu, Dan [I.] Moldovan, Christine Clark, Mitchell Bowden, Andrew Hickl and Patrick Wang. Employing two question answering systems in TREC-2005. In *Proceedings of The Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, 2006. URL `http://trec.nist.gov/pubs/trec14/papers/lcc-sanda.qa.pdf`; 2007/01/16.

Sanda [M.] Harabagiu, Dan [I.] Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Răzvan [C.] Bunescu, Roxana Gîrju, Vasile Rus and Paul Morărescu. Falcon: Boosting knowledge for answer engines. In *Proceedings of The Ninth Text REtrieval Conference (TREC 9)*, Gaithersburg, MD, 2001b. URL `http://trec.nist.gov/pubs/trec9/papers/smu.pdf`; 2006/02/16.

Mariikka Haapalainen and Ari Majorin. GERTWOL: Ein System zur automatischen Wortformerkennung deutscher Wörter. `http://www.lingsoft.fi/doc/gertwol/intro/gertwol.txt`; 2006/07/29, 1994. In German.

Mariikka Haapalainen and Ari Majorin. GERTWOL und Morphologische Disambiguierung für das Deutsche. In *Proceedings of the 10th Nordic*

Conference on Computational Linguistics (Nordiska datalingvistdagarna, NoDaLiDa 95), Helsinki, 1995. URL `http://www.lingsoft.fi/doc/gercg/NODALIDA-poster.html`; 2006/07/29. In German.

V[olker] Haarslev and R[alf] Möller. Description of the RACER system and its applications. In *Proceedings International Workshop on Description Logics (DL-2001)*, pages 131–141, Stanford, CA, 2001a. URL `http://www.sts.tu-harburg.de/~r.f.moeller/papers/2001/HaMo01e.pdf`; 2006/10/24.

V[olker] Haarslev and R[alf] Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proceedings of Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 161–166, Seattle, WA, 2001b. URL `http://www.racer-systems.com/technology/contributions/2001/HaMo01c.pdf`; 2006/03/16.

Aria D. Haghighi, Andrew Y. Ng and Christopher D. Manning. Robust textual inference via graph matching. In *Proceedings of the Conference on Human Language Technology and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-05)*, Vancouver, 2005. URL `http://www.aclweb.org/anthology/H05-1049`; 2006/11/08.

Per-Kristian Halvorsen and Ronald M. Kaplan. Projections and semantic descriptions in Lexical-Functional Gramar. In Mary Dalrymple, Ronald M. Kaplan, John T. Maxwell III and Annie Zaenen, editors, *Formal Issus in Lexical-Functional Grammar*, number 47 in CSLI Lecture Notes, pages 279–292. CSLI Publications, Stanford, CA, 1995.

C[harles] L. Hamblin. Questions. *Australasian Journal of Philosophy*, 36(3): 159–168, 1958.

C[harles] L. Hamblin. Questions in Montague English. *Foundations of Language*, 10:41–53, 1973.

Birgit Hamp and Helmut Feldweg. Germanet – a lexical-semantic net for german. In *Proceedings of the ACL/EACL-97 Wordkshop "Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications"*, Madrid, 1997. URL `http://www.aclweb.org/anthology/W97-0802`; 2006/08/09.

David Harrah. The logic of questions. In D[ov] M. Gabbay and F[ranz] Guenthner, editors, *Handbook of Philosophical Logic*, volume II, pages 715–764. Kluwer, Dordrecht, 1984.

Sven Hartrumpf. Coreference resolution with syntactico-semantic rules and corpus statistics. In *Proceedings of the Fifth Computational Natural Language Learning Workshop (CoNLL-2001)*, pages 137–144, Toulouse, 2001. URL `http://www.aclweb.org/anthology/W01-0717`; 2005/03/16.

Sven Hartrumpf. *Hybrid Disambiguation in Natural Language Analysis*. Der Andere Verlag, 2003. Also Ph.D. Dissertation, Fachbereich Informatik, Fernuniversität Hagen.

Sven Hartrumpf. Question answering using sentence parsing and semantic network matching. In *Working Notes for the CLEF 2004 Workshop*, Bath, UK, 2004. URL `http://clef.isti.cnr.it/2004/working_notes/WorkingNotes2004/47.pdf`; 2005/02/03.

Roland Hausser and Dietmar Zaefferer. Questions and answers in a context-dependent montague grammar. In F[ranz] Guenthner and S[iegfried] J. Schmidt, editors, *Formal Semantics and Pragmatics for Natural Languages*, number 4 in Synthese Language Library, pages 339–358. Reidel, Dordrecht, 1979.

Gerhard Helbig and Joachim Buscha. *Deutsche Grammatik. Ein Handbuch für den Ausländerunterricht*. VEB Verlag Enzyklopädie, Leipzig, 12., unveränderte Auflage, 1989.

H[ermann] Helbig and C[arsten] Gnörlich. Multilayered extended semantic networks as a language for meaning representation in NLP systems. In *Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2002)*, volume 2276 of *LNCS*, pages 69–85, Ciudad de México, 2002. Springer. URL `http://www.springerlink.de/content/9r06f9lx6m43njpj`; 2006/11/23.

Gary G. Hendrix, Earl D. Sacerdoti, Daniel Sagalowicz and Jonathan Slocum. Developing a natural language interface to complex data. *ACM Transactions on Database Systems*, 3(2):105–147, 1978. URL `http://doi.acm.org/10.1145/320251.320253`; 2006/01/11.

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink and Ying Shi. Recognizing textual entailment with LCC's GROUNDHOG system. In Magnini and Dagan (2006). URL `http://ir-srv.cs.biu.ac.il:64080/RTE2/proceedings/14.pdf`; 2006/10/29.

James Higginbotham. The semantics of questions. In Lappin (1996), chapter 14, pages 361–383.

James Higginbotham and Robert May. Questions, quantifiers, and crossing. *Linguistic Review*, 1:41–79, 1981.

Thomas Hillenbrand. CITIUS ALTIUS FORTIUS: lessons learned from the theorem prover WALDMEISTER (invited paper). In *Proceedings of the 4th International Workshop on First-Order Theorem Proving (FTP)*, Valencia, 2003. URL `http://www.mpi-sb.mpg.de/~hillen/documents/Hil03.ps`; 2006/08/09.

L[ynette] Hirschman and R[obert] Gaizauskas. Natural language question answering: the view from here. *Natural Language Engineering*, 7(4):275–300, 2001. URL `http://dx.doi.org/10.1017/S1351324901002807`; 2005/06/13.

Lynette Hirschman and Henry S. Thompson. Overview of evaluation in speech and natural language processing. In Cole et al. (1997), pages 409–414.

Jerry R. Hobbs. Resolving pronoun references. *Lingua*, 44:311–338, 1978.

Tilman Höhle. Topologische Felder. `http://www.linguistik.uni-tuebingen.de/hoehle/manuskripte/Topologische_Felder.pdf`; 2006/08/30, March 1983. Manuscript.

John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

Chiori Hori, Takaaki Hori, Hajime Tsukada, Hideki Isozaki, Yutaka Sasaki and Eisaku Maeda. Spoken interactive ODQA system: SPIQA. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, Poster and Demonstration Session*, Sapporo, 2003. URL `http://www.aclweb.org/anthology/P03-2028`; 2005/12/23.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin and Deepak Ravichandran. Toward semantics-based answer pinpointing. In *Proceedings of the DARPA Human Language Technology conference (HLT)*, San Diego, CA, 2001. URL `http://www.isi.edu/natural-language/projects/webclopedia/pubs/01hlt.pdf`; 2004/03/08.

Eduard Hovy, Ulf Hermjakob and Deepak Ravichandran. A question/answer typology with surface text patterns. In *Proceedings of the DARPA Human Language Technology conference (HLT)*, San Diego, CA, 2002a. URL `http://www.isi.edu/natural-language/projects/webclopedia/pubs/02hlt.pdf`; 2004/03/08.

