Semi-supervised Semantic Role Labeling via Graph Alignment

Dissertation
zur Erlangung des akademischen Grades eines
Doktors der Philosophie
der Philosophischen Fakultäten der Universität des Saarlandes

Vorgelegt von Hagen Fürstenau
aus Bonn-Bad Godesberg

Dekan der Philosophischen Fakultät II:	Prof. Dr. Erich Steiner
Berichterstatter:	Prof. Dr. Manfred Pinkal
	Dr. Mirella Lapata
	Prof. Dr. Dietrich Klakow
Tag der letzten Prüfungsleistung:	10. Februar 2011

Im Gedenken an meinen Großvater
Karl-Heinz Fürstenau
(21. Februar 1917 – 16. März 2011)

Abstract

Semantic roles, which constitute a shallow form of meaning representation, have attracted increasing interest in recent years. Various applications have been shown to benefit from this level of semantic analysis, and a large number of publications has addressed the problem of *semantic role labeling*, i.e., the task of automatically identifying semantic roles in arbitrary sentences. A major limiting factor for these approaches, however, is the need for large *manually labeled semantic resources* to train semantic role labeling systems in the supervised learning paradigm. Consequently, the application of such systems is still limited to the small number of languages and domains for which sufficiently large semantic resources are available.

This thesis addresses the *knowledge acquisition problem* of semantic role labeling, i.e., the substantial *annotation effort* required for the creation of semantic resources that can be used to train state-of-the-art semantic role labeling systems.

Our main contribution is to formulate a *semi-supervised approach* to semantic role labeling, which requires only a *small manually labeled corpus* of role-annotated sentences. This initial seed corpus is augmented with annotation instances generated automatically from a large unlabeled corpus. The augmented corpus is used as training data for a supervised role labeler, to improve labeling performance over what can be attained when training on the manually labeled sentences alone. Our approach therefore *reduces the annotation effort* required to attain satisfactory performance and thus alleviates the knowledge acquistion problem, especially for languages and domains where the cost of annotating large semantic resources is prohibitive.

The key idea of our semi-supervised approach is to measure the similarity between labeled sentences from the manually annotated resource and sentences from a large unlabeled corpus. Similarity is conceptualized in terms of *optimal graph alignments*, which are employed to *project annotations* from labeled to unlabeled sentences. To select a set of novel training instances, similarity is operationalized as a measure of confidence, allowing us to limit the adverse effect of erroneous annotations. The optimization problem is formulated as an *integer linear program* and solved efficiently.

The thesis broadly consists of two parts. In the theoretical part, our semi-supervised approach to semantic role labeling is described in detail.

The empirical part then evaluates the effect of this method on various corpora extracted from existing semantic resources for English and German. These experiments show that the additional training data generated by our method can indeed improve the performance of a semantic role labeler and thus reduce annotation effort in practice.

Zusammenfassung

In den letzten Jahren hat sich ein wachsendes Interesse an semantischen Rollen, einer flachen Form von Bedeutungsrepräsentation, entwickelt. Es hat sich gezeigt, dass verschiedene Anwendungen von dieser Ebene der semantischen Analyse profitieren können, und eine große Zahl an Publikationen hat sich mit dem Problem der *automatischen rollensemantischen Annotation* befasst, also der Aufgabe, semantische Rollen in beliebigen Sätzen automatisch zu identifizieren. Ein Hindernis für solche Verfahren ist der Mangel an umfangreichen, *semantisch annotierten Ressourcen*, wie sie benötigt werden, um rollensemantische Annotationssysteme nach überwachten Lernverfahren zu trainieren. Die Anwendung solcher Systeme ist daher noch auf eine kleine Zahl von Sprachen und Domänen begrenzt, für die hinreichend große semantische Ressourcen zur Verfügung stehen.

Die vorliegende Arbeit beschäftigt sich mit dem Problem der *Wissensaquise* für rollensemantische Annotation, d.h. mit dem erheblichen *Annotationsaufwand*, der mit der Erstellung von semantischen Ressourcen für leistungsfähige rollensemantische Annotationssysteme verbunden ist.

Der Hauptbeitrag der Arbeit liegt in der Formulierung eines *halbüberwachten Ansatzes* für rollensemantische Annotation, der lediglich auf ein *kleines manuell mit semantischen Rollen annotiertes Korpus* angewiesen ist. Dieses Initialkorpus wird durch Annotationsinstanzen ergänzt, die mit Hilfe eines umfangreichen unannotierten Korpus automatisch generiert werden. Das erweiterte Korpus wird dann verwendet, um ein überwachtes Rollenannotationssystem zu trainieren und dessen Leistung im Vergleich zu einem System, dem nur das Initialkorpus zur Verfügung steht, zu verbessern. Unser Ansatz reduziert daher den Annotationsaufwand, der nötig ist, um zufriedenstellende Annotationsqualität zu erreichen, und mildert so das Problem der Wissensaquise, insbesondere für Sprachen und Domänen, für die die hohen Annotationskosten ein großes Hindernis darstellen.

Die Grundidee unseres halbüberwachten Verfahrens ist es, die Ähnlichkeit zwischen annotierten Sätzen des Initialkorpus und unannotierten Sätzen des großen Erweiterungskorpus zu quantifizieren. Wir drücken diese Ähnlichkeit mit Hilfe von *optimalen Graphalinierungen* aus, die wir verwenden, um semantische Information von annotierten auf unannotierte Sätze zu *projizieren*. Um eine Auswahl von neuen Trainingsinstanzen zu treffen, verwenden wir diese Ähnlichkeit als Konfidenzmaß, was uns erlaubt, Annotationsfehler zu vermeiden. Das Optimierungsproblem wird als Aufgabe der ganzzahligen linearen Programmierung formuliert und effizient gelöst.

Die Arbeit gliedert sich grob in zwei Teile. Der theoretische Teil beschreibt detailliert unseren halbüberwachten Ansatz zur rollensemantischen Annotation. Im empirischen Teil wird dann der Effekt dieser Methode auf verschiedenen Korpora evaluiert, die auf existierenden semantischen Ressourcen für Englisch und Deutsch basieren. Diese Experimente zeigen, dass die zusätzlichen Trainingsdaten, die unsere Methode erzeugt, in der Tat die Leistung eines semantischen Rollenannotationssystems verbessern und somit den Annotationsaufwand reduzieren können.

Acknowledgements

There are a number of people without whom this thesis would not have taken its present shape — or none at all. While only those most directly involved are mentioned here, my thanks go to everyone who helped me make the quest for my PhD enjoyable and ultimately successful.

Naturally, the first to mention is my supervisor Manfred Pinkal. Ten years ago, he let me take my first glimpse into Computational Linguistics, sparking an interest deep enough for me to come to Saarbrücken a few years later and venture into a new field. Guiding me on this way, he always left me enough room to learn from my own mistakes, but also time and again helped me take the essential "step back" from my work, and see the difference between results and insights.

I am also much indebted to my second supervisor Mirella Lapata. During my time in Edinburgh and afterwards, she helped me shape and focus the topic of my thesis, and taught me much about how to go about planning, conducting, and presenting good research. Of course, where this is not reflected in the present work, the responsibility is entirely mine.

Out of the many other people who shared their experience with me, I am especially grateful for the guidance I received from Sabine Schulte im Walde and Caroline Sporleder during the earlier stages of my studies. Further I want to thank Martin Forst for patiently helping me solve my problems with LFG parsing, as well as Sebastian Padó and Alexander Koller for some stimulating discussions. I was also fortunate to enjoy an excellent research environment, supported first by the DFG International Research Training Group 715 *"Language Technology and Cognitive Systems"* and later by the SALSA Project (DFG grant PI 154/9-3), complemented with excellent technical support by Christoph Clodo and the *Systemgruppe*.

Finally, my most special thanks go to Suhee, whose love and patience and unfaltering belief supported me on this whole journey, and make everything worthwhile.

Contents

List of Figures and Tables

Chapter 1

Introduction

Weigh the meaning
and look not at the words.

Ben Jonson

One of the fundamental goals of computational linguistics is to formulate representations of the meaning conveyed by linguistic entities such as words and sentences which lend themselves to automatic derivation and manipulation. In one form or another such meaning representations are essential for most tasks in natural language processing (NLP). In machine translation, usually considered the founding task of the field, it is evident that in order to translate a sentence such as

(1.1) *She decided to treat herself to a fig from the bowl.*

some notion of its meaning is indispensable. For example, the translation of the English word *treat* into most other languages will require at least an implicit notion of its *word senses* (such as *"to interact in a certain way"*, *"to provide with a gift"*, *"to care for medically"*, *"to have as a topic"*, and several others). Likewise, to generate a translation with adequate word order and function words, the relations between the meanings of the words are decisive. These cannot be formulated in syntactic terms alone since syntactic structures are specific to individual languages: that *a fig from the bowl* is expressed in a prepositional phrase of the verb *treat* in English does not preclude it from being realized as a direct object of the verb *gönnen* in a possible German translation (*"Sie entschied, sich eine Feige aus der Schale zu gönnen"*). In both cases, however, it plays the same *semantic role* relative to the denoted action, namely that of the "enjoyed thing", in contrast to the other occuring role of the "enjoying person". A fully adequate treatment of the translation task must reflect this.

Similar requirements are apparent when considering other NLP problems. To automatically find or generate answers for natural language questions, we need an understanding of the content and composition of the question sentence, allowing us, e.g., to generalize over synonyms , distinguish between the different senses of homonyms, and recognize the relations holding between the given words (including the question word). Without any such notion, it is not conceivable that the question (1.2) could be associated with the answer candidate (1.3):

(1.2) *How much did Google pay for YouTube?*

(1.3) *Google snapped up YouTube for $1.65 billion.*

Here, we need to determine that *"snap up"* in this context refers to a commercial transaction and its object ("*YouTube*") corresponds to a prepositional phrase when the payment is expressed by the verb *"pay"*.

Examples could be given for a number of other NLP applications. To allow for complex reasoning and inference, meaning representations have to not only comprise word senses and semantic roles, but also take into account other semantic aspects such as modality, time, negation, quantification, etc. Computational semantics has long sought to address such problems through the derivation of *logical representations* for natural language sentences. The long tradition of formal logic in the history of western thought, the sophisticated level it has reached in modern philosophy and mathematics, as well as its apparent suitability to computational concepts and contemporary technology have all contributed to the appeal of formal logic to represent meaning. *Formal semantics* has therefore concentrated on the derivation of logical formulas from text, usually assisted by syntactic theory. Originating in the seminal work of Montague (1973) these efforts have yielded complex and sophisticated formal systems. Nevertheless, they have largely failed to deliver on their promise of making reasoning about natural language feasible in practice: lack of coverage and robustness have turned out not to be transient shortcomings of these approaches, but rather inherent in the discrepancy between the immensity of the required knowledge and the limitations of manually encoding it.

In computational semantics, as in other subfields, the last two decades have seen these obstacles addressed more and more by statistical approaches, which seek to represent meaning by the quantitative analysis of language data. Variously hailed for bringing empiricism to linguistics and bemourned for dispensing with linguistic insight, this development has indubitably been essential for the practical success of most NLP applications today. Meaning representations, however, are less clearly identifiable in statistical approaches. Often, meaning is encoded implicitly in the models learned from the data, which do not easily lend themselves to interpretation. If, for example, a statistical machine translation model learns to translate *treat* differently depending on whether the following words are *"herself to"* or *"him badly"*, it does make a distinction correlating to word senses. These senses, however, are never made explicit. Moreover, large amounts of data have to be available to account for lexical variability, so that other phrases such as *"treat the guests to a wonderful meal"* or *"treat the animal with caution"* can be handled adequately. Finally, it is questionable whether all the information necessary for natural language understanding can be acquired from corpora alone. Common sense knowledge in particular is frequently assumed, but hardly ever expressed.

Recognizing the shortcomings of both purely formal methods dependent on manually codified rules on the one hand and statistical approaches trying to extract information solely from naturally occuring texts on the other hand, much of current research focuses on a middle ground. In computational semantics, the concepts of word senses and semantic roles in particular have proved amenable to such treatment. Both taxonomies of word senses and theories of semantic roles have been complemented by annotated corpora, which allow *supervised learning* of statistical models to predict linguistically defined categories, giving rise to the active research fields of word sense disambiguation and semantic role labeling. These semantic representations have proved beneficial to various NLP applications, such as information extraction (Surdeanu et al., 2003), question answering (Shen and Lapata, 2007), and machine translation (Wu and Fung, 2009). Tasks such as the recognition of textual entailment would also profit if higher accuracy could be attained (Burchardt et al., 2009). However, while annotating corpora with word senses or semantic roles requires less effort and expertise than devising a rule-based system, it is still a major limiting factor for the performance of NLP systems, and has thus become known as the "knowledge acquisition bottleneck". In practice, large annotated corpora, which are required for high-quality statistical systems, are available for but a small number of languages (most notably English) and domains. A possible solution to this problem are *semi-supervised learning approaches*, which in addition to manually annotated corpora also utilize information gained from unlabeled corpora. With large amounts of text nowadays readily available in electronic form, semi-supervised methods hold promise for improving the performance of NLP systems at little or no cost. They can thus reduce the required annotation effort and make many NLP tasks feasible for a broad range of languages or special domains with limited resources.

This thesis presents a semi-supervised approach to semantic role labeling (SRL), a task which has received much attention in recent years, yet is still dominated by supervised methods. In particular, we focus on the theory of *Frame Semantics*, which will require us to also address, to some extent, the problem of word sense disambiguation. We will show how our method, making use of both annotated Frame Semantic resources and large unlabeled corpora, is able to significantly improve SRL performance compared with state-of-the-art supervised approaches. This directly translates into reduced annotation costs for a given level of accuracy.

In the following sections, we first provide some background on semantic roles in linguistics and computational linguistics, describe the most important role semantic resources, and briefly survey the state of the art in SRL. We then discuss a range of semi-supervised learning approaches. Finally, we give an overview over the structure of the thesis.

1.1 Semantic Roles

The notion of semantic roles can be traced back to the Aṣṭādhyāyī of Pāṇini, an ancient Sanskrit grammarian, who probably lived in the 4th century BCE. A fundamental concept in his grammar is that of six categories, called *kāraka*, into which things may be classified relative to an action. Following Cardona (1976), these are

1. *katṛ* (agent): "the *kāraka* which functions independently with respect to other participants in a given action"
2. *karman* (object): "that participant in an action which the agent most wishes to reach through the action in question"
3. *karaṇa* (instrument): "that *kāraka* which, more than any other participant in a given action, serves as means for its accomplishment"
4. *adhikaraṇa* (locus): "the *kāraka* which functions as substrate relative to an action"
5. *saṃpradāna*: "that *kāraka* which the agent intends as goal through the object of the action in which he participates"
6. *apādāna*: "that *kāraka* which functions as a point of departure"

Here, the linguistic significance of the relationship between an action and its participants is clearly recognized. It is disputed, however, to what extent the rules for classifying participants into the various *kāraka* are purely semantic or partly influenced by their function in deriving the syntax of Sankrit. Cardona (1976) discusses that, e.g., a "person towards whom anger is felt" is variously designated as either *saṃpradāna* or *karman*, depending on the specific verb and preverb of the sentence. Similarly, in the sentence "The axe is cutting the tree", the reason for classifying "the axe" as *katṛ* instead of *karaṇa* is arguably also a sytactic one. Pāṇini is not alone in compounding syntactic and semantic criteria for the classification of participants in actions. This can be similarly observed in classical Greek and Latin grammar, where syntactic cases and their (semantic) uses were traditionally conflated into descriptions such as *Dative of Purpose* or *Ablative of Cause.*

The first attempt to go beyond this level of analysis was made by Fillmore (1968). In his *case grammar* he seeks to identify "deep-structure cases", which different languages may realize in various morphological and syntactic surface forms. A preliminary set of six such semantically defined cases is proposed:

1. *Agentive*: "the case of the perceived instigator of the action identified by the verb, typically animate"

2. *Instrumental*: "the case of the inanimate force or object causally involved in the action or state identified by the verb"

3. *Dative*: "the case of the animate being affected by the state or action identified by the verb"

4. *Factitive*: "the case of the object or being resulting from the action or state identified by the verb, or understood as a part of the meaning of the verb"

5. *Locative*: "the case which identifies the location or spatial orientation of the state or action identified by the verb"

6. *Objective*: "[...] the case of anything representable by a noun whose role in the action or state identified by the verb is identified by the semantic interpretation of the verb itself [...]"

Fillmore notes that this enumeration is not to be taken as exhaustive. Subsequent attempts at completing the picture, however, did not yield consensus on a universal set of "deep cases" or "thematic roles", leading Dowty (1991) to conclude that the only universal categories seem to be *Proto-Agent* and *Proto-Patient*, which arguments can be associated with to varying degrees.

Recognizing the problems arising in the definition of a universally valid set of roles, Fillmore (1976) developed the theory of *Frame Semantics*. It assumes that the understanding of language necessarily requires background knowledge about prototypical events and situations, organized in *frames*. Each frame is associated with a specific set of semantic roles or *frame elements*. In a sentence, a frame is evoked by a particular lexical unit or construction, which is called a *frame evoking element (FEE)*. Often, this is a verb referring to an action or state, with frame elements denoting the participating or involved entitites, but frames may also be evoked by words belonging to other parts of speech. Due to polysemy, many lexical units may evoke a number of different frames dependening on the context they occur in. For example, the verb *treat* may evoke, among others, the TOPIC, GIVING, and CURE frames, while in the example sentence (1.1) the context makes it clear that the GIVING frame is evoked.

By virtue of their definition, frames generalize across different lexical units. Both *"pay"* in (1.2) and *"snapped up"* in (1.3) refer to the same situation, which is captured by the COMMERCIAL_TRANSACTION frame. Similarly, the frame elements *Buyer* and *Goods* of this frame generalize over the concrete syntactic realizations in the two sentences. Frame Semantics therefore offers a semantic representation that takes into account polysemy, synonymy , and variability in syntactic realization. Important cases of syntactic variability are *diathesis alternations*, i.e., cases where syntactic realizations are subject to systematic variation. A large number of such alternations has been documented by Levin (1993). As an example, we consider the following sentences:

(1.4) [The burglar]$_{\textit{Agent}}$ [**broke**]$_{\textsc{Cause\_to\_fragment}}$ [the window]$_{\textit{Whole\_patient}}$ [with a hammer]$_{\textit{Instrument}}$.

(1.5) [The hammer]$_{\textit{Instrument}}$ [**broke**]$_{\textsc{Cause\_to\_fragment}}$ [the window]$_{\textit{Whole\_patient}}$.

Here we have indicated the frame CAUSE_TO_FRAGMENT evoked by the word *broke* and its frame elements *Agent*, *Whole_patient*, and *Instrument*. The Frame Semantic analysis abstracts away from the syntactic surface realization of the *Instrument*, once in a prepositional phrase and once as the subject of *break*, and thus expresses the common underlying meaning.

1.2 Resources

Before addressing the question of how to automatically derive frames and roles[1] for given sentences, we first present the major resources that have made the data-driven modeling of the problem possible.

FrameNet. The FrameNet project (Baker et al., 1998) is an effort to identify and document an inventory of frames suitable for the analysis of naturally occuring English text. Its first aim is to build a *frame lexicon*, defining frames and their associated roles. As an example, the definition of the CAUSE_HARM frame is shown in Figure 1.1. It includes a definition text, explaining the concept represented by the frame, a set of *core roles*, which are specific to the frame, and a number of *non-core roles*, which cover more general and incidental information. Finally, all lexical units that may evoke this frame are listed.

In addition to the creation of a frame lexicon, the FrameNet project further seeks to exemplify frames by naturally occuring sentences, drawn from the British National Corpus (BNC) and the LDC North American Newswire

[1]We will often use the terms "role" or "semantic role" instead of "frame element", as it is more common in the SRL literature.

Frame: Cause_harm	
Definition: The words in this frame describe situations in which an *Agent* or a *Cause* injures a *Victim*. The *Body_part* of the *Victim* which is most directly affected may also be mentioned in the place of the *Victim*.	
Core Roles	
Agent:	*Agent* is the person causing the *Victim*'s injury.
Body_part:	The *Body_part* identifies the location on the body where the bodily injury takes place.
Cause:	The *Cause* marks expressions that indicate some non-intentional, typically non-human, force that inflicts harm on the *Victim*.
Victim:	The *Victim* is the being or entity that is injured.
Non-core Roles	
Degree:	*Degree* is the degree to which the *Agent* causes harm to the *Victim*.
Reason:	A fact or action somehow related to the *Victim* that the *Agent* responds to by causing harm to the *Victim*.
Means:	An intentional action performed by the *Agent* that accomplishes the action indicated by the target.
Lexical Units	
bash.v, batter.v, bayonet.v, beat_up.v, beat.v, belt.v, biff.v, bludgeon.v, boil.v, break.v, bruise.v, buffet.v, burn.v, butt.v, cane.v, chop.v, claw.v, clout.v, club.v, crack.v, crush.v, cudgel.v, cuff.v, cut.v, elbow.v, electrocute.v, electrocution.n, flagellate.v, flog.v, fracture.v, gash.v, hammer.v, hit.v, horsewhip.v, hurt.v, impale.v, injure.v, jab.v, kick.v, knee.v, knife.v, knock.v, lash.v, maim.v, maul.v, mutilate.v, pelt.v, poison.v, poisoning.n, pummel.v, punch.v, slap.v, slice.v, smack.v, smash.v, spear.v, squash.v, stab.v, sting.v, stone.v, strike.v, swipe.v, thwack.v, torture.v, transfix.v, welt.v, whip.v, wound.v	

Figure 1.1: An abridged version of the frame Cause_harm as defined by FrameNet.

Corpora. For this purpose, occurences of lexical units are manually annotated with the frames they evoke in given sentential contexts. The textual representations of roles are identified and labeled accordingly. The following sentences are examples of the Cause_harm frame:

1. [Lee]$_{Agent}$ [**punched**]$_{\text{Cause\_harm}}$ [John]$_{Victim}$ [in the eye]$_{Body\_part}$.

2. [A falling rock]$_{Cause}$ [**crushed**]$_{\text{Cause\_harm}}$ [my ankle]$_{Body\_part}$.

3. [She]$_{Agent}$ [**slapped**]$_{\text{Cause\_harm}}$ [him]$_{Victim}$ [hard]$_{Degree}$
[for his change of mood]$_{Reason}$.

4. [Rachel]$_{Agent}$ [**injured**]$_{\text{CAUSE\_HARM}}$ [her friend]$_{Victim}$ [by closing the car door on his left hand]$_{Means}$.

In these examples, the words *punched*, *crushed*, *slapped* and *injured* are all annotated as evoking the CAUSE_HARM frame. This constitutes a form of word sense annotation. The word *crush* may in a different context evoke, e.g., the GRINDING frame, while *slap* may also evoke the IMPACT frame. In addition, each sentence is annotated with all applicable roles, such as *Agent*, *Victim*, or *Cause*. Technically, the annotation of an FEE or a role consists of a (not necessarily contiguous) textual span, specified in terms of character positions, and correponding frame and role labels.

In its latest version (release 1.3), FrameNet contains almost 140,000 example sentences for a total of 502 frames, which are evoked by over 5,000 different lexical units (predominantly verbs, nouns and adjectives) in the annotated sentences. However, in spite of the considerable effort that went into its creation, amounting to a large number of person-years over the space of more than a decade, the resource is still incomplete. Frames and lexical units are missing, and even for listed lexical units there may be few or no example sentences.

SALSA. The frames defined in the English FrameNet project have been successfully used to annotate sentences in a number of other languages, empirically confirming the intuition that frames are to a large degree language-independent. While FrameNets for Spanish (Subirats and Petruck, 2003), Japanese (Ohara et al., 2004), and Swedish (Borin et al., 2009) are of limited size, the German SALSA corpus (Burchardt et al., 2006) contains about 20,000 annotated sentences for verbal lexical units, which is roughly a third of the number for FrameNet (excluding other parts of speech). In contrast to the annotation procedure employed for FrameNet, the SALSA corpus is not based on manually selected example sentences, but on the *exhaustive annotation* of a number of verbs over a text corpus. The number of sentences for a verb therefore reflects its corpus frequency. Specifically, the TIGER treebank (Brants et al., 2002), comprising news texts from the German "Frankfurter Rundschau", was labeled with frames and roles of 491 verbs (compared to 2,115 verbal lexical units in FrameNet), thus adding a layer of semantic annotations to the syntactic treebank information. As a result, the corpus is more homgeneous in domain than FrameNet, which draws on the balanced collection of the BNC. Moreover, the annotation of roles more closely corresponds to syntactic constituents, as the units of annotation are not words or characters, but non-terminals and terminals in the phrase structure trees of the treebank.

Finally, the exhaustive annotation of all instances of the selected verbs encountered many verb senses for which a frame has not yet been defined in FrameNet. In these cases a total of 444 proto-frames were defined. These indicate verb senses not covered by FrameNet, but do not generalize across predicates as fully defined frames do. This limits their practical use and calls for a lexicographic effort to identify common underlying frames, which could unify proto-frames of different lexical units.

PropBank. The PropBank corpus (Palmer et al., 2005) consists of the "Wall Street Journal" portion of the Penn Treebank (Marcus et al., 1993) with an additional layer of semantic role annotations. However, the project follows a different approach than FrameNet, avoiding any commitment to a theory of semantic roles. Relative to a verb in a given word sense, a sentential constituent may fill an argument role ARG-n (with $n \geq 0$) or any of a small number of adjunct-like roles (like ARGM-LOC, ARGM-TMP, or ARGM-MNR for location, time, or manner). While ARG-0 and ARG-1 are meant to correspond to Dowty's Proto-Agent and Proto-Patient respectively, the argument types ARG-n for $n \geq 2$ characterize semantic roles *specific to individual verbs*. Descriptions of their semantics are provided for each verb, but do not generalize over different verbs as frames in Frame Semantics do. This allows abstraction over syntactic variability like the diathesis alternation shown in (1.4) and (1.5), but does not address the problems raised by lexical variation, as shown in (1.2) and (1.3). A complementary project to PropBank is NomBank (Meyers et al., 2004), which similarly addresses the annotation of nominal predicates and their arguments.

1.3 Semantic Role Labeling

With the availability of large manually annotated corpora like FrameNet and PropBank, *supervised learning* approaches to SRL became practically feasible. In the supervised paradigm, a statistical model is learned from a manually annotated *training corpus* and then applied to automatically analyze the role semantic structure of input sentences. Recent years have seen a multitude of publications on supervised SRL approaches. In this section, we give a broad overview over the different lines of research. For an extensive survey, we refer the reader to Palmer et al. (2010) or Màrquez et al. (2008).

The first supervised learning approach to SRL was proposed by Gildea and Jurafsky (2002). Their model uses FrameNet data and estimates the probabilities of role labels conditioned on a range of features, which are extracted from phrase structure trees. Many of the features they proposed still live on in current state-of-the-art systems. Among them are the phrase type of an argument, its governing category, the path in the parse tree between a predicate and its argument, the head word of an argument, etc. Interest in SRL has rapidly spread ever since, as witnessed by the shared tasks of CoNLL-2004 (Carreras and Màrquez, 2004), Senseval-3 (Litkowski, 2004), CoNLL-2005 (Carreras and Màrquez, 2005), SemEval-2007 (Litkowski and Hargraves, 2007; Màrquez et al., 2007; Pradhan et al., 2007; Baker et al., 2007), CoNLL-2008 (Surdeanu et al., 2008), and SemEval-2010 (Ruppenhofer et al., 2010a).

Various general-purpose machine learning techniques have been applied to the task, such as decision trees (Surdeanu et al., 2003), maximum entropy classifiers (Fleischman et al., 2003), and support vector machines (Pradhan et al., 2005). Often the identification of the substrings that realize roles and their classification into specific role types are separated into two consecutive stages. In SRL systems for Frame Semantics, frame classification is an additional preceding stage (Erk and Padó, 2006). While fast and easy to implement, this "pipeline architecture" is not without problems. Once a decision has been made, it cannot be corrected at a later stage. More complex architectures therefore employ methods of joint inference, e.g., by solving an integer linear program which ensures that structural constraints are respected (Punyakanok et al., 2008), re-ranking of candidate analyses (Toutanova et al., 2008), generative modeling (Thompson et al., 2003), formulation as a sequence tagging problem (Màrquez et al., 2005a; Pradhan et al., 2005), or modeling by conditional random fields (Cohn and Blunsom, 2005).

Besides the choice of system architecture, feature engineering has played an important role in improving the performance of supervised SRL systems. Commonly, heuristics such as the pruning rules of Xue and Palmer (2004) are applied to exclude words or constituents which are unlikely to be arguments of a given predicate, thus increasing precision and reducing processing time. Some novel feature types have also been expressed by tree kernels, which are employed in support vector machines to measure the similarity between instances by way of counting common substructures in their parse trees (Moschitti et al., 2008).

Attention has also been paid to the syntactic representations providing feature values for the various machine learning approaches. While initially parsers generating phrase structure trees were employed, attention has to some extent shifted towards dependency structure, as witnessed by the CoNLL-2008 Shared Task on "Joint Learning of Syntactic and Semantic Dependencies". SRL systems based on dependency graphs obtain similar performance as those based on phrase structure trees (Johansson, 2008). Màrquez et al. (2005b) show that even a system based on partial syntactic analyses can perform relatively well.

Supervised learning algorithms and the annotation they require can be integrated in a process of *active learning*, which selects sentences for human annotation based on the uncertainty of a classifier. This avoids wasting annotation effort on cases that can be handled by the model already, while focusing attention on problematic cases. While several publications have addressed active learning for word sense disambiguation (Chen et al., 2006;

Chan and Ng, 2007; Zhu and Hovy, 2007; Zhu et al., 2008), it has not yet been explored for semantic role labeling.

1.4 Semi-supervised Learning

Problems of resource scarcity in NLP are increasingly addressed by approaches that are able to extract useful information from unlabeled data. *Unsupervised learning* methods try to utilize regularities in unlabeled data and thus make predictions without any manually annotated training data. An example of this class of algorithms are clustering approaches, which partition given data points into a fixed or variable number of classes based on similarity in a suitable feature space. More advanced methods build statistical models based on hidden variables, which are probabilistically inferred from the observable data. For example, in Latent Dirichlet Allocation (Blei et al., 2003), underlying topics are inferred from the words of a text.

There have been few attempts to apply unsupervised learning to the task of SRL. The role identification task has been addressed by Abend et al. (2009), who estimate the collocation strength between predicates and possible arguments to decide whether a role relation holds or not. Their approach only relies on part-of-speech annotations. Grenager and Manning (2006) address the task of role classification by a structured probabilistic model and apply the EM algorithm (Dempster et al., 1977) to learn its parameters in an unsupervised fashion. The same problem is addressed by Lang and Lapata (2010), who present an algorithm that is able to determine non-standard linkings between semantic roles and their syntactic realizations.

A problem of completely unsupervised techniques is that they may be able to group instances together with some accuracy, but not to associate them with class labels as defined by the task, e.g., semantic role labels. This connection has still to be made either with the help of information from a manually created lexicon or by manually labeling a small set of instances. Swier and Stevenson (2004, 2005) present an approach relying on information from VerbNet (Kipper et al., 2000), a verb lexicon documenting alternation behavior of the kind studied by Levin (1993), which constitutes a weak form of supervision. They make initial unambiguous labeling decisions and then iteratively update a probability model in a bootstrapping procedure.

In contrast to unsupervised learning, the *semi-supervised learning* paradigm assumes that manual creation of a small labeled resource of *seed annotations* is feasible, and that in addition to these labeled instances there is a large amount of *unlabeled instances*. Many unsupervised probabilistic models can take into account labeled instances by way of observable variables in their learning process, thus improving model quality. Besides such general statistical models allowing for training on both labeled and unlabeled instances, there are various specific semi-supervised learning methods. One of the most straightforward procedures is *self-training*, where a supervised classifier is first trained on the seed data and then applied to the unlabeled data, with the resulting labeled instances added to the seed data. The classifier is then retrained on the augmented seed set. This procedure has been shown to be effective for some NLP problems (Mihalcea, 2004; McClosky et al., 2006). It does, however, generally seem to depend on different models or "views" of the problem informing each other. This notion is formalized in the approach of *co-training* (Blum and Mitchell, 1998). Here, such different and independent views are explicitly assumed. Different classifiers are trained and constrained to agree with each other in the learning process. This has the effect that the individual classifiers can learn from each other, to the extent that not all of them make the same mistakes.

There are various more advanced semi-supervised learning approaches. *Transductive support vector machines* (Joachims, 1999) extend standard support vector machines by finding a hyperplane which not only separates labeled instances of different classes but also cuts through regions in the feature spacer showing low density of unlabeled instances, which can thus improve the quality of the learned model. A similar idea is formalized in *graph-based semi-supervised approaches*. Here, the graph of all labeled and unlabeled instances is considered, with edges between them labeled by some kind of similarity values. A global optimization task is then solved to disconnect the graph into components, each containing only labeled instances of one class. For example, Blum and Chawla (2001) formulate this task as a mincut problem, removing graph edges to disconnect instances of two classes and minimizing the total similarity represented by the removed edges. Another semi-supervised technique is *structural learning* (Ando and Zhang, 2005), which defines auxiliary problems on the unlabeled data and then uses their solutions to inform the main task.