Eduard Hovy, Margaret [Maghi] King and Andrei Popescu-Belis. Principles of context-based machine translation evaluation. *Machine Translation*, 17(1): 43–75, 2002b. URL `http://www.springerlink.com/content/n2hw078662189615/`; 2007/01/09.

Ray S. Jackendoff. *Semantic Interpretation in Generative Grammar*. MIT Press, Cambridge, MA, 1972.

Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics (ROCLING X)*, pages 19–33, Taipei, 1997. URL `http://arxiv.org/pdf/cmp-lg/9709008`; 2006/06/13.

René Kager. *Optimality Theory*. Cambridge University Press, Cambridge, 1999.

Michael Kaisser and Tilman Becker. Question answering by searching large corpora with linguistic methods. In *Proceedings of The Thirteenth Text REtrieval Conference (TREC 2004)*, Gaithersburg, MD, 2005. URL `http://trec.nist.gov/pubs/trec13/papers/saarlandu.qa.pdf`; 2005/04/15.

Megumi Kameyama. Recognizing referential links: An information extraction perspective. In *Proceedings of the ACL/EACL '97 Workshop 'Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts'*, pages 46–53, Madrid, 1997. URL `http://www.aclweb.org/anthology/W97-1307`; 2006/08/24.

Hans Kamp and Uwe Reyle. *From Discourse to Logic. Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*, volume 42 of *Studies in Linguistics and Philosophy*. Kluwer, Dordrecht, 1993.

Lauri Karttunen. Syntax and semantics of questions. *Linguistics and Philosophy*, 1(1):3–44, 1977. URL `http://www.springerlink.com/openurl.asp?genre=article&issn=0165-0157&volume=1&issue=1&spage=3`; 2006/08/09. Reprinted as Karttunen (2002).

Lauri Karttunen. Syntax and semantics of questions. In Paul Portner and Barbara H. Partee, editors, *Formal Semantics. The Essential Readings*, volume 2 of *Linguistics: The Essential Readings*, pages 382–420. Blackwell Publishing, Oxford and Malden, MA, 2002.

Lauri Karttunen and Stanley Peters. Conventional implicature. In Choon-Kyu Oh and David A. Dinneen, editors, *Presupposition*, volume 11 of *Syntax and Semantics*, pages 1–56. Academic Press, New York, NY, San Francisco, CA, London, 1979. URL `http://www2.parc.com/istl/members/karttune/publications/ConvImp.pdf`; 2006/03/09.

Lauri Karttunen and Stanley Peters. Interrogative quantifiers. In Christian Rohrer, editor, *Time, Tense, and Quantifiers*, number 83 in Linguistische Arbeiten, pages 181–205. Niemeyer, Tübingen, 1980. URL `http://www2.parc.com/istl/members/karttune/publications/archive/interrogquant.pdf`; 2006/01/31.

Lauri Karttunen and Annie Zaenen. Veridicity. In *Proceedings of the Dagstuhl Seminar "Annotating, Extracting and Reasoning about Time and Events"*, Dagstuhl, 2005. URL `http://drops.dagstuhl.de/opus/volltexte/2005/314`; 2006/11/08.

Tsuneaki Kato, Jun'ichi Fukumoto and Fumito Masui. An overview of NTCIR-5 QAC3. In *Proceedings of the Fifth NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access*, Tōkyō, 2005. URL `http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings5/data/QAC/NTCIR5-OV-QAC-KatoT.pdf`; 2006/01/19.

Boris Katz, Matthew Bilotti, Sue Felshin, Aaron Fernandes, Wesley Hildebrandt, Roni Katzir, Jimmy Lin, Daniel Loreto, Gregory Marton, Federico Mora and Ozlem Uzuner. Answering multiple questions on a topic from heterogeneous resources. In *Proceedings of The Thirteenth Text REtrieval Conference (TREC 2004)*, Gaithersburg, MD, 2005. URL `http://trec.nist.gov/pubs/trec13/papers/mit.qa.pdf`; 2004/02/26.

Boris Katz and Beth Levin. Exploiting lexical regularities in designing natural language systems. In *Proceedings of the International Conference on Computational Linguistics (COLING-88)*, Budapest, 1988. URL `http://www.aclweb.org/anthology/C88-1065`; 2006/01/11.

Boris Katz and Jimmy Lin. Selectively using relations to improve precision in question answering. In *Proceedings of the EACL 2003 Workshop on Natural Language Processing for Question Answering*, Budapest, 2003. URL `http://www.umiacs.umd.edu/~jimmylin/publications/Katz-Lin-EACL03.pdf`; 2004/10/14.

Boris Katz, Jimmy Lin, Daniel Loreto, Wesley Hildebrandt, Matthew Bilotti, Sue Felshin, Aaron Fernandes, Gregory Marton and Federico Mora. Integrating web-based and corpus-based techniques for question answering. In *Proceedings of The Twelfth Text REtrieval Conference (TREC 2003)*, Gaithersburg, MD, 2004. URL `http://trec.nist.gov/pubs/trec12/papers/mit.qa.pdf`; 2005/12/22.

Daisuke Kawahara, Nobuhiro Kaji and Sadao Kurohasi. Question and answering system based on predicate-argument matching. In *Proceedings of the Third NTCIR Workshop*, Tōkyō, 2002. URL `http://citeseer.ist.psu.edu/564640.html`; 2003/07/09.

Martin Kay. The concrete lexicon and the abstract dictionary. In *Dictionaries in the Electronic Age: Fifth Annual Conference of the University of Waterloo Centre for the New OED*, pages 35–41, Oxford, 1989.

Edward L. Keenan and Robert D. Hull. The logical presuppositions of questions and answers. In János S. Petöfi and Dorothea Franck, editors, *Präsuppositionen in Philosophie und Linguistik*, volume 7 of *Linguistische Forschungen*, pages 441–466. Athenäum, Frankfurt am Main, 1973.

Andrew Kehler. A discourse copying algorithm for ellipsis and anaphora resolution. In *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics (EACL'93)*, Utrecht, 1993. URL `http://www.aclweb.org/anthology/E93-1025`; 2006/03/06.

Christopher Kennedy and Branimir Boguraev. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *Proceedings of the 16th International Conference on Computational Linguistics (Coling'96)*, volume 1, pages 113–118, København, 1996. URL `http://www.aclweb.org/anthology/C96-1021`; 2004/10/15.

Adam Kilgarriff. I don't believe in word senses. *Computers and Humanities*, 31(2):91–113, 1997. URL `http://www.springerlink.com/content/l02645r0805180l5/?p=5f886008a4594e87b65f2f6ac8d9ef6c&pi=1`; 2007/02/04. Also as ITRI-Report ITRI-97-12.

Pekka Kilpeläinen. *Tree Matching Problems with Applications to Structured Text Databases*. PhD thesis, Department of Computer Science, University of Helsinki, 1992. URL `http://www.cs.helsinki.fi/TR/A-1992/6/A-1992-6.ps.gz`; 2006/06/23. Also Report A-1992-6.

Margaret [Maghi] King. Users and user needs: Problems in the evaluation of knowledge discovery systems. In [Margaret] Maghi King and Nancy Underwood, editors, *Proceedings of the LREC 2004 Workshop "User-Oriented Evaluation of Knowledge Discovery Systems"*, pages 2–7, Lisboa, 2004.

Kimmo Koskenniemi. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. Department for General Linguistics. University of Helsinki, Finland, Helsinki, 1983. URL `http://www.ling.helsinki.fi/~koskenni/doc/Two-LevelMorphology.pdf`; 2006/07/29.

Milen Kouylekov and Bernardo Magnini. Recognizing textual entailment with tree edit distance algorithms. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment (RTE 1)*, Southampton, 2005. URL `http://www.cs.biu.ac.il/~glikmao/rte05/kouylekov_and_magnini.pdf`; 2006/01/05.

Milen Kouylekov and Bernardo Magnini. Tree edit distance for recognizing textual entailment: Estimating the cost of insertion. In Magnini and Dagan (2006). URL `http://ir-srv.cs.biu.ac.il:64080/RTE2/proceedings/12.pdf`; 2006/11/08.

Emiel Krahmer, Sebastiaan van Erk and André Verleg. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72, 2003. URL `http://www.aclweb.org/anthology/J03-1003`; 2006/04/19.

Manfred Krifka. For a structured account of questions and answers. In Caroline Féry and Wolfgang Sternefeld, editors, *Audiatur vox sapientiae. A Festschrift for Arnim von Stechow*, pages 287–319. Akademie Verlag, Berlin, 2001. URL `http://vivaldi.sfs.nphil.uni-tuebingen.de/~arnim10/Festschrift/Krifka-5-komplett\%20fertig.pdf`; 2006/01/11.

Claudia Kunze and Lothar Lemnitzer. Adapting GermaNet for the web. In *Proceedings of the First Global WordNetConference*, pages 174–181, Mysore, 2002a. URL `http://www.sfs.uni-tuebingen.de/~lothar/publ/GWA_GN_paper.ps`; 2006/08/09.

Claudia Kunze and Lothar Lemnitzer. Germanet – representation, visualization, application. In *Proceedings of the LREC 2002*, volume V, pages 1485–1491, Las Palmas, 2002b. URL `http://www.sfs.uni-tuebingen.de/~lothar/publ/LREC_main_paper.ps`; 2004/04/19.

Claudia Kunze and Andreas Wagner. Integrating GermaNet into EuroWordNet, a multi-lingual lexical-semantic database. *Sprache und Datenverarbeitung*, 23(2):5–20, 1999.

Julian Kupiec. Murax: a robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '93)*, pages 181–190, Pittsburgh, PA, 1993. ACM Press. URL `http://doi.acm.org/10.1145/160688.160717`; 2005/03/29.

Namhee Kwon, Michael Fleischman and Eduard Hovy. Frame-Net based semantic parsing using maximum entropy models. In *Proceedings of the 21st International Conference on Computational Linguistics (COLING)*, Genève, 2004a. URL `http://www.mit.edu/~mbf/COLING_2004.pdf`; 2004/07/05.

Namhee Kwon, Michael Fleischman and Eduard Hovy. SENSEVAL automatic labeling of semantic roles using maximum entropy models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, SENSEVAL-3 Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, 2004b. URL `http://www.mit.edu/~mbf/senseEval_04.pdf`; 2004/07/04.

Shalom Lappin, editor. *The Handbook of Contemporary Semantic Theory*. Blackwell, Cambridge, MA, 1996.

Shalom Lappin and Herbert J. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561, 1994. URL `http://www.aclweb.org/anthology/J94-4002`; 2004/10/18.

Benoit Lavoie and Owen Rambow. A fast and portable realizer for text generation systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP'97)*, Washington, DC, 1997. URL `http://www.aclweb.org/anthology/A97-1039`; 2006/08/22.

Claudia Leacock and Martin Chodorow. Combining local context and WordNet similarity for word sense identification. In Fellbaum (1998c), pages 265–283.

Wendy G. Lehnert. *The Process of Question Answering. A Computer Simulation of Cognition*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1978.

Marc Light, Gideon S. Mann, Ellen Riloff and Eric [J.] Breck. Analyses for elucidating current question answering technology. *Natural Language Engineering*, 7(4):325–342, 2001. URL `http://journals.cambridge.org/action/displayFulltext?type=1&fid=96170&jid=&volumeId=&issueId=04&aid=96169`; 2006/08/09.

Dekang Lin. Dependency-based evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems*, Granada, 1998a. URL `http://www.cs.ualberta.ca/~lindek/papers/granada.ps`; 2006/03/01.

Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, Madison, WI, 1998b. URL `ftp://ftp.cs.umanitoba.ca/pub/lindek/papers/sim.ps.gz`; 2006/01/05.

Dekang Lin. Dependency-based evaluation of MINIPAR. In Abeillé (2003), pages 317–329. URL `http://www.cfilt.iitb.ac.in/archives/minipar_evaluation.pdf`; 2006/03/01.

Dekang Lin and Patrick Pantel. Dirt – discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 323–328, San Francisco, CA, 2001a. URL `http://doi.acm.org/10.1145/502512.502559`; 2006/01/05.

Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360, 2001b. URL `http://journals.cambridge.org/action/displayFulltext?type=1&fid=96162&jid=&volumeId=&issueId=04&aid=96161`; 2006/08/09.

Jimmy Lin, Dennis Quan, Vineet Sinha, Karun Bakshi, David Huynh, Boris Katz and David R. Karger. What makes a good answer? The role of context in question answering. In *Proceedings of the Ninth IFIP TC13 International Conference on Human-Computer Interaction (INTERACT 2003)*, Zürich, 2003. URL `http://www.umiacs.umd.edu/~jimmylin/publications/Lin-etal-INTERACT03.pdf`; 2004/10/14.

David Luckham and Nils J. Nilsson. Extracting information from resolution proof trees. *Artificial Intelligence*, 2:27–54, 1971.

Bernardo Magnini and Ido Dagan, editors. *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment (RTE*

2), Venezia, 2006. URL `http://ir-srv.cs.biu.ac.il:64080/RTE2/proceedings/RTE-2Proceedings.pdf`; 2006/08/08.

Bernardo Magnini, Danilo Giampiccolo, Pamela Forner, Christelle Ayache, Valentin Jijkoun, Petya Osenova, Anselmo Peñas, Paulo Rocha, Bogdan Sacaleanu and Richard Sutcliffe. Overview of the CLEF 2006 multilingual question answering track. In *Working Notes for the CLEF 2006 Workshop*, Alicante, 2006. URL `http://www.clef-campaign.org/2006/working_notes/workingnotes2006/magniniOCLEF2006.pdf`; 2006/12/15.

Bernardo Magnini, Simone Romagnoli, Alessandro Vallin, Jesús Herrera, Anselmo Peñas, Víctor Peinado, Felisa Verdejo and Marten de Rijke. Creating the DISEQuA corpus: a test set for multilingual question answering. In *Working Notes for the CLEF 2003 Workshop*, Trondheim, Norway, 2003. URL `http://www.clef-campaign.org/2003/WN_web/37.pdf`; 2006/02/15.

Bernardo Magnini, Alessandro Vallin, Christelle Ayache, Gregor Erbach, Anselmo Peñas, Marten de Rijke, Paulo Rocha, Kiril Simov and Richard Sutcliffe. Overview of the CLEF 2004 multilingual question answering track. In *Working Notes for the CLEF 2004 Workshop*, Bath, UK, 2004. URL `http://www.clef-campaign.org/2004/working_notes/WorkingNotes2004/35.pdf`; 2006/02/15.

Christopher D. Manning. Local textual inference: It's hard to circumscribe, but you know it when you see it – and NLP needs it. `http://nlp.stanford.edu/~manning/papers/LocalTextualInference.pdf`; 2006/10/25, 2006.

Daniel Marcu and Ana-Maria Popescu. Towards developing probabilistic generative models for reasoning with natural language representations. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2005)*, pages 88–99, Ciudad de México, 2005. URL `http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=3406&spage=88`; 2006/02/09.

Katja Markert and Udo Hahn. Metonymies in discourse. *Artificial Intelligence*, 135(1–2):145–198, 2002. URL `http://www.comp.leeds.ac.uk/markert/Papers/AI2002.pdf`; 2006/08/09.

Erwin Marsi, Emiel Krahmer, Wauter Bosma and Mariët Theune. Normalized alignment of dependency trees for detecting textual entailment. In Magnini and Dagan (2006), pages 50–55. URL `http://ir-srv.cs.biu.ac.il:64080/RTE2/proceedings/10.pdf`; 2006/11/08.