As in the case of unsupervised methods, applications of semi-supervised approaches to SRL have been few and far between. First attempts at using both labeled and unlabeled sentences to train SRL models did not yield convincing results. Self-training and co-training were not successful (He and Gildea, 2006), and semi-supervised generative modeling yielded only small improvements (Thompson, 2004). Some methods, however, have been developed to address resource scarcity in general. Gordon and Swanson (2007) use surrogate training material of syntactically similar verbs to label PropBank roles of verbs with only one fully annotated sentence. Padó et al. (2008) propose a method for training on verbal annotation data to label event nominalizations, establishing a mapping between roles of verbs and their nominalizations.

A different strategy is followed by methods of *cross-lingual annotation projection*. Those seek to derive annotations for one language by projecting role semantic annotations of another language via aligned bi-texts. Padó

(2007) applies such techniques to project Frame Semantic annotations from English to German and French. Similarly, Johansson and Nugues (2006) project from English to Swedish. Fung and Chen (2004) avoid the need for parallel corpora and infer annotations for Chinese by associating FrameNet information with HowNet, an ontology for Chinese.

To our knowledge, this thesis presents the first semi-supervised approach to SRL that infers new annotation instances by projection from a labeled to an unlabeled corpus of the same language. An important feature of our method is that it procures additional training data for an arbitrary supervised SRL system. Learning an SRL model is therefore orthogonal to utilizing unlabeled data, and overall performance can profit from future advances in either field.

1.5 Thesis Overview

This thesis addresses the knowledge acquisition problem of SRL described in the preceding sections. It proposes a novel semi-supervised approach based on the acquisition of additional training instances for a supervised SRL system.

Chapter 2 introduces the concepts which our semi-supervised algorithm is based on, and describes how the required resources, i.e., a seed corpus of semantic annotations and syntactic analyses produced by a dependency parser, are preprocessed into a suitable input format for all later processing stages.

Chapter 3 describes the general framework of the expansion algorithm driving our semi-supervised method. A similarity measure based on optimal graph alignments is derived and employed both for the projection of annotations from labeled to unlabeled sentences and for the selection of suitable novel annotation instances. A number of parameters are left unspecified, such as the measures of lexical and syntactic similarity, a weight parameter between the two, and a parameter determining the amount of generated annotation data.

Chapter 4 presents specific measures of lexical and syntactic similarity that can be used in the general framework of Chapter 3. For lexical similarity, two common approaches from the literature are discussed, one based on distributional similarity and the other based on WordNet information. Efficient algorithms for the computation of these similarity measures are presented. For syntactic similarity, a simple definition is given. A more complex approach is also discussed, but found to be unsatisfactory.

Chapter 5 shows how the optimization problem of Chapter 3 can be formulated as an integer linear program. The complexity of the problem is analyzed both theoretically and empirically, and an algorithm is described that efficiently solves the problem in practice.

Chapter 6 describes the supervised SRL system which is employed in the evaluation of the expansion algorithm in Chapters 7 and 8. Following a state-of-the-art system from the literature, the tasks of frame and role labeling are implemented by machine learning algorithms in a pipeline architecture.

Chapter 7 reports on the results of applying our semi-supervised SRL approach to generate novel annotation instances for predicates that are exemplified by a small number of manually labeled sentences. First, an optimal instantiation of the framework of Chapter 3 is determined, choosing one of the two alternative definitions of lexical similarity presented in Chapter 4 and optimizing the weight parameter. In the main experiments of the chapter, we investigate how the sizes of the manually labeled corpus and the set of generated instances influence the performance of our method. Significant improvements are found for small and medium sized seed sets. Finally, we compare our method to a self-training approach, which shows that our similarity-based formulation is essential for the observed improvements.

Chapter 8 focuses on the harder task of labeling frames and roles of predicates for which no annotated sentences at all are available. To process such predicates, we first need to determine frame candidates for them. We present two alternative methods for this task, one based on distributional similarity and one making use of WordNet information. We then show how instances generated by the expansion algorithm can improve frame and role labeling performance.

Chapter 9 concludes the thesis by summarizing the main contributions and giving a brief outlook on possible directions for future research.

Relevant Publications

Results of the research presented in this thesis have been reported on in the following publications:

- Hagen Fürstenau. 2008.
  Enriching Frame Semantic Resources with Dependency Graphs.
  In *Proceedings of the 6th International Language Resources and Evaluation Conference*, 1478–1484. Marrakech, Morocco.

- Hagen Fürstenau and Mirella Lapata. 2009.
  Semi-supervised Semantic Role Labeling.
  In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, 220–228. Athens, Greece.

- Hagen Fürstenau and Mirella Lapata. 2009.
  Graph Alignment for Semi-supervised Semantic Role Labeling.
  In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing.* 11–20, Singapore.

Chapter 2

Preprocessing

Before anything else,
preparation is the key to success.

Alexander Graham Bell

In this chapter, we detail the preprocessing steps necessary for both the application and the evaluation of our semi-supervised semantic role labeling approach. We first introduce some concepts and notation used throughout the following chapters, most importantly that of a semantically labeled dependency graph. We then describe a general method for generating such graphs by combining annotations of a Frame Semantic resource with the output of a dependency parser. These graphs will later function as seeds for our expansion algorithm. We evaluate the quality of the produced graphs and show how the impact of parser errors, the most important source of incorrect labeled graphs, can be significantly reduced.

2.1 Dependency Syntax

Dependency syntax (Tesnière, 1959) describes the structural relations between the words of a sentence in terms of *dependencies*, i.e., binary relations between words, where one word is the *head*, the other is the *dependent* and a certain *grammatical relation* (e.g., *subject* or *direct object*) holds between the two.

We can visualize dependency structures as directed graphs like the one shown in Figure 2.1. All words of the sentence are represented by graph nodes, while grammatical relations show up as directed labeled edges from heads to dependents. Various kinds of additional information, extending the original notion of dependency syntax, may be associated with the graph nodes. For our purposes we store information about word form, lemma, part of speech tag, voice (for verbs) and linear position within the sentence.

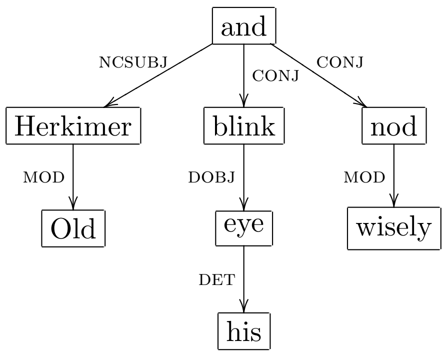


Figure 2.1: Dependency graph for the sentence *"Old Herkimer blinked his eye and nodded wisely"* taken from the FrameNet corpus. Graph edges are labeled with grammatical relations: NCSUBJ for non-clausal subject, CONJ for conjunct, MOD for modifier, DOBJ for direct object, and DET for determiner.

Other morphological information like case, number or tense is not relevant for our purposes. For simplicity, we will only show the lemma of each word when drawing dependency graphs.

While general linguistic criteria have been formulated to decide whether two words stand in a dependency relation, which direction it takes and how it is to be labeled (Mel'čuk, 2003), there are a number of phenomena for which analyses vary among different syntactic theories. For example, there is no consensus on how to analyse coordination, and to consider the coordinating word the head, as in Figure 2.1, is only one possibility (see Nilsson et al. (2006) for a discussion of alternative analyses).

Given a pair of words, the head generally should be the word that determines the syntactic behaviour of the phrase to a greater extent. This would imply that in a phrase consisting of a noun and an attributive participle, such as *"broken window"*, the noun should be the syntactic head. However, some dependency parsers, such as the RASP system (Briscoe et al., 2006), which will be used in our experiments for English, include a relation in the opposite direction. This reflects the fact that a predication is expressed by the verb *break*, which is therefore considered the head word. This solution is clearly inspired by the semantics of the phrase more than by its syntax. If relations in both directions are included, the graph contains a directed cycle, as illustrated in Figure 2.2.

In fact, much of the variance that can be observed in dependency parser output stems from attempts at producing more "semantic" analyses. Another example for this is the treatment of raising and control verbs. Raising verbs are verbs that realize a semantic argument of a subordinate clause as a direct syntactic dependent. A prominent example is the verb *seem*. In the sentence *"He seems to work a lot"*, the subject *he* is a syntactic de-

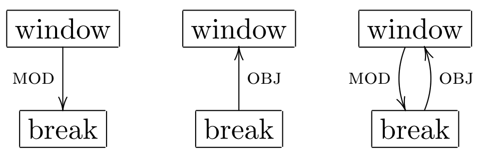


Figure 2.2: Three different analyses for the phrase *"broken window"*, the third one showing a directed cycle.

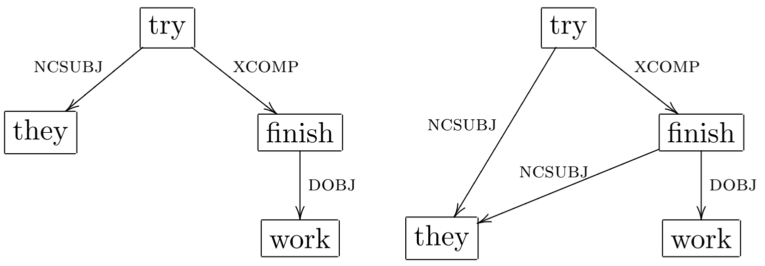


Figure 2.3: Two different analyses for the sentence *"They tried to finish the work"*, the second graph exhibiting an undirected cycle. The edge labels denote non clausal subject (NCSUBJ), unsaturated VP complement (XCOMP), and direct object (DOBJ). The words *"to"* and *"the"* are omitted.

pendent of *seem*, but semantically an argument of *work*. A similar case are control verbs, for which the syntactic dependent is a semantic argument of *both* verbs, as *they* in *"They tried to finish the work"*. Again, some dependency parsers will try to encode the semantic information and analyze the argument in question as a dependent of the subordinate verb — or even of both verbs. The latter case results in a dependency graph that is not a tree, because the dependent will have two heads, and the dependency graph therefore exhibits an undirected cycle, as shown in Figure 2.3.

These two examples highlight that, while dependency syntax is often formally defined to exclude multiple heads so that the resulting graphs are actually trees, in practice we may have to deal with analyses containing undirected or even directed cycles. Undirected cycles do not present significant problems for our algorithms, and we will therefore allow dependency analyses to be general directed acyclic graphs (DAGs). Directed cycles, however, pose problems for many algorithms involving graph traversal, since sets of "seen nodes" have to be maintained to ensure termination. We therefore remove directed cycles as a step of preprocessing, using a small number of rules that are specific to the parser we employ. For example, for RASP we remove edges from a non-verb to a verb if there also is an edge in the opposite direction. As an additional preprocessing step, we ensure that subjects of verbs in the passive voice are marked by distinct grammatical relations (e.g., NCSUBJ_PASS).

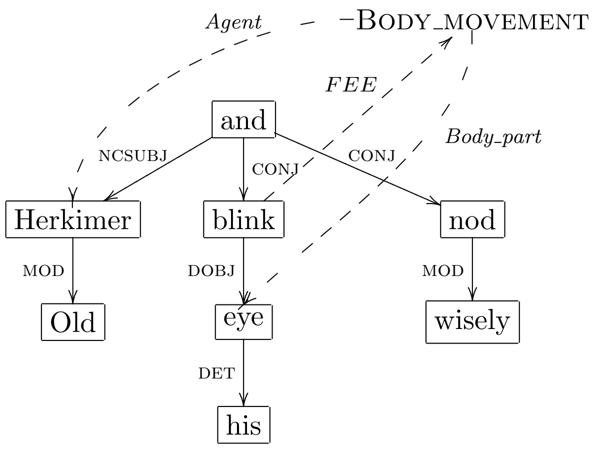


Figure 2.4: Semantically annotated dependency graph: the word *blink* evokes the frame Body_movement, which features the roles *Agent* and *Body_part*.

2.2 Semantically Labeled Dependency Graphs

Neither of the two Frame Semantic corpora to which we will apply our expansion method contains dependency syntax annotation. The annotations in the FrameNet corpus have limited syntactic information about phrase types and the grammatical relation between the FEE and the roles (distinguishing only the functions *external*, *object* and *dependent* for verbal FEEs). The SALSA corpus does contain detailed manual syntactic annotation, as it was built on top of the TIGER treebank (Brants et al., 2002). However, this treebank holds phrase structure analyses, and while a small subset has been converted into a dependency bank (Forst et al., 2004), this process involves a significant amount of manual work, making it infeasible for the entire SALSA corpus.

Our goal therefore is to produce semantically annotated dependency graphs by merging the Frame Semantic analyses with dependency graphs produced by an automatic parser. The result will be semantically annotated dependency graphs, where FEEs are marked with the frames they evoke and each role of a frame is associated with graph nodes representing its instantiation in the sentence. Specifically, we will add a frame structure to a dependency graph by linking a special *frame node* to its FEE by an edge labeled *"FEE"*. Each role is then represented by an edge between the frame node and the head of the phrase realizing this role, labeled with the role name. An example is given in Figure 2.4.

We now describe in turn how to map the annotations of FEEs and roles onto dependency graphs.

Mapping FEEs. For every annotated FEE in the Frame Semantic corpus we identify a corresponding node in the dependency graph. We make the following simplifying assumptions:

1. In this thesis we will focus on SRL for verbal predicates, which have received more attention in the literature than other parts of speech. In the FrameNet corpus, they make up 60,666 of the 139,439 annotated sentences, while in the SALSA corpus all 19,494 annotations are for verbs. Each sentence in the two corpora features the annotation of a single FEE and its roles. As we will apply our semi-supervised SRL approach to these two corpora, we can thus assume that **in each sentence there is exactly one annotated verbal lexical unit**.

2. In FrameNet, a single lexical unit always evokes exactly one frame. In SALSA, however, approximately 7% of the instances contain FEEs annotated with multiple frames. These express underspecification, where more than one frame may be applicable. We chose to ignore this phenomenon in our work and treat each annotated frame as a separate annotation instance. Accounting for underspecification at the frame level would make evaluation of our results less obvious (e.g., we would have to decide whether to give partial scores if some of the annotated frames are correctly predicted) and complicate exposition. We thus assume that **each FEE evokes exactly one frame**.

3. A Frame Semantic lexical unit may consist of more than one word. A frequent example in English are particle verbs such as *put up*. A dependency parser will typically analyze two tokens here, with *up* a dependent (of type *particle* or similar) of *put*. Following Frame Semantics, we would therefore have to mark two graph nodes as constituting the FEE. We avoid this complication and only mark the verb (or, more generally, the first part of a multi-word expression) as the FEE. This should be of little practical consequence: an SRL system typically has access to the entire dependency graph and its grammatical function labels, and can therefore recover this information. Consequently, we assume that **each FEE consists of a single word, corresponding to a single node in the dependency graph**.

Having made these assumptions, the identification of a single graph node corresponding to the FEE is straightforward: we simply choose the graph node which coincides with the annotated FEE in its textual span. There may be cases, however, where no such graph node exists, e.g., because the tokenization performed by the parser differs from the annotation in FrameNet. In such cases we choose the graph node having the largest *overlap* with the textual span of the FEE instead, which we count in number of characters. This makes our preprocessing more robust to minor incompatibilities of the employed tools. We record the number of mismatched characters (i.e., characters which are in the node span, but not the FEE span or vice versa) as a *mismatch score* for this instance. This score will later be used to discard unreliable graphs from the seed set of our expansion algorithm.

Mapping Roles. To associate frame roles with nodes in the dependency graph, we identify the node whose *yield* corresponds to the annotated textual role span. Here, the yield of a node is the textual span of the words associated with itself or any of its direct or indirect dependents. For example, in Figure 2.4 the yield of the node *eye* would be the substring *"his eye"* of the given sentence. The yield of a node does not have to be a contiguous substring of the sentence (dependency graphs for which it always is are called *projective*). In fact, if textual spans of graph nodes are represented by character indices in the sentence string, the union of multiple spans will usually not be contiguous due to missing whitespace characters or punctuation marks (which are often omitted from dependency graphs). To compare such textual spans to the ones marked in FrameNet, we employed heuristics to ignore mismatches on such "irrelevant" character positions. There is, however, still no guarantee that for a given role span a single node with a matching yield can be found. The reasons for this fall into three classes:

1. As in the case of mapping FEEs, there may be minor incompatibilities in tokenization or invalid offsets of string indices due to encoding errors. We handle these problems robustly by an approach equivalent to the one outlined above and increase the mismatch score by the number of mismatching characters for each role. In the end, it therefore represents the sum of the character mismatches in mapping the FEE and all role instantiations, and a mismatch score of 0 indicates that no problems were encountered.

2. The assumption that a role span corresponds to the yield of a single graph node is a simplification of the situation found in Frame Semantic annotation. It is violated in a number of cases in the FrameNet corpus, where semantic roles are annotated across syntactic phrase boundaries. The most frequent case is caused by the annotation guideline that a role annotated on a relative pronoun also include the referent of the relative pronoun in the containing clause. Other cases are less systematic and often subject to idiosyncratic annotation decisions. (For example, in the sentence *"These genes are duplicated along with the chromosomes"*, both *"these genes"* and *"along with the chromosomes"* are annotated with the role *Original* of the frame DUPLICATION.) Under our approach, roles will be imperfectly mapped in these cases, as only the part of the role span overlapping most with the yield of a graph node will be represented in the dependency graph.

3. The majority of cases in which no perfectly matching node can be found are due to parser errors. If a parser error leads to an incorrect yield for a node that should receive a role label, the comparison of this yield to the textual span of the role will fail.

Interestingly, this last type of mismatches opens up a possibility of detecting parser errors. This will be discussed in the next section. It is also the reason why we do not generalize the method to allow one role to be represented by more than one graph node, which might improve coverage of the phenomena of the second type discussed above. Allowing an arbitrary number of nodes to represent a role would make the matching succeed in almost all cases, thus rendering this indicator of parser errors useless. Some overall accuracy might be gained by a compromise of allowing a small number (e.g., two or three) head nodes per role span. For ease of exposition, however, we leave exploration of this parameter to future work.

2.3 Parse Selection

A significant source of errors for any SRL system relying on syntactic preprocessing are parser errors, which affect both the training and the application of an SRL system. In the former case, these errors negatively impact the quality of the learned model, while in the latter case applicability of the model to the badly parsed test sentences is impaired.

In the context of our semi-supervised approach to SRL, there is a third process suffering from parser errors: since we infer new training instances from seed sentences, a single incorrectly parsed seed sentence may lead to a large number of badly inferred training instances, introducing noise into the learning process and thus counteracting our attempts at improving the training data. In the worst case the detrimental effect of this noise may be larger than the benefit from capturing novel information. Avoiding parser errors is therefore even more crucial to our approach than to conventional SRL architectures.

We address this problem by performing a form of parse selection during preprocessing. The key observation here is that many parser errors are caused by syntactic ambiguity that could be resolved with semantic information. A prime example for this is the attachment problem of prepositional phrases (PPs). In the sentence *"I eat my meal with a fork"* there is a syntactic ambiguity between attaching the PP *"with a fork"* either to the verb *"eat"* or to the noun *"meal"*. Semantically, however, only the first choice makes sense. Such attachment ambiguities are one of the chief causes of parser errors (Collins, 2003). In our case, however, the semantic information which the parser lacks in order to make the right disambiguation decision is often implicit in the Frame Semantic annotation. In the given example sentence, we would see *"my meal"* annotated as a role span. This leads to a mapping mismatch with any analysis attaching *"with a fork"* to *"meal"*, while for the correct analysis the spans match perfectly. In fact, the correct attachment is inherent in the semantic annotation decision of leaving *with a fork* out of the role span.

Since there is no easy way of incorporating this information into the parsing process itself, we resort to a *reranking* approach on n-best lists of analyses. Specifically, we set the parser to output the 20 parses with the highest probability and try to map the Frame Semantic annotation to each of them in turn as described in Section 2.2, proceeding in order of decreasing parse probability. As soon as we find a perfect match, we terminate and map the frame onto this dependency graph. If the frame cannot be mapped perfectly to any of the 20 graphs, we instead choose the analysis leading to the least *mismatch score* as defined above.

2.4 Evaluation with Different Parsers

As the quality of the semantically annotated dependency graphs is crucial for all later processing stages, we evaluate our preprocessing algorithm for coverage and accuracy, taking into account different parsers for English and German.

We consider the PARC XLE system with the English LFG developed in the ParGram project (Riezler et al., 2003), which is a hand-crafted grammar with statistical parse ranking. It produces c-structures, resembling classical phrase structure trees, and f-structures, which are structurally close to dependency graphs. An example of an f-structure is shown in Figure 2.5. It covers four different readings of the sentence *"They saw the girl with the telescope"* in a packed representation. We do not need to go into detail about the specific information contained in it. It suffices to note that it is possible to convert f-structures into dependency graphs. For our experiments we applied a set of rewriting rules[1] for this purpose. Figure 2.6 shows dependency graphs for two of the readings in our example, representing the attachment ambiguity of the prepositional phrase. In the other two readings, the verb *to see* is replaced by the verb *to saw*.

As a second system for English, we considered RASP (Briscoe et al., 2006), which is a toolchain consisting of a tokenizer, a part of speech tagger, a lemmatizer, a parser based on a manually built grammar, and a ranking model. It produces both phrase structure trees and dependency output. An example of the latter has already been shown in Figure 2.1. Briscoe and Carroll (2006) find that the LFG and RASP perform similarly at 79.6% and 79.7% F_1 score, respectively, on the PARC 700 Dependency Bank (King et al., 2003).

[1]kindly provided to us by Tracy H. King

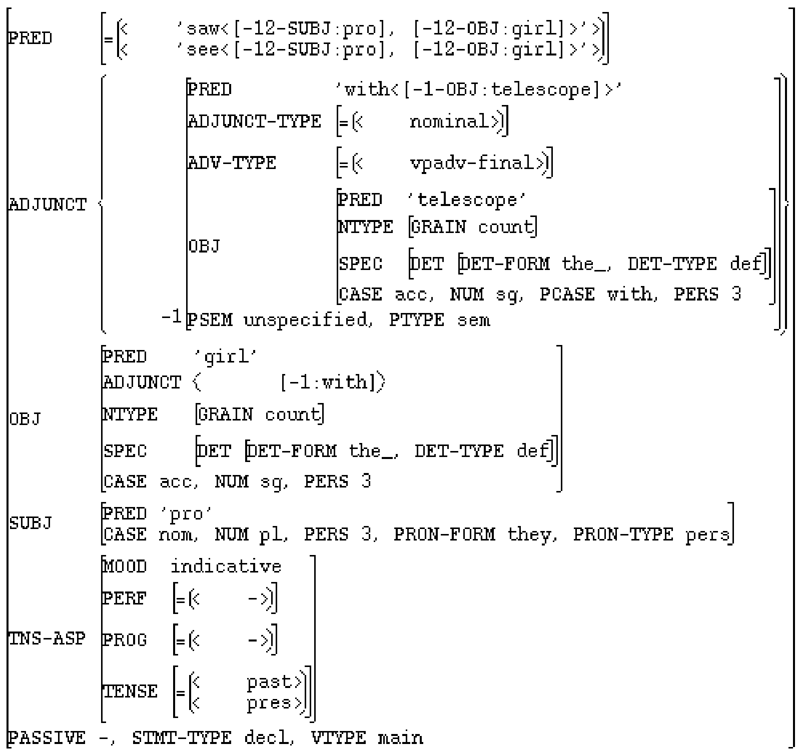

PRED
=< 'saw<[-12-SUBJ:pro], [-12-OBJ:girl]>'>
< 'see<[-12-SUBJ:pro], [-12-OBJ:girl]>'>
ADJUNCT
PRED 'with<[-1-OBJ:telescope]>'
ADJUNCT-TYPE =< nominal>
ADV-TYPE =< vpadv-final>
OBJ
PRED 'telescope'
NTYPE GRAIN count
SPEC DET DET-FORM the_, DET-TYPE def
CASE acc, NUM sg, PCASE with, PERS 3
-1 PSEM unspecified, PTYPE sem
OBJ
PRED 'girl'
ADJUNCT [-1:with]
NTYPE GRAIN count
SPEC DET DET-FORM the_, DET-TYPE def
CASE acc, NUM sg, PERS 3
SUBJ
PRED 'pro'
CASE nom, NUM pl, PERS 3, PRON-FORM they, PRON-TYPE pers
TNS-ASP
MOOD indicative
PERF =< ->
PROG =< ->
TENSE =< past> < pres>
PASSIVE -, STMT-TYPE decl, VTYPE main

Figure 2.5: LFG f-structure for the sentence *"They saw the girl with the telescope."*

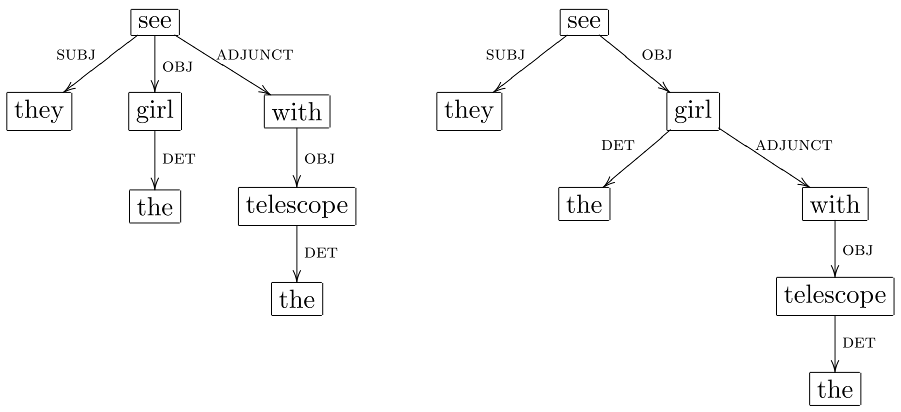


Figure 2.6: Dependency graphs converted from two of the four readings encoded in the f-structure of Figure 2.5

(a) FrameNet (verbal FEEs)

Parser	n-best	Mappable
LFG	1	49.3%
RASP	1	44.4%
LFG	20	61.1%
RASP	20	65.1%

(b) SALSA

Parser	n-best	Mappable
LFG	1	56.7%
LFG	20	64.5%

Table 2.7: Proportions of sentences for which the Frame Semantic annotation could be mapped without any mismatches

For German, we considered the LFG developed again as part of the ParGram project (Dipper, 2003; Rohrer and Forst, 2006), together with the statistical disambiguation model of Forst (2007), which achieves an F_1 score of 83.01% on the TiGer Dependency Bank (Forst et al., 2004). As in the case of English, we converted f-structures to dependency graphs with the help of rewriting rules[2].

We first analyze how many of the instances in the FrameNet and SALSA corpora can be mapped perfectly (i.e., with mismatch score 0) onto dependency graphs produced by different parsers. We apply the procedure described in Section 2.2 to map frames and roles onto nodes of dependency graphs generated by the three parsers. For each parser, we consider either only the most probable parse or select among the 20 most probable analyses as described in Section 2.3. The results are shown in Table 2.7.

We can see that parse selection improves coverage in terms of perfectly mappable instances by about 8 to 21 percentage points, depending on the corpus and parser employed. The difference between the two parsers for English is rather small. Moreover, their relative order is reversed between the 1-best and 20-best scenarios, with RASP obtaining the higher coverage with parse selection. This might indicate that a parser based on a deep linguistic grammar can be more accurate for our purposes, but does not gain as much from n-best parse selection. It should be noted, however, that general conclusions about the relative performance of syntactic parsers are outside the scope of this thesis.

To ascertain whether higher coverage really corresponds to higher quality of preprocessing, we next randomly selected samples of 100 sentences from both the FrameNet and SALSA corpora. We then manually analysed the successfully mapped annotations, classifying them as *correct* or *incorrect*. Here, an annotated dependency graph was considered *correct* if the part of the sentence constituting the semantic predicate-argument structure described by the frame was correctly analysed in its syntax and additionally the FEE and all roles were correctly associated with graph nodes. Due to the size of the sample and the fact that there was only one annotator (the

[2]kindly provided to us by Martin Forst

(a) FrameNet (verbal FEEs)

Parser	n-best	Correct
LFG	1	76% (±9%)
RASP	1	78% (±9%)
LFG	20	77% (±9%)
RASP	20	78% (±9%)

(b) SALSA

Parser	n-best	Correct
LFG	1	81% (±8%)
LFG	20	80% (±8%)

Table 2.8: Manual evaluation of the correctness of the mapped annotations. Confidence intervals ($p < 0.05$) are given in brackets.

author), this cannot give more than a rough indication of the quality of the annotated dependency graphs. The results are shown in Table 2.8. Across corpora and parsers no significant deterioration of accuracy is discernible within the confidence intervals. This means that the significant increase in coverage achieved by selecting among the 20 best parses does not incur an obvious loss in accuracy, which confirms the suitability of our parse selection approach.

Comparison to conventional parser evaluation is not straighforward. The simplest measure of parser performance is *exact match*, which counts how many of the analyses produced by a parser coincide perfectly with those in a gold standard. Often, however, more fine-grained measures are employed, such as the popular PARSEVAL metric (Black et al., 1991), which considers *precision and recall* of individual relations, or extensions of it that also take into account relation labels. For an overview over dependency parser evaluation we refer the interested reader to Dridan (2010). Accuracy for our purposes of mapping Frame Semantic annotation lies somewhere in between strict measures like exact match and more lenient ones like precision and recall of individual relations. For complex sentences, usually only a small part of the dependency graph will be relevant to a given semantic frame. Therefore it is certainly a more lenient measure than exact match. It does, however, usually require multiple syntactic relations to be correct, which is stricter than relation-based measures.

Ultimately, an accuracy of about 80% for dependency graphs with a mismatch score of 0 seems to be the best that can be achieved in preprocessing. This still means that in the application of our expansion method about one in five seed sentences can be expected to contain some relevant parser error. Our algorithms therefore have to be designed so that they are robust in the face of noise. It is also interesting to note that, while better parser accuracy would be welcome, SRL would benefit especially from accuracy improvements in local analyses, namely in the parts of a parse covered by a single semantic predicate-argument structure. This is not necessarily the objective which statistical parsers are optimized for when tuned on popular evaluation metrics.

(a) FrameNet (verbal FEEs)

Parser	n-best	Parser Errors
LFG	1	82% (±8%)
RASP	1	76% (±9%)
LFG	20	83% (±8%)
RASP	20	78% (±9%)

(b) SALSA

Parser	n-best	Parser Errors
LFG	1	81% (±8%)
LFG	20	80% (±8%)

Table 2.9: Proportion of the unmappable sentences that fail due to parser errors. Confidence intervals ($p < 0.05$) are given in brackets.

In a final evaluation, we considered those sentences which could not be mapped perfectly either to the best or to any of the 20 best parses. Table 2.9 shows that roughly 80% of those failures were due to parser errors. (The remaining failures are caused by the other reasons discussed in Section 2.2, such as tokenization errors, roles spanning more than one subgraph, etc.) Comparing this with the 20% parser errors among mappable sentences, we see that failure to map the semantic annotation is indeed a strong indicator for an invalid syntactic analysis and therefore acts as a filter in our preprocessing. We will therefore not use any sentences with a mismatch score > 0 as seeds in our expansion algorithm. As described in Section 6.4, we will, however, include such sentences in the test sets to guarantee an unbiased evaluation of our method.

For all further experiments with the English FrameNet corpus we will use annotated dependency graphs produced by RASP and parse selection among the 20 best analyses.

2.5 Syntactic Information in the SALSA Corpus

In the case of the SALSA corpus, the semantic annotation is already associated with tree nodes of phrase structure parses. We would therefore expect annotated dependency graphs of higher quality if it was possible to avoid reparsing these sentences and instead convert the phrase structure trees into dependency graphs while maintaining frame and role labels. This would also make the mapping procedure unnecessary, as the converted structures would already be equipped with semantic annotation. Unfortunately, there are two drawbacks to this approach:

First, conversion from phrase structure trees to dependency graphs is ambiguous in some cases. In the creation of the TiGer Dependency Bank, this necessitated a manual disambiguation procedure, which is not feasible on the complete SALSA corpus. The statistical disambiguation system we employed on parsed structures cannot be applied to converted graphs either, as they lack specific information required by that model. In such cases, we

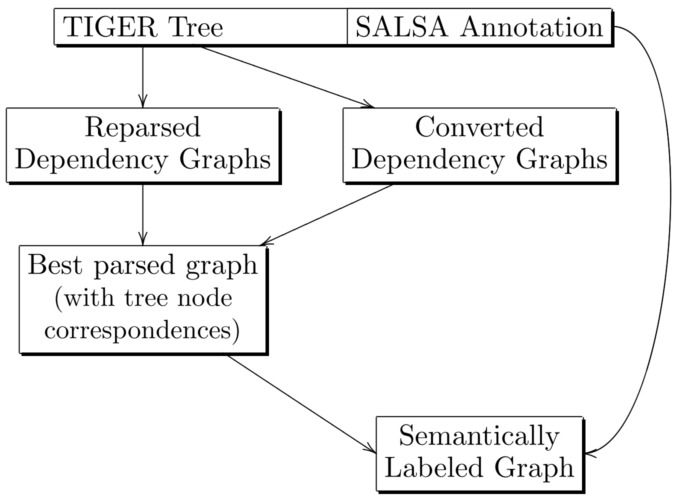

TIGER Tree
SALSA Annotation
Reparsed Dependency Graphs
Converted Dependency Graphs
Best parsed graph (with tree node correspondences)
Semantically Labeled Graph

Figure 2.10: Schematic representation of the combined method employed on the SALSA corpus

can therefore only choose one of the converted graphs at random, which introduces a moderate amount of errors.