Paul Martin, Douglas E. Appelt, Barbara J. Grosz and Fernando Pereira. Team: An experimental transportable natural-language interface. In *Proceedings of 1986 ACM Fall Joint Computer Conference (ACM'86)*, pages 260–267, Dallas, TX, 1986. URL `http://portal.acm.org/ft_gateway.cfm?id=324576&type=pdf&coll=GUIDE&dl=ACM&CFID=11111111&CFTOKEN=2222222`; 2006/08/09.

Megumi Matsuda and Jun'ichi Fukumoto. Answering questions of IAD task using reference resolution of follow-up questions. In *Proceedings of the Fifth NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access*, Tōkyō, 2005. URL `http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings5/data/QAC/NTCIR5-QAC-MatsudaM.pdf`; 2006/08/09.

Cynthia Matuszek, John Cabral, Michael Witbrock and John DeOliveira. An introduction to the syntax and content of Cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, Stanford, CA, 2006. URL `http://www.cyc.com/doc/white_papers/AAAI06SS-SyntaxAndContentOfCyc.pdf`; 2006/08/24.

Mark T. Maybury. Toward a question answering roadmap. `http://www.mitre.org/work/tech_papers/tech_papers_02/maybury_toward/maybury_toward_qa.pdf`; 2004/04/02, 2002.

Mark T. Maybury, editor. *New Directions in Question Answering*. AAAI Press, Cambridge, MA, 2004a.

Mark [T.] Maybury. Question answering: An introduction. In Maybury (2004a), pages 3–14.

William McCune. Otter 3.3 reference manual. ANL/MCS-TM- 263, Argonne National Laboratory, Argonne, IL, August 2003. URL `http://www.cs.unm.edu/~mccune/otter/Otter33.pdf`; 2006/10/24. 2003.

Igor A. Mel'čuk. *Dependency Syntax: Theory and Practice*. State University of New York Press, Albany, NY, 1988.

Igor [A.] Mel'čuk and Alexander Zholkovsky. The explanatory combinatorial dictionary. In Martha Walton Evens, editor, *Relational Models of the Lexicon. Representing Knowledge in Semantic Networks*, Studies in Natural Language Processing, pages 41–74. Cambridge University Press, Cambridge, 1988.

Wolfgang Menzel. Robust processing of natural language. In *Proceedings of the 19th Annual German Conference on Artificial Intelligence (KI '95)*, pages 19–34, Bielefeld, 1995. URL `http://arxiv.org/pdf/cmp-lg/9507003`; 2006/08/08.

B[runo] T. Messmer and H[orst] Bunke. Subgraph isomorphism in polynomial time. Technical Report IAM 95-003, Institut für Informatik und angewandte Mathematik, Universität Bern, Bern, 1995. URL `http://www.iam.unibe.ch/publikationen/techreports/1995/iam-95-003/file/at_download`; 2006/04/19.

Rada Mihalcea and Dan [I.] Moldovan. eXtended WordNet: Progress report. In *Proceedings of NAACL Workshop on WordNet and Other Resources*, pages 95–100, Pittsburgh, PA, 2001. URL `http://www.cs.unt.edu/~rada/papers/mihalcea.naacl.wn01a.ps`; 2006/09/20.

George A. Miller. Foreword. In Fellbaum (1998c), pages xv–xxii.

George A. Miller. Nouns in WordNet. In Fellbaum (1998c), pages 23–46.

George A. Miller and Florentina Hristea. Wordnet nouns: Classes and instances. *Computational Linguistics*, 32(1):1–3, 2006.

Katherine J. Miller. Modifiers in WordNet. In Fellbaum (1998c), pages 47–67.

Scott Miller, Michael Crystal, Heidi Fox, Lance Ramshaw, Richard Schwartz, Rebecca Stone, Ralph Weischedel and the Annotation Group. Algorithms that learn to extract information – BBN: Description of the sift system as used for muc-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, Washington, DC, 1998. URL `http://www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_proceedings/bbn_muc7.pdf`; 2006/08/09.

Ruslan Mitkov. *Anaphora Resolution*. Pearson, London, 2002.

Dan [I.] Moldovan, Christine Clark, Sanda [M.] Harabagiu and Steve[n J.] Maiorano. COGEX: A logic prover for question answering. In *Proceedings of HLT-NAACL 2003*, pages 87–93, Edmonton, 2003a. URL `http://www.aclweb.org/anthology/N03-1022`; 2004/08/16.

Dan [I.] Moldovan, Sanda [M.] Harabagiu, Roxana Girju, Paul Morarescu, Finley Lacatusu, Adrian Novischi, Adriana Badulescu and Orest Bolohan. LCC tools for question answering. In *Proceedings of The Eleventh Text REtrieval Conference (TREC 2002)*, Gaithersburg, MD, 2003b. URL `http://trec.nist.gov/pubs/trec11/papers/lcc.moldovan.pdf`; 2004/02/02.

Dan [I.] Moldovan and Adrian Novischi. Lexical chains for question answering. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei, 2002. URL `http://www.aclweb.org/anthology/C02-1167`; 2005/12/15.

Dan I. Moldovan and Vasile Rus. Logic form transformation of WordNet and its applicability to question answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, 2001. URL `http://www.aclweb.org/anthology/P01-1052`; 2004/02/11.

Diego Mollá and Stephen Wan. Macquarie University at DUC 2006: Question answering for summarisation. In *Proceedings of the Document Understanding Conference (DUC 2006)*, New York, NY, 2006. URL `http://www.ics.mq.edu.au/~diego/publications/DUC2006.pdf`; 2007/01/12.

Richard Montague. The proper treatment of quantification in ordinary english. In K. J[aakko] J. Hintikka, J[ulius] M. E. Moravcsik and P[atrick] Suppes, editors, *Approaches to Natural Languages: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*, pages 221–242. D. Reidel, Dordrecht, 1973.

Richard Montague. The proper treatment of quantification in ordinary english. In Richmond H. Thomason, editor, *Formal Philosophy*, pages 247–270. Yale University Press, New Haven, CT, London, 1974. Reprint of Montague (1973).

M[anuel] Montes-y-Gómez, A[lexander] Gelbukh, A[urelio] López-López and R[icardo] Baeza-Yates. Flexible comparison of conceptual graphs. In *Proceedings of the 12th International Conference and Workshop on Database and Expert Systems Applications (DEXA)*, volume 2113 of *Lecture Notes in Computer Science*, pages 102–111, München, 2001. URL `http://www.gelbukh.com/CV/Publications/2001/DEXA-2001-Flexible.pdf`; 2005/03/02.

Christof Monz. Document retrieval in the context of question answering. In *Proceedings of the 25th European Conference on Information Retrieval Research (ECIR-03)*, number 2633 in Lecture Notes in Computer Science, pages 571–579, Pisa, 2003a. Springer. URL `http://www.dcs.qmul.ac.uk/~christof/publications/ecir03.pdf`; 2005/10/28.

Christof Monz. *From Document Retrieval to Question Answering*. PhD thesis, Institue for Logic, Language and Computation, Universiteit van Amsterdam, 2003b.

Christof Monz and Marten de Rijke. Light-weight entailment checking for computational semantics. In *Proceedings of the 3rd Workshop on Inference in Computational Semantics (ICoS-3)*, pages 59–72, Siena, 2001. URL `http://www.dcs.qmul.ac.uk/~christof/publications/icos3.pdf`; 2006/01/05.

Alessandro Moschitti, Paul Morărescu and Sanda [M.] Harabagiu. Open domain information extraction via automatic semantic labeling. In *Proceedings of the 2003 Special Track on Recent Advances in Natural Language at the 16th International FLAIRS Conference*, St. Augustine, FL, 2003. URL `http://ai-nlp.info.uniroma2.it/moschitti/articles/flairs03.pdf`; 2004/09/07.

Christoph Müller and Michael Strube. Annotating anaphoric and bridging relations with MMAX. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*, pages 1–6, Aalborg, Denmark, 2001a. URL `http://www.aclweb.org/anthology/W01-1612`; 2006/10/25.