More limiting for our purposes is the fact that dependency graphs converted from phrase structure trees differ in many details (such as exact lemmatization or part of speech tags) from those produced by a parser. Our ultimate goal, however, is to use annotated sentences as seeds in the expansion algorithm and compare them to parsed sentences from an unlabeled corpus. It is therefore essential that seeds and expansion sentences are represented by comparable dependency graphs.

Therefore, we chose not to use the converted structures themselves, although annotated dependency graphs can be generated for almost all of them (97.4%) and accuracy is high at 94%(±5%). Instead, we follow an approach of *combining* converted and reparsed structures. For this we operationalize the *converted structures* for disambiguation of the *parsed structures*. The process is shown schematically in Figure 2.10.

The idea here is to compare all parses of a sentence to all readings of the ambiguous converted structure and choose the parse which is most similar to one of the readings. If several parses rank equally in similarity, the statistical parse ranking model is employed to choose the most probable analysis. The similarity between parsed and converted structures is computed on the basis of shared f-structure facts, which are the individual pieces of information collected in f-structures.[3] Using this approach, we selected

[3]Comparison was performed with a script provided by Martin Forst. Because of computational costs, the sets of structures had to be restricted in cases where they were too large (a maximum of 10,000 combinations of a converted and a parsed analysis was allowed). Furthermore, parses with low similarity scores (matching less than 50% of the f-structure facts in the converted analysis) were discarded.

a dependency parse for each sentence, while at the same time maintaining the information of which node in the phrase structure tree corresponds to which node in the dependency graph. This allowed us to obtain labeled dependency graphs from the original SALSA annotations, without recourse to a mapping procedure.

Evaluation shows that 84.4% of the sentences in the SALSA corpus can be converted in this way (for the remaining ones some incompatibility in the syntactic representations causes the annotation information to be lost or corrupted). This is an increase of about 20% over the best result (64.5%) achieved with the mapping algorithm of Section 2.2, which ignores the treebank information. Accuracy of the produced annotated dependency graphs on the other hand does not suffer. It remains essentially stable at 79%(±8%).

All further experiments with the German SALSA corpus will therefore make use of annotated dependency graphs produced by this combined conversion method. The higher success rate of 84.4% will ensure a sufficient number of seed instances for the SALSA corpus, which is considerably smaller than the FrameNet corpus.

2.6 Summary

In this chapter, we have introduced syntactic dependency graphs, and shown how they can be combined with Frame Semantic annotations in a procedure that maps frame and role labels onto graphs nodes. We have discussed the sources of errors in this procedure, identifying parser errors as the major problem. An approach to parse selection was proposed, taking advantage of the disambiguating information inherent in the semantic analyses. We evaluated the performance of the mapping procedure in terms of coverage and accuracy on syntactic analyses of three different parsers, two for English and one for German, showing that selection among the 20 most probable parses substantially reduces the number of mapping failures, while maintaining high accuracy. For the German SALSA corpus, we have described a combined approach of parsing and treebank conversion, which takes advantage of the phrase structure trees available for SALSA sentences and further reduces the number of mapping failures.

In our experiments in Chapters 7 and 8, we will employ labeled dependency graphs as seed corpora for our expansion algorithm. For the English FrameNet corpus, we will make use of the annotation instances mapped onto dependency graphs produced by RASP, while for the German SALSA corpus, those generated from treebank information and LFG parses in the combined method of Section 2.5 will be used. In the following chapter, we now describe the general framework of our semi-supervised approach to SRL.

Chapter 3

General Expansion Framework

This chapter describes the general framework of our semi-supervised SRL approach. We first motivate the basic idea of the expansion algorithm and provide a technical description of its structure. The following sections then describe our notion of sentence similarity and its definition by way of graph alignments. Finally, we discuss the procedure of projecting annotations.

3.1 Motivation

The key idea of our semi-supervised approach to SRL is to automatically generate novel training instances for a supervised classifier by projecting annotations from labeled *seed sentences* to unlabeled sentences. This requires a (possibly rather small) labeled *seed corpus* and a much larger *expansion corpus* without semantic annotation. For example, our expansion corpus might contain a sentence like the following:

(3.1) The rest of his body thumped against the front of the cage.

To infer a Frame Semantic annotation for this sentence we might find a sentence like the following one in our seed corpus:

(3.2) [His back]$_{\textit{Impactor}}$ [**thudded**]$_{\text{IMPACT}}$ [against the wall]$_{\textit{Impactee}}$.

We now wish to determine that the two sentences are similar to the degree that they share a Frame Semantic analysis. If we are also able to associate the different parts of the two sentences with each other in a suitable way, we can then *project* the annotation of the seed sentence onto the unlabeled sentence, and thus obtain a labeled version of (3.1):

(3.3) [The rest of his body]$_{\textit{Impactor}}$ [**thumped**]$_{\text{IMPACT}}$
[against the front of the cage]$_{\textit{Impactee}}$.

Our confidence in the validity of this projection will depend on the similarity of the two sentences. In order to maximize our chances of obtaining a correct annotation, we should therefore look for a seed sentence with maximum similarity to the given unlabeled sentence. For example, the following seed sentence is not substantially similar and therefore projection of its annotation onto (3.1) is not appropriate:

(3.4) [He]$_{Agent}$ [**thumped**]$_{\text{CAUSE\_IMPACT}}$ [his fists]$_{Impactor}$ [on his knees]$_{Impactee}$.

Projecting the annotation from this sentence would erroneously annotate the Cause_impact frame instead of the correct Impact frame shown in (3.3). Moreover, any projection of the role *Agent* would be incorrect.

Modeling Similarity. As we have seen, the way sentence similarity is modeled plays an important role in our approach. However, while high similarity increases our confidence that annotation projection is warranted, we also have to allow for novel information to be acquired in the automatically generated annotation instances. We consider two kinds of such novel information:

As state-of-the-art SRL systems learn their models from annotated training sentences alone, without recourse to additional resources like taxonomies or ontologies, their ability to generalize over different lexical items is limited by the example sentences on which they are trained. We should therefore aim at providing additional training instances exhibiting *lexical variation*, which enable the classifier to learn better models of selectional preferences and semantic classes implicit in the data.

Manually labeled corpora, which are of limited size, cannot exhaustively document the syntactic behaviour of predicates and often exhibit limited coverage of diathesis alternations as shown in Section 1.1, or variability of syntactic realizations in general. Without access to a syntactic verb lexicon, a SRL classifier is therefore severely limited in fully modeling such phenomena. New training instances showing *syntactic variation* would alleviate this problem by exemplifying various syntactic realizations.

While high lexical and syntactic similarity are good indicators that annotation projection is possible, it may therefore nevertheless be desirable to allow new instances to deviate to some extent from labeled seeds in their lexical material and syntactic configuration. To account for this, but still maintain overall high confidence in the validity of our inference, we will model sentence similarity as a trade-off between lexical and syntactic similarity: lower lexical similarity can be counterbalanced by higher syntactic similarity and vice versa.

Selecting Expansion Sentences. Finding similar seed sentences for annotation projection is only part of the problem of generating novel training data. We also need to decide which of the sentences of the expansion corpus should receive annotations and thus become new training instances. This is necessary, because for many sentences in the expansion corpus there will be no sufficiently similar seed sentence, making it impossible to project a correct annotation. Even if all unlabeled sentences could be annotated with reasonable accuracy, there would still be the problem that the expansion corpus is typically orders of magnitude larger than the seed corpus. Including automatically inferred annotations for all of those sentences would lead to an imbalance between a small number of clean manual annotations and a much larger number of noisy automatically inferred annotations, which would inevitably have a negative impact on classifier performance.

We therefore need a procedure to select a subset of the expansion corpus that is suitable as additional training data. We will again base this selection procedure on our confidence in the accuracy of the projected annotations, as estimated by the similarity between the source and target sentences of the projection. Our solution will feature a parameter determining the size of the resulting set of additional training instances. The experiments performed in Chapters 7 and 8 will show that there is a trade-off between the amount and the accuracy of the inferred annotation instances.

A simple alternative to our procedure of measuring similarity and projecting annotations would be to apply a supervised SRL system to label sentences from the expansion corpus and then add them to the training set in a process of self-training. We will show the advantage of our approach over self-training in Section 7.5.

3.2 Structure of the Algorithm

In this section we give a technical overview of our semi-supervised framework. Let L denote a set of sentences labeled with Frame Semantic frames and roles (the *seed corpus*) and U a (much larger) set of unlabeled sentences (the *expansion corpus*), from which we wish to automatically create novel annotated instances. The basic structure of our algorithm is shown in Figure 3.1. It broadly consists of two parts: a *labeling stage* (lines 1 to 20) and a *selection stage* (lines 21 to 26).

Labeling Stage. In the labeling stage, we find the most similar seed instance for each unlabeled instance. Here, an unlabeled instance consists of a sentence $u \in U$ together with a target predicate t in that sentence. Complex sentences may feature several target predicates and thus give rise to different annotation instances, as each of them may evoke a frame.

A given pair (u, t) is compared to each seed $l \in L$. The target predicate of the seed is always its annotated FEE, so formally we are comparing the pairs (u, t) and $(l, \text{FEE}(l))$. To carry out the comparison we first determine an

```
1: Initialize $X_l \leftarrow \emptyset$ for all $l \in L$
2: for $u \in U$ do
3:     for each target predicate $t$ in $u$ do
4:         $s^* \leftarrow 0$
5:         for $l \in$ relevant seeds from $L$ do
6:             $M \leftarrow$ alignment domain of FEE in $l$
7:             $N \leftarrow$ alignment range of $t$ in $u$
8:             $(\sigma, s) \leftarrow$ optimal alignment of $M$ and $N$ and its score
9:             if $(s > s^*)$ and ($\sigma$ covers all roles) then
10:                $l^* \leftarrow l$
11:                $\sigma^* \leftarrow \sigma$
12:                $s^* \leftarrow s$
13:            end if
14:        end for
15:        if $s^* > 0$ then
16:            $u' \leftarrow u$ with annotation projected via $\sigma^*$ from $l^*$
17:            add $(u', s^*)$ to $X_l$
18:        end if
19:    end for
20: end for
21: $X \leftarrow \emptyset$
22: for $l \in L$ do
23:     for $(u', s) \in X_l$ do
24:         add $u'$ to $X$ if $s$ is among the $k$ highest scores in $X_l$
25:     end for
26: end for
27: return $X$
```

Figure 3.1: Expansion algorithm: given a set L of labeled seed sentences and a set U of unlabeled sentences it produces a labeled expansion set X containing the k nearest neighbours of each seed.

alignment domain and an alignment range in the dependency graphs representing u and l. These represent the relevant predicate-argument structures of t and FEE(l) in these graphs. Their definition will be detailed in the next section. We then compare these subgraphs by determining their optimal alignment according to our similarity measure. Section 3.4 will detail this process. This gives us a similarity score for each seed $l \in L$, so that we can determine the highest-scoring seed l^*, its alignment σ^* to u and its score s^*. The seed l^* is then used to project a frame and a set of roles to u. This projection and some additional measures to ensure that alignments cover all role-bearing nodes are described in Section 3.5.

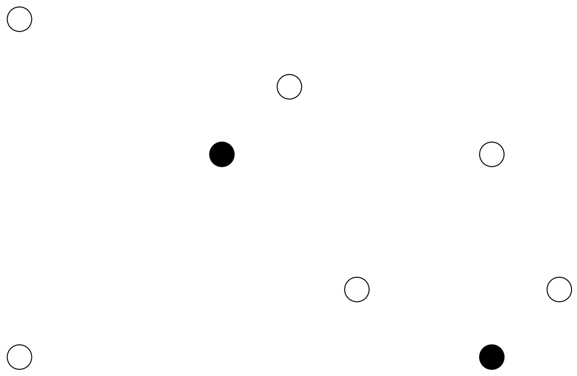
In principle, each unlabeled sentence u is thus compared to each labeled seed l. In practice, however, we reduce the number of comparisons by demanding that u und l have identical or at least similar targets. This effectively reduces computational complexity by avoiding nonsensical comparisons. Details will be given in Chapters 7 and 8, where our experimental setups are described.

The steps of the algorithm are shown schematically in Figure 3.2. In Figure 3.2(a) two seeds and six unlabeled sentences are visualized as solid and empty circles, respectively. The distance between them conveys their mutual similarity, with shorter distances representing higher similarity. Note that this representation is for visualization purposes only and the similarity metric does not actually include any notion of geometric distances. Figure 3.2(b) shows the result of the labeling stage: each of the unlabeled sentences is associated with its most similar seed, indicated by dashed lines. The thick solid line shows how two seeds divide the space of possible sentences into two areas, as any unlabeled sentence is either closer to one or the other seed (we ignore the border case of equal similarity values). After the labeling stage, each unlabeled sentence (with a few exceptions to be detailed in Section 3.5) has obtained an annotation by projection.

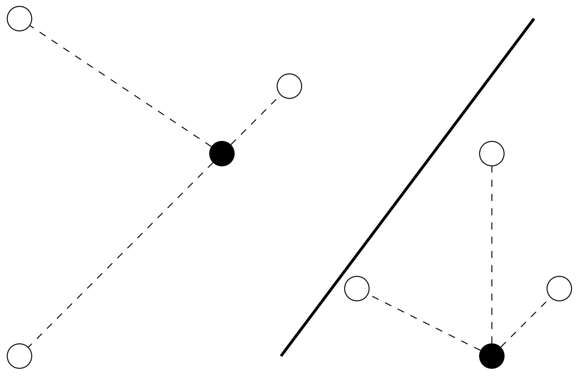
Selection Stage. In the selection stage we now take the similarity between each new instance and its most similar seed, which gave rise to its annotation, as a measure of our confidence in the projected annotation. The instances of the final expansion corpus are then chosen based on this confidence score. A requirement of this selection process is that the number of new instances and the proportions of different FEEs, frames and roles should not primarily depend on the properties of the unlabeled corpus. Favoring frequent instances, for example, would not be justified by the requirements of supervised learning algorithms, which are much more in need of examples for infrequent phenomena. This suggests an approach that selects a number of unlabeled sentences *per seed instance*. We therefore collect, out of all new annotations originating from a particular seed, the k instances scoring highest in their confidence values. Taken together for all seeds, this produces our expansion set X.

Returning to our schematic representation, Figure 3.2(c) shows the result of selecting the $k = 2$ most similar neighbours of each seed. The four selected sentences are shown in gray. Note how our two-stage procedure differs from simply selecting the two most similar unlabeled instances for each seed. Under such an approach the unlabeled sentence indicated by the arrow would be the second-closest neighbour of either of the two seeds, leading to conflicting evidence as to which of the two its annotation should be projected from. In our approach, however, the preceding labeling stage has already uniquely identified a source of projection for each sentence and the selection only takes place *among those instances.*

(a) Schematic view of the distances between two seeds (black circles) and six unlabeled instances (white circles).



(b) Associating each unlabeled instance with its most similar seed for projection. The thick line indicates equal distance from both seeds.



(c) Selecting the two most similar new instances for each seed. The node indicated by the arrow is the second nearest neighbour of both seeds.

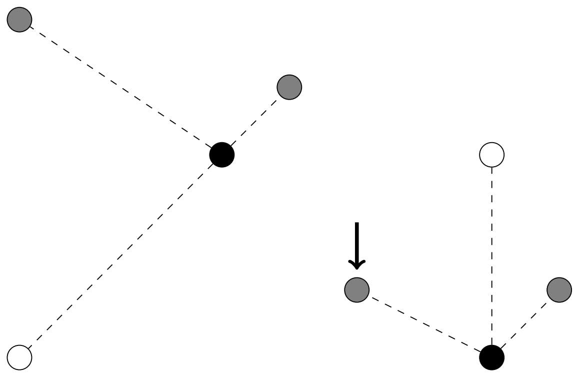


Figure 3.2: Schematic view of the labeling and selection stages

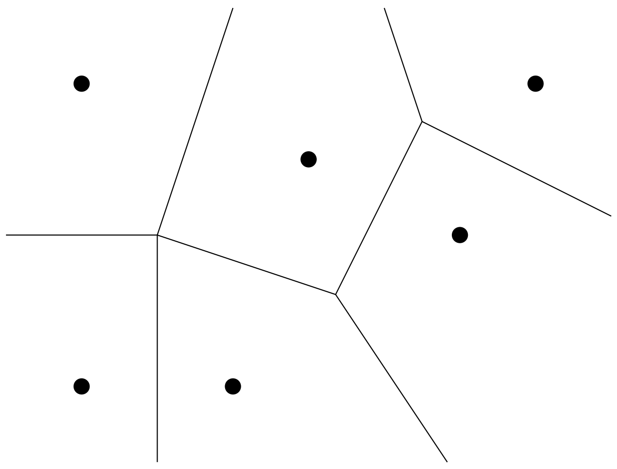


Figure 3.3: Voronoi cells of six seeds

More generally, we can say that the set of seeds divides the space of possible sentences into regions corresponding to their Voronoi cells. In analogy to its geometric definition, we define the Voronoi cell of a seed to be the set of all possible sentences more similar to this seed than to any of the other seeds. An example with 6 seeds is visualized in Figure 3.3. Here, it is obvious that the Voronoi cells are not symmetric around the seeds, and a sentence relatively close to a particular seed may fall within a different cell while a less similar one may still be in the same cell. Under this interpretation, the labeling stage of our algorithm corresponds to determining for each unlabeled sentence the Voronoi cell it falls into, while the selection stage collects the k most similar neighbours of each seed *within its own cell.* This clearly shows that in our approach the manually labeled seed instances define the structure of the space of example sentences and the unlabeled sentences fill this structure with additional pieces of evidence.

In the following sections we will detail our definition of similarity between a labeled and an unlabeled sentence. In Section 3.3, we first define the concepts of graph alignments between an alignment domain and an alignment range. This will enable us to formulate a similarity score in Section 3.4. The details of the projection procedure are then described in Section 3.5.

3.3 Graph Alignments

Before we give a definition of our similarity metric in the next section, we first have to introduce the concept of an alignment between specific subgraphs in the dependency analyses of a labeled and an unlabeled sentence.

Alignments are a widely employed concept in computational linguistics. In the simplest case, the task is to find an optimal relation between the objects of two sets without internal structure. An example of this is the *assignment problem*, which can be interpreted as the assignment of tasks to agents, with each agent capable of performing at most one task and each assignment of a task to an agent incurring a specific cost. This problem can be solved efficiently, e.g., with the so called Hungarian Algorithm (Kuhn, 1955). If linear orders on the two sets have to be respected, the problem becomes one of *sequence alignment*. This is, e.g., the case for edit distances like the Levenshtein distance. Transforming one string into another by successive substitutions, deletions and insertions can be viewed as a sequence alignment between the two strings where alignment of different characters incurs the substitution cost and unaligned characters of either string incur deletion or insertion costs, respectively. Further examples of sequence alignment problems abound in NLP, such as alignment of sentences, phrases, or words of bilingual corpora in statistical machine translation systems, alignment between speech signals and phonetic representations in speech recognition and synthesis, or alignment between different instances of event sequences in the temporal analysis of discourse structure, to name but a few.

In *graph alignment* the complexity of the structure imposed on the aligned sets is taken a step further. Instead of linear order, we now assume an arbitrary binary relation, defining the edges between graph nodes. Consequently, when aligning two graphs, we optimize a measure based on the similarity of *aligned nodes* and the degree to which the graph structure is respected. In an alignment that is perfectly compatible with the graph structure, a pair of nodes would be connected by a graph edge if and only if their aligned partners in the other graph are also connected by an edge. Any deviation from this structural compatibility will incur a corresponding cost. We apply the concept of graph alignments to our problem by defining alignments between specific subgraphs of the dependency analyses of labeled and unlabeled sentences.

Alignment Domain. We do not try to align complete dependency graphs, because we are only interested in a single predicate-argument structure of each sentence and other parts of the sentence are not relevant to the similarity comparison. We first characterize the *alignment domain* M of a labeled seed sentence, which is a subgraph of its dependency graph and corresponds to the predicate-argument structure of its FEE. Let p be the FEE node of the graph. If there are no mismatches between what constitutes semantic and syntactic arguments, we expect all roles in the graph to be instantiated by syntactic dependents of p. While this is often the case, it does not always hold, e.g., because of the way the dependency parser analyses raising, control or coordination structures. We therefore cannot simply define M as the set of direct dependents of the predicate, but rather also have to consider *complex paths* between p and role-bearing nodes. An example is given in

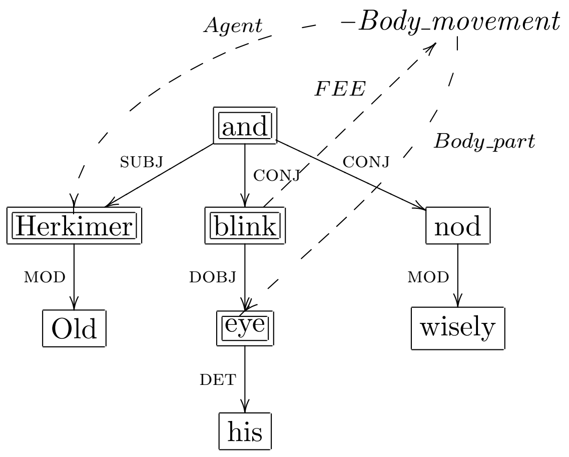


Figure 3.4: Annotated dependency graph of the sentence *"Old Herkimer blinked his eye and nodded wisely"* with the annotation domain indicated by double frames.

Figure 3.4, where the role *Agent* is filled by a node which is not dominated by the FEE *blink*; instead, it is connected to *blink* by the complex path (CONJ^{-1}, SUBJ). This reflects the fact that *Old Herkimer* is subject of both of the coordinated predicates *blink* and *nod*. For a given sentence we build a list of all such complex paths to any role-bearing node and include all nodes connected to p by any of these paths in the alignment domain M. We thus define the subgraph M to contain

(1) the predicate node p

(2) all direct dependents of p, except auxiliaries

(3) all nodes on complex paths originating in p

(4) single direct dependents of any preposition or conjunction node which is in (2) or end-point of a complex path covered in (3)

The first two items require little justification. The predicate node is naturally part of the predicate-argument structure and its direct syntactic dependents are prime candidates for semantic arguments, while auxiliaries do not contribute information relevant to Frame Semantic analyses. Note that it is not essential to exclude all syntactic dependents which are not semantic arguments. Remaining adjuncts may realize non-core semantic roles or provide lexical and syntactic context information even when not realizing any semantic role. The inclusion of nodes on complex paths in (3) is necessary for obtaining a connected subgraph: as long as the role-bearing end point of the path is included, the path leading there must also be considered in order to capture the complete syntactic relation between the predicate and its argument.

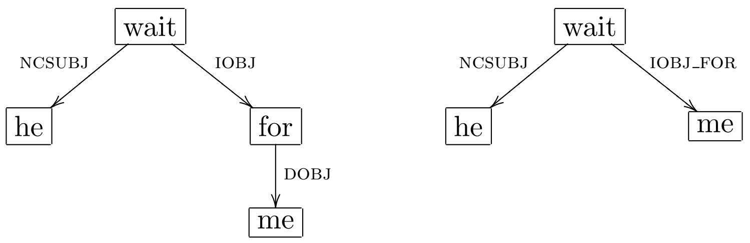


Figure 3.5: Example of merging the preposition in the sentence *"He waited for me"*. In the merged version on the right, the edge label IOBJ_FOR includes the preposition word *for*.

Our procedure in general does not include indirect dependents of the predicate node, e.g., dependents of its dependents. This reflects the fact that typically the syntactic heads of the dependent constituents correspond to the semantic heads of arguments. Including indirect dependents would therefore lead to overly specific comparisons of role instantiations, picking up various modifiers. For example, instead of comparing the words *cat* and *dog*, we might have to compare subgraphs representing the phrases *Abyssinian cat* and *lap dog*. It is not unreasonable to assume that such modifiers can have significant influence on the sense of the semantic head, either by disambiguating an ambiguous head or by forming an idiomatic expression together with it. A *hot dog*, e.g., should be less similar to an *Abyssinian cat* than a *lap dog*. For simplicity, we only take into account specific forms of this phenomenon: the syntactic analyses of prepositional phrases and subordinate clauses.

Prepositional phrases are usually analyzed with the preposition as the syntactic head of its nominal complement. The semantic status of prepositions between function words and content words is a topic of ongoing research (Saint-Dizier, 2006; Litkowski and Hargraves, 2005). However, we typically want to consider the head of the nominal complement of a preposition as the semantic head of the entire prepositional phrase. A common approach to deal with this mismatch between syntax and semantics in SRL is to *merge* preposition words into grammatical relations. Figure 3.5 shows the RASP analysis of the sentence *"He waited for me"* on the left and a merged version of the dependency graph on the right. Here, the preposition word *for* has been merged with the relation IOBJ, which occurs between preposition words and their heads, yielding the composite relation IOBJ_FOR. This allows us to view the head of the nominal complement (in this case the word *me*) as a direct syntactic dependent.

Our graph alignment approach allows us to leave the original dependency graph intact and instead include both the preposition word and its nominal head into the alignment domain. This has the advantage that preposition words can be compared lexically in the same way as any other graph node,

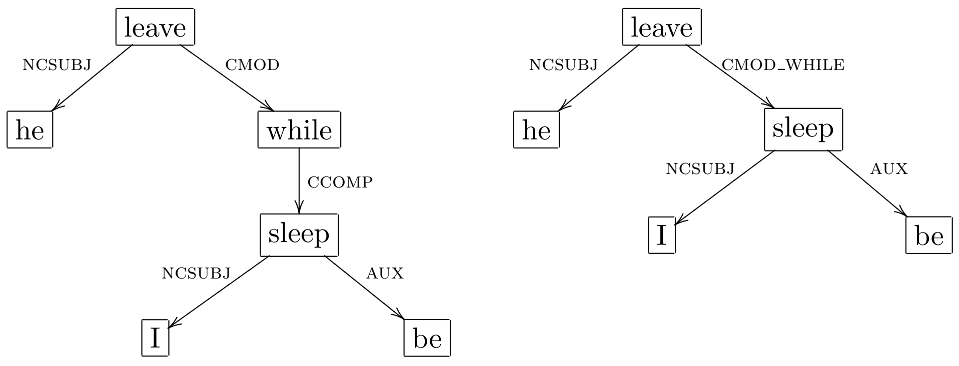


Figure 3.6: Example of merging the conjunction word in the sentence *"He left while I was sleeping"*. Here the word *while* is merged into the new edge label CMOD_WHILE.

while still capturing the information of the nominal complement head. A very similar situation arises for subordinate clauses headed by conjunctions. Figure 3.6 shows an analysis of the sentence *"He left while I was sleeping"* and a version obtained by merging the conjunction word *while* with the corresponding relation CMOD. Again, we can avoid the ad-hoc solution of modifying the dependency graph by simply including dependents of conjunctions in the alignment domain. For both prepositions and conjunctions, this is achieved by rule (4) above.

Alignment Range. Having defined the alignment domain in the semantically labeled dependency graph, we must now define a corresponding subgraph in the unlabeled dependency graph. This *alignment range* represents the predicate-argument structure of the *target predicate*.

While the process for identifying nodes of this subgraph would ideally follow the exact same procedure as for the labeled graph, the definition of complex paths forces us to introduce an asymmetry. We cannot determine a list of complex paths for the unlabeled graph because complex paths are defined by the syntactic relations between the predicate and role-bearing nodes. Since an unlabeled graph has no role annotations, the appropriate complex paths are therefore also unknown.

The simplest solution to this problem would be to ignore complex paths altogether in the case of unlabeled graphs and only include syntactic dependents in the alignment range. However, this approach assumes that unlabeled sentences are structurally simpler than labeled ones. This assumption, for which there is no justification, would lead to the incorrect alignment of complex unlabeled sentences to structurally simpler seed sentences, or to their omission due to the lack of a suitable alignment. Both outcomes are detrimental to the quality of the resulting training set, because they introduce a bias towards simple structures for the new training instances, either by missing semantic roles or through the systematic exclusion of complex example sentences. Such a bias in the training data would lead an SRL classifier to learn skewed models and ultimately result in sub-optimal performance.

On a more formal level, ignoring complex paths for unlabeled graphs entails a similarity metric under which a sentence does not necessarily align perfectly to itself. Guided by this intuitive requirement, we address the problem by reusing the list of complex paths extracted from the labeled partner of an unlabeled sentence. This solution is not ideal either: it makes the comparison asymmetrically dependent on the annotation of the labeled sentence. However, in this regard, it only reflects the asymmetry inherent in comparing semantically labeled and unlabeled sentences.

We therefore define the alignment range N in exact analogy to the alignment domain M, i.e., following the rules (1) to (4) above, with the only difference that the notion of complex paths is taken from the labeled partner in the comparison being performed.

Alignments. An *alignment* between an alignment domain M and an alignment range N can be formalized as a function $\sigma : M \to N \cup \{\epsilon\}$. A node $x \in M$ is said to be *aligned* to a node $x' \in N$, if $\sigma(x) = x'$. We do not require σ to be *onto* N or surjective, so that nodes in N may be unaligned. Similarly, a node $x \in M$ may be unaligned, which is expressed by $\sigma(x) = \epsilon$. To ensure that no node in either set is aligned to more than one in the other, we require that $\sigma(x) = \sigma(x') \neq \epsilon$ implies $x = x'$.

Figure 3.7 shows an example of an alignment for the two sentences *"His back thudded against the wall"* and *"The rest of his body thumped against the front of the cage"*. The labeled graph of the first sentence is shown on the left, annotated with the Impact frame. The nodes in its alignment domain are indicated by double frames. To the right, the unlabeled graph of the second sentence is shown, with the alignment range indicated by double frames. One possible alignment σ is indicated by waved arrows between nodes. There are many more possibilities. In principle, any node in one of the two subgraphs could be aligned to any in the other one or be left unaligned, as long as no node has more than one aligned partner.

To compute the total number of possible alignments of two graphs with m and n nodes respectively, we first assume that $m \leq n$. Let us further assume that $k \leq m$ nodes of the first graph are aligned to nodes of the second graph. According to basic combinatorics, there are $\binom{m}{k} = \frac{m!}{(m-k)!k!}$ ways to choose those k nodes. For each of these choices there are $n \cdot (n-1) \cdots (n-k+1) = \frac{n!}{(n-k)!}$ ways to align them to k different nodes of the second graph. The factors in this product correspond to the number of choices for the k individual alignment links, which successively decreases as nodes of the second graph are aligned and thus become unavailable. There-

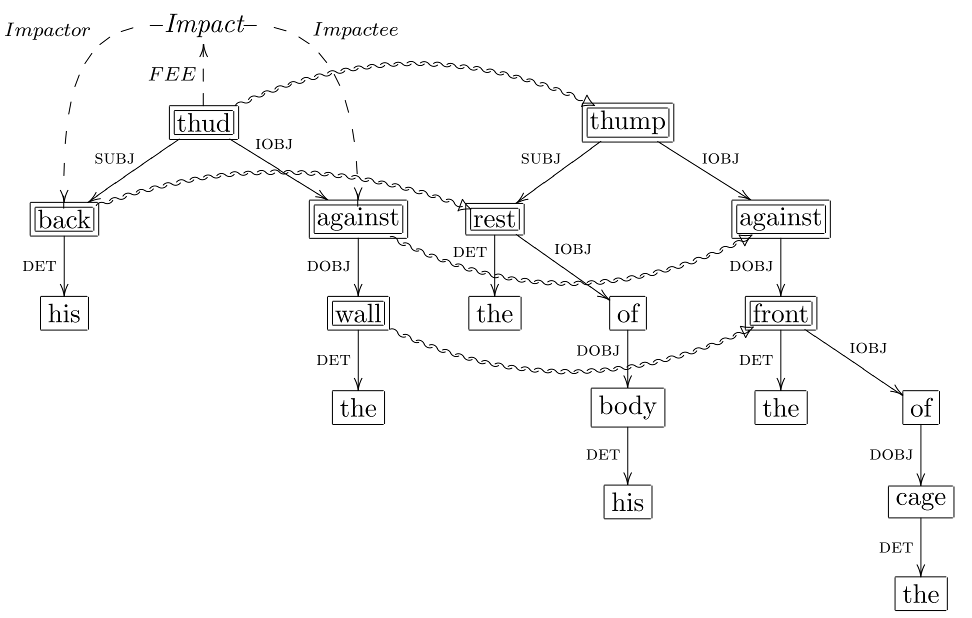


Figure 3.7: Alignment between a labeled and an unlabeled dependency graph. Alignment domain and alignment range are indicated by double frames, with waved arrows linking aligned nodes.

fore, we have $\frac{m!n!}{(m-k)!(n-k)!k!}$ choices for an alignment of k nodes. This term is symmetric in m and n, which means that our initial assumption of $m \leq n$ is unnecessary: if $m > n$, we simply swap the notation and still obtain the same result. An alignment may have any number of aligned nodes between 0 and m (for $m \leq n$), or 0 and n (for $m > n$), and therefore k may range between 0 and $\min(m, n)$. The overall number of alignments can be expressed as the sum of the numbers of alignments for each k:

$$c(m,n) := \sum_{k=0}^{\min(m,n)} \frac{m!n!}{(m-k)!(n-k)!k!} \tag{3.5}$$

We will impose only one further restriction upon graph alignments, namely that the nodes of the FEE and the target predicate be aligned to each other. Any other alignment of those two nodes would violate the basic assumption that our subgraphs represent predicate-argument structures, i.e., contain well-defined predicate nodes. With one node on either side pinned down, the number of possible alignments therefore is $c(m-1, n-1)$.

3.4 Similarity Scores

Based on the concept of graph alignments, we now develop a measure of similarity between two predicate-argument structures. Our basic assumption is that it is possible to identify *corresponding words and grammatical relations* in two comparable predicate-argument structures, and express these correspondences in an *optimal alignment.* We will first formulate a scoring function expressing the similarity reflected by this optimal alignment. Afterwards we will show that in general the optimal alignment can be expected to maximize this scoring function. This will give us a way of finding the optimal alignment in practice by maximizing the scoring function over all possible alignments.

Scoring Function. Given two predicate argument structures represented by an alignment domain M and an alignment range N, we assume for the moment that we know their optimal alignment σ^*, reflecting the most appropriate correspondences between nodes and edges. The easiest way of quantifying the similarity expressed by σ^* is to sum similarity values for each individual aligned node and edge pair:

$$\text{score}(\sigma^*) := \sum_{\substack{x \in M \\ \sigma^*(x) \neq \epsilon}} \text{lex}\left(x, \sigma^*(x)\right) + \alpha \cdot \sum_{\substack{(x_1,x_2) \in E(M) \\ (\sigma^*(x_1),\sigma^*(x_2)) \in E(N)}} \text{syn}\left(r^{x_1}_{x_2}, r^{\sigma^*(x_1)}_{\sigma^*(x_2)}\right) \tag{3.6}$$

Here, lex and syn are arbitrary measures of similarity between words and grammatical relations, respectively. The first sum comprises similarity values for each pair of an aligned node x in the alignment domain M and its partner $\sigma^*(x)$ in the alignment range N. It therefore stands for the overall lexical similarity expressed by σ^*. In the second sum, $E(M)$ and $E(N)$ are the sets of edges in the alignment domain and alignment range, respectively. For each edge (x_1, x_2) in $E(M)$, the aligned nodes $\sigma^*(x_1)$ and $\sigma^*(x_2)$ in N may or may not be connected by an edge, i.e., $(\sigma^*(x_1), \sigma^*(x_2))$ may or may not be in $E(N)$.[1] If they are, the edge pair contributes a syntactic score corresponding to the similarity of the two corresponding edge labels $r^{x_1}_{x_2}$ and $r^{\sigma^*(x_1)}_{\sigma^*(x_2)}$. The factor α scales the contribution of the syntactic similarity measure to the overall score. This accounts for the fact that the two measures lex and syn represent fundamentally different kinds of information, whose relative weight may need adjusting. In our experiments we will determine the value of the weight parameter α in a parameter tuning procedure.

The functional form of this similarity measure satisfies the requirements formulated in Section 3.1. It depends both on lexical and syntactic similarity and expresses the two in a trade-off: lower lexical similarity can be directly compensated by higher syntactic similarity and vice versa. In Chapter 4, we will provide different instantiations of lex, alternatively based on distributional similarity or WordNet information, and discuss possible definitions of syn, settling on a binary measure. The simple formulation of similarity in

[1]If x_1 or x_2 is not aligned, then formally $\sigma^*(x_1) = \epsilon$ or $\sigma^*(x_2) = \epsilon$ and therefore $(\sigma^*(x_1), \sigma^*(x_2)) \notin E(N)$.

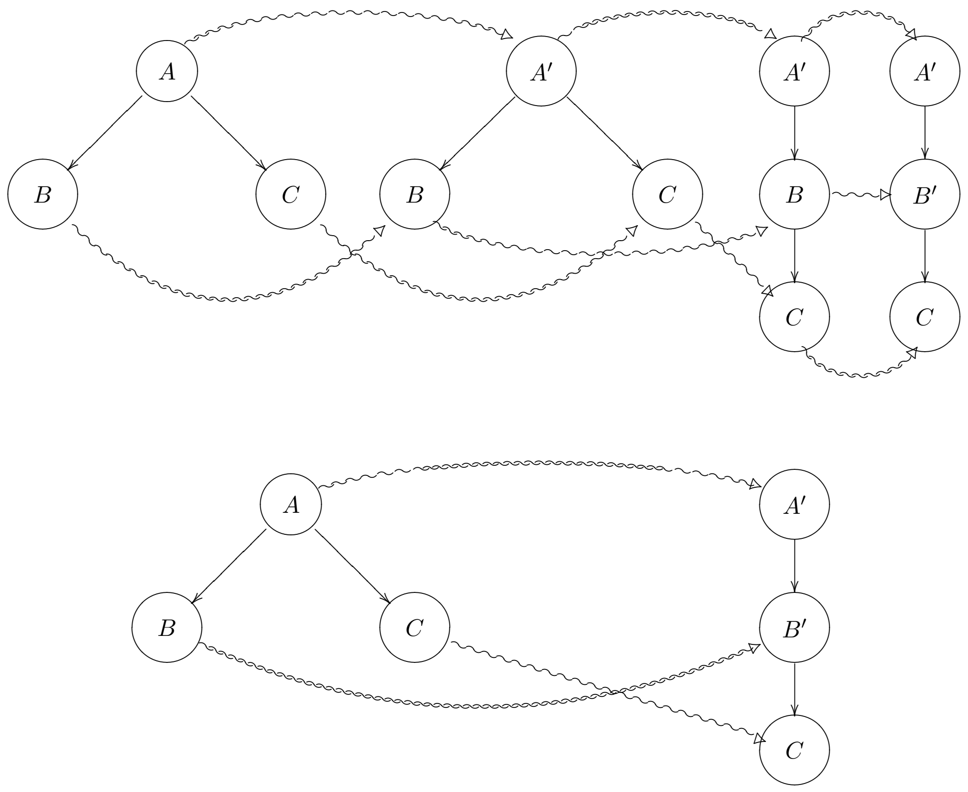


Figure 3.8: Illustration of optimal graph alignments as the result of a step-wise transformation process. Graph edges are drawn as solid arrows while waved arrows show node correspondences.

terms of a weighted sum over node and edge pairs will allow us to employ efficient linear programming algorithms in Chapter 5.

The Optimal Alignment. Since our definition of similarity between predicate-argument structures is based on optimal alignments , we have to address the question of how to determine such an optimal alignment in the first place. To find correspondences between nodes in two dependency graphs, we imagine a transformation process similar to the one underlying the definition of classical edit distance metrics. Starting with one of the two graph, we apply lexical and syntactic changes until it has been transformed into the other one. The identity of nodes can be maintained throughout such an imaginary transformation, yielding an alignment of corresponding nodes. This is the optimal alignment between the two graphs.

Figure 3.8 shows a simple schematic illustration of such a transformation process. In the top row, a graph is transformed in four steps. Going from left to right, first the node label A is substituted by A', then the graph structure is changed so that C is now a child node of B rather than A, and finally B is substituted by B'. The bottom part of the figure shows the resulting alignment between the original and the final version of the graph. We may assume that A and A', as well as B and B', stand for comparatively similar words. Consequently, an alternative alignment, e.g., associating A with B' and B with A' would result in lower overall similarity in terms of the score (3.6). As long as the applied changes are not too big, the resulting alignment thus maximizes the similarity score, since any change in the alignment of a node can be expected to lower lexical or syntactic similarity. Although we do not completely formalize these transformations or precisely delineate the class of "small changes" for which our conclusions hold, this observation strongly motivates a definition of the optimal alignment as that alignment which maximizes the similarity score (3.6):

$$\sigma^* = \arg\max_{\sigma}(\text{score}(\sigma)) \tag{3.7}$$

Graph Similarity. We can now define the final similarity measure between the subgraphs M and N as:

$$\text{sim}(M, N) = \frac{1}{C}\,\text{score}(\sigma^*) = \frac{1}{C}\max_{\sigma}(\text{score}(\sigma)) \tag{3.8}$$

Here, C is a normalization factor, whose purpose is to make similarity scores of different pairs of sentences comparable. Without such normalization, alignments involving complex predicate-argument structures would tend to receive higher similarity scores than those between simpler ones, because a larger number of nodes and edges will tend to make the sums of lexical and syntactic scores larger. This is, of course, counter-intuitive. A straightforward solution is to normalize "self-similarity", i.e., the similarity of a graph to itself, which is an upper bound for the similarity to any other sentence. Assuming that lex and syn take a maximum of 1 on identical words and grammatical relations, the self-similarity of an alignment domain M is $|M| + \alpha|E(M)|$. We could use this term as a normalization factor. However, this would only account for the size of the alignment domain and ignore the size of the alignment range. A suitable normalizing factor should treat both graphs symmetrically. We achieve this by replacing the term $|M|$ by the geometric mean $\sqrt{|M| \cdot |N|}$ of $|M|$ and $|N|$, and similarly the term $|E(M)|$ by the geometric mean $\sqrt{|E(M)| \cdot |E(N)|}$. This results in a normalization factor

$$C := \sqrt{|M| \cdot |N|} + \alpha\sqrt{|E(M)| \cdot |E(N)|} \tag{3.9}$$

which is symmetric in the two graphs, while still normalizing self-similarity to 1, since

$$\frac{1}{\sqrt{|M|^2} + \alpha\sqrt{E(M)^2}}\,(|M| \cdot 1 + \alpha \cdot |E(M)| \cdot 1) = 1$$

Although exploration of different normalization factors may be worthy of some further attention, it is unlikely to have a large impact on the scoring function, for which the concrete measures lex and syn are more important.

3.5 Annotation Projection

We have defined the similarity between the predicate-argument structure of the FEE of a labeled graph and that of the target predicate of an unlabeled graph in terms of the score of the optimal graph alignment between them. This allows us to determine the most similar labeled seed l^* for any unlabeled sentence u. What remains to be described is the projection procedure leading to a labeled version of u.

Given the optimal alignment σ^* between an alignment domain M and an alignment range N, the basic definition of semantic projection from l^* to u is rather straightforward: as described in Section 2.2, frame names are associated with the nodes of their FEEs and role names with the nodes of their role filler heads. By definition, all these nodes are in the alignment range M. We can therefore simply label $\sigma(x) \in N$ with the same role as the one carried by x, for each role-bearing node $x \in M$. The only complicating factor is the possibility of unaligned nodes, i.e., nodes x with $\sigma(x) = \epsilon$. While the notion of unaligned nodes is useful for ignoring irrelevant words, we have to decide what to do if the optimal alignment σ leaves a role-bearing node unaligned. (This problem cannot occur for the FEE, as we have assumed that it is always aligned to the target predicate node.)

A possible solution would be not to project roles annotated on unaligned nodes, so that only roles associated with aligned nodes show up in the inferred annotation. Unfortunately, allowing such partial projections introduces a systematic bias in favour of simple structures: roles annotated on seed sentences can be discarded, but new roles cannot arise, so that the generated instances will on average show fewer roles than the seed sentences. When these new instances are used for training a role labeler, they will in turn bias the classifier towards under-annotating roles and thus decrease performance.

The most straightforward way to enforce the constraint that $\sigma(x) \neq \varepsilon$ for all role-bearing nodes x would be to impose it on the alignments in the maximization (3.7), i.e., to find a (possibly lower scoring) solution satisfying the condition that each role-bearing node is aligned. However, in cases where it leads to a different alignment, this approach would ignore the information that for at least one role-bearing node no good correspondence could be found. Rather than resort to a lower-ranking alignment, it is therefore much more appropriate to conclude that the given seed l^* is not a good source of information for the labeling of u. As our overall approach calls for precision rather than recall, we therefore choose to dismiss the seed l^* as a partner for u if the optimal alignment σ^* fails to align all role-bearing nodes. The unlabeled sentence u may still receive an annotation projected from a different seed in L.

In some cases it is possible that an unlabeled sentence u will not find any suitable seed for annotation projection. This happens when there is no seed for which the optimal alignment covers all role-bearing node and has a similarity score greater than 0. In these cases, u is discarded, as no suitable annotation can be inferred. The majority of unlabeled sentences, however, will obtain some annotation and thus be available to the selection stage, in which the most similar neighbours of each seed are determined.

3.6 Summary

In this chapter, we have described a general framework for semi-supervised semantic role labeling. We have motivated our projection approach based on lexical and syntactic similarity and given an overview over the structure of the algorithm. The two stages of *labeling* and *selection* were interpreted in terms of the space of possible sentences, which is structured by labeled seeds and then filled with more examples in the form of unlabeled sentences. We have then detailed the individual parts of the algorithm, introducing alignments between the relevant predicate-argument structures of labeled and unlabeled sentences, formulating a similarity measure in terms of optimal alignments, and finally describing the process of annotation projection.

In the following chapter we will now present definitions of the similarity measures lex and syn introduced in (3.6), while Chapter 5 will address the optimization problem and describe an efficient solution algorithm.

Chapter 4

Lexical and Syntactic Similarity

In the previous chapter, we have described the general framework of our semi-supervised SRL approach. The scoring function defined in Section 3.4 features two measures, called lex and syn, of lexical and syntactic similarity, which were left unspecified. In this chapter, we now present concrete instantiations of these two measures. We will provide two alternative definitions of the measure lex, one based on a vector space model of distributional similarity and one relying on the WordNet taxonomy. The latter has the advantage of taking into account high-quality information from a manually created resource. On the other hand, this dependence precludes its use for languages where such a broad-coverage resource is not available. In Chapter 7, we will compare the results of employing either of the two measures in our expansion algorithm. For the measure of syntactic similarity, we will give a simple definition based on the identity of grammatical relations, and discuss possibilities of more gradual definitions.

An important requirement for the two measures is that their computation must be efficient, as it is performed on a large number of word and edge pairs. Syntactic similarity can be determined efficiently by precomputing a table of all possible pairs of grammatical relations. However, this is not practical for lexical similarity, as the number of possible word pairs is prohibitively large. Apart from memory requirements, the skewedness typical of word distributions would cause most of the entries in a table of lexical similarity values to be accessed only once or not at all, so that no performance benefit can be expected from a precomputed table. We therefore compute the lexical similarity measures on-line.

In the following sections, we will introduce the different similarity measures, discuss their computational complexity, and present data structures and algorithms for their efficient computation.[1] Afterwards, we will briefly discuss the relative weight of lexical and syntactic similarity.

4.1 Vector Space Model of Lexical Similarity

Vector space models of word meaning (Lund and Burgess, 1996; Landauer and Dumais, 1997) are an increasingly popular method of measuring lexical similarity. Their basic idea goes back to the very inception of the field of computational linguistics, when Weaver (1955) claimed that the problem of polysemy encountered in automatic translation could be resolved by taking neighboring words into consideration. This intuition becomes concrete in the distributional hypothesis, put forward by Harris (1968), which states that words occuring in similar contexts are themselves similar and is succinctly expressed in the quotation "You shall know a word by the company it keeps" of Firth (1957).

In vector space models, the meaning of a word is represented by a vector whose components correspond to co-occuring words and take values reflecting the frequency of co-occurence. The simplest context to consider for such co-occurence is a *context window* consisting of up to n words to the left and to the right. Commonly, a numeric similarity value for a pair of such vectors is derived by computing the cosine of the angle between them. Although there is no compelling theoretical justification for a geometric interpretation, this measure has proved successful in various applications and comparisons with other measures (Lee, 1999; Weeds et al., 2004), and additionally has the advantage of being easy to compute.

More complex variants of this basic approach have been proposed. For example, contexts may vary in size from a few words up to whole documents, as in latent semantic analysis (Landauer and Dumais, 1997). Instead of simple frequency counts, association measures such as pointwise mutual information have been proposed to account for chance co-occurence. Furthermore, there is a wide range of functions of vector similarity, inlcuding Euclidean distance (or L^2 norm), Manhattan distance (or L^1 norm), Jaccard coefficient, as well as information theoretic measures such as Kullback-Leibler divergence or Jensen-Shannon divergence. Another direction for the extension of simple vector space models is the inclusion of *syntactic information*. Padó and Lapata (2007) give a general framework for how to select context words based on their grammatical relations to the word being characterized.

A fundamental problem of classical vector space models is their insensitivity to polysemy. By aggregating co-occurence counts over all occurences of a word, they fail to provide separate representations of the different senses

[1]A free implementation computing the two lexical similarity measures is available at `http://www.coli.uni-saarland.de/~hagenf/software/svectors/`.

a word may have. The resulting vector representation therefore combines these senses in a weighted average, which favours the (often very dominant) most frequent sense of a word. Recently, a number of approaches have been proposed to integrate the word sense disambiguation task into the derivation of vector representations. This is achieved by modifying or *contextualizing* the vector of a word depending on the context of its concrete occurrence. Mitchell and Lapata (2008) compare various operations to contextualize vectors in simple models based on context windows, achieving good results with component-wise vector multiplication. Erk and Padó (2008) take syntactic relations into account and contextualize words with the selectional preferences of syntactically related words. Recently, Thater et al. (2010) have proposed a model based on first- and second-order syntactic co-occurences.

Definition. For the corpus-based formulation of our lexical similarity measure, we employ a relatively simple vector space model, found to correspond well to human similarity judgements (Mitchell and Lapata, 2011). However, within our flexible framework this could easily be replaced by a more advanced vector space model. Specifically, we build vectors whose dimensions correspond to the 2,000 most frequent words in the unlabeled expansion corpus and consider co-occurence within a context window extending 5 words to the left and 5 words to the right. For a given word w, the vector component corresponding to the context word w' is set to be the following ratio of probabilities of w and w':

$$v_{w,w'} := \frac{p(w,w')}{p(w)p(w')} = \frac{p(w'|w)}{p(w')} \tag{4.1}$$

The first form of this ratio may be interpreted as the degree to which the probability $p(w,w')$ of w and w' co-occuring in the context window is higher than the probability of chance co-occurence, which is proportional to $p(w)p(w')$. The second form allows an asymmetric interpretation: values greater than 1 reflect the degree to which the probability of encountering w' is higher in the context of w than in general. $v_{w,w'}$ is closely related to the pointwise mutual information (PMI) of w and w', by $\mathrm{PMI}(w,w') = \log(v_{w,w'})$.

We compute the probabilities by maximum likelihood estimation as the ratio of frequency counts over the unlabeled expansion corpus: $p(w'|w)$ is estimated as the number of occurences of w' in the context of w divided by the number of occurences of w, while for $p(w')$ we divide the number of occurences of w' by the total number of word tokens in the corpus. As the part of speech (POS) tags of all words in the corpus are available from their parses (which are needed for our expansion procedure), our vector space model takes into account these POS tags as well. For example, the noun *shock*, represented as *shock.N*, and the verb *to shock*, represented as *shock.V*, are different word types in our model. We make only coarse-grained POS distinctions between nouns (N), verbs (V), adjectives (J), prepositions (I), conjunctions (C), and all other categories (X).

Based on two lemmatized words w_1 and w_2 and their coarse-grained POS tags p_1 and p_2, we define lexical similarity as follows:

$$\mathrm{lex}(w_1, w_2) := \begin{cases} \cos(w_1, w_2) & \text{if } p_1 = p_2 \in \{N, V, J\} \\ \delta_{w_1,w_2} & \text{if } p_1 = p_2 \in \{I, C\} \\ 0 & \text{else} \end{cases} \tag{4.2}$$

Here, δ_{w_1,w_2} takes the value 1 or 0 depending on whether $w_1 = w_2$ or not. This reflects the fact that vector space models are not appropriate for modeling the semantics of closed class words such as prepositions and conjunctions. Pairs of nouns, verbs, and adjectives are compared by way of the cosine function, while all pairs with other or mixed POS tags are given a similarity value of 0, as they are unlikely to raise our confidence in the given alignment.

Efficient Computation. We now discuss how to efficiently compute cosine similarity between two word vectors (the other cases of definition (4.2) are trivial). Let N be the number of dimensions of the vector space (in our case 2,000) and two vectors $\vec{v}$ and $\vec{w}$ be described by real numbers a_i and b_i ($1 \leq i \leq N$), respectively. The cosine between the two vectors can be expressed as

$$\begin{aligned} \cos(\vec{v}, \vec{w}) &= \frac{\vec{v} \cdot \vec{w}}{||\vec{v}|| \cdot ||\vec{w}||} \\ &= \frac{\sum_{i=1}^{N} a_i b_i}{\sqrt{\sum_{j=1}^{N} a_j^2} \cdot \sqrt{\sum_{j=1}^{N} b_j^2}} \\ &= \sum_{i=1}^{N} \left(\frac{a_i}{\sqrt{\sum_{j=1}^{N} a_j^2}} \cdot \frac{b_i}{\sqrt{\sum_{j=1}^{N} b_j^2}} \right) \end{aligned} \tag{4.3}$$

The first obvious optimization here is to store *normalized* versions of the vectors $\vec{v}$ and $\vec{w}$ instead of the vectors themselves. Having precomputed

$$\tilde{a}_i := \frac{a_i}{\sqrt{\sum_{j=1}^{N} a_j}} \tag{4.4}$$

for the vectors of all lexical items, it takes only N multiplications and $N - 1$ additions to compute:

$$\cos(\vec{v}, \vec{w}) = \sum_{i=1}^{N} \tilde{a}_i \tilde{b}_i \tag{4.5}$$

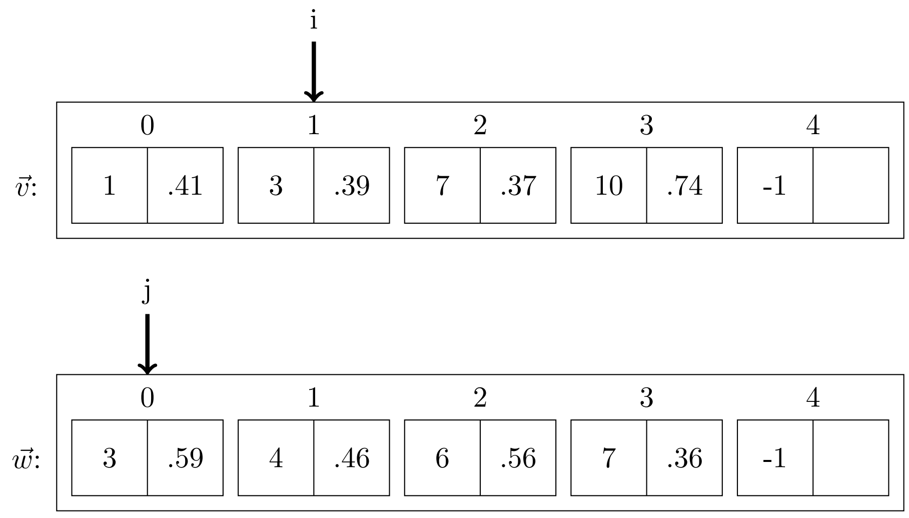


Figure 4.1: Illustration of computing cosine similarity on sparse vector representations, containing pairs of a key and a value. Here i and j point at elements with the same key, so the product $.39 \times .59 = .2301$ would be added to the result.

Efficiency can be further improved by observing that lexical vectors tend to be sparse, i.e., many of the values are 0. When representing vectors as fixed arrays of N values, we can improve performance by only executing multiplication and addition for an index i if neither $\tilde{a}_i$ nor $\tilde{b}_i$ is 0. This saves one multiplication and one addition in the other cases, at the cost of an additional test for each index i.

A more substantial improvement can be achieved by storing vectors in *sparse representations* instead of fixed-size arrays. Figure 4.1 shows such representations of two vectors in a 10-dimensional vector space. Each has 4 non-zero values, which are stored in a list of (key, value) pairs, with an (arbitrary) numerical identifier of the dimension as key and the corresponding (normalized) vector component as value. A key of -1 indicates the end of the list. By storing these lists *sorted by key*, the computation of the cosine becomes very efficient. Figure 4.2 shows an algorithm that traverses both lists and only executes multiplication and addition whenever coinciding keys are found. For each of the lists an index is maintained, starting at the first element. If the keys of the two referenced elements are equal, their corresponding values are multiplied and added to the result. Otherwise the index referencing the element with the lower key is incremented. Through iteration, all pairs of elements with coinciding keys are processed, before finally one of the indices reaches the end of its list.

Obviously, the loop will never be executed more than $m+n$ times, where m and n are the numbers of non-zero elements in the two vectors, since in

```
 1: result $\leftarrow 0$
 2: $i \leftarrow 0$
 3: $j \leftarrow 0$
 4: while $(v[i].key \neq -1)$ and $(w[j].key \neq -1)$ do
 5:     if $v[i].key = w[j].key$ then
 6:         result $\leftarrow$ result + $v[i].value \cdot w[j].value$
 7:         $i \leftarrow i + 1$
 8:         $j \leftarrow j + 1$
 9:     else if $v[i].key < w[j].key$ then
10:         $i \leftarrow i + 1$
11:     else
12:         $j \leftarrow j + 1$
13:     end if
14: end while
15: return result
```

Figure 4.2: Algorithm for computing cosine similarity on sparse vector representations v and w

each iteration at least one index is incremented. For vectors with few non-zero values relative to N, this is significantly more efficient than traversing a fixed-size list of length N. But even if m and n are not much smaller than N, performance does not suffer relative to the naive approach: if $m + n > N$, there have to be at least $k := m+n-N$ "overlapping" dimensions with non-zero values in both vectors. For each of these, the loop advances both indices, which saves k iterations relative to the upper bound of $m + n$ given above. We can thus lower the bound to $m + n - k = N$ in this case. Consequently, the total number of iterations is always bounded by $\min(m + n, N)$. Our approach is therefore never worse than the naive one, and superior to it if $m + n < N$ for at least some vector pairs.

4.2 WordNet-based Model of Lexical Similarity

A fundamentally different approach to measuring lexical similarity resorts to manually created taxonomies that categorize words according to semantic relations, such as synonymy , hypernymy, meronymy, antonymy etc. The most widely used such taxonomy for English is WordNet (Fellbaum, 1998), and in particular its structure of synonymy and hypernymy relations. Words are grouped together in *synsets* if they are synonymous in at least some contexts. The hypernymy relations between synsets then constitute a directed acyclic graph. In this section, we will review a number of WordNet-based similarity measures, leading up to the definition of Jiang-Conrath similarity, which we will employ for the definition of the lexical similarity measure lex.

WordNet relations have been used in different ways to quantify similarity between word meanings. The simplest approach is to determine the distance of two synsets in the hypernymy graph, i.e., the number of edges on the shortest path between them. This has the drawback of depending heavily upon the granularity of the WordNet hierarchy: the fact that two synsets are separated by many others may not reflect their semantic distance but rather a high degree of detail in this part of WordNet. As the degree of granularity inevitably varies over the hierarchy, the simple measure of path length is relatively unreliable. In particular, related concepts deep in the hierarchy should get higher similarity scores than general concepts near the top. This is reflected in scaled measures such as the one proposed by Wu and Palmer (1994), which sets the path length measure d in relation to the depth of the least common ancestor $s_{1,2}$ of two synsets s_1 and s_2:

$$\mathrm{sim}_{WP}(s_1, s_2) = \frac{2 \cdot \mathrm{depth}(s_{1,2})}{d(s_1, s_{1,2}) + d(s_2, s_{1,2}) + 2 \cdot \mathrm{depth}(s_{1,2})} \tag{4.6}$$

If we assume that the hierarchy is a tree, the shortest path between two nodes has to pass through their least common ancestor, so that $d(s_1, s_{1,2}) + d(s_2, s_{1,2}) = d(s_1, s_2)$. For the reciprocal of sim_{WP}, representing distance instead of similarity, we therefore have:

$$\frac{1}{\mathrm{sim}_{WP}(s_1, s_2)} = \frac{d(s_1, s_2) + 2 \cdot \mathrm{depth}(s_{1,2})}{2 \cdot \mathrm{depth}(s_{1,2})} = 1 + \frac{d(s_1, s_2)}{2 \cdot \mathrm{depth}(s_{1,2})} \tag{4.7}$$

This shows that the simple path length distance d is scaled by the depth of the least common ancestor, so that distances at the bottom of the hierarchy become smaller.

Information content. A more empirical approach of utilizing the WordNet hypernymy hierarchy was proposed by Resnik (1995). He quantifies the similarity of two concepts s_1 and s_2 by way of their shared information. This is expressed in their most specific common hypernym, which is their lowest common ancestor $s_{1,2}$ in the hierarchy. Following Shannon's self-information (Shannon, 1948), its *information content* is quantified by

$$\mathrm{IC}(s_{1,2}) = -\log p(s_{1,2}) \tag{4.8}$$

where $p(s)$ is the probability of the occurence of the concept s or any of its hyponyms. In accordance with its information theoretical interpretation as "surprisal" at the occurence of s, the information content $\mathrm{IC}(s)$ therefore quantifies the specificity of s, growing unboundedly for $p(s) \to 0$ and falling to 0 for $p(s) = 1$.

A maximum likelihood estimate of the probability $p(s)$ can be derived from a suitable corpus by taking the ratio between the number of occurences of the concept s or any of its hyponyms and the overall number of concepts:

$$p(s) = \frac{\sum_{s' \in \text{hypo}(s)} f(s')}{\sum_{s'} f(s')} \tag{4.9}$$

Here, $f(s)$ is the frequency of the sense represented by the synset s, and hypo(s) is the set containing s itself and its direct and indirect hyponyms. The numerator sums the frequencies of all those synsets, while the denominator is the constant sum over all WordNet synsets. If there is a root of the hierarchy (such as the universal hypernym *entity*), this synset will thus always have an information content of $-\log 1 = 0$. Similarly, general terms will tend to have low information content values, as for them the numerator gets large contributions from hyponyms, while specific and infrequent terms will have small numerators and consequently high values of IC(s).

Since sufficiently large corpora with WordNet sense annotations are not available, $f(s)$ usually has to be estimated from the frequencies of the words $w \in s$ making up the synset s. The easiest method to derive sense frequencies from word frequencies is to equally distribute the frequency value of a word across the synsets in which it occurs. The assumption that the senses of a word show a uniform distribution is obviously not realistic. Sense distributions in a corpus tend to be strongly skewed towards a few frequent senses. Modeling of a non-uniform distribution, however, would have to rely on statistics from sense annotated corpora, which may not match the corpus from which word counts are extracted. On the other hand, it may be argued that over-estimation and under-estimation of sense frequencies cancel each other out to some degree over the different words in a synset. We thus estimate

$$f(s) = \sum_{w \in s} \frac{f(w) + 1}{|\text{synsets}(w)|} \tag{4.10}$$

where synsets(w) is the set of synsets containing w. Here, we also employed add-1 smoothing (Lidstone, 1920), adding 1 to each word frequency. This makes sure that the probability of words which occur in WordNet but not in the corpus is not estimated as 0.

Jiang-Conrath Measure. The definition given by Resnik was developed further by Jiang and Conrath (1997), who measure similarity of two concepts not by the amount of shared information, but rather by the specific information *separating* them, i.e., the information added by each of them over that of their most specific common hypernym. This can be expressed as the differences $\text{IC}(s_1) - \text{IC}(s_{1,2})$ and $\text{IC}(s_2) - \text{IC}(s_{1,2})$. The sum of these two terms therefore measures the distance between s_1 and s_2, and a similarity measure can thus be given by their reciprocal:

$$\text{JC}(s_1, s_2) = \frac{1}{\text{IC}(s_1) + \text{IC}(s_2) - 2\,\text{IC}(s_{1,2})} \tag{4.11}$$

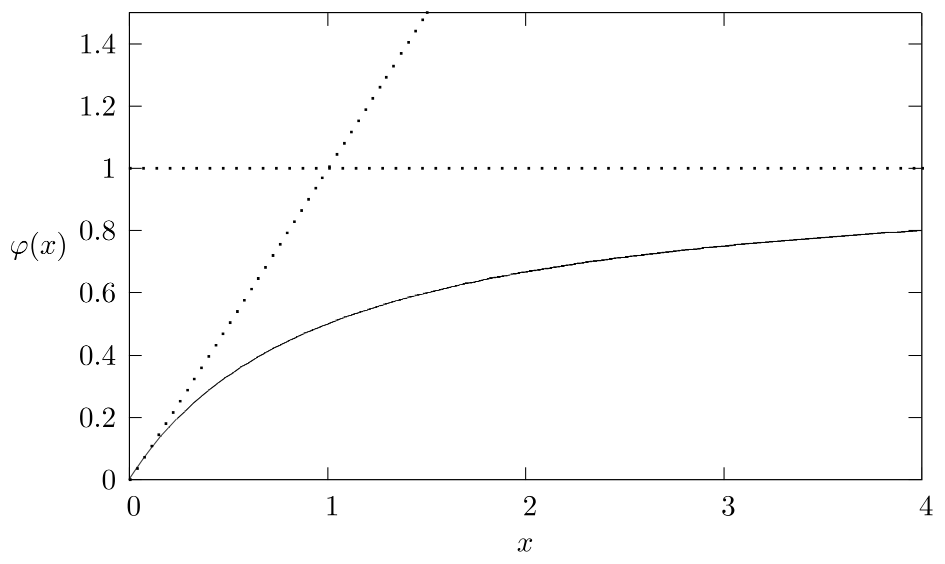


Figure 4.3: Transformation φ of the unbounded range of the Jiang-Conrath similarity measure into the bounded interval $[0, 1)$. The dotted lines show the tangents in $x = 0$ and $x = \infty$.

Budanitsky and Hirst (2006), comparing five WordNet-based similarity measures, found that the Jiang-Conrath similarity measure performed best in several evaluation settings.