Christoph Müller and Michael Strube. MMAX: A tool for the annotation of multi-modal corpora. In *Proceedings of the 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 45–50, Seattle, WA, 2001b. URL `http://www.eml-research.de/english/Research/NLP/ijcaiws01.ps.gz`; 2006/10/25.

Stefan Müller and Walter Kasper. HPSG analysis of german. In *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 238–253. Springer, Berlin, 2000.

Rowan Nairn, Cleo Condoravdi and Lauri Karttunen. Computing relative polarity for textual inference. In *Proceedings of the 5th International Workshop on Inference in Computational Semantics (ICoS-5)*, Buxton, 2006. URL `http://www.cs.man.ac.uk/~ipratt/ICoS-5/ICOS-5-PDF/Nairn.pdf`; 2006/05/09.

Srini Narayanan, Charles J. Fillmore, Collin F. Baker and Miriam R. L. Petruck. FrameNet meets the semantic web: A DAML+OIL frame representation. In *Proceedings of the 8th National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. URL `http://framenet.icsi.berkeley.edu/~framenet/papers/semweblr.pdf`; 12-jun-2003.

Srini Narayanan and Sanda [M.] Harabagiu. Answering questions using advanced semantics and probabilistic inference. In *Workshop on Pragmatics of Question Answering, HLT-NAACL*, Boston, MA, 2004a. URL `http://www.icsi.berkeley.edu/\~snarayan/prag.pdf`; 2005/11/28.

Srini Narayanan and Sanda [M.] Harabagiu. Question answering based on semantic structures. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-2004)*, Genève, 2004b. URL `http://www.hlt.utdallas.edu/papers/837.pdf`; 2005/11/28.

Günter Neumann and Feiyu Xu. Mining answers in German web pages. In *Proceedings of The International Conference on Web Intelligence (WI 2003)*, Halifax, Canada, 2003. URL `http://www.dfki.de/~neumann/publications/new-ps/neumanng_german.pdf`; 2004/02/24.

Eamonn Newman, Nicola Stokes, John Dunnion and Joe Carthy. UCD IIRG approach to the textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment (RTE 1)*, Southampton, 2005. URL `http://www.cs.biu.ac.il/~glikmao/rte05/newman_et_al.pdf`; 2006/11/08.

Vincent Ng and Claire Cardie. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 104–111, Philadelphia, PA, 2002. URL `http://www.aclweb.org/anthology/P02-1014`; 2006/08/24.

H. Penny Nii. Blackboard systems. Technical Report STAN-CS-86-1123, Knowledge Systems Laboratory, Departments of Medical and Computer Science, Stanford University, Stanford, CA, 1986. URL `ftp://reports.stanford.edu/pub/cstr/reports/cs/tr/86/1123/CS-TR-86-1123.pdf`; 2006/08/08. Also appeared in AI Magazine 7(2), 38–53, and 7(3), 82–106.

Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS'01)*, pages 2–9, Ogunquit, ME, 2001. URL `http://doi.acm.org/10.1145/505168.505170`; 2006/08/09.

Sergei Nirenburg and Victor Raskin. *Ontological Semantics*. MIT Press, Cambridge, MA, London, 2004.

Hans de Nivelle. A resolution decision procedure for the guarded fragment. In *Proceedings of the Conference on Automated Deduction (CADE-15)*, number 1421 in Lecture Notes in Computer Science, pages 191–204. Springer, 1998. URL `http://www.springerlink.com/content/4e7u1d8lkr6847yy/`; 2006/10/24.

Geoffrey Nunberg, Ivan A. Sag and Thomas Wasow. Idioms. *Language*, 70(3), 1994. URL `http://lingo.stanford.edu/sag/papers/idioms.pdf`; 2005/09/16.

Eric Nyberg, John [D.] Burger, Scott Mardis and David Ferrucci. Software architectures for advanced QA. In Maybury (2004a), pages 19–29.

Stephan Oepen and John Carroll. Ambiguity packing in constraint-based parsing practical results. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, Seattle, WA, 2000. URL `http://www.aclweb.org/anthology/A00-2022`; 2006/08/30.

Oracle. Oracle database 10g release 2. XML DB. an oracle white paper. `http://download.oracle.com/otndocs/tech/xml/xmldb/TWP_XML_DB_10gR2_long.pdf`; 2006/11/23, May 2005.

Sebastian Padó and Mirella Lapata. Cross-linguistic projection of role-semantic information. In *Proceedings of HLT/EMNLP 2005*, Vancouver, BC, 2005. URL `http://www.aclweb.org/anthology/H05-1108`; 2006/07/05.

Martha Palmer, Daniel Gildea and Paul Kingsbury. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31 (1):71–105, 2005. URL `http://verbs.colorado.edu/mpalmer/palmer/papers/prop.pdf`; 2006/08/09.

Siddharth Patwardhan, Satanjeev Banerjee and Ted Pedersen. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, volume 2588 of *Lecture Notes in Computer Science*, Ciudad de México, 2003. URL `http://www.d.umn.edu/~tpederse/Pubs/cicling2003-3.pdf`; 2005/01/07.

Adam Pease and Ian Niles. IEEE Standard Upper Ontology: A progress report. *Knowledge Engineering Review*, 17:65–70, 2002. URL `http://home.earthlink.net/~adampease/professional/KER.ps`; 2006/08/09. Special Issue on Ontologies and Agents.

Ted Pedersen, Siddharth Patwardhan and Jason Michelizzi. WordNet::Similarity – measuring the relatedness of concepts. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 04)*, pages 38–41, Boston, MA, 2004. URL `http://acl.eldoc.ub.rug.nl/mirror/hlt-naacl2004/demos/pdf/pedersen2.pdf`; 2006/06/13.

Manfred Pinkal. Definite noun phrases and the semantics of discourse. In *Proceedings of the 11th Conference Computational Linguistics (ACL'86)*, New York, NY, 1986. URL `http://www.aclweb.org/anthology/C86-1088`; 2006/08/30.

Manfred Pinkal. Radical underspecification. In *In Proceedings of the 10th Amsterdam Colloquium*, pages 587–606, Amsterdam, 1995. URL `ftp://ftp.coli.uni-sb.de/pub/coli/claus/claus72.ps.gz`; 2006/08/10.

Massimo Poesio, Olga Uryupina, Renata Vieira, Mijail Alexandrov-Kabadjov and Rodrigo Goulart. Discourse-new detectors for definite description resolution: A survey and a preliminary proposal. In *Proceedings of the ACL Workshop on Reference Resolution*, Barcelona, 2004. URL `http://cswww.essex.ac.uk/staff/poesio/publications/ACL04_refres.pdf`; 2005/04/08.

Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. University of Chicago Press, Chicago, IL, London, 1994.

Ana-Maria Popescu, Oren Etzioni and Henry Kautz. Towards a theory of natural language interfaces to databases. In *Proceedings of the Conference on Intelligent User Interfaces (IUI'03)*, pages 149–157, Miami, FL, 2003. URL `http://doi.acm.org/10.1145/604045.604070`; 2006/02/28.

M[artin] F. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

M[artin] F. Porter. An algorithm for suffix stripping. In Karen Sparck Jones and Peter Willett, editors, *Readings in Information Retrieval*, pages 313–316. Morgan Kaufmann, 1997. Reprint of Porter (1980).

Oana Postolache and Corina Forăscu. A coreference resolution model on excerpts from a novel. In *Proceedings of the Ninth European Summer School in Logic Language and Information Student Session (ESSLLI'2004)*, Nancy, 2004. URL `http://www.coli.uni-saarland.de/~oana/publications/postolache-forascu-04.pdf`; 2004/09/09.

John M. Prager, Jennifer Chu-Carroll, Eric W. Brown and Krzysztof Czuba. Question answering by predictive annotation. In Strzalkowski and Harabagiu (2006), pages 349–382.