For very specific concepts s_1 and s_2, which are in very different branches of the WordNet hierarchy – reflected by a very general lowest common ancestor $s_{1,2}$ – this value may approach 0. On the other hand, there is no upper bound for it. $\mathrm{IC}(s_1)$ and $\mathrm{IC}(s_2)$ may be arbitrarily close to $\mathrm{IC}(s_{1,2})$, which means that $\mathrm{JC}(s_1, s_2)$ may get arbitrarily large. For $s_1 = s_2$ we may define it by $\mathrm{JC}(s, s) = \infty$. Obviously, in the context of our graph alignment approach it is not appropriate for the measure lex to be unbounded. We will frequently compare identical or very similar words of two sentences, and thus a very large similarity contribution would unduly dominate the overall score, practically excluding similarity information contributed by other node pairs. We therefore formulate a *bounded version* of Jiang-Conrath similarity. Our goal is to apply a monotonous transformation of the unbounded interval $[0, \infty)$ into the bounded interval $[0, 1)$. One of the simplest transformations realizing this is:

$$\varphi : x \mapsto \frac{x}{1 + x} \tag{4.12}$$

Its graph for $0 \leq x \leq 4$ is shown in Figure 4.3. Small values of x are hardly changed by φ, as $x - \varphi(x) = \frac{x^2}{1+x}$ which is very small for small x. For $x \to \infty$, however, $\varphi(x)$ approaches its upper bound of 1. We define bounded

Jiang-Conrath similarity by $\mathrm{JC}_b := \varphi \circ \mathrm{JC}$, so that:

$$\begin{aligned}\mathrm{JC}_b(s_1, s_2) = \varphi(\mathrm{JC}(s_1, s_2)) &= \frac{\mathrm{JC}(s_1, s_2)}{1 + \mathrm{JC}(s_1, s_2)} \\ &= \frac{1}{1 + \mathrm{IC}(s_1) + \mathrm{IC}(s_2) - 2\,\mathrm{IC}(s_{1,2})}\end{aligned} \tag{4.13}$$

Dealing with Ambiguity. When comparing words in labeled and unlabeled sentences to align their dependency graphs, the meaning of each word may be ambiguous between several WordNet synsets. So far, however, we have only defined similarity on the level of synsets. We could employ general word sense disambiguation techniques to determine the right synset for each word. However, most successful approaches are supervised (Navigli, 2009), which entails that we would need training data annotated with WordNet senses. Dependence on another type of manual annotation in addition to FrameNet seed instances, however, contradicts the basic goals of our approach, which aims at minimizing annotation effort.

We therefore follow a different approach and exploit the fact that we have to make disambiguation choices not for single words, but for *pairs of words.* Our key assumption is that, among all the senses of two words, the respective senses closest to each other in the WordNet hierarchy are the most probable ones. There are different ways of defining these “closest senses”. The most obvious way would be to maximize bounded Jiang-Conrath similarity over all sense pairs. For two words w_1 and w_2 with respective senses $s_{1,i}$ and $s_{2,j}$, we would thus choose the two senses s_1 and s_2 that maximize $\mathrm{JC}_b(s_{1,i}, s_{2,j})$.

While this is a valid approach, we choose a slightly different definition, which more directly reflects closeness of two synsets in WordNet and has the additional advantage of allowing very efficient maximization. Instead of choosing the senses to maximize JC_b, we pick the pair of senses s_1 and s_2 maximizing $\mathrm{IC}(s_{1,2})$, i.e., those senses of w_1 and w_2 with the most specific lowest common ancestor. Intuitively, we determine the most specific subset of WordNet containing one sense of w_1 and one of w_2. Formally, given w_1 and w_2 we choose

$$(s_1, s_2) := \operatorname*{arg\,max}_{(s_{1,i}, s_{2,j})} \left[\mathrm{IC}(\mathrm{lca}(s_{1,i}, s_{2,j}))\right] \tag{4.14}$$

where lca stands for the lowest common ancestor of two synsets. If the maximum is not unique, we choose the most general synsets s_1 and s_2, i.e., those with the lowest information content values. With these synsets we can now give our definition of lex based on WordNet information in analogy to the earlier definition (4.2):

$$\mathrm{lex}(w_1, w_2) := \begin{cases} \mathrm{JC}_b(s_1, s_2) & \text{if } p_1 = p_2 \in \{N, V\} \\ \delta_{w_1, w_2} & \text{if } p_1 = p_2 \in \{I, C\} \\ 0 & \text{else} \end{cases} \tag{4.15}$$

Here, p_1 and p_2 are again the POS tags of the lemmatized words w_1 and w_2. While similarity of noun and verb pairs is measured by JC_b, there is no corresponding hypernymyhypernym hierarchy for adjectives, which forces us to leave them out. This is not a big problem as adjectives do not usually occur as role filler heads, and so their number in alignment domains and alignment ranges is comparatively low.

Efficient Computation. We now describe a data structure and an algorithm to efficiently determine the information content of the maximizing synsets s_1 and s_2 of (4.14) and their lowest common ancestor $s_{1,2}$ for a given pair of words w_1 and w_2. This then directly allows the computation of JC_b by (4.13). Surprisingly, we can use a data structure very similar to that employed for sparse vector representations in Section 4.1.

We precompute a database with an entry for each word w. This entry is a list of triples $(h, s, \mathrm{IC}(h))$, where h and s are (unique identifiers of) synsets and $\mathrm{IC}(h)$ is the information content of h. To build this list, we collect all direct and indirect hypernyms of all synsets s_i containing w and store them together with their information content values and the respective sense of w which gave rise to them:

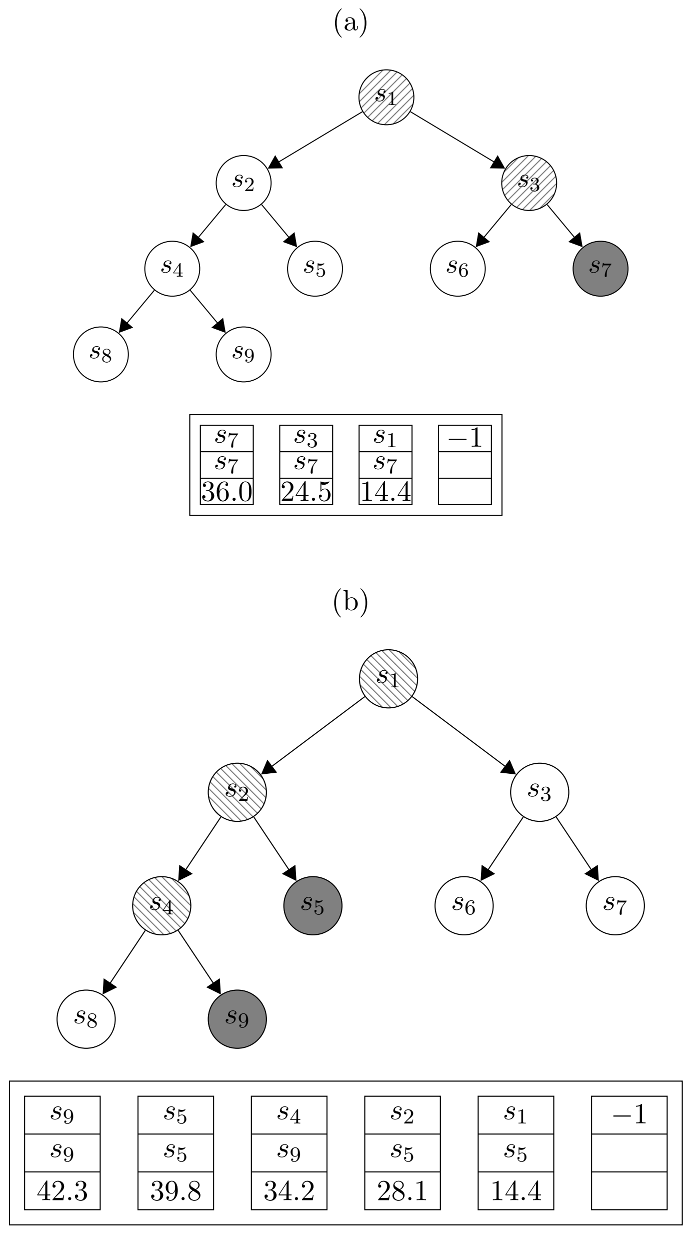
$$H(w) := \bigcup_i \{(h, s_i, \mathrm{IC}(h)) \mid h \in \mathrm{hyper}(s_i)\} \tag{4.16}$$

Here, $\mathrm{hyper}(s)$ denotes the set of all direct and indirect hypernyms of a synset s, including s itself. Now the database entry of w is the list of the elements of $H(w)$ sorted by descending information content values.[2] However, among all triples $(h, s, IC(h))$ with the same h, only the one with the most general synset s (least information content) is retained.

Two schematic examples of database entries are shown in Figure 4.4. The upper half shows an easy case where a word is unambiguous and only occurs in synset s_7, drawn as a gray node in the hierarchy. Its hypernyms s_3 and s_1 are drawn with a hatched pattern. The corresponding list of triples, shown below the graph, therefore consists of entries for s_7, s_3, and s_1, ordered by descending information content. While the concrete numbers are arbitrary, their order is determined by the fact that according to definition (4.9) a synset can never have lower information content than its hypernym. The end of the list is marked by a value of -1. As the represented word has only one sense, the second triple component is s_7 in all three triples.

Figure 4.4(b) shows a more complicated example. Here the given word is ambiguous between the two senses s_5 and s_9, drawn in gray. The resulting list of triples contains triples for those two synsets and their three hypernyms

[2]To ensure consistency in the algorithm described below, triples with identical information content values are sorted by their hypernym synsets according to an arbitrary fixed order on all WordNet synsets.



(a)

s_7	s_3	s_1	-1
s_7	s_7	s_7	
36.0	24.5	14.4	

(b)

s_9	s_5	s_4	s_2	s_1	-1
s_9	s_5	s_9	s_5	s_5	
42.3	39.8	34.2	28.1	14.4	

Figure 4.4: Two examples of the precomputed data structures used in the Jiang-Conrath similarity computation. Sorted lists of triples are built from structural WordNet information and information content values that have been derived from corpus statistics.

s_4, s_2, and s_1, ordered by descending information content. Here, the concrete numbers and the position of s_5 relative to s_9 and s_4 are arbitrarily chosen for the example. The second triple components are set to either s_5 or s_9, depending on which of the two gave rise to this triple. In the cases of s_2 and s_1, which are hypernyms of both s_5 and s_9, the synset with the lower information content was chosen, which here is s_5.

When computing $\text{lex}(w_1, w_2)$, we traverse the lists of w_1 and w_2 using the algorithm shown in Figure 4.5. It finds the first element $(h, s_{1,i}, \text{IC}(h))$ in the first list that has a corresponding element $(h, s_{2,j}, \text{IC}(h))$ in the second list representing the same synset h. This means that h is a common ancestor of $s_{1,i}$ and $s_{2,j}$. In our example, taking Figure 4.4(a) to represent an unambiguous word w_1 and Figure 4.4(b) to represent a word w_2 with two senses, this common ancestor would be the synset s_1. Because of the ordering of the lists, it is also a lowest common ancestor of the two synsets: if h' is another common ancestor of $s_{1,i}$ and $s_{2,j}$, then $\text{IC}(h') \leq \text{IC}(h)$, which means that h' cannot be a hyponym of h. Moreover, among the lowest common ancestors of all pairs of senses of w_1 and w_2, this particular lowest ancestor of $s_{1,i}$ and $s_{2,j}$ is also maximal with regard to information content, which again follows from the ordering of the lists: if the lowest common ancestor of another sense pair had higher information content, it would already have been found in the list traversal. Finally, if there is more than one pair of senses with this lowest common ancestor, our construction ensures that $s_{1,i}$ and $s_{2,j}$ are minimal with regard to their information content, which is in accord with our definition above. We have therefore simultaneously solved the maximization problem of Equation (4.14) and found the information content $\text{IC}(s_{1,2})$ of the lowest common ancestor $s_{1,2} = h$. To compute $\text{JC}_b(s_1, s_2)$ according to (4.13), we only have to look up the values $\text{IC}(s_1)$ and $\text{IC}(s_2)$, which is done by traversing the two lists again individually.

The efficiency of this algorithm is due to the fact that a synset usually has a limited number of hypernyms in the WordNet hierarchy. In a tree there is a unique path from the root to any node and therefore the number of hypernyms is equal to the depth of the synset in the hierarchy. In a perfectly balanced tree with constant numbers of children per node, the depth is bounded by the logarithm of the number of nodes. While WordNet is certainly not a perfectly balanced tree, the number of hypernyms of a given synset is still very small compared to the overall number of synsets: in the noun hypernymy hierarchy, a synset on average has only about 9 hypernyms, including itself, whereas there is a total of 82,115 synsets. The largest number of hypernyms for any synset is 29 (for the synset {scat singing, scat}, a kind of improvisation in vocal jazz).

The number of elements in the precomputed list for word w is bounded by $|\text{synsets}(w)| \cdot |\text{hyp}(w)|$. As the maximum polysemy in WordNet is 33 (for the word *head*), we can therefore give $33 \cdot 29 = 957$ as an upper bound for the list length. As each iteration of the loop increments one of the two list

```
$i \leftarrow 0$
$j \leftarrow 0$
while $(v[i].key \neq -1)$ and $(w[i].key \neq -1)$ and $(v[i].h \neq w[j].h)$ do
    if $v[i].IC < w[j].IC$ then
        $j \leftarrow j + 1$
    else if $v[i].IC > w[j].IC$ then
        $i \leftarrow i + 1$
    else if $v[i].h < w[j].h$ then                # arbitrary order on synsets
        $j \leftarrow j + 1$
    else
        $i \leftarrow i + 1$
    end if
end while
if $(v[i].key = -1)$ or $(w[i].key = -1)$ then
    return 0                                      # no common ancestor
else
    $s_1 \leftarrow v[i].s$
    $s_2 \leftarrow w[j].s$
    $IC_{1,2} \leftarrow v[i].IC$                 # same as $w[j].IC$
    $i \leftarrow 0$
    while $(v[i].h \neq s_1)$ do
        $i \leftarrow i + 1$
    end while
    $IC_1 \leftarrow v[i].IC$
    $j \leftarrow 0$
    while $(w[j].h \neq s_2)$ do
        $j \leftarrow j + 1$
    end while
    $IC_2 \leftarrow w[j].IC$
    return $\frac{1}{1+IC_1+IC_2-2 \cdot IC_{1,2}}$
end if
```

Figure 4.5: Algorithm to efficiently compute bounded Jiang-Conrath similarity from precomputed lists v and w of hypernyms

indices, it is therefore executed at most 1914 times. Typically, the number of executions will be much smaller, especially for relatively similar words, where the first lowest common ancestor occurs early in both lists. The individual traversal of both lists to look up the information content of the two maximizing concepts according to (4.14) is similarly bounded by the maximum list length and will be even faster as the synsets themselves occur early in the respective lists.

4.3 Syntactic Similarity

Compared with research on lexical similarity, study of syntactic similarity is a much less clearly delineated field. This may be due to the fact that, intuitively, there is a small set of discrete syntactic classes, whereas the meanings of words more obviously fill a complex "semantic space", which researchers have then sought to model. Traditional grammar theories have included a small number of syntactic classes, such as subjects or direct and indirect objects. The same is true for many variants of phrase structure grammars, going back to Chomsky (1957), or dependency grammars, following Tesnière (1959), which are often based on small inventories of phrase types or grammatical relation labels. In some grammar theories, such as head-driven phrase structure grammar (Pollard and Sag, 1994) or combinatory categorial grammar (Steedman, 1987), syntactic categories have proliferated. To our knowledge, however, no systematic study of their similarity, independent of their role in syntactic composition processes, has been undertaken.

It is not obvious whether graded similarity values for grammatical relations are warranted at all, or how these should be computed. A simple definition of syntactic similarity can be formulated by making a binary distinction: identical relation types are "similar" and different ones are "dissimilar", with no intermediate values. Formally, this can be expressed as:

$$\mathrm{syn}(r_1, r_2) = \delta_{r_1,r_2} = \begin{cases} 1 & \text{if } r_1 = r_2 \\ 0 & \text{else} \end{cases} \tag{4.17}$$

Although individual edge pairs in graph alignments under this definition are subject to binary classification, dependency paths containing more than one edge can still show graded similarity. For example, the path ($\textsc{conj}^{-1}$, $\textsc{subj}$) in Figure 3.4 compares perfectly to an identical path, contributing a syntactic similarity of 2, but comparison with a simple path ($\textsc{subj}$) will still result in a similarity contribution of 1. In this regard, this measure is similar to methods based on tree kernels (Moschitti et al., 2008), which also measure the degree of similarity by counting common substructures.

We consider one alternative to the simple binary definition of (4.17). The RASP parser employed in our experiments for English produces grammatical relations for which a hierarchy has first been proposed in Carroll et al. (1998).

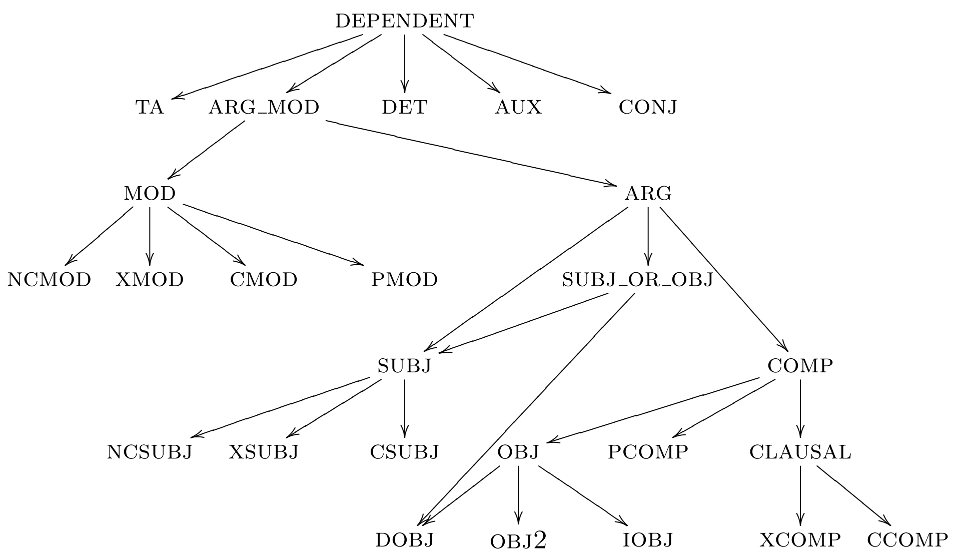


Figure 4.6: Hierarchy of grammatical relations defined for RASP

We reproduce the revised hierarchy from Briscoe et al. (2006) in Figure 4.6. It is designed to give the parser fall-back options in case of unresolvable ambiguity and to facilitate comparison between different formalisms. We follow the intuition that closeness in this hierarchy represents similarity of grammatical relations. As a measure of closeness, we again employ Jiang-Conrath similarity.

Table 4.7 shows the different grammatical relations (GRs) occuring in our expansion corpus, i.e., the BNC parsed with RASP. A total number of 86,286,092 GR tokens was counted. For each GR, its relative frequency, the cumulative frequency of itself and all its subordinate GRs in the hierarchy, and the information content (IC), computed in analogy to Equation (4.9), is shown. From the information content values, the bounded Jiang-Conrath similarity according to Equation (4.13) – with GRs instead of synsets – is computed, which allows us to define:

$$\mathrm{syn}(r_1, r_2) = \mathrm{JC}_b(r_1, r_2) \tag{4.18}$$

The similarity values between the most frequent GRs are shown in Table 4.8. Intuitively, those results do not look very promising in the context of our expansion algorithm. For example, non-clausal subjects (NCSUBJ) and direct objects (DOBJ) are given a relatively high similarity score, while in most cases their distinction will be essential in role labeling, and potentially much more so than the distinction between modifiers and arguments. Confirming this intuition, preliminary experiments with the definition (4.18) did not show any improvements over the simple binary measure (4.17). We therefore do not consider it any further. An alternative way of defining a

GR	**Rel. Freq.**	**Cumulative**	**IC**
NCMOD	24.67%	24.67%	1.400
DOBJ	16.94%	16.94%	1.776
DET	12.87%	12.87%	2.050
NCSUBJ	10.81%	10.81%	2.225
CONJ	7.70%	7.70%	2.563
IOBJ	6.28%	6.28%	2.768
XCOMP	4.77%	4.77%	3.043
AUX	4.76%	4.76%	3.044
CCOMP	4.12%	4.12%	3.189
TA	2.40%	2.40%	3.731
XMOD	1.39%	1.39%	4.274
CMOD	1.38%	1.38%	4.283
ARG_MOD	0.47%	72.27%	0.325
OBJ	0.46%	23.97%	1.428
PCOMP	0.38%	0.38%	5.579
OBJ2	0.30%	0.30%	5.815
PMOD	0.13%	0.13%	6.666
CSUBJ	0.08%	0.08%	7.130
XSUBJ	0.06%	0.06%	7.353
COMP	0.04%	33.28%	1.100
ARG	0.00%	44.23%	0.816
SUBJ	0.00%	10.95%	2.212
CLAUSAL	0.00%	8.89%	2.420
DEPENDENT	0.00%	100.00%	0.000
SUBJ_OR_OBJ	0.00%	34.92%	1.052
MOD	0.00%	27.57%	1.289

Table 4.7: Corpus statistics and computed information content (IC) values for the grammatical relations (GRs) produced by RASP

	NCMOD	DOBJ	DET	NCSUBJ	CONJ	IOBJ	XCOMP
NCMOD	1.0000	0.2836	0.2247	0.2516	0.2015	0.2213	0.2086
DOBJ		1.0000	0.2072	0.3452	0.1873	0.3721	0.2764
DET			1.0000	0.1896	0.1781	0.1719	0.1641
NCSUBJ				1.0000	0.1728	0.2293	0.2157
CONJ					1.0000	0.1579	0.1514
IOBJ						1.0000	0.2169
XCOMP							1.0000

Table 4.8: Bounded Jiang-Conrath similarity between the most frequent grammatical relations

syntactic similarity measure would be to optimize its values on pairs of GRs in a tuning procedure. However, a symmetric similarity measure between N grammatical relations with $\text{syn}(r, r) = 1$ has $\frac{N(N-1)}{2}$ free parameters. For the 26 GRs produced by RASP, this would be 325 parameters, resulting in a huge search space, which would have to be restricted in some way. Addressing this problem is beyond the scope of this thesis.

4.4 The Weight Parameter

An important parameter in the formulation of our similarity score (see Equation (3.6) on page 44) is the relative weight α of syntactic similarity compared with lexical similarity. We consider its influence on the final similarity score, which includes the normalization factor (see Equation (3.9) on page 46). Abbreviating the sums of lexical and syntactic scores for simplicity, we have:

$$\text{sim}(M, N) = \frac{\sum \text{lex} + \alpha \sum \text{syn}}{\sqrt{|M| \cdot |N|} + \alpha \sqrt{|E(M)| \cdot |E(N)|}} \tag{4.19}$$

Obviously, values of α below 0 do not make sense in our model as our similarity measure would then penalize genuinely similar syntactic structure. In the special case of $\alpha = 0$ we have

$$\text{sim}(M, N) = \frac{\sum \text{lex}}{\sqrt{|M| \cdot |N|}} \tag{4.20}$$

which means that syntactic information is ignored. Progressively larger values $\alpha > 0$ reflect growing influence of syntactic agreement on the overall similarity score, but an analogous case in which only syntactic information is considered while lexical similarity is ignored only occurs in the limit of $\alpha \to \infty$. In this case we have

$$\begin{aligned} \text{sim}(M, N) &= \frac{\alpha^{-1} \sum \text{lex} + \sum \text{syn}}{\alpha^{-1} \sqrt{|M| \cdot |N|} + \sqrt{|E(M)| \cdot |E(N)|}} \\ &\to \frac{\sum \text{syn}}{\sqrt{|E(M)| \cdot |E(N)|}} \end{aligned} \tag{4.21}$$

The parameter α may thus range between 0 and ∞, expressing a scale of weights between the extremes of only considering lexical or only considering syntactic similarity. To make this range of values symmetric we will consider the *logarithmic weight parameter* $\log \alpha$, which correspondingly ranges from $\log \alpha = -\infty$ (i.e., $\alpha = 0$, only lexical information is taken into accont), to $\log \alpha = \infty$ (i.e., the limit of $\alpha \to \infty$, only syntactic information is taken into account).

We expect that for an appropriate definition of our similarity measure both kinds of information should be taken into account, and $\log \alpha$ should therefore be neither too small nor too large. Its not clear, however, how to derive a specific numeric value theoretically. Both measures lex and syn may in principle take any value. Even when they are normalized to the interval $[0, 1]$, which is the case for the definitions given in the present chapter, the distributions of their values within these intervals may be very different. The absolute values of the cosine similarity measure, for example, depend very much on the sparseness of vectors, which in turn depends on the dimensionality of the employed vector space. In Section 7.3 we will therefore determine an optimal value of $\log \alpha$ empirically.

4.5 Summary

In this chapter, we have given definitions of our measures of lexical and syntactic similarity. For lexical similarity, we have discussed two fundamentally different approaches: vector space models based on co-occurence statistics and similarity measures based on a taxonomy like WordNet. In both cases we have derived the definition of a suitable measure lex and shown how to compute it efficiently in the context of our task, i.e., given the prevalence of sparse vectors for cosine similarity and relatively shallow hierarchies for WordNet-based similarity.

Addressing syntactic similarity, we have first proposed a simple binary measure. We have then given an alternative definition by applying the taxonomy-based approach to a hierarchy of syntactic relations and deriving similarity scores for them. However, this did not yield an appropriate similarity measure. Future approaches to measure graded syntactic similarity might have to be defined on fine-grained syntactic categories, provided by a suitable parser.

In the following chapter, we will address the problem of finding graph alignments which maximize the combined lexical and syntactic similarity score.

Chapter 5

Solving the Optimization

In this chapter, we describe a solution algorithm for the optimization problem formulated in Section 3.4. We first discuss the scope and complexity of the problem, showing that naive enumeration of all possible solutions is infeasible. We then formulate the problem as an integer linear program (ILP) and discuss its properties, showing that a standard approach of reduction to a problem solvable in polynomial time is not possible, as the problem is NP-hard. We therefore propose an adapted branch-and-bound algorithm, which renders the problem efficiently solvable in the vast majority of cases.

5.1 Scope of the Problem

In Section 3.4 we defined the similarity of two predicate-argument structures as the maximum score of any alignment between them. For our expansion algorithm, we need to determine this optimal alignment and its score according to equation (3.6) for a large number of labeled and unlabeled dependency graphs. The number $c(m,n)$ of all possible alignments between two graphs with m and n nodes respectively was derived as follows (repeated here from equation (3.5)):

$$c(m,n) := \sum_{k=0}^{\min(m,n)} \frac{m!n!}{(m-k)!(n-k)!k!} \tag{5.1}$$

Assuming $m \leq n$, we can give a lower bound for this expression by picking only the term for $k = \lceil \frac{m}{2} \rceil$ from this sum and thus estimating:

$$c(m,n) \geq \frac{m!}{(m - \lceil \frac{m}{2} \rceil)!} \cdot \binom{n}{\lceil \frac{m}{2} \rceil} \geq \frac{m!}{(\lfloor \frac{m}{2} \rfloor)!} \geq 2^m \tag{5.2}$$

The last inequality can easily be proved inductively, noting that $(\lfloor \frac{m+1}{2} \rfloor)! \leq \frac{m+1}{2}(\lfloor \frac{m}{2} \rfloor)!$ and therefore:

$$\frac{(m+1)!}{(\lfloor \frac{m+1}{2} \rfloor)!} \geq \frac{(m+1)m!}{\frac{m+1}{2} \cdot (\lfloor \frac{m}{2} \rfloor)!} = 2 \cdot \frac{m!}{(\lfloor \frac{m}{2} \rfloor)!} \tag{5.3}$$

For reasons of symmetry, we similarly have $c(m,n) \geq 2^n$ in the case of $n \leq m$ and therefore

$$c(m,n) \geq 2^{\min(m,n)} \tag{5.4}$$

This shows that $c(m,n)$ is at least exponential in $\min(m,n)$. The given lower bound, however, is actually very weak. Figure 5.1 shows that even for small m and n the number of possible alignments $c(m,n)$ already becomes very large. To assess how this relates to our problem, we computed a histogram over the 60,666 instances of verbal predicates in the FrameNet corpus, counting the frequency of alignment domains with given numbers of nodes. The result is shown in Figure 5.2. The average number of nodes in an alignment domain is 5.7, but 18% of the alignment domains have more than 7 nodes and 3% even more than 10 nodes. Due to the treatment of complex paths described in Section 3.3, alignment ranges depend on the alignment domains of labeled partners considered in the concrete experimental setting. We therefore do not give statistics of their sizes. Given the symmetry of the definitions, however, they can be expected to be of similar sizes as the alignment domains. This means that in a significant part of similarity comparisons, subgraphs of 10 or more nodes are compared and an optimal solution among $c(10,10) = 234,662,231$ alignments or more has to be found. It is evident that such large numbers of evaluations of the scoring function are infeasible, and maximization by simple enumeration therefore unrealistic.

We are thus facing a large discrete optimization problem. Such problems occur in many areas of NLP, such as syntactic parsing or translation. Often it turns out that their solution can only be determined approximately, and heuristics must be applied to prune unpromising parts of the search space, as is the case, e.g., in beam search. Other approaches relax a discrete problem into a continuous one in order to apply optimization strategies for continuous functions. An example for this can be found in Klau (2009), whom we will also follow in our formulation of the ILP for the graph alignment problem. The obvious disadvantage of any approximation method is that it introduces an additional source of errors. Approximation errors have to be distinguished from modeling errors in that the former constitute failures to find the optimal solution within the model, while the latter arise when the model insufficiently represents the modeled phenomenon and therefore fails to characterize the correct solution of the problem. In the following sections we will describe an *exact solution* of the maximization problem (3.7), which exludes the possibility of approximation errors. Any remaining errors are therefore due to our model, including the general approach of similarity-based projection and the concrete measures of similarity, which facilitates error analysis.

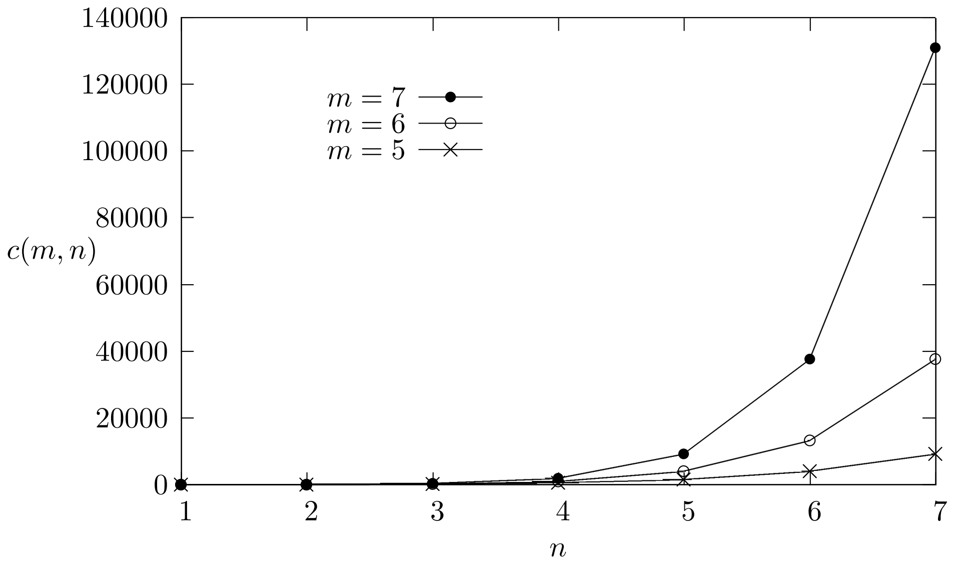


Figure 5.1: Values of the number $c(m,n)$ of all possible alignments for different values of m and n. The three graphs show the functions $n \mapsto c(m,n)$ for $m = 5$, $m = 6$ and $m = 7$.

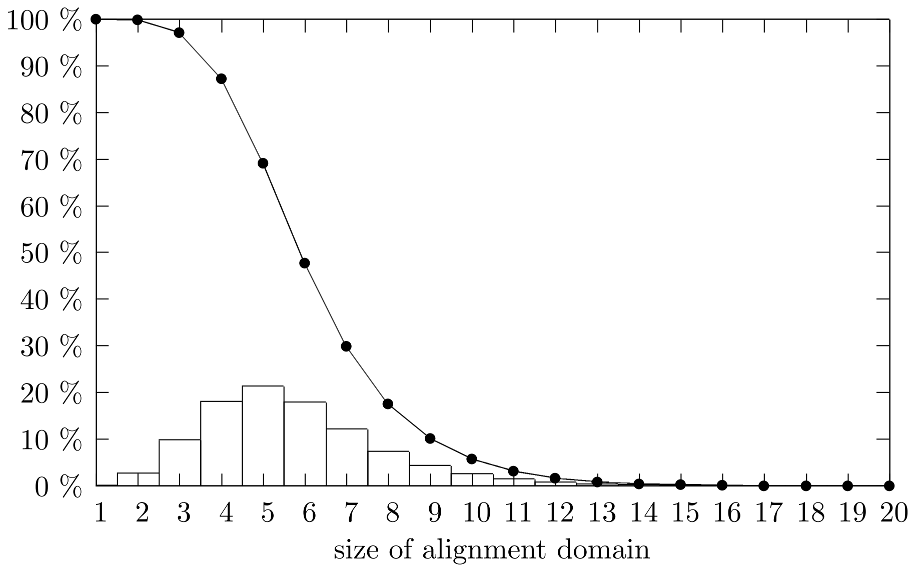


Figure 5.2: Histogram of the size of alignment domains over labeled seeds from the FrameNet corpus. The bars show relative frequencies of alignment domains of the given size, while the curve shows the relative numbers of alignment domains of *at least* the given size.

5.2 Formulation as Integer Linear Program

To formulate our optimization problem as an integer linear program (ILP) we characterize alignments by sets of *indicator variables.* Given an alignment σ from an alignment domain $M = \{n_1, \dots, n_m\}$ to an alignment range $N = \{n'_1, \dots, n'_n\}$ (where n_1 and n'_1 represent the FEE and the target predicate, respectively, and the order of the other nodes is fixed arbitrarily), we define binary variables x_{ij} as:

$$x_{ij} := \begin{cases} 1 & \text{if } \sigma(n_i) = n'_j \\ 0 & \text{else} \end{cases} \tag{5.5}$$

Each of these variables thus indicates whether the two nodes $n_i \in M$ and $n'_j \in N$ are aligned or not, so that the set of all variables x_{ij} uniquely determines σ. Not any assignment of those variables, however, corresponds to a valid alignment. We have to ensure that no node of either graph is aligned to more than one node in the other graph. This can be expressed by the following constraints:

(1) $\forall_i : \sum_{1 \leq j \leq n} x_{ij} \leq 1$

(2) $\forall_j : \sum_{1 \leq i \leq m} x_{ij} \leq 1$

Constraint (1) ensures that for each i at most one of the binary variables $x_{i1}, \dots, x_{in}$ takes a value of 1, which corresponds to the constraint that each node in M is aligned to at most one node in N. Constraint (2) enforces the same in the opposite direction. Due to the universal quantification over i and j, (1) and (2) together actually represent $m + n$ linear constraints on the variables x_{ij}. We can now express the scoring function

$$\text{score}(\sigma^*) := \sum_{\substack{n \in M \\ \sigma^*(n) \neq \epsilon}} \text{lex}\,(n, \sigma^*(n)) + \alpha \cdot \sum_{\substack{(n_1,n_2) \in E(M) \\ (\sigma^*(n_1),\sigma^*(n_2)) \in E(N)}} \text{syn}\left(r_{n_2}^{n_1}, r_{\sigma^*(n_2)}^{\sigma^*(n_1)}\right) \tag{5.6}$$

(repeated here from equation (3.6)) in terms of the indicator variables x_{ij} instead of the function $\sigma : M \cup \{\epsilon\} \to N$:

$$\text{score}(\mathbf{x}) = \sum_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} \text{lex}\left(n_i, n'_j\right) x_{ij} + \alpha \cdot \sum_{\substack{1 \leq i,k \leq m \\ 1 \leq j,l \leq n}} \text{syn}\left(r_{n_k}^{n_i}, r_{n'_l}^{n'_j}\right) x_{ij} x_{kl} \tag{5.7}$$

To see that these two forms are indeed equivalent, we first consider the sum of lex values over the indices i and j. For each aligned $n_i \in M$ there is exactly one j' with $x_{ij'} = 1$, so that the sum over j reduces to a single term $\text{lex}(n_i, n'_{j'})$, while for an unaligned $n_i \in M$ all x_{ij} are 0 and this sum is therefore empty. The remaining sum over i is then identical with the corresponding sum in (3.6), adding lexical similarity values for all aligned node pairs. The second sum over i, j, k, and l reduces similarly: the product $x_{ij}x_{kl}$ is 1 if and only if $\sigma(n_i) = n'_j$ and $\sigma(n_k) = n'_l$. The sum over all combinations of i, j, k, and l therefore comprises syntactic similarity values for all edge pairs between the relevant node pairs and coincides with the corresponding term in (3.6).

Equation (5.7) expresses the maximization problem in terms of binary variables. However, due to the products of the form $x_{ij}x_{kl}$ the scoring function is not linear in its variables. To formulate an integer linear program, we therefore introduce another set of auxiliary binary variables y_{ijkl}, obeying the following additional constraints:

(3) $\forall_{i,j,k,l} : y_{ijkl} \leq x_{ij}$

(4) $\forall_{i,j,k,l} : y_{ijkl} \leq x_{kl}$

(5) $\forall_{i,j,k,l} : y_{ijkl} \geq x_{ij} + x_{kl} - 1$

The $2m^2n^2$ constraints in (3) and (4) ensure

$$(y_{ijkl} = 1) \Rightarrow (x_{ij} = 1 \wedge x_{kl} = 1)$$

while the m^2n^2 constraints in (5) enforce the reverse implication. For binary variables they are therefore equivalent to stating $y_{ijkl} = x_{ij}x_{kl}$. This means that we have linearized the scoring function, which can now be written as:

$$\text{score}(\mathbf{x}, \mathbf{y}) = \sum_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} \text{lex}\left(n_i, n'_j\right) x_{ij} + \alpha \cdot \sum_{\substack{1 \leq i,k \leq m \\ 1 \leq j,l \leq n}} \text{syn}\left(r^{n_i}_{n_k}, r^{n'_j}_{n'_l}\right) y_{ijkl} \tag{5.8}$$

Finally, as was already mentioned (Section 3.3, page 43), we want to ensure that the FEE of the labeled sentence is aligned to the target predicate of the unlabeled sentence. Since we assume that the FEE and the target predicate are represented by n_1 and n'_1, this is expressed by the single constraint:

(6) $x_{11} = 1$

The problem of finding an optimal alignment has thus been reduced to that of maximizing the objective function (5.8) in the $mn + m^2n^2$ variables x_{ij} and y_{ijkl}, subject to the $m+n+3m^2n^2+1$ constraints listed in (1) to (6). Integer linear programs in general assume arbitrary integer values for the variables. Formally we therefore have to include the additional constraint $x_{ij} \geq 0$ for each x_{ij}. Together with (1) this implies $x_{ij} \leq 1$ and therefore $x_{ij}, y_{ijkl} \in \{0, 1\}$.

5.3 Complexity of the Integer Linear Program

Exact optimization for the general ILP problem is NP-hard, as can be shown by reduction from the boolean satisfiability problem, which is NP-hard (Cook, 1971). There is an important subclass of ILPs, however, which are solvable in polynomial time. For those problems the solution of the linear program (LP), i.e., the problem without the integer constraint, automatically takes integer values. A solution of the LP is therefore also a solution of the ILP. One of the earliest algorithms proposed for the solution of the general LP is the Simplex Algorithm (Dantzig, 1963). Although its worst-case performance is exponential in the number of variables (Klee and Minty, 1972), it is typically very efficient in practice. The problem was finally proved to be solvable in polynomial time by Khachiyan (1980). His Ellipsoid algorithm, however, performs poorly in practice, and the Simplex Algorithm is still widely used today, besides newer algorithms such as interior point methods (Karmarkar, 1984; Mehrotra, 1992).

To our knowledge, there is no general method for deciding whether an ILP belongs to the class of problems for which the integer constraint is satisfied automatically and which are therefore solved by the solution of the corresponding LP. A sufficient condition, however, can be given with the help of the *constraint matrix.* The constraints of an LP or ILP in the variables $z_1, \ldots, z_N$ can be written in a normalized form where the i-th constraint ($i = 1, \ldots, M$) is given by

$$a_{i,1}z_1 + \cdots + a_{i,N}z_N \leq b_i \tag{5.9}$$

The matrix of all coefficients $a_{i,j}$ is then called the constraint matrix. It can be shown that, if an LP has a *totally unimodular* constraint matrix, i.e., the determinant of any of its square submatrices is in $\{-1, 0, 1\}$, and if the b_i are all integral, the solution of the corresponding LP is also integral and therefore also a solution of the ILP. A proof for this can be found, e.g., in Sierksma (2001).

This test allows the reduction of a number of comparatively simple ILPs, such as the linear assignment problem, to efficiently solvable LPs. Unfortunately, our problem cannot be reduced in this way. We show this by proving that the constraint matrix of our ILP always has a square submatrix with a determinant that is not -1, 0, or 1. For a non-trivial problem, we may assume $n \geq 2$. We can number our $N := mn + m^2n^2$ variables x_{ij} and y_{ijkl} by defining $z_1, \ldots, z_N$ through

$$z_{(i-1)n+j} := x_{ij} \tag{5.10}$$

for $1 \leq i \leq m$ and $1 \leq j \leq n$ and

$$z_{mn+(i-1)mn^2+(j-1)mn+(k-1)n+l} := y_{ijkl} \tag{5.11}$$

for $1 \leq i, k \leq m$ and $1 \leq j, l \leq n$. The $M := m + n + 3m^2n^2 + 1$ constraints (1) to (6) can be normalized and similarly numbered, so that we obtain a $M \times N$ constraint matrix. We consider the following three constraints, taken from (1) with $i = 1$, from (3) with $i = j = k = 1$ and $l = 2$, and from (4) with $i = j = k = 1$ and $l = 2$:

$$x_{11} + x_{12} + \cdots + x_{1n} \leq 1 \tag{5.12}$$

$$y_{1112} \leq x_{11} \tag{5.13}$$

$$y_{1112} \leq x_{12} \tag{5.14}$$

Normalized and expressed in terms of the z_i, these are

$$1 \cdot z_1 + 1 \cdot z_2 + \ldots + 1 \cdot z_n \qquad \leq 1 \tag{5.15}$$

$$-1 \cdot z_1 \qquad +1 \cdot z_{mn+2} \leq 0 \tag{5.16}$$

$$- 1 \cdot z_2 \qquad +1 \cdot z_{mn+2} \leq 0 \tag{5.17}$$

Taking the rows corresponding to these constraints and the columns corresponding to z_1, z_2, and z_{mn+2} from the constraint matrix, we thus obtain the following 3×3 submatrix:

$$A = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{pmatrix} \tag{5.18}$$

Its determinant is 2, which shows that the constraint matrix is not totally unimodular. It should be noted, however, that the normalization of (5.15), (5.16), and (5.17) is not unique, as these inequalities could be multiplied by arbitrary non-zero factors a, b, and c, and still be of the form (5.9). The resulting matrix

$$B = \begin{pmatrix} a & a & 0 \\ -b & 0 & b \\ 0 & -c & c \end{pmatrix} \tag{5.19}$$

then has a determinant of $2abc$. Since $a, b, c \neq 0$, at least one of them would have to be non-integral for $2abc$ to be in $\{-1, 0, 1\}$. But this would yield a trivial 1×1 submatrix with non-integral determinant, and thus again contradict total unimodularity.

This discussion shows that our problem does not satisfy the sufficient condition for the existence of a polynomial-time solution as stated above, but does not yet prove anything about its hardness. However, it can be shown that it is indeed NP-hard. For the proof by reduction from the maximum subgraph problem we refer the reader to Klau (2009). This result means that no polynomial-time algorithm for the exact solution of our ILP is available. However, in the next section we describe a solution algorithm which is efficient in all but a negligible number of cases in practice.

5.4 Solution of the Integer Linear Program

One of the most popular algorithms to make the solution of general ILPs practically feasible is the branch-and-bound algorithm (Land and Doig, 1960). Its basic idea is to avoid unnecessary evaluations of the objective function by using a hierarchical search strategy. Before entering a "branch" of the search space, an upper "bound" for the best solution within this branch is estimated. If this shows that the current best solution cannot be improved upon, the branch is skipped. This may lead to significant performance improvements compared to naive enumeration of all possibilities. In this section, we describe a version of the branch-and-bound algorithm adapted to the special structure of our problem. We will highlight the differences to the general algorithm, where the structure of our ILP allows for some early exclusion of impossible branches and tighter bound estimates, leading to further performance improvements.

Figure 5.3 shows pseudocode for our algorithm. For a given alignment domain $M = \{n_1, \dots, n_m\}$ and alignment range $N = \{n'_1, \dots, n'_n\}$ it determines the alignment σ^* maximizing (3.6). Here, we employ a notation using sets of alignment pairs $n_i \mapsto n'_j$ or $n_i \mapsto \epsilon$. A set with m such alignment pairs describes an alignment, while smaller sets stand for *partial alignments*, in which the alignment of the missing nodes $n_i \in M$ is still unspecified. Such nodes must be distinguished from unaligned nodes; nodes with unspecified alignment represent intermediate states in which an alignment decision has not yet been made, while an alignment pair $n_i \mapsto \epsilon$ expresses the decision that n_i be left unaligned. This way of representing alignments is solely for notational convenience and equivalent to the representation by the binary indicator variables x_{ij} and y_{ijkl}.

At the beginning of the search process, we initialize σ^* with the trivial solution which aligns n_1 to n'_1 and leaves all other nodes unaligned. This gives a score of $\text{lex}(n_1, n'_1)$, which is stored in s^*. To find better solutions we start with an initial partial alignment σ_0, which contains only the alignment pair $n_1 \mapsto n'_1$ and leaves the alignments of all other $n \in M$ unspecified. As in the general branch-and-bound algorithm, the space of all alignments is then searched recursively by branching on the alignment decision for each remaining node. A branch is left as soon as a bound on the achievable score indicates that the current best solution cannot be improved upon within this branch.

Given a partial alignment σ_0 (the initial or any subsequent one) defined on some subset of M, we estimate a suitable bound by extending σ_0 to a complete function σ on all nodes in M: each of the remaining nodes is aligned to a partner in N in such a way that lex is maximized for the pair. If no positive value can be achieved for lex, the node is defined as unaligned. We then define the bound s as the score of σ_0 together with the lexical scores of the newly created alignments and a hypothetical syntactic score

```
1: $\sigma^* \leftarrow \{n_1 \mapsto n'_1, n_2 \mapsto \epsilon, \ldots, n_m \mapsto \epsilon\}$
2: $s^* \leftarrow \mathrm{lex}(n_1, n'_1)$
3: Initialize stack with single item $(\{n_1 \mapsto n'_1\}, \mathrm{lex}(n_1, n'_1))$
4: while stack not empty do
5:     pop $(\sigma_0 = \{n_1 \mapsto n'_1, \ldots, n_k \mapsto \sigma_0(n_k)\}, s_0)$ from stack
6:     $\sigma \leftarrow \sigma_0$
7:     $s \leftarrow s_0$
8:     for $i$ from $k+1$ to $m$ do
9:         $n' \leftarrow \arg\max_{n' \in N} \mathrm{lex}(n_i, n')$
10:        if $\mathrm{lex}(n_i, n') > 0$ then
11:            $\sigma \leftarrow \sigma \cup \{n_i \mapsto n'\}$
12:            $s \leftarrow s + \mathrm{lex}(n_i, n') + \alpha \cdot \mathrm{syn}^* \cdot |\mathrm{neighbours}(n_i)|$
13:        else
14:            $\sigma \leftarrow \sigma \cup \{n_i \mapsto \epsilon\}$
15:        end if
16:    end for
17:    if $s > s^*$ then
18:        if $\mathrm{valid}(\sigma, k)$ and $\mathrm{syn\_max}(\sigma, k)$ then
19:            $\sigma^* \leftarrow \sigma$
20:            $s^* \leftarrow s$
21:        else
22:            for $n' \in (N - \mathrm{range}(\sigma_0)) \cup \{\epsilon\}$ do
23:                $\sigma_1 \leftarrow \sigma_0 \cup \{n_{k+1} \mapsto n'\}$
24:                $s_1 \leftarrow s_0 + \mathrm{lex}(n_{k+1}, n')$
25:                for $i$ from 1 to $k$ do
26:                    $s_1 \leftarrow s_1 + \alpha \cdot \mathrm{syn}\left(r_{n_i}^{n_{k+1}}, r_{\sigma_0(n_i)}^{n'}\right)$
27:                    $s_1 \leftarrow s_1 + \alpha \cdot \mathrm{syn}\left(r_{n_{k+1}}^{n_i}, r_{n'}^{\sigma_0(n_i)}\right)$
28:                end for
29:                push $(\sigma_1, s_1)$ onto stack
30:            end for
31:        end if
32:    end if
33: end while
34: return $(\sigma^*, s^*)$
```

$\mathrm{valid}(\sigma, k) := \forall_{k<i<j\leq m} : \sigma(n_i) \neq \sigma(n_j) \vee \sigma(n_i) = \epsilon$

$\mathrm{syn\_max}(\sigma, k) := \forall_{(n_i,n_j)\in E(M)} : (i \leq k \wedge j \leq k) \vee \left(\mathrm{syn}\left(r_{n_j}^{n_i}, r_{\sigma(n_j)}^{\sigma(n_i)}\right) = \mathrm{syn}^*\right)$

Figure 5.3: Pseudocode of our adaptation of the branch-and-bound algorithm to find an optimal alignment σ^*. σ_0 and σ_1 denote partial solutions, while completions are built in σ. syn^* is the maximum possible value of syn, i.e., $\mathrm{syn}^* = 1$ for our binary measure.

which assumes that each of the newly considered edges is aligned perfectly, i.e., with the maximum value syn* attainable by syn. This is a lower bound than the one a naive application of the branch-and-bound algorithm to our ILP would compute. It accounts for our knowledge of the relation between node and edge alignments and avoids syntactic scores of edge pairs which, given σ_0, cannot be aligned any longer.

Of course, σ need not fulfill the constraints of the ILP, and s need not be an attainable score. It is, however, an upper bound for the score of any alignment extending σ_0. If it is not greater than the current best score s^*, we leave the current branch. Otherwise, we check if σ is a valid aligment with score s, i.e., if it satisfies the constraints of the ILP and s is its score (which means that the assumption of perfect syntactic scores on all newly considered edge pairs was correct). If this is the case, we have a new current optimum and do not need to follow the current branch any more either.

If, however, the bound s is greater than the current optimum s^*, but σ violates some constraint or does not achieve the score s because it contains new imperfect syntactic alignments, we have to branch on the decision of how to extend σ_0 by an additional alignment link. We consider the next node with unspecified alignment and recursively apply the algorithm to extensions of σ_0. Each extension σ_1 aligns this node to a partner in N which is not yet taken. This simple check of constraint (1), which extends the general branch-and-bound algorithm, avoids recursing into branches that cannot contain any valid solutions. The partial score s_1 corresponding to σ_1 is computed by taking into account the consequences of the new alignment to the lexical and syntactic scores.

The algorithm terminates when no further branches need to be considered. Because of the theoretical considerations in Section 5.3, no polynomial bound can be given for the number of iterations of the outer loop. This number is trivially bounded by n^{m-1}, as each of the $m-1$ nodes $n_2, \ldots, n_m$ may be aligned to any of the $n-1$ nodes $n'_2, \ldots, n'_n$, or be left unaligned. But this does not, of course, constitute a useful bound. Including all invalid alignments, it is much larger than the number of valid alignments $c(m-1, n-1)$. The algorithm can therefore only be efficient in practice, if the estimated bounds allow it to exclude large parts of the search space from further consideration. The benefit of the bounds has to outweigh the disadvantage of theoretically having to consider a much larger number of (both valid and invalid) alignments. In the next section we will present experimental evidence that this is indeed the case for our problem.

5.5 Average Time Complexity

In this section we estimate the average time complexity of the algorithm in Figure 5.3 on problem instances practically occuring within our expansion

iter. <	10	100	1,000	10,000	100,000	1,000,000
inst.	22.08%	70.08%	93.47%	98.64%	99.73%	99.95%

Table 5.4: Proportion of solvable problem instances for different limits on the number of iterations of our branch-and-bound algorithm

	$m = 5$	$m = 6$	$m = 7$	$m = 8$	$m = 9$	$m = 10$
$n = 5$	13.66%	13.42%	12.71%	13.73%	13.42%	13.57%
$n = 6$	10.20%	9.76%	8.74%	9.38%	8.93%	8.98%
$n = 7$	7.80%	7.21%	6.19%	6.50%	5.98%	5.84%
$n = 8$	6.20%	5.54%	4.54%	4.62%	4.07%	2.46%
$n = 9$	5.09%	4.44%	3.46%	3.42%	(1.39%)	(0.32%)
$n = 10$	4.22%	3.56%	2.66%	1.46%	(0.31%)	(0.06%)

Table 5.5: Average number of iterations of our algorithm relative to number of all valid alignments. Numbers in parantheses exclude timed-out instances.

framework. Specifically, we present data derived from the expansion of a FrameNet development set. The concrete setup of this experiment will be detailed in Section 7.3. For now it suffices to note that it is representative of realistic applications of our framework.

In the given application of the expansion algorithm, graph alignments were performed on $69,982,452$ pairs of a labeled and an unlabeled sentence. For each of them, we determined the number of iterations of our algorithm until the solution was found. We found this distribution to be highly skewed, with the vast majority of problems solvable within a relatively small number of iterations. Table 5.4 shows how many of the total number of instances could be solved within given numbers of iterations, ranging from 10 to 1,000,000 on a logarithmic scale. To prevent rare instances requiring very high numbers of iterations from inordinately slowing down the overall process, we set a time-out of $1,000,000$ iterations. Compared to a time-out in terms of actual CPU time, this has the advantage of making the results of our experiments deterministic and independent of the available hardware. For the 0.05% of problems exceeding this bound, we do not return any alignment, but rather skip the graph pair in question.

Table 5.5 compares the number of iterations of our algorithm to the number of possible alignments, which would have to be enumerated in a naive approach. For various values of m, the size of the alignment domain, and n, the size of the alignment range, we show the ratio $\frac{b(m,n)}{c(m,n)}$, where $b(m,n)$ is the empirical average number of iterations of our variant of the branch-and-bound algorithm and $c(m,n)$ is the number of alignments from Equation (5.1). Although computational cost for one iteration in the two algorithms cannot be compared directly, we can see that the ratio decreases significantly for growing m and n. For example, for $m = 7$ the number of iterations falls from 12.71% of the number of possible alignments for $n = 5$ to only 2.66% for $n = 10$. This shows that in a realistic setting the average performance of our algorithm scales qualitatively better to larger problems than naive enumeration. The relative performance gains of several orders of magnitude make its application in our framework feasible.

It is interesting to note that Table 5.5 is not symmetric in m and n, whereas the problem definition and $c(m, n)$ are. This could be used to optimize an implementation by swapping the two graphs whenever $m > n$. As the benefit is rather small, we did not implement this, though.

5.6 Summary

In this chapter, we have analyzed the optimization problem raised by our definition of similarity between predicate-argument structures. We have shown that the size of the problem makes a naive approach to its solution infeasible. Therefore, a formulation as an integer linear program was proposed. Analysis of this problem showed that no solution algorithm with a polynomial worst-case complexity is available (nor possible, if $P \neq NP$). However, we have presented an adapted branch-and-bound algorithm that efficiently solves all but a negligible number of instances of our problem in practice.

We have now completely specified the expansion algorithm of our semi-supervised approach to SRL. In the next chapters of the thesis we will proceed to evaluate it in a number of experimental settings. In the following chapter, we first describe the supervised SRL system that will be employed for this evaluation.

Chapter 6

Supervised Semantic Role Labeler

Every individual matters.
Every individual has a role to play.
Every individual makes a difference.

Jane Goodall

In this chapter, we describe the supervised frame and role labeling system that forms the basis for the evaluation of our semi-supervised SRL approach. In the experiments conducted in Chapters 7 and 8, this system is trained on different data sets, either comprising only manually annotated sentences or augmented by application of the expansion algorithm described in Chapter 3. This will allow us to assess the benefit of the automatically generated trainind data.

We will first explain our choice of architecture for the supervised SRL system, and then describe how it realizes frame labeling, role recognition and role classification. Finally, we discuss how to evaluate SRL performance.

6.1 Architecture

A wide range of supervised SRL systems has been proposed in the literature since the task was first addressed by Gildea and Jurafsky (2002). For an extensive survey of the state of the art we refer the reader to Palmer et al. (2010). Frequently, the problem is split into a number of subtasks, which are then solved sequentially in a "pipeline architecture". For the task of labeling FrameNet frames and roles, we can identify three such stages:

1. **Frame Labeling**: Choosing the right frame for a given FEE.

2. **Role Recognition**: Deciding whether a given substring realizes a role of a given frame or not.

3. **Role Classification**: Choosing the right role label for a given substring realizing a role.

In principle, a complete system would have to perform an additional step before frame labeling, namely recognize which words actually evoke a frame. We leave this stage out of our SRL architecture and evaluation because the FrameNet corpus in its present state does not contain full text annotations that could be used to train or test the recognition of FEEs, but rather only exemplifies a single frame on each annotated sentence. Recently, some fully annotated data sets have become available from the FrameNet project and the SemEval 2010 Shared Task on "Linking Events and their Participants in Discourse" (Ruppenhofer et al., 2010a), but they are still too limited in size to be a basis for our evaluation, comprising not more than a few hundred sentences.

The first of the three stage, frame labeling, is similar to a supervised word sense disambiguation task. While most approaches to word sense disambiguation are based on the sense inventory provided by WordNet, a lexical unit capable of evoking different frames is ambiguous between different senses defined by the frame inventory. The stages of role recognition and role classification are executed separately for technical reasons. In principle, they could be combined into one classifier deciding between a number of role labels or the special label *none*. As training data for such a classifier would be heavily biased towards this *none* class, however, it has proved beneficial to filter out this class in the role recognition stage and let a dedicated classifier decide between the actual role labels in the role classification stage (Pradhan et al., 2005).

Although popular, this pipeline architecture for SRL is not universally used and more integrated models have been proposed to allow later stages to correct mistakes of earlier ones. On the other hand, a key advantage of the pipeline architecture is its efficiency. This makes it suitable for evaluation in our experiments, which will involve retraining of the system on many different training sets.

Most existing SRL systems rely on the output of a syntactic parser and extract various lexical and syntactic features from training instances and input sentences. Early approaches to SRL were based on phrase structure trees, but recent attention has shifted towards the output of dependency parsers (see Section 2.1), as witnessed, e.g., in the Shared Task on "Joint Parsing of Syntactic and Semantic Dependencies" (Surdeanu et al., 2008). There are multiple reasons for this, including advances in research on dependency parsing and availability of parser implementations for various languages. Perhaps the most convincing advantage is that syntactic dependency structure is conceptually closer to the semantic predicate-argument structures that SRL systems try to extract. This has already been observed by Fillmore (1968), who notes that the concept of predication in classical logic seems for a long time to have fostered the view that there is a fundamental distinction between subject NPs and other arguments of the verb (collected in the VP), with the former not requiring further analysis regarding their semantic function. The structure of classical phrase structure grammars reflects this view. Dependency syntax, on the other hand, treats all dependents of a predicate equally, whether they stand in a subject, object, or other relation. It therefore better corresponds to the principles of Frame Semantics, which describe the relationships between a predicate (specifying an action or situation) and all of its arguments (denoting the participants).

Since automatic conversion between phrase structure trees and dependency graphs is feasible with reasonable accuracy, they may to some degree be viewed as different representations of the same information. Experimental results confirm this view, showing that SRL systems utilizing dependency information achieve similar performance as those built on phrase structure trees (Johansson, 2008). This of course means that in practice the decision between the formalisms will often be made according to the availability of tools and resources for a given language and domain. With our semi-supervised approach based on dependency graphs, it is therefore natural to choose a dependency-based SRL system for our evaluation.

6.2 Frame Labeling

To predict the frame type evoked by an FEE in a given sentence, we implemented a classifier based on features extracted from dependency graphs. Our choice of features is modeled after those presented in Johansson and Nugues (2007a).

The task of choosing the right one out of hundreds of frames with sometimes very fine-grained semantic distinctions is substantially difficult. It is significantly simplified if an accurate frame lexicon such as the one included with FrameNet is available, listing for each predicate the frames it can possibly evoke. But even if manually encoded information is unavailable for a given lexical unit, it can still be approximated by inferring frame candidates automatically. We will describe several such approaches when we are faced with the task of labeling unknown predicates in Chapter 8. For now, we assume that our frame labeler is provided with a small set of evokable frames for each target lemma type.

A complete list of all features used by our frame labeler is given in Table 6.1. The features *voice* and *parent_has_obj* are binary, all others take values from (sometimes much) larger finite sets. Following standard practice in machine learning we replace them by corresponding binary features before

Feature	**Type**	**Description and example value**
target_lemma	atomic	lemma of the target node (savour)
frames	set	set of frames that can be evoked by the target verb ({PERCEPTION_ACTIVE, EXPERIENCER_SUBJ})
voice	binary	voice of the target node (active)
parent_word	set	lemmata of the parents of the target node ({drink})
parent_POS	set	parts of speech of the parents of the target node ({VVD})
rel_to_parent	set	grammatical relations between the target node and its parents ({XMOD})
parent_has_obj	binary	whether or not any of the parents has an outgoing "object" relation (no)
dsubcat	atomic	subcategorization frame, i.e., the multi-set of all outgoing relations of the target node (DOBJ, IOBJ)
child_word_set	set	set of lemmata of all children of the target node ({flavour, with})
child_dep_set	set	set of all outgoing relations of the target node ({DOBJ, IOBJ})
child_word_dep_set	set	set of pairs (lemma, relation) for all children of the target node ({(flavour, DOBJ), (with, IOBJ)})

Table 6.1: Features used by the frame classifier. Example values for the target predicate *savour* in the annotated graph of Figure 6.2 are given in parantheses.

feeding them to the learning algorithm. Each of the set-valued features (indicated by the type "set" in the table) is represented by a number of binary features corresponding to the possible elements of the set. A value of 1 indicates the presence of this element in the set, while a value of 0 indicates its absence. This is a natural form of subset encoding: each of the 2^k subsets of a set with k elements is encoded by a specific configuration of k binary feature values.

The features *target_lemma* and *dsubcat* take atomic values. For the latter feature in particular this means that each subcategorization frame is viewed as a separate value independent of all others, even if it shares some grammatical relations with them. Consequently, we represent each possible feature value by an independent binary feature. Of these features, exactly one will take the value 1, and all other the value 0. This is known as 1-of-k coding in the machine learning literature (Bishop, 2007).

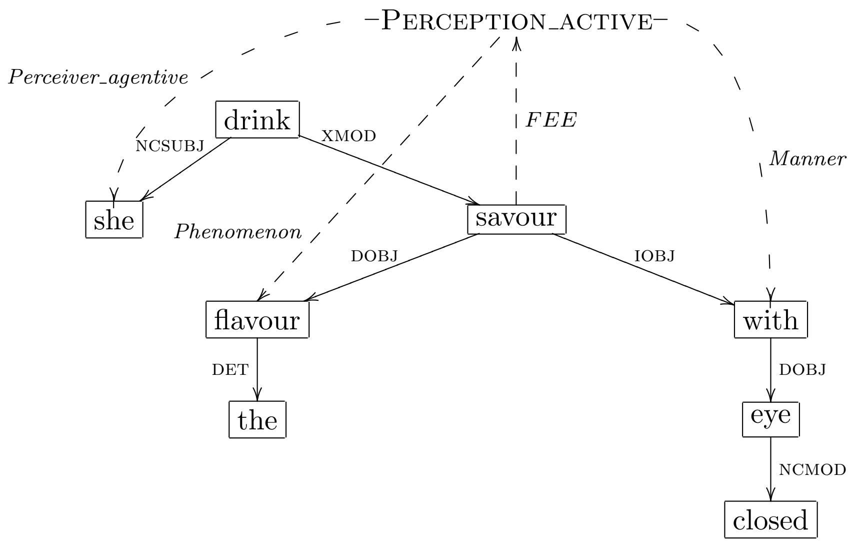


Figure 6.2: Annotated dependency graph for the sentence *"She drank, savouring the flavour with closed eyes"*. The word *savour* evokes the PERCEPTION_ACTIVE frame with the three roles *Perceiver_agentive*, *Phenomenon*, and *Manner*. Extracted features are shown as examples in Tables 6.1 and 6.3.

The extracted features are used to train a linear support vector machine (SVM) model. We employed the LIBLINEAR implementation by Fan et al. (2008) with the cost parameter C set to 1.0 (the default). We follow the one-versus-one approach(Friedman, 1996) for multi-classification. This means that one SVM is trained for each pair of frames, with the binary prediction interpreted as a vote for either one or the other of the pair. Each classifier is trained on all instances of the two frames it chooses between. When labeling a target predicate, the predictions of all classifiers deciding between any two of the evokable frames are considered, and the frame with the most votes is chosen.

Note that under this approach it is possible to apply the frame labeler to unseen predicates, i.e., predicates for which there are no annotated sentences among the training data, provided a set of frame candidates can be specified. However, while the model may be able to infer some information by generalizing across predicates, the quality of those predictions can be expected to be low. We will address this issue in detail in Chapter 8.

6.3 Role Recognition and Classification

The two stages of role labeling follow a similar approach as the preceding frame labeling stage. The task now consists of deciding for each node of a given dependency graph whether it carries a role and, if so, what role label should be assigned to it.

The same set of features is used for both tasks. A complete list is shown in Table 6.3. The upper part lists those features which are not specific to the graph node being classified. Those overlap to a large extent with the features used in frame labeling. New features include the set feature *roles* which depends on the frame chosen in the preceding frame labeling stage and incorporates information from the frame lexicon, and the atomic feature *target_POS*. The lower part lists features which are specific to the given graph node. All of them except *function* are atomic features describing the argument head itself and its surroundings in the sentence. A very important feature is *path*. It captures the syntactic relationship between the FEE and the argument head. As in the case of frame labeling, the selection of these features follows Johansson and Nugues (2007a).