Uta E. Priss. The formalization of WordNet by methods of relational concept analysis. In Fellbaum (1998c), pages 179–196. URL `http://upriss.org.uk/papers/mitpaper.pdf`; 2006/08/09.

Vasin Punyakanok, Dan Roth and Wen-tau Yih. Natural language inference via dependency tree mapping: An application to question answering. *Computational Linguistics*, 2007. URL `http://l2r.cs.uiuc.edu/~danr/Teaching/CS598-04/Papers/PRY-ACLSquib.pdf`; 2006/01/05. In Submission.

Yonggang Qiu and H[ans] P[eter] Frei. Concept based query expansion. In *Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval*, pages 160–169, Pittsburgh, PA, 1993. URL `http://doi.acm.org/10.1145/160688.160713`; 2006/02/28.

J. Joachim Quantz. Semantische Repräsentation anaphorischer Bezüge in terminologischen Logiken. KIT Report 96, Technische Universität Berlin, Berlin, 1992. URL `ftp://ftp.cs.tu-berlin.de/pub/local/kit/documents/KIT-Reports/r096.ps.gz`; 2004/10/18. In German.

Uwe Quasthoff, Christian Biemann and Christian Wolff. Named entity learning and verification: Expectation maximization in large corpora. In *Proceedings of the Sixth Workshop on Computational Language Learning (CoNLL-2002)*, Taipei, 2002. URL `http://wortschatz.informatik.uni-leipzig.de/asv/publikationen/Quast-Biem-Conll-2002.pdf`; 2004/04/15.

Deepak Ramachandran, Pace Reagan and Keith Goolsbey. First-orderized researchcyc: Expressivity and efficiency in a common-sense ontology. In *Papers from the AAAI Workshop on Contexts and Ontologies: Theory, Practice and Applications*, Pittsburgh, PA, 2005. URL `http://www.cyc.com/doc/white_papers/folification.pdf`; 2006/08/30.

Deepak Ravichandran and Eduard Hovy. Learning surface patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 41–47, Philadelphia, PA, 2002. URL `http://www.isi.edu/natural-language/projects/webclopedia/pubs/02ACL-patterns.pdf`; 2004/02/02.

Ingo Reich. *Frage, Antwort und Fokus*. Number 55 in studia grammatica. Akademie Verlag, Berlin, 2003. In German.

Ehud Reiter. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the 7th International Workshop on Natural Language Generation*, pages 163–170, Kennebunkport, ME, 1994. URL `http://www.aclweb.org/anthology/W94-0319`; 2006/08/30.

Ehud Reiter. NLG vs. templates. In *Proceedings of the 5th European Workshop on Natural Language Generation*, Leiden, 1995. URL `http://arxiv.org/pdf/cmp-lg/9504013`; 2006/08/30.

Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87, 1997. URL `http://www.ics.mq.edu.au/~rdale/publications/papers/1997/jnle97.pdf`; 2006/08/30.

Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press, Cambridge, 2000.

Philip Resnik. Using information content to evaluate semantic similarity. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, 1995. URL `http://arxiv.org/pdf/cmp-lg/9511007`; 2006/06/16.

Uwe Reyle. Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics*, 10(2):123–179, 1993. URL `http://jos.oxfordjournals.org/cgi/reprint/10/2/123`; 2006/08/10.

Ellen Riloff. Little words can make a big difference for text classification. In *Proceedings of the 18th Annual International ACM SIGIR Conference On Research and Development in Information Retrieval (SIGIR '95)*, pages 130–136, Seattle, WA, 1995. URL `http://doi.acm.org/10.1145/215206.215349`; 2006/08/24.

Ellen Riloff and Rosie Jones. Learning dictionaries for information extraction by multi-level boot-strapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 1044–1049, Orlando, FL, 1999. URL `http://www.cs.utah.edu/~riloff/pdfs/aaai99.pdf`; 2006/02/28.

Marc Rössler. Using Markov models for named entity recognition in German newspapers. In *Proceedings of the ESSLLI'02 Workshop on Machine Learning Approaches in Computational Linguistics*, Trento, 2002. URL `http://cl.informatik.uni-duisburg.de/roessler/esslli02.pdf`; 2004/04/15.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck and Christopher R. Johnson. Framenet: Theory and practice. `http://framenet.icsi.berkeley.edu/book/book.html`; 2006/08/08, July 2005.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson and Jan Scheffczyk. FrameNet II: Extended theory and practice. `http://framenet.icsi.berkeley.edu/book/book.pdf`; 2006/10/05, August 2006.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence. A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 1995.

Ivan A. Sag, Francis Bond, Ann Copestake and Dan Flickinger. Multiword expressions. LinGO Working Paper 2001-01, CSLI Linguistic Grammars Online (LinGO) Lab at Stanford University, 2001. URL `http://lingo.stanford.edu/pubs/WP-2001-01.pdf`; 2005/09/16.

Yutaka Sasaki, Hsin-Hsi Chen, Kuang-hua Chen and Chuan-Jie Lin. Overview of the NTCIR-5 cross-lingual question answering task (CLQA1). In *Proceedings of the Fifth NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access*, Tōkyō, 2005. URL `http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings5/data/CLQA/NTCIR5-OV-CLQA-SasakiY.pdf`; 2006/01/19.

Roger C. Schank. Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology*, 3(4): 552–631, 1972. URL `http://www.sciencedirect.com/science/article/B6WCR-4D6RJBY-21/2/296eac0de8ec69a2125e5e241129ac72`; 2006/10/05.

Roger C. Schank. Identification of conceptualizations underlying natural language. In Roger C. Schank and Kenneth Mark Colby, editors, *Computer Models of Thought*, pages 187–247. Morgan Freeman, San Francisco, CA, 1973.

Roger C. Schank and Robert P. Abelson. *Scripts, Plans, Goals and Understaning. An Inquiry into Human Knowledge Structures*. Lawrence Erlbaum, Hillsdale, NJ, 1977.

Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing (NEMLAP'94)*, Manchester, 1994. URL `http://www.ims.uni-stuttgart.de/projekte/gramotron/PAPERS/MISC/NEMLAP97-TreeTagger.ps.gz`; 2006/08/30.

Helmut Schmid. LoPar. Design and implementation. Arbeitspapiere des Sonderforschungsbereichs 340 "Linguistic Theory and the Foundations of Computational Linguistics" 149, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, 2000. URL `http://www.ims.uni-stuttgart.de/projekte/gramotron/PAPERS/MANUALS/lopar.ps.gz`; 2006/08/09.

Sabine Schulte im Walde. *Experiments on the Automatic Induction of German Semantic Verb Classes*. PhD thesis, Institut für maschinelle Sprachverarbeitung, Universität Stuttgart, 2003. URL `http://www.coli.uni-saarland.de/~schulte/Theses/PhD-Thesis/phd-thesis.pdf`; 2006/08/09.

John R. Searle. *Speech Acts. An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, New York, New Rochelle, Melbourne, Sydney, 1969. Reprint, 1990.

Chung-chieh Shan and Balder ten Cate. The partition semantics of questions, syntactically. In *Proceedings of the 14th European Summer School in Logic, Language and Information (ESSLLI), Student Session*, Trento, 2002. URL `http://staff.science.uva.nl/~bcate/papers/esslli02ss.pdf`; 2006/03/22.

Sergej Sizov, Michael Biwer, Jens Graupmann, Stefan Siersdorfer, Martin Theobald, Gerhard Weikum and Patrick Zimmer. The BINGO! system for information portal generation and expert web search. In *First Semiannual Conference on Innovative Data Systems Research (CIDR)*, Asilomar, CA, 2003. URL `http://www.mpi-inf.mpg.de/~sizov/sources/2003/cidr2003.pdf`; 2002/12/04.