Role recognition is handled by a single linear SVM with cost parameter of $C = 0.1$. This value proved superior to the standard of $C = 1.0$ in preliminary experiments. Role labeling is performed by a one-versus-one set-up of linear SVMs with $C = 1.0$ and a voting approach similar to the one employed for frame labeling. The previous frame labeling decision is taken into account by training frame-specific sets of classifiers. This reflects the basic concept of Frame Semantics that roles are specific to the frames they are defined for.

6.4 Evaluation Measures

As SRL constitutes an intermediate task, deriving semantic representations, but not on its own addressing a specific NLP problem, there are various ways of evaluating the quality of an automatic SRL system. The most straightforward approach is to count the number of sentences in a manually annotated test corpus for which a completely accurate analysis was produced, i.e., for which all annotation decisions match those of the manually annotated gold standard. In our context this means that the right frame was annotated on the given target predicate, and the role-labeled substrings coincide exactly with those in the gold standard. Counting the relative number of sentences in the test set for which this is the case, we obtain the evaluation measure of *exact match*.

It is important to note that in order to avoid a systematic bias we have to perform all evaluations on the *original form of manual annotation*, and not, e.g., on the labeled dependency graphs produced by our preprocessing procedure described in Chapter 2. If we were to compare on that level, we might implicitly make the task easier and so inflate evaluation results. For example, there may be sentences in the test set for which a role substring cannot be matched perfectly with a subgraph of the dependency graph. We could now either exclude such sentence from the test set or map the relevant

Feature	**Type**	**Description and example value**
target_lemma	atomic	lemma of the FEE (savour)
target_POS	atomic	part of speech of the FEE (VVG)
roles	set	set of roles that can feature in the given frame ({*Perceiver_agentive*, *Phenomenon*, *Manner*, ...})
voice	binary	voice of the FEE (active)
parent_word	set	lemmata of the parents of the FEE ({drink})
parent_POS	set	parts of speech of the parents of the FEE ({VVD})
rel_to_parent	set	grammatical relation between the FEE and its parents ({XMOD})
parent_has_obj	binary	whether or not any of the parents has an outgoing "object" relation (no)
dsubcat	atomic	subcategorization frame, i.e., the multi-set of all outgoing relations of the FEE (DOBJ, IOBJ)
child_dep_set	set	set of all outgoing relations of the FEE ({DOBJ, IOBJ})
arg_word	atomic	lemma of the argument head (flavour)
arg_POS	atomic	part of speech of the argument head (NN1)
position	atomic	position of the argument head in the sentence (before, on, or after), relative to the FEE (after)
left_word	atomic	lemma of the word to the left of the argument head in the sentence (the)
left_POS	atomic	part of speech of the word to the left of the argument head in the sentence (AT)
right_word	atomic	lemma of the word to the right of the argument head in the sentence (with)
right_POS	atomic	part of speech of the word to the right of the argument head in the sentence (IW)
path	atomic	path of grammatical relations from the FEE to the argument head, omitting steps upwards in the graph to verbal heads (DOBJ)
function	set	set of relations between the argument head and all its heads ({DOBJ})

Table 6.3: Features used by the role classifiers. Example values for the target argument head *flavour* in the annotated graph of Figure 6.2 are given in parantheses.

role to the closest matching graph node according to the procedure detailed in Chapter 2. In both cases we would influence the evaluation. Skipping such sentences might systematically exclude hard cases from the test set, while a gold standard instance with an imperfectly mapped role might be easier to predict than the original role-bearing substring, which could not be mapped in the first place. For an unbiased evaluation, we therefore evaluate on the level of the annotated substrings of the FrameNet corpus and demand that these substrings match exactly. In the case of the SALSA corpus, the original annotation has been carried out on the phrase structure trees of the TIGER treebank (Brants et al., 2002). Evaluating on the corresponding substrings, however, is equivalent to evaluating on sets of terminal and non-terminal nodes, so we apply the same procedure here.

While the evaluation measure of exact match is easy to define, it has the disadvantage of treating all kinds of errors in the labeling of a sentence equally. It does not distinguish between a severe mistake, such as labeling an FEE with a wrong frame or predicting a completely wrong set of roles, and minor ones, such as the mislabeling of a single role or the precise extent of a substring. Often, however, it is better to get a partially correct solution than none at all. In this case, exact match will make it harder for us to compare different SRL systems: a system might score lower than another one, i.e., produce fewer perfect analyses, but at the same time also make fewer fatal mistakes. The latter would not be reflected by the exact match measure, although – depending on the application – it might make the lower scoring system preferable.

The most popular way of defining more fine-grained evaluation metrics is to evaluate the labeling decisions individually and then aggregate the results. For an SRL system based on Frame Semantics, the labeling of frames is one of these decisions. As a multi-class labeling decision, it can be evaluated by *frame labeling accuracy*, i.e., the relative number of correctly labeled FEEs in the test corpus. If the correct frame for an FEE is identified, we then have to evaluate the correctness of the role labeling task. We will first discuss how to evaluate the recognition of the correct role-bearing substrings. Usual measures for such a recognition task are *precision* and *recall*. They are defined by counting the number of *true positive (TP)*, *false positive (FP)*, and *false negative (FN)* role instances over the test set. A true positive instance is a role-bearing substring which occurs both in the gold standard and the prediction for a particular sentence. A false positive accordingly is a substring which occurs in the prediction, but not in the gold standard, while a false negative conversely occurs in the gold standard, but not in the prediction. Precision (P) and recall (R) are then defined as:

$$\mathrm{P} = \frac{TP}{TP + FP} \tag{6.1}$$

$$\mathrm{R} = \frac{TP}{TP + FN} \tag{6.2}$$

Precision thus quantifies the number of true predictions relative to the number of all predictions, while recall quantifies the same number of true predictions relative to the number of all predictions that should have been made according to the gold standard. Often, precision and recall are combined into the F_β score (Rijsbergen, 1979)

$$F_\beta = \frac{(1+\beta^2) \cdot P \cdot R}{\beta^2 \cdot P + R} \tag{6.3}$$

Here β determines the weight of precision relative to recall. Most commonly, the two are weighted equally ($\beta = 1$), thus giving rise to the F_1 score:

$$F_1 = \frac{2PR}{P + R} \tag{6.4}$$

This is the harmonic mean of precision and recall. Thus far, these measures do not take into account the role label, but only the recognition of role-bearing substrings. We will therefore call them *unlabeled precision*, *unlabeled recall*, and *unlabeled F_1 score*, respectively. If the right role-bearing substrings were identified, the labeling is again a multi-class decision, which can be evaluated in terms of *role labeling accuracy.* We therefore have three types of measures:

(1) Frame labeling accuracy

(2) Unlabeled precision, recall, and F_1 score (for role recognition)

(3) Role labeling accuracy

These correspond to the three stages of our pipeline architecture and therefore evaluate the individual components of our SRL system. It should be stressed, however, that they are equally applicable to any other SRL architecture: evaluating certain aspects of the result does not require these to be handled independently in prediction. However, these measures nonetheless serialize the evaluation process. We can only sensibly determine the measures in (2) on sentences labeled with correct frames, as determining whether roles were correctly recognized does not make sense if the predicted frame was wrong in the first place. Similarly, we cannot assess the correctness of labels on invalid substrings, so (3) requires a correct solution of the role recognition task.

There are various ways to address these dependencies. We could base the evaluation of any stage on gold standard annotations of the preceding stages. This allows fine-grained evaluation of the components of the SRL system, but does not well assess overall performance of the system: it might be better in practice for the sets of instances on which the individual components work well to have large overlap. As an extreme example, consider a frame labeler correctly predicting one half of the instances. If the role recognition also works well on one half of the instances, it would be best for these two sets to coincide. In the worst case they are disjoint, so that no correct role is predicted. Evaluation on gold standard annotations does not capture such interdependencies. An alternative is to evaluate each stage not on the complete test set, but on the subset of instances correctly handled by all previous stages. This takes into account correlation in the errors of different components, but leads to problems when comparing different systems. For example, a change in our SRL system or its training data may change the set of instances for which a correct frame is predicted. This, however, is exactly the test set for the subsequent evaluation of role recognition. If the evaluation measures of role recognition improve, we therefore do not know whether this is actually due to better role recognition, or rather because the instances with a correct frame label are now in some way "easier" for the role recognition task than before.

Ultimately, the best solution for our purposes seems to be the definition of scores aggregating over all labeling decisions. This is commonly done for role recognition and role labeling by the definition of *labeled precision* and *labeled recall*. These measures differ from their unlabeled variants in that a true positive (TP) is only counted when a correctly recognized substring was also labeled with the correct role label. Otherwise, we count it both as a false positive (FP), since an incorrect role was predicted, and as a false negative (FN), reflecting the fact that the correct role was missed. This may seem like penalizing one mistake twice, but is actually necessary for consistency, as $TP + FP$ must reflect the total number of predicted roles, while $TP+FN$ is the total number of roles in the gold standard. Decreasing TP by 1, we therefore have to increase both FP and FN by 1, since the totals stay the same. The *labeled F_1 score* is then defined in terms of labeled precision and labeled recall according to (6.4).

We extend the common notion of these labeled scores to also include the frame labeling decision. This means that a predicted role counts as a true positive only if its label is correct *and* the correct frame was predicted for the sentence in question. Otherwise it is again counted as both a false positive and a false negative. Following this definition, a frame labeling mistake causes all the predicted roles to be wrong. This conforms well to the basic concept of frames in Frame Semantics: if we do not adequately characterize the *situation* as expressed in the frame, then any further role prediction is bound to be misleading and should be regarded as a mistake.

In the evaluation of our experiments in Chapters 7 and 8, we will concentrate on the labeled F_1 score and on frame labeling accuracy, which can be assessed on its own as it does not depend on any prededing stage. In the appendix, we also report results in terms of labeled precision, labeled recall, and exact match.

6.5 Summary

In this chapter, we have described the overall architecture of our supervised SRL system and the implementation of the consecutive stages of frame labeling, role recognition, and role classification. Various evaluation metrics were discussed, ranging from the coarse-grained measure of exact match to fine-grained ways of evaluating individual system components. We have argued that measures of intermediate granularity, representing composite scores over the different stages but still sensitive to the quality of individual labeling decisions, are most appropriate for our purpose. This led to the definition of labeled precision, recall, and F_1 score taking into account both frame and role labeling decisions.

The supervised SRL system and the evaluation measures presented here will be used in the evaluation of our experiments in the following Chapters 7 and 8.

Chapter 7

Semi-supervised Learning for Known Predicates

It ain't what you don't know
that gets you into trouble.
It's what you know for sure
that just ain't so.

Mark Twain

In this chapter, we describe a first set of experiments, in which we apply our expansion framework of Chapter 3 to generate novel training instances for *known predicates*, i.e., predicates for which some sentences with manual semantic annotations are available.

We first describe the evaluation procedure carried out to assess the benefit of our semi-supervised SRL approach. Then we instantiate our general framework by choosing one of the two lexical similarity measures presented in Sections 4.1 and 4.2, and tuning the weight parameter α discussed in Section 4.4. To assess the performance of this instantiation of our model, we then conduct a number of experiments on subsets of different sizes, sampled from the English FrameNet and the German SALSA corpus. Finally, we compare our method to a self-training approach.

7.1 Evaluation Procedure

The goal of our semi-supervised SRL approach is to improve the prediction quality of a supervised SRL system. A suitable evaluation procedure should therefore measure the performance of an SRL system when trained on an augmented training set, generated by applying our expansion algorithm to a seed set of manually annotated sentences. Comparison with the same SRL system trained only on the original seed data will then show if and to which extent our expansion algorithm was able to effect an increase in performance. Our evaluation is thus carried out as follows:

(1) Apply our expansion algorithm from Chapter 3 to a seed set, augmenting it to a *training set* for the SRL system.

(2) Train our SRL system from Chapter 6 on this training set.

(3) Apply the trained system to a test set.

(4) Evaluate labeling performance on the test set by comparing the predicted annotations with the held-out gold standard annotations.

The seed and test sets in this procedure will vary depending on the specific experiment. In Sections 7.2 and 7.3 we will carry out experiments to find a suitable instance of the framework described in Chapter 3: We will choose between two definitions for the lexical similarity measure lex and determine an optimal value for the weight parameter α. In the subsequent sections we will then evaluate the quality of this particular instance of our framework. It is essential that the choice of this instance is based on a *development set* independent of the data used to evaluate its performance. In a realistic application of our semi-supervised approach we would not be able to choose our model based on the specific data we want to derive semantic analyses for, as this would require gold standard annotations for the very data we want to label. Choosing an instance of our framework based on a separate development set therefore ensures the general validity of our evaluation results. For our experiments, we chose a random sample of 20% of the instances of verbal FEEs in the FrameNet corpus as a *development seed set* (12,134 sentences) and another 10% as a *development test set* (6,066 sentences). For experiments on the German SALSA corpus, we will apply the same similarity measure and value of α and therefore do not require a separate development set. Throughout this and the following chapter, we will use the entire BNC, comprising 6,026,276 sentences, as our unlabeled expansion corpus for experiments on English, with RASP parses extracted from the corpus described in Andersen et al. (2008). For experiments on German, we will employ a corpus of 20,742,146 sentences of newpaper texts from “Süddeutsche Zeitung” (years 1995 to 2003). All these sentences were parsed with the German LFG parser described in Section 2.4.

The present chapter concentrates on role labeling performance, reporting labeled precision, labeled recall, and labeled F_1 scores on instances with gold standard frame labels. In the case of known predicates, which is considered here, the frame labeling task is much less interesting than the role labeling task. Typically, a predicate has a very dominant frame, corresponding to its most frequent word sense. Moreover, FrameNet does not typically contain annotated instances for all or even most senses of a predicate. Results on subsets of the FrameNet corpus therefore do no represent general frame labeling performance very well, yielding overly high accuracy results in the 80% range (Johansson and Nugues, 2007a) or even 90% range (Erk and Padó, 2006), with majority baselines not much lower. The remaining labeling errors often reflect gaps in the annotation corpus, e.g., predicate senses not exemplified by any annotated sentences in the training set. Such problems cannot be solved by our similarity-based approach, as it is not able to induce frames that are not annotated in the first place. Proposals like that of Ruppenhofer et al. (2010b), who coarsen the frame structure of FrameNet by merging similar frames, may be more suitable to alleviate this problem in practice. We will address the potential of our approach with regard to frame labeling in the next chapter, where we show that in the case of *unknown predicates* there is ample room for improvement.

7.2 Choice of Lexical Similarity Measure

In this section, we compare the influence of the two alternative definitions of our lexical similarity measure lex that were given in Sections 4.1 and 4.2. For clarity, we call the measure based on cosine similarity in a distributional vector space lex_{cos} (see Equation (4.2) on page 52), and the measure based on bounded Jiang-Conrath similarity in the WordNet hierarchy lex_{JC} (see Equation (4.15) on page 58). To compare these two measures empirically, we carry out the evaluation procedure described in Section 7.1 on the development seed and test sets. The parameter k of our expansion algorithm is set to 1, so that we augment the development seed set with the nearest neighbours of the seeds. In Section 7.4, the influence of this parameter will be examined in detail.

Table 7.1 shows labeled precision, labeled recall, and labeled F_1 scores on the development test set for the two measures lex_{cos} and lex_{JC}. For syntactic similarity, the binary measure of equation (4.17) is employed. A range of different values of the logarithmic weight parameter between $\log \alpha = -3.0$ and $\log \alpha = 3.0$ is considered. Additionally, the special cases $\log \alpha = -\infty$ (i.e., $\alpha = 0$, only lexical information) and $\log \alpha = \infty$ (i.e., the limit for $\alpha \to \infty$, only syntactic information) are shown, which were discussed in Section 4.4. As a baseline, we also show the performance of the SRL system trained on the development seed set without any additional training instances. The results for the baseline and $\log \alpha = \infty$ do not depend on the lexical similarity measure and are therefore identical in the columns for lex_{cos} and lex_{JC}.

The most striking result is the fact that, in spite of incorporating information from a manually built taxonomy, the measure lex_{JC} performs consistently worse than lex_{cos}, both in labeled precision and labeled recall. Whereas lex_{cos} is able to improve over both the baseline and the purely syntactic similarity measure, the addition of any amount of information from lex_{JC} leads to substantially lower performance. For lex_{cos}, on the other hand,

	lex_{cos}			lex_{JC}		
$\log\alpha$	P/%	R/%	F_1/%	P/%	R/%	F_1/%
$-\infty$ (lex)	47.27	36.84	41.41	42.70	32.79	37.10
-3.0	47.71	37.52	42.00	42.61	33.10	37.26
-2.0	**48.23**	38.13	42.59	42.78	33.26	37.42
-1.0	47.92	38.06	42.42	42.89	33.52	37.63
0.0	47.83	38.27	42.52	43.13	33.96	38.02
1.0	48.02	**38.39**	**42.67**	42.81	33.77	37.76
2.0	47.46	38.12	42.35	42.83	33.71	37.73
3.0	47.73	38.15	42.41	42.91	33.81	37.82
∞ (syn)	47.47	38.09	42.27	47.47	**38.09**	**42.27**
baseline	48.16	36.88	41.77	**48.16**	36.88	41.77

Table 7.1: Labeled precision, recall, and F_1 score for the two alternative measures lex_{cos} and lex_{JC} under various values for the logarithmic weight parameter $\log\alpha$. Performance on the unexpanded seed set constitutes the baseline.

best results in precision and recall are attained for intermediate values of $\log\alpha$, balancing syntactic and semantic information. To explain this perhaps counterintuitive result, we analyzed the differences in the annotation instances generated under lex_{cos} and lex_{JC}. While their exact effect on the machine learning algorithm used in the supervised SRL system cannot be predicted with certainty, two observations can be made:

First, the limited coverage of the WordNet-based measure lex_{JC} may cause annotation errors. This is exemplified in the following sentences, showing the SOCIAL_EVENT frame with its roles *Host*, *Manner*, *Social_event*, and *Attendee*:

(7.1) In May, [Dundee Port Authority]$_{\textit{Host}}$ [generously]$_{\textit{Manner}}$ [**hosted**]$_{\text{SOCIAL\_EVENT}}$ [a reception]$_{\textit{Social\_event}}$ [for around 50 Council members based in the Tayside area]$_{\textit{Attendee}}$.

(7.2) In April, [SCOTVEC]$_{\textit{Host}}$ [also]$_{\textit{Manner}}$ [**hosted**]$_{\text{SOCIAL\_EVENT}}$ [an event]$_{\textit{Social\_event}}$ [for policy makers in higher education]$_{\textit{Attendee}}$.

(7.3) In April, SCOTVEC [also]$_{\textit{Manner}}$ [**hosted**]$_{\text{SOCIAL\_EVENT}}$ [an event]$_{\textit{Social\_event}}$ [for [policy makers in higher education]$_{\textit{Host}}$]$_{\textit{Attendee}}$.

The sentence (7.1) is manually annotated and functions as a seed for our expansion algorithm. Under both lexical similarity measures, the same unlabeled sentence is selected as the most similar neighbour. However, lex_{cos} leads to the correct annotation (7.2), while under lex_{JC} the incorrect labeling (7.3) is inferred. In (7.2) *SCOTVEC* is correctly aligned to *Authority*, heading the phrase *"Dundee Port Authority"*, which leads to correct projection of the role *Host*. Under lex_{JC}, however, *SCOTVEC* cannot contribute any lexical similarity score as it is not in WordNet. *Authority* is therefore aligned to the suboptimal node *makers*, heading the phrase *"policy makers in higher education"*. This causes the incorrect nested annotation in (7.3).[1] For the rest of the sentence, the two measures lead to identical alignments. The role *Social_event* is projected correctly, and the adverb *also* is mislabeled as *Manner* in both cases. The alignment of *May* and *April* raises the similarity scores, but does not lead to role projection, as *"In May"* – perhaps incorrectly – was not annotated with any role in FrameNet. The example therefore shows how coverage limitations of WordNet can lead to incorrect alignments. By contrast, the distributional model, being completely unsupervised, can be trained on the seed and expansion corpora and therefore capture information about any lexical item occuring in the graph alignments.

A second potential advantage of the measure lex_{cos} is its better suitability for the acquisition of substantially new lexical material. The following example sentences illustrate this:

(7.4) When he and my mother came to live at the house he bought [five Aylesbury ducks]$_{\textit{Self\_mover}}$ to [**swim**]$_{\text{SELF\_MOTION}}$ [on the pond]$_{\textit{Area}}$ and clear it of weed.

(7.5) [A cloud]$_{\textit{Self\_mover}}$ [**swam**]$_{\text{SELF\_MOTION}}$ [on a cloud-reflecting tile]$_{\textit{Area}}$.

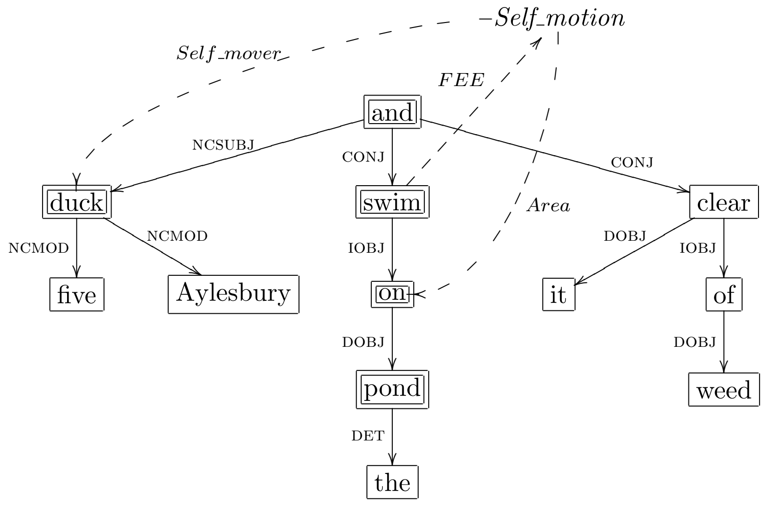
(7.6) [Ducks]$_{\textit{Self\_mover}}$ [**swam**]$_{\text{SELF\_MOTION}}$ about [on the lake]$_{\textit{Area}}$, beside which we would sometimes sit of a summer evening after supper, before going back on duty.

Here, the seed sentence (7.4) gives rise to a substantially novel instance (7.5) under lex_{cos}, while lex_{JC} cannot detect the validity of this inference and comes up with the nearest neighbour (7.6), which is correctly annotated, but much more closely follows the seed – even repeating the word *ducks* – and therefore does not provide as much new information. To see how this comes about, we have to look into the specific graph alignments performed here. Figure 7.2 shows a labeled dependency graph for the seed sentence (7.4) at the top, with the five nodes of the alignment domain indicated by double frames. A complex path (CONJ^{-1}, NCSUBJ) occurs, but is not present in either of the other two sentences (7.5) or (7.6), whose unlabeled dependency graphs are shown below. It therefore does not enlarge their alignment ranges , which are again indicated by double frames.

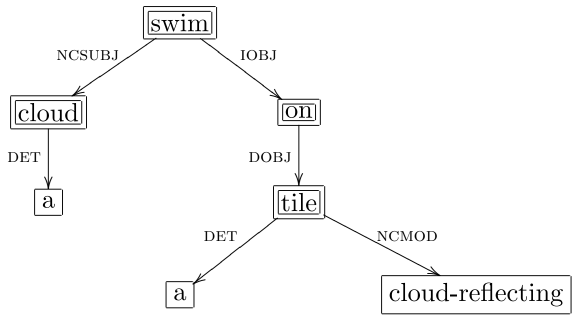
For both sentences, the two lexical similarity measures lead to the same optimal alignments . Their similarity scores, however, differ. For illustration, we compute these scores for a weight parameter of $\alpha = 1$. Table 7.3(a)

[1] Nested labelings need not be invalid in principle. For example, they often occur when body parts are mentioned, such as in the sentence *[[Her]$_{Agent}$ mouth]$_{Body\_part}$ [**dropped**]*$_{\text{BODY\_MOVEMENT}}$ *[open]$_{Result}$*.

(a) Labeled dependency graph for sentence (7.4)



(b) Dependency graph for sentence (7.5)



(c) Dependency graph for sentence (7.6)

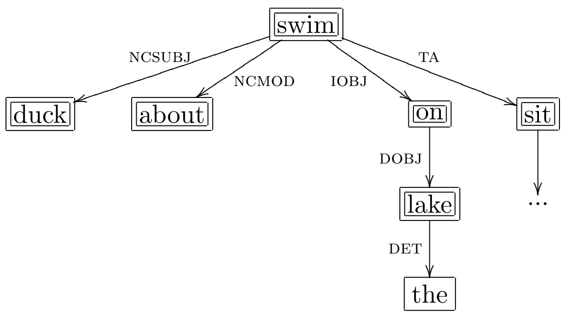


Figure 7.2: Dependency graphs of three sentences featuring the predicate *swim*. The dependent TA (for a text adjunct delimited by punctuation) in (c) is due to a parser error.

(a) Alignment between (7.4) and (7.5)

n	n'	$\text{lex}_{\cos}(n, n')$	$\text{lex}_{\text{JC}}(n, n')$
and	—	0.0000	0.0000
duck	cloud	0.2482	0.0588
swim	swim	1.0000	1.0000
on	on	1.0000	1.0000
pond	tile	0.2389	0.0586
$\sum$ lex		2.4871	2.1174
score		**0.5654**	**0.5188**

(b) Alignment between (7.4) and (7.6)

n	n'	$\text{lex}_{\cos}(n, n')$	$\text{lex}_{\text{JC}}(n, n')$
and	sit	0.0000	0.0000
duck	duck	1.0000	1.0000
swim	swim	1.0000	1.0000
on	on	1.0000	1.0000
pond	lake	0.3495	0.8104
—	about	0.0000	0.0000
$\sum$ lex		3.3495	3.8104
score		**0.5377**	**0.5840**

Table 7.3: Alignments between a seed and two different unlabeled sentences, one favoured by $\text{lex}_{\cos}$, the other by lex_{JC}

shows the alignment between the 5 nodes of the alignment domain of the seed (7.4) and the 4 nodes of the alignment range of sentence (7.5), together with the resulting lexical similarity scores under the two measures. Not surprisingly, lex_{JC} rates similarity between *duck* and *cloud* as well as *pond* and *tile* very low: the corresponding concepts are far from each other in the WordNet taxonomy. $\text{lex}_{\cos}$, on the other hand, gives somewhat higher similarity scores. Inspection of the corresponding vectors shows that context words such as *sky* and *fly* contribute most strongly to the cosine similarity for *duck* and *cloud*. Similarly, the vectors of *pond* and *tile* are connected by context words such as *surface*, *edge*, and *shape*, which characterize their similarity in our context quite well: it is the shape of a plane surface, which makes both a pond and a tile suitable areas for "swimming", either in the literal or the metaphorical sense. Table 7.3(a) also shows the resulting similarity scores according to equation (3.8) from Chapter 3, taking into account the syntactic similarity of the two aligned edges (IOBJ and DOBJ) and the normalization factor (3.9).

Turning to the unlabeled sentence (7.6), we show alignments to the seed (7.4) and lexical similarity scores for both measures in Table 7.3(b). Here the alignment between *duck* and *duck* achieves a maximal score of 1 under both measures. lex_{JC} also rates the similarity between *pond* and *lake* very high (the synset {pond, pool} is a of the synset {lake}), while lex_{cos} gives a somewhat lower score on distributional grounds (top context words for *pond* are *fish*, *garden*, *bottom*, and *pool*, while for *lake* the context words *district*, *mountain*, *river*, and *valley* are most frequent). Syntactic similarity again contributes scores for two edge agreements. The normalization factor, however, is larger than for sentence (7.5), as the alignment range has two more nodes: *sit* and *about* are not well accounted for. The first one, being the result of a parser error, is aligned to *and* without any effect on the similarity score, while the second one is unaligned. As a result, the total score for sentence (7.6) is slightly lower than for (7.5) under lex_{cos}. For lex_{JC} it is the other way around: The higher similarity for the word pairs (*duck*, *duck*) and (*pond*, *lake*) compared with the pairs (*duck*, *cloud*) and (*pond*, *tile*) can compensate for the larger normalization factor, so that in total sentence (7.6) scores higher than (7.5) under lex_{JC}. This is the cause of the difference in the choice of nearest neighbours.

While in general we cannot expect our method to allow correct inference for metaphorical usage as seen in this example, there seem to be circumstances under which distributional similarity more effectively models role filler similarity than a hypernymy hierarchy does. Observing that our two measures lex_{cos} and lex_{JC} seem to model complementary aspects of argument similarity, we also experimented with various ways of combining them, e.g., taking averages, maxima, minima, or falling back from lex_{JC} to lex_{cos} in case of missing lexical items. None of these combinations nor a number of alternative WordNet-based measures led to better results than lex_{cos} on its own, confirming our finding that this measure most flexibly models role filler similarity. Possible role fillers do not seem to be confined to a restricted subgraph of the WordNet taxonomy, but rather cut across it according to a number of different semantic criteria. This phenomenon has been studied in the context of *selectional restrictions* and *selectional preferences*, i.e., the predicate-specific semantic properties of valid arguments. Li and Abe (1998), building on the information-theoretic formulation by Resnik (1996), have proposed to model selectional preferences by tree cuts in a taxonomy. This allows an argument type to be characterized by a set of different subregions of the taxonomy. However, it is not clear how such a model could be integrated into our approach. For the remainder of the present chapter and the following one, we will therefore only consider $\text{lex} = \text{lex}_{\text{cos}}$.

7.3 Tuning the Weight Parameter

So far, we have not yet settled on a fixed value of the weight parameter α. Before evaluating our semi-supervised approach in different realistic set-ups we therefore perform a tuning procedure to determine an optimal value for it. There are different possible objective functions which could be used to optimize α. The most straightforward approach is to measure the quality of a parameter value by the improvement it brings in terms of our evaluation metrics, such as labeled F_1 score. A drawback of this objective function is the high computational complexity of its evaluation. For a given value of α, we have to perform the whole evaluation procedure, including our expansion algorithm as well as training and application of the supervised SRL system, to determine performance on the test set.

There are other possible choices of objective functions, which would be computationally less expensive. For example, instead of generating new instances from an unlabeled corpus, we could employ our expansion algorithm to infer annotations for sentences of a labeled test set. Ignoring their true annotations, our algorithm would infer a projected annotation for each of them, which could then be compared with the held-out gold annotations to quantify the accuracy of the generated training data. This would avoid training and testing of an SRL model and therefore be much cheaper computationally. On the other hand, there are two disadvantages. First, this evaluation does not involve the selection stage of our algorithm: each projected annotation would influence the estimated performance of the semi-supervised approach, even if in a realistic application of the method it would never be selected into the actual augmented training set. Second, the accuracy of the novel instances may be a weak proxy for measuring the success of the approach. In an extreme case, we may tune α to generate perfectly accurate new instances, which nonetheless fail to improve an SRL system because they do not contain new information. Tuning for accuracy of the generated new instances may thus not reflect our actual objective. We therefore choose to stay with the end-to-end evaluation described above, and carry out a tuning procedure for this computationally expensive objective function.

Optimization Problem. The problem of tuning the weight parameter α is different from the optimization problem discussed in Chapter 5 in several ways. First, it has to be solved only once for given similarity measures lex and syn, which makes a computationally expensive empirical objective function feasible. Another important difference is that we are now dealing with a *continuous* parameter, whose infinite set of possible values rules out simple enumeration. Many kinds of continuous optimization methods, however, are inapplicable to our problem. Exact analytic optimization relies on the differentiability (often even the existence of the second derivative) of the objective function and on analytic solvability for the roots of the derivative. These are high requirements, seldom met in practice. Many approximative approaches such as gradient descent methods similarly rely on differentiability. Our objective function, on the other hand, is non-continuous and piecewise constant. This can be seen as follows:

Let α_1 and α_2 be two parameter values. If the corresponding values of the objective function described above differ for these two values, the F_1 scores of the SRL system on the test set have to differ depending on which of the two parameter values was used in the expansion process generating the training data. This is only possible if the training sets differ, i.e., if at least one instance differs. This in turn implies that either the relative similarity of two unlabeled sentences to one seed sentence was changed, leading to a different selection of the seed's neighbours in the *selection stage* of our expansion algorithm, or the optimal alignment for a graph pair changed, resulting in a different role projection for an unlabeled sentence in the *labeling stage*. In either case, the properties of the scoring function (3.6) from Chapter 3 allow us to find a lower bound for the difference of parameter values $|\alpha_1 - \alpha_2|$, except in the case that one of them is in a certain discrete set A of parameter values. This set represents the parameter values for which there are ties between two unlabeled sentences or two competing graph alignments, and an arbitrarily small change of α may thus deliver a different outcome of the expansion algorithm.[2] Outside this discrete set of parameter values we have therefore shown that a change in the value of the objective function implies a minimum change in α. Conversely, this means that the objective function is locally constant outside A. As the discrete set A divides the set of real numbers into open intervals, it is thus constant on any of these intervals, with non-continuities in each point of A.

As a piecewise constant function, our objective function cannot be expected to be maximized by any method involving gradient estimation. Gradient-free algorithms, such as Powell's method (Powell, 1964), try to find an optimum of a function of several variables by a clever choice of search directions. They do not offer any benefit in the case of a single variable, though. We therefore apply a simple tuning procedure based on grid search, which evaluates the objective function on a set of equidistant parameter values. As described in Section 4.4, it is natural to consider the logarithmic weight parameter $\log\alpha$, ranging from $-\infty$ to ∞. Addition of a constant c then corresponds to multiplication of the parameter α with a factor e^c. We choose an additive step size of $c = 0.2$ for $\log\alpha$, which means that the weight of syntactic similarity is increased by a factor of $e^{0.2} = 1.22$ or about 22% in each step, and evaluate our objective function for those parameter values between $\log\alpha = -3.0$ and $\log\alpha = 3.0$ that have not already been computed for Table 7.1.