Wojciech Skut and Thorsten Brants. Chunk tagger – statistical recognition of noun phrases. In *Proceedings of the ESSLLI Workshop on Automated Acquisition of Syntax and Parsing*, Saarbrücken, 1998a. URL `http://www.coli.uni-saarland.de/~thorsten/publications/Skut-Brants-ESSLLI-Parsing98.pdf`; 2006/08/09.

Wojciech Skut and Thorsten Brants. A maximum-entropy partial parser for unrestricted text. In *Proceedings of the Sixth Workshop on Very Large Corpora*, Montreal, 1998b. URL `http://www.coli.uni-saarland.de/~thorsten/publications/Skut-Brants-WVLC98.pdf`; 2006/08/09.

Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1–3):233–272, 1999. URL `http://springerlink.metapress.com/openurl.asp?genre=article&issn=0885-6125&volume=34&issue=1&spage=233`; 2005/12/15.

Wee Meng Soon, Daniel Chung Yong Lim and Hwee Tou Ng. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544, 2001. URL `http://www.aclweb.org/anthology/J01-4004`; 2006/08/30.

Martin M. Soubbotin. Patterns of potential answer expressions as clues to the right answers. In *Proceedings of The Tenth Text REtrieval Conference (TREC 2001)*, Gaithersburg, MD, 2002. URL `http://trec.nist.gov/pubs/trec10/papers/insight_trec10.pdf`; 2004/02/02.

Martin M. Soubbotin and Sergei M. Soubbotin. Use of patterns for detection of answer strings: A systematic approach. In *Proceedings of The Eleventh Text REtrieval Conference (TREC 2002)*, Gaithersburg, MD, 2003. URL `http://trec.nist.gov/pubs/trec11/papers/insightsoftm.sergei.pdf`; 2003/07/09.

Karen Sparck Jones and Julia R. Galliers. *Evaluating Natural Language Processing Systems. An Analysis and Review*. Number 1083 in Lecture Notes in Artificial Intelligence. Springer, Berlin, 1996.

Rohini [K.] Srihari and Wei Li. A question answering system supported by information extraction. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP)*, Seattle, WA, 2000. URL `http://www.aclweb.org/anthology/A00-1023`; 2005/11/29.

Rohini K. Srihari, Wei Li and Xiaoge Li. Question answering supported by multiple levels of information extraction. In Strzalkowski and Harabagiu (2006), pages 349–382.

Arnim von Stechow. Focusing and backgrunding operators. In Werner Abraham, editor, *Discourse Particles*, volume 12 of *Pragmatics and Beyond. New Series*, pages 37–84. John Benjamins, Amsterdam, Philadelphia, PA, 1991.

Manfred Stede. The search for robustness in natural language understanding. *Artificial Intelligence Review*, 6:383–414, 1992. URL `http://www.springerlink.com/openurl.asp?genre=article&issn=0269-2821&volume=6&issue=4&spage=383`; 2006/08/08.

Mark Stevenson and Robert Gaizauskas. Improving named entity recognition using annotated corpora. In *Proceedings of the LREC Workshop "Information Extraction Meets Corpus Linguistics"*, pages 24–30, Athens, 2000. URL `ftp://ftp.dcs.shef.ac.uk/home/robertg/papers/lrec00_iews_ne.ps`; 2006/08/09.

Oliviero Stock, Rino Falcone and Patrizia Insinnamo. Island parsing and bidirectional charts. In *Proceedings of the International Conference on Computational Linguistics (Coling 1988)*, volume 2, Budapest, 1988. URL `http://www.aclweb.org/anthology/C88-2132`; 2006/10/13.

Michael Strube, Stefan Rapp and Christoph Müller. The influence of minimum edit distance on reference resolution. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 312–319, Philadelphia, PA, 2002. URL `http://www.aclweb.org/anthology/W02-1040`; 2005/04/08.

Tomek Strzalkowski and Sanda [M.] Harabagiu, editors. *Advances in Open Domain Question Answering*, volume 32 of *Text, Speech and Language Technology*. Springer, Dordrecht, 2006.

Tomek Strzalkowski, Sharon Small, Hilda Hardy, Paul Kantor, Wu Min, Sean Ryan, Nobuyuki Shimizu, Liu Ting, Nina Wacholder and Boris Yamron. Question answering as dialogue with data. In Strzalkowski and Harabagiu (2006), pages 149–188.

Roland Stuckardt. Design and enhanced evaluation of a robust anaphor resolution algorithm. *Computational Linguistics*, 27(4):479–506, 2001. URL `http://portal.acm.org/citation.cfm?id=972600`; 2004/10/15.

Fabian M. Suchanek. Ontological reasoning for natural language understanding. Master's thesis, Computer Science, Saarland University, 2005. URL `http://www.mpi-inf.mpg.de/~suchanek/publications/ORNLU.pdf`; 2006/03/22.

Renxu Sun, Jing Jiang, Yee Fan Tan, Hang Cui, Tat-Seng Chua and Min-Yen Kan. Using syntactic and semantic relation analysis in question answering. In *Proceedings of The Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, 2006. URL `http://trec.nist.gov/pubs/trec14/papers/nus.qa.pdf`; 2007/01/16.

M[arta] Tatu, B[randon] Iles, J[ohn] Slavik, A[drian] Novischi and D[an I.] Moldovan. COGEX at the Second Recognizing Textual Entailment Challenge. In Magnini and Dagan (2006). URL `http://ir-srv.cs.biu.ac.il:64080/RTE2/proceedings/17.pdf`; 2006/10/29.

Lucien Tesnière. *Eléments de syntaxe structurale*. Klincksieck, Paris, 1959.

Lucien Tesnière. *Grundzüge der strukturalen Syntax*. Klett-Cotta, Stuttgart, 1980. Herausgegeben und übersetzt von Ulrich Engel; German translation of Tesnière (1959).

Bozena H[enisz] Thompson and Frederick B. Thompson. Introducing ASK, a simple knowledgeable system. In *Proceedings of the First Conference on Applied Natural Language Processing*, pages 17–24, Santa Monica, CA, 1983. URL `http://www.aclweb.org/anthology/A83-1003`; 2005/12/12.

Bozena Henisz Thompson and Frederick B. Thompson. ASK is transportable in half a dozen ways. *ACM Transactions on Information Systems*, 3(2): 185–203, 1985. URL `http://doi.acm.org/10.1145/3914.3983`; 2005/12/12.

Henry S. Thompson and David McKelvie. Hyperlink semantics for stand-off markup of read-only documents. In *Proceedings of SGML Europe'1997*, Barcelona, 1997. URL `http://www.ltg.ed.ac.uk/~ht/sgmleu97.html`; 2006/08/09.

Masaru Tomita. An efficient augmented-context-free parsing algorithm. *Computational Linguistics*, 13(1–2):31–46, 1987. URL `http://www.aclweb.org/anthology/J87-1004`; 2006/10/13.

David Traum and Nizar Habash. Generation from lexical conceptual structures. In *Proceedings of Workshop on Applied Interlinguas, NAACL/ANLP2000*, Seattle, WA, 2000. URL `http://www.aclweb.org/anthology/W00-0207`; 2006/02/28.

TREC. Trec 2004 judgments for TREC 2004 factoid questions. `http://trec.nist.gov/data/qa/2004_qadata/04.factoid_judgments.txt`; 2006/03/23, 2004a.

TREC. TREC QA 2004 testset. `http://trec.nist.gov/data/qa/2004_qadata/QA2004_testset.xml`; 2005/07/26, 2004b.

A[lan] M. Turing. On computable numbers, with an application to the *Entscheidungsproblem*. *Proceedings of the London Mathematical Society, Series 2*, 42:230–265, 1936. URL `http://www.abelard.org/turpap2/tp2-ie.asp`; 2006/10/25.

Brian Ulicny. Bringing commercial question answering to the web. In Maybury (2004a), pages 31–45.

J[ulian] R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, 1976. URL `http://doi.acm.org/10.1145/321921.321925`; 2006/06/21.

Olga Uryupina. High-precision identification of discourse new and unique noun phrases. In *Proceedings of the ACL Student Workshop*, Sapporo, Japan, 2003. URL `http://www.coli.uni-sb.de/~ourioupi/ury_acl_fin.pdf`; 2004/02/06.

Olga Uryupina. Coreference resolution with and without linguistic knowledge. In *Proceedings of the 5th Edition of the International Conference on Language Resources and Evaluation (LREC 2006)*, Genova, 2006.

Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory* *of Computing (STOC ’82)*, pages 137–146, San Francisco, CA, 1982. URL `http://doi.acm.org/10.1145/800070.802186`; 2006/06/21.

Maria Vargas-Vera and Enrico Motta. Aqua – Ontology-based question answering system. In *Proceedings of the Third International Mexican Conference on Artificial Intelligence (MICAI-2004)*, number 2972 in Lecture Notes in Computer Science, pages 468–477, Mexico City, 2004. URL `http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=2972&spage=468`; 2005/12/12.

Renata Vieira and Massimo Poesio. An empirically based system for processing definite descriptions. *Computational Linguistics*, 26(4):539–593, 2000. URL `http://www.aclweb.org/anthology/J00-4003`; 2006/03/04.

Marc Vilain, John [D.] Burger, John Aberdeen, Dennis Connolly and Lynette Hirschman. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Conference on Message Understanding (MUC-6)*, Columbia, Maryland, 1995. URL `http://portal.acm.org/citation.cfm?id=1072405`; 2005/12/21.

María Begoña Villada Moirón. *Data-driven Identification of Fixed Expressions and Their Modifiability*. PhD thesis, Fakulteit van de Lettern, Rijksuniversieit Groningen, Groningen, 2005. URL `http://odur.let.rug.nl/~begona/thesis.pdf`; 2005/09/30.

Luciano Vitacolonna. ‘Text’/‘discourse’ definitions. In János S. Petöfi, editor, *Text and Discourse Constitution. Empirical Aspects, Theoretical Approaches*, volume 4 of *Research in Text Theory. Untersuchungen zur Texttheorie*, pages 421–439. Walter de Gruyter, 1988.

Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13 (2):260–269, 1967.

Martin Volk and Simon Clematide. Learn-filter-apply-forget. Mixed approaches to named entity recognition. In *Proceedings of 6th International Workshop on Applications of Natural Language for Information Systems*, Madrid, 2001. URL `http://www.ifi.unizh.ch/cl/volk/papers/Madrid_2001.pdf`; 2004/04/15.

Ellen M. Voorhees. The TREC-8 question answering track report. In *Proceedings of The Eighth Text REtrieval Conference (TREC 8)*, Gaithersburg, MD,

2000. URL `http://trec.nist.gov/pubs/trec8/papers/qa_report.pdf`; 2004/02/06.

Ellen M. Voorhees. The TREC question answering track. *Natural Language Engineering*, 7(4):361–378, 2001. URL `http://journals.cambridge.org/action/displayFulltext?type=1&fid=96166&jid=&volumeId=&issueId=04&aid=96165`; 2006/08/09.

Ellen M. Voorhees. Evaluating answers to definition questions. In *Proceedings of the HLT-NAACL 2003*, Edmonton, Canada, 2003. URL `http://www.aclweb.org/anthology/N03-2037`; 2007/01/16.

Ellen M. Voorhees. Overview of the TREC 2004 question answering track. In *Proceedings of The Thirteenth Text REtrieval Conference (TREC 2004)*, Gaithersburg, MD, 2005. URL `http://trec.nist.gov/pubs/trec13/papers/QA.OVERVIEW.pdf`; 2005/10/07.

Ellen M. Voorhees. Evaluating question answering system performance. In Strzalkowski and Harabagiu (2006).

Ellen M. Voorhees and Hoa Trang Dang. Overview of the TREC 2005 question answering track. In *Proceedings of The Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, MD, 2006. URL `http://trec.nist.gov/pubs/trec14/papers/QA.OVERVIEW.pdf`; 2007/01/09.

Piek Vossen. Introduction to EuroWordNet. *Computers and the Humanities*, 32(2–3):73–89, 1998. URL `http://www.springerlink.com/openurl.asp?genre=article&issn=0010-4817&volume=32&issue=2&spage=73`; 2006/08/09.

W3C. XML path language (XPath). Version 1.0. `http://www.w3.org/TR/xpath`; 2005/05/25, November 1999.

W3C. Resource Description Framework (RDF) schema specification 1.0, 2000. URL `http://www.w3.org/TR/2000/CR-rdf-schema-20000327`; 2001/05/08.

W3C. Extensible markup language (XML) 1.0 (third edition). `http://www.w3.org/TR/2004/REC-xml-20040204`; 2006/07/28, February 2004a.

W3C. Owl web ontology language. overview. `http://www.w3.org/TR/owl-features/`; 2006/10/24, February 2004b.

W3C. XQuery 1.0: An XML query language. `http://www.w3.org/TR/xquery/`; 2006/11/23, November 2006.

Stephan Walter and Manfred Pinkal. Automatic extraction of definitions from german court decisions. In *Proceedings of the ACL 2006 Workshop 'Information Extraction Beyond The Document'*, Sydney, 2006. URL `http://nlp.shef.ac.uk/result/iebd06/papers/IEBD0603.pdf`; 2006/10/18.

Stephen Warshall. A theorem on boolean matrices. *Journal of the ACM*, 9 (1):11–12, 1962. URL `http://doi.acm.org/10.1145/321105.321107`; 2006/06/21.

Bonnie Lynn Webber. Questions, answers and responses: Interacting with knowledge-base systems. In Brodie and Mylopoulos (1986), pages 365–402.

Christoph Weidenbach, Uwe Brahm, Thomas Hillenbrand, Enno Keen, Christian Theobald and Dalibor Topic. SPASS version 2.0. In *Proceedings of the 18th International Conference on Automated Deduction (CADE-18)*, number 2392 in Lecture Notes in Computer Science, pages 275–, København, 2002. Springer. URL `http://www.springerlink.com/content/lkc7vwuy37e99ww0/`; 2006/10/24.

W[illiam] A. Woods. Progress in natural language understanding – an application to lunar geology. In *American Federation of Information Processing Societies (AFIPS) Conference Proceedings*, volume 42, pages 441–450, Montvale, NJ, 1973.

W[illiam] A. Woods. Lunar rocks in natural english: Explorations in natural language question answering. In Zampolli (1977), pages 521–569.

Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138, Las Cruces, NM, 1994. URL `http://www.aclweb.org/anthology/P94-1019`; 2006/06/21.

Roman Yangarber and Ralph Grishman. NYU: Description of the Proteus/PET system as used for MUC-7 ST. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, Washington, DC, 1997. URL `http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_proceedings/nyu_st_paper.pdf`; 10. Juli 2003.

Annie Zaenen, Lauri Karttunen and Richard [Dick] Crouch. Local textual inference: can it be defined or circumscribed? In *ACL 2005 Workshop on Empirical Modelling of Semantic Equivalence and Entailment*, Ann Arbor, MI, 2005. URL `http://www.aclweb.org/anthology/W05-1206`; 2006/01/09.

Antonio Zampolli, editor. *Linguistic Structure Processing*, volume 5 of *Fundamental Studies in Computer Science*. North-Holland, Amsterdam, New York, NY, Oxford, 1977.

K[aizhong] Zhang and D[ennis] Shasha. Tree pattern matching. In Alberto Apostolico and Zvi Galil, editors, *Pattern Matching Algorithms*, pages 341–371. Oxford University Press, Oxford and New York, 1997.

Heike Zinsmeister, Jonas Kuhn and Stefanie Dipper. Utilizing LFG parses for treebank annotation. In *Proceedings of the LFG02 Conference*, Athens, 2002. URL `http://csli-publications.stanford.edu/LFG/7/lfg02zinsmeisteretal-num.pdf`; 2004/07/05.