[2] The specific required property of the scoring function is that it has a continuous derivative as a function of α. If this was not the case, there could be an α_0 for which the difference in similarity values oscillates around 0 with increasing frequency for $\alpha \to \alpha_0$ (an example would be $\alpha \mapsto \alpha \sin\frac{1}{\alpha}$ around $\alpha_0 = 0$, which is differentiable, but whose derivative is not continuous). This would mean that in any arbitrary small neighbourhood of α_0 there would be parameter values leading to either of the sentences being chosen, which means that A would not be discrete.

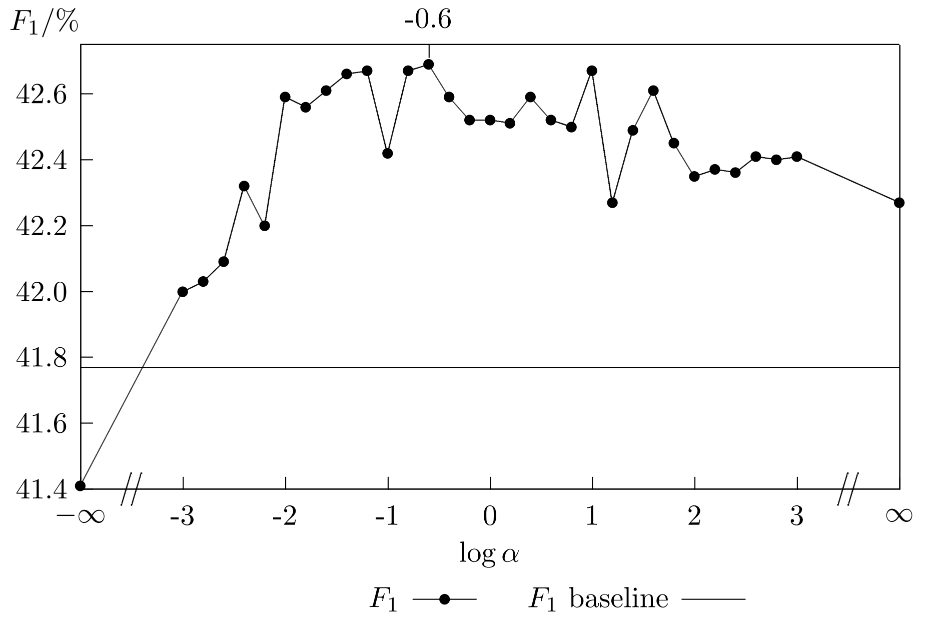


Figure 7.4: Performance of the expansion method on the development set for various values of α. The baseline is a classifier trained on the unexpanded seed set.

Tuning Results. Results of the tuning procedure in terms of labeled F_1 scores are shown in Figure 7.4. For comparison, the baseline performance of an SRL system trained only on the development seed set is indicated by the horizontal line. It can be seen that except for $\log \alpha = -\infty$ all augmented training sets lead to improved performance. This confirms the preliminary results from Table 7.1 that our semi-supervised approach is successful in improving labeling quality, even for non-optimal parameter settings. Furthermore, it is clear that neither lexical information alone nor syntactic information alone lead to an optimal similarity measure. The former actually reduces labeling quality, while basing similarity on syntax alone achieves sub-optimal performance. Finally, while there is some variability in performance depending on the exact value of $\log \alpha$, it is clearly the intermediate range of values that lead to best performance. With the exception of $\log \alpha = -1$ (which is probably due to chance), the objective function is relatively stable between $\log \alpha = -2$ and $\log \alpha = 1$. The empirical maximum in our experiment is at $\log \alpha = -0.6$, corresponding to $\alpha = e^{-0.6} \approx 0.55$. This means that lex is weighted about twice as strongly as syn. We will thus set α to this value for all following experiments.

7.4 Experiments on Corpora of Different Sizes

In this section, we present results of applying our method to seed corpora of different sizes. This allows us to assess in which situations and to what extent our semi-supervised approach can improve the quality of SRL. To show its general applicability, we conduct experiments on both the English FrameNet and the German SALSA corpus.

Experiments on FrameNet. For the evaluation of our results we first set aside a random sample of 10% of the annotations with verbal FEEs in the FrameNet corpus (disjoint from the development sets) as a fixed *test set* (6,066 sentences). The remaining 60% of those annotations, which are neither in this test set nor in one of the development sets used in Sections 7.2 and 7.3, constitute our *complete seed set* (36,400 sentences). From this we extract smaller seed sets according to the following procedure:

For each predicate annotated in FrameNet, we extract a random sample of n annotation instances. If there are less than n annotated instances of a predicate in the complete seed set, all of them are extracted. This procedure yields seed sets of different sizes for different values of n. To facilitate comparison between the different sets, we enforce monotonicity of those sets, i.e., we draw the samples in such a way that the seed corpus for n is a subset of the seed corpus for $n + 1$. This can easily be achieved by randomly drawing one sentence per predicate at a time, incrementally building the seed corpora. The resulting sizes of these corpora can be expected to be roughly linear in n, at least as long as the pool of annotation instances for the majority of predicates is not yet exhausted. We apply our expansion algorithm to each of the seed corpora, producing differently sized augmented sets by variation of the parameter k of our expansion algorithm, which controls how many nearest neighbours of a seed are selected. We thus control corpus size by two parameters: n for the seed corpus size and k for the amount of generated novel instances.

Figure 7.5 shows the resulting corpus sizes for a number of different combinations of n and k. The solid line indicates the sizes of the original seed sets for $n = 1, \ldots, 6, 8, 10$, while the dotted lines represent the sizes of the augmented sets for $k = 1, \ldots, 6$. While they show a regular pattern, the sizes of the augmented corpora are not integral multiples of the sizes of the respective seed corpora. One reason for this is that in the application of the expansion algorithm we disregard unrealiable seed instances, i.e., dependency graphs which could not perfectly be equipped with semantic annotations, receiving a mismatch score > 0 in the preprocessing stage described in Chapter 2. The number of seeds actually used in the expansion is therefore smaller than the number of instances in the seed corpus. When selecting the k nearest neighbours of each of them according to our similarity measure, we therefore do not produce a corpus $k + 1$ times the size of

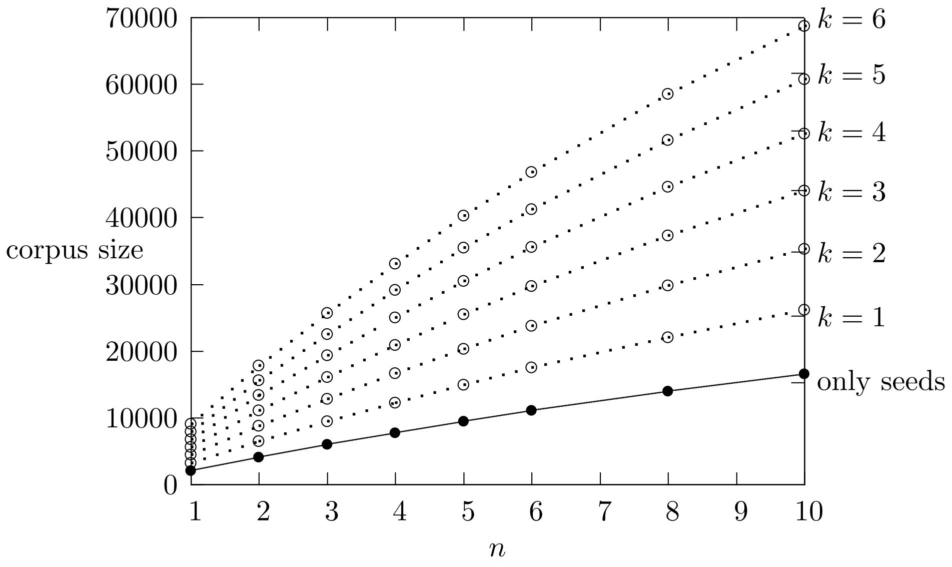


Figure 7.5: Corpus sizes of various training sets: the solid line shows the sizes of the seed sets for $n = 1, \ldots, 10$, while the dotted lines represent the same sets augmented by $k = 1, \ldots, 6$ nearest neighbours.

the seed corpus. Another reason is that even a seed entering the expansion process may fail to produce k new instances. This can happen when it is not the nearest neighbour of a sufficient number of unlabeled sentences and therefore does not have k neighbours in the selection stage. For example, in the augmented corpus with $n = 10$ and $k = 6$ only 83% of the seeds give rise to a full 6 new instances. The average number of new instances per seed in this case is about 5.4.

We apply our expansion algorithm to compare each unlabeled sentences only to labeled seed sentences featuring the *same predicate*. Compared to considering all pairs of predicates, this reduces computational complexity considerably and increases accuracy of the resulting generated instances. In the following chapter, dealing with unknown predicates, we will have to relax this restriction.

Results. We evaluate performance for each of the seed sets and augmented seed sets by training our supervised SRL system on the respective set and then applying it to the fixed test set, which makes results between different training sets comparable. Figure 7.6 shows results in terms of labeled F_1 score for the FrameNet corpus. Each of the horizontal dotted lines indicates the baseline performance when training on one of the seed sets for $n = 1, \ldots, 6$. The solid lines then show how this performance improves by

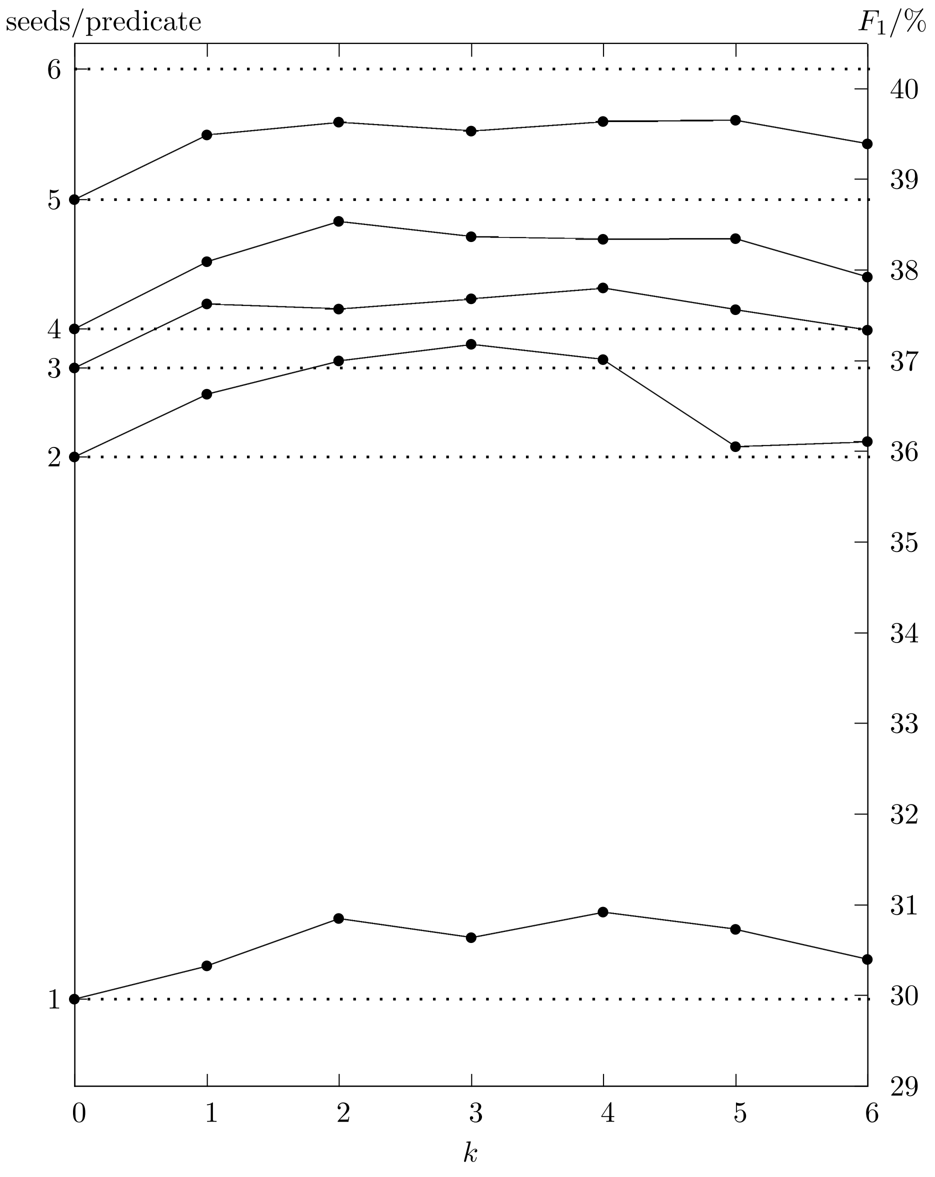


Figure 7.6: Improvements in role labeling F_1 obtained by expanding seed corpora of different sizes: the dotted lines show performance of unexpanded classifiers trained on 1 to 6 seed instances per predicate. Each of the solid lines starts from such a baseline at $k = 0$ and for $k > 0$ shows the improvements obtained by adding the k nearest neighbours of each seed to the respective baseline corpus.

augmenting the seed set. Each of them starts on its respective baseline for $k = 0$ (no expansion), but then rises above it for $k = 1, \ldots, 6$.

It can be seen that each expansion leads to improved performance of the classifier. All improvements for $1 \leq k \leq 5$ (except $k = 5$ for 2 seeds per predicate) are statistically significant ($p < 0.05$, for details see appendix), as determined by stratified shuffling (Noreen, 1989). The curves generally show the largest improvements for k between 2 and 4, declining for higher values of k. This reflects the trade-off between the acquisition of larger amounts of novel information and the inevitable introduction of noise due to incorrectly inferred annotations. For progressively less similar neighbours, the positive effect of the former is outweighed by the detrimental effect of the latter. The progressively lower accuracy of the annotations on more distant neighbours is illustrated by the following example:

(seed) In other academic areas it is assumed that [the teacher]$_{\textit{Communicator}}$ knows more than the student, and is there to [**convey**]$_{\text{SUCCESSFULLY\_COMMUNICATE\_MESSAGE}}$ [this knowledge]$_{\textit{Message}}$, whether as a corpus or a skill.

(1) However, what is most apparent generally in the provisions described is that deaf children are unable to interact, do not contribute to class lessons through speech, are subjected to distorted and exaggerated mouthings by [teachers]$_{\textit{Communicator}}$ and pupils in order to [**convey**]$_{\text{SUCCESSFULLY\_COMMUNICATE\_MESSAGE}}$ [specific information]$_{\textit{Message}}$ (i.e. not natural language interaction) and are unlikely to have secure peer group friendships.

(2) Elaborate wood and iron work, overhanging eaves, portes cochères, and [all sorts of architectural ornamentation]$_{\textit{Communicator}}$ all [**conveyed**]$_{\text{SUCCESSFULLY\_COMMUNICATE\_MESSAGE}}$ [a sense of the romance of travel]$_{\textit{Message}}$, identified the station as landmark, and offered the various companies the opportunity to distinguish themselves by particular 'house' characteristics.

(3) For [a moment]$_{\textit{Communicator}}$ her eyes and her smile, turned to me, [**conveyed**]$_{\text{SUCCESSFULLY\_COMMUNICATE\_MESSAGE}}$ [a hint of past emotions]$_{\textit{Message}}$.

(4) At first [Skip]$_{\textit{Communicator}}$ [**conveys**]$_{\text{SUCCESFULLY\_COMMUNICATE\_MESSAGE}}$ [to us]$_{\textit{Message}}$ that he was competitive and that he had hopes for the boat.

We show the sentences unabridged to convey a sense of the difficulty they present to the parser. As a consequence, there are several errors in their syntactic analyses. Nonetheless, syntactic similarity (direct objects of the verb *convey*) and lexical information (identity of the word *teacher*) work together to yield an almost perfect annotation of the most similar neighbour (1), only missing the extension of the *Communicator* role to *"and pupils"*. In the second nearest neighbour (2), the choice of frame seems a little bit awkward, but is actually the best which can be achieved given the seed data: the only alternative frame annotated in FrameNet for *convey* is BRINGING, which is clearly not applicable here. Neighbours (3) and (4), on the other hand show clear role labeling errors in the *Communicator* and *Message* roles, respectively. This is reflected in the lower similarity between the aligned role filler heads (*teacher*/*moment* and *knowledge*/*to*). The example therefore illustrates how lower similarity corresponds to lower annotation accuracy.

Another interesting observation that can be made in Figure 7.6 is that the addition of automatically generated instances often has a positive effect on role labeling performance similar to – or even exceeding – the addition of one manually labeled instance per predicate. For example, the corpus with n=2 seeds per predicate, expanded by $k = 2$, 3 or 4 nearest neighbours per seed, leads to better performance than the unexpanded corpus with $n = 3$ manually labeled seeds per predicate. Similarly, an expanded version of the $n = 5$ seeds/predicate corpus closes about 60% of the gap to the $n = 6$ seeds/predicate corpus.

Generally, the positive effect of our expansion method is largest for corpora with only a few seed instances per predicate. We did not observe significant improvements for corpora with 6 or more seeds per predicate. Such predicates can be considered adequately represented by FrameNet, at least to the extent to which their word senses are covered. A complete collection of evaluation results for all training sets can be found in the appendix.

Experiments and Results on SALSA. To confirm the validity of our results on a different corpus and language, we carried out an analogous experiment on the German SALSA corpus. We randomly selected 10% of its instances as a test set (1,949 sentences) and sampled seed sets from the remaining 90% (17,545 sentences) as described above. All other parameters are identical to those in the FrameNet experiment. This avoids the necessity of another development set and allows us to verify that the optimal values generalize to another corpus.

Figure 7.7 shows evaluation results of this experiment on seed corpora with $n = 1, 5, 10,$ and 20 sentences per predicate and on the complete seed set ("all"). We observe a significant increase in labeled F_1 scores for all corpus sizes, even on the complete seed set ($p < 0.001$ for $n \leq 20$ and $p < 0.05$ for all but one expansion of the complete set, determined by stratified shuffling, for details see appendix). Probably as a result of the smaller corpus sizes, improvements are much more pronounced than in the case of FrameNet, with augmented versions of the $n = 10$ seeds/predicate corpus outperforming the $n = 20$ seeds/predicate corpus, i.e., compensating an additional 10 manu-

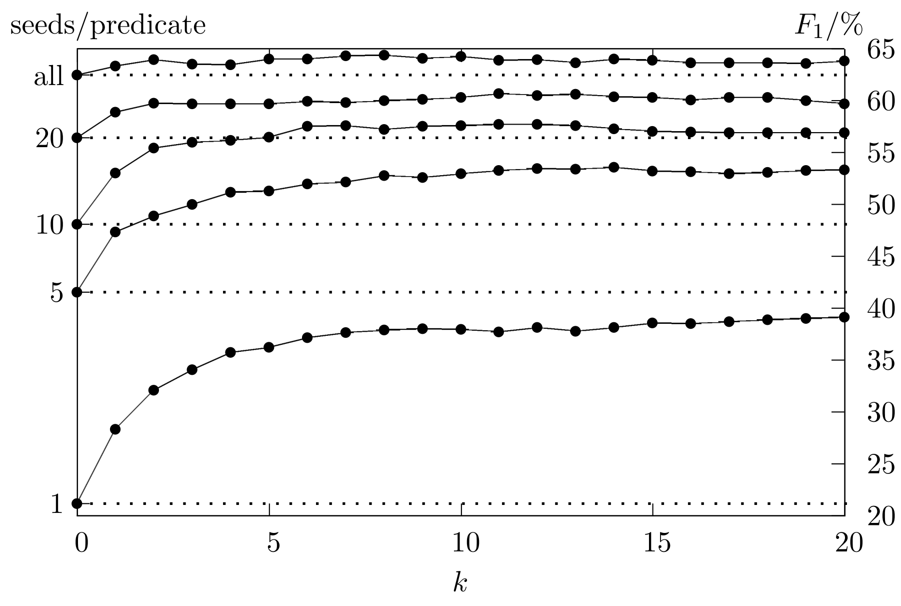


Figure 7.7: Results on the SALSA corpus, depiction like in Figure 7.6

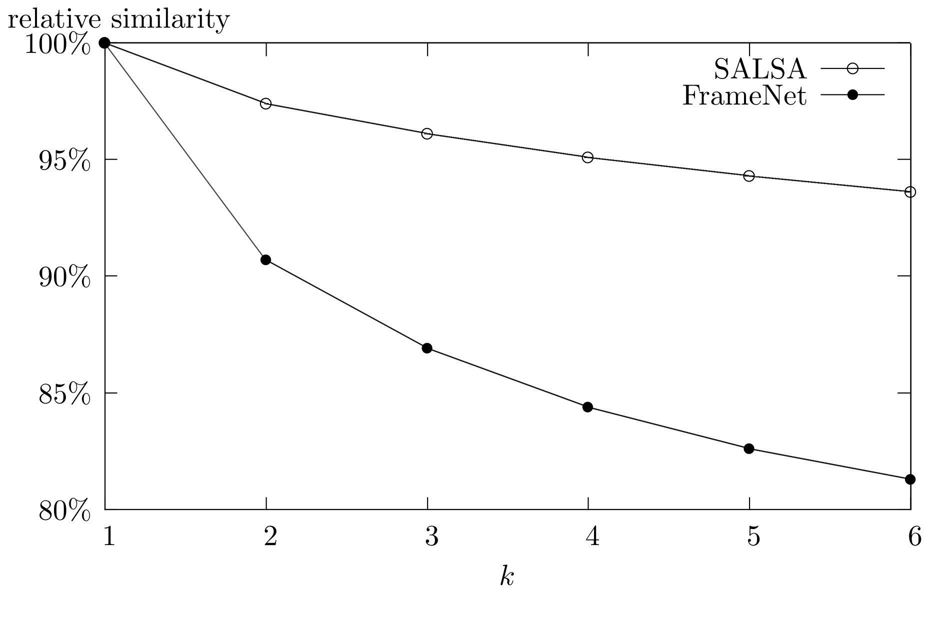


Figure 7.8: Similarity of the k-th nearest neighbour relative to that of the nearest neighbour in the FrameNet and SALSA experiments

ally annotated instances per predicate. The smaller corpus sizes also make it feasible to explore relatively high values for k, yet even there we did not observe significantly declining performance, but rather a leveling-off of labeled F_1 scores. This is most likely due to the more than three times larger and more homogeneous expansion corpus (more than 20 million sentences of Süddeutsche Zeitung compared with about 6 million sentences from the BNC). Here, more distant neighbours are not likely to be much less similar to the seed than closer ones. The following example shows a seed for the predicate *anschießen* (*to shoot*) and some of its neighbours. English word-for-word glosses are provided.

(seed) Wenige Stunden später wurde in Belfast [ein Katholik]$_{\textit{Target}}$ [**angeschossen**]$_{\text{HIT\_TARGET}}$ und schwer verletzt.

Few hours later was in Belfast [a catholic]$_{\textit{Target}}$ [**shot**]$_{\text{HIT\_TARGET}}$ and severely wounded.

(1) [Eine 23-jährige Frau]$_{\textit{Target}}$ ist in Köln auf offener Straße [**angeschossen**]$_{\text{HIT\_TARGET}}$ und verletzt worden.

[A 23-year-old woman]$_{\textit{Target}}$ was in Cologne on open street [**shot**]$_{\text{HIT\_TARGET}}$ and wounded.

(12) [Die Waffenlobby der USA]$_{\textit{Target}}$ ist [**angeschossen**]$_{\text{HIT\_TARGET}}$.

[The US gun lobby]$_{\textit{Target}}$ is [**shot**]$_{\text{HIT\_TARGET}}$.

(19) Noch drei Tage später wollen Jagdaufseher im Wald [[**angeschossene**]$_{\text{HIT\_TARGET}}$ und verletzte Tiere]$_{\textit{Target}}$ aufgespürt haben.

Even three days later claim gamekeepers in the forest [[**shot**]$_{\text{HIT\_TARGET}}$ and wounded animals]$_{\textit{Target}}$ found have.

Coming from the news domain, all of the 11 nearest neighbours closely follow the seed in structure and word sense. Only a sentence as distant as the 12th nearest neighbour shows a different pattern, using *anschießen* metaphorially in a play on words, which our expansion algorithm is unable to handle. A syntactically different sentence is not found earlier than the 19th nearest neighbour. Here, the target is correctly identified as a modifying participle of the role filler head.

The intuition that similarity to the seed declines much less rapidly than in our FrameNet experiments is confirmed by Figure 7.8. Here, we show the average of the ratios between the similarity scores of the k-th nearest neighbour and the first nearest neighbour. The values are extracted from the augmented versions of the seed corpus for $n = 6$ in both cases. With increasing k the similarity ratio falls much less steeply for SALSA than for FrameNet, indicating that the expansion corpus is more slowly exhausted of sentences that our expansion algorithm can confidently annotate.

Flexible Selection of Neighbours We tried various methods of making the selection of novel instances more flexible than simply choosing the k most similar neighbours for each seed sentence. One possibility is to set absolute or relative similarity thresholds, below which a novel annotation instance is discarded. Alternatively, one could conjecture that a cluster of relatively similar neighbours can be distinguished from less reliable instances by a detectable "gap" in absolute or relative similarity scores. None of our experiments with such methods, however, led to any improvements over the results shown here. Apparently, the similarity scores themselves are much less predictive of annotation quality than their relative order for a given seed sentence.

7.5 Comparison with Self-training

Finally, we want to assess whether the improvements shown in the previous sections are due to our formulation of projection confidence in terms of sentence similarity, or whether a simpler method of generating new training instances could work similarly well. We therefore compare our approach to self-training, a semi-supervised method that has been successfully applied to some NLP problems (Mihalcea, 2004; McClosky et al., 2006). Here, instead of projecting annotations we employ the supervised SRL system, trained on a seed corpus, to label parts of the expansion corpus. We then add the resulting annotated sentences to the seed corpus and retrain the SRL system.

We consider the seed corpora used in the FrameNet experiments of Section 7.4. Instead of adding the k nearest neighbours of each seed according to our similarity score, however, we randomly select k sentences from the expansion corpus for each seed sentence. These must feature the same predicate as the seed sentence, so that the resulting corpus contains annotation instances for different predicates in the same proportions as in our earlier experimente. The selected sentences were labeled by the supervised SRL system trained on the corresponding seed corpus. As before, the augmented seed corpus was used to retrain the supervised SRL system, which was then applied to the fixed test set. Results parallel to those for our earlier experiment are shown in Figure 7.9. As can be seen, the additional annotated sentences obtained with the self-training method lead to considerably lower labeled F_1 scores in all cases. Performance decreases with growing amounts of additional training data, showing that the negative effect of annotation errors in the new instances outweighs the benefit of new information.

This result confirms that our definition of similarity and the corresponding projection approach cannot be replaced by a simple method of self-training, and are thus crucial for the success of our method and the improvements shown in Section 7.4.

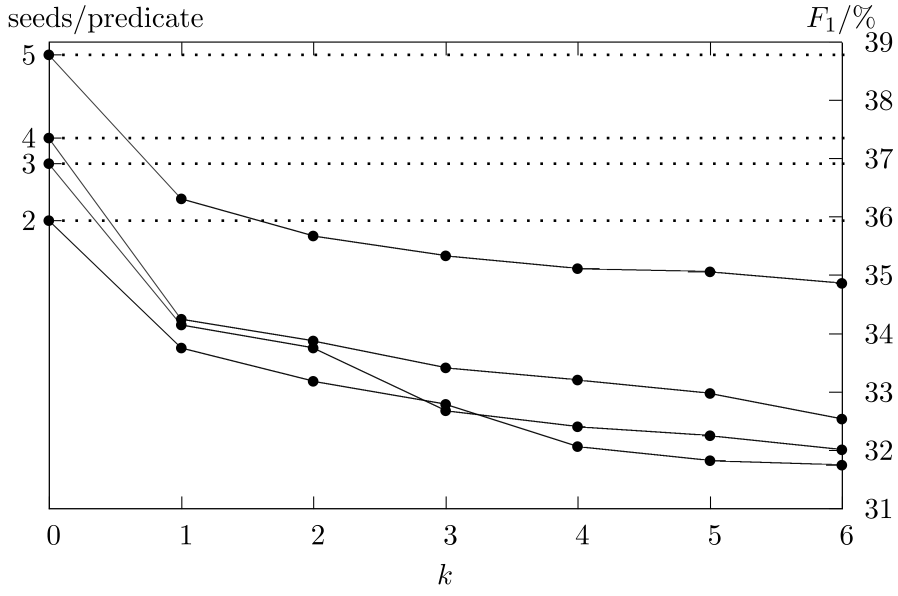


Figure 7.9: In the same set-up as shown in Figure 7.6, self-training leads to lower F_1 scores

7.6 Summary

In this chapter, we have described the set-up and reported on results of several experiments in which we applied our semi-supervised SRL approach to the task of labeling known predicates, i.e., predicates for which some manually annotated sentences are available.

We first described an experimental set-up to determine the extent to which our expansion algorithm improves labeling performance of a supervised SRL system. We then conducted experiments to find a suitable instantiation of our expansion framework described in Chapter 3. Comparing different definitions of our lexical similarity measure, we found distributional similarity to be superior to WordNet-based similarity measures. This surprising result was analysed theoretically and illustrated by different examples. Addressing the weight parameter α employed in our similarity score, we discussed a tuning procedure for it and the properties of an appropriate objective function. An optimal parameter value was determined by grid search, empirically confirming the intuition that both lexical and syntactic similarity contribute to the performance of our method. In experiments on various subsets of both the English FrameNet and the German SALSA corpus, we found that our method is able to significantly improve role labeling performance, especially for smaller corpora. These improvements were compared to the effect of additional manually annotated instances, showing that our approach can replace manual annotation to a substantial degree and therefore has the potential of reducing annotation effort in practice. Comparison with a simple self-training set-up showed that the formulation of our expansion framework is crucial for this result.

In the following chapter, we will address the complementary task of labeling predicates without any labeled training data, and apply our expansion algorithm to generate training instances for them.

Chapter 8

Semi-supervised Learning for Unknown Predicates

We still do not know
one thousandth of one percent
of what nature has revealed to us.

Albert Einstein

In this chapter, we describe a second set of experiments in which we apply our semi-supervised SRL approach to acquire novel instances for *unknown predicates*, i.e., predicates for which no manually labeled instances are available. Such predicates present a major problem for existing supervised SRL systems, which due to the lack of specific training material perform poorly on them, or are even unable to predict frames and roles at all.

We first address the problem of finding *frame candidates* for an unknown predicate, proposing two different approaches based on methods from the literature. We then empirically evaluate our expansion method on the tasks of frame and role labeling of unknown predicates, drawing on data from the English FrameNet and the German SALSA corpus.

8.1 Frame Candidates

In the application of our expansion algorithm in Chapter 7, we compared each unlabeled sentence only to seed sentences featuring the same predicate. This is not strictly necessary in our framework, but reduces its computational cost considerably, and ensures higher precision than allowing the comparison of sentences with arbitrary predicates. In the acquisition of instances of unknown predicates, however, this simplification is no longer possible, as there are no annotated instances of these predicates in the seed corpus. While this means that we now have to compare predicate-argument structures of different verbs, the vast majority of seeds will still be inappropriate for a given unlabeled sentence, as two arbitrary predicates are likely to pertain to enirely different situations. To maintain high precision, and also to make expansions computationally feasible, we therefore employ a *filtering stage*, determining which seeds may possibly be relevant for a given unknown predicate type. Only those sentence pairs passing this initial filter will then be further considered by the graph alignment algorithm.

We could address the problem by comparing the lemmata of the two predicates, e.g., measuring their lexical similarity. On the other hand, the Frame Semantic annotation of the seed sentences allows us to compare unknown predicates to the *frames* evoked by known predicates, taking advantage of the disambiguating information provided in the manual annotation. We follow this approach and determine a set of *frame candidates* for each unknown predicate. An unlabeled sentence is then compared only to seed sentences annotated with one of the frame candidates of the unknown predicate.

Several methods have been proposed to address the coverage problem of Frame Semantic resources and infer the possible frames of an unknown predicate. Burchardt et al. (2005) propose a rule-based system making use of WordNet information, while Johansson and Nugues (2007b) train classifiers to predict whether a predicate may evoke a particular frame or not, also employing WordNet information. In this section, we focus on the work of Pennacchiotti et al. (2008), who present two methods, one based on WordNet and one that does not make use of lexical resources and is thus more portable across languages and domains. We describe these two methods and propose variants of them, which show superior performance in a type-based evaluation procedure. In the subsequent sections, the frame candidates predicted by these methods will then be employed in the application of our expansion algorithm.

Vector-based Methods. Pennacchiotti et al. (2008) propose a vector-based method for the derivation of frame candidates. The vector space model they use is similar to the one we defined in Section 4.1. We therefore employ this model to represent predicates by co-occurcence vectors and compute their cosine similarity. Their method first computes a frame vector $\vec{f}$ for each FrameNet frame f as the *weighted centroid* of the vectors $\vec{v}$ for each predicate v that may evoke f:

$$\vec{f} = \sum_{v \in f} w_{v,f} \vec{v} \tag{8.1}$$

They define the weight $w_{v,f}$ as the relative frequency of v among the predicates evoking f, counted over the same corpus used to build the vector space model (which in our case is the unlabeled expansion corpus). This allows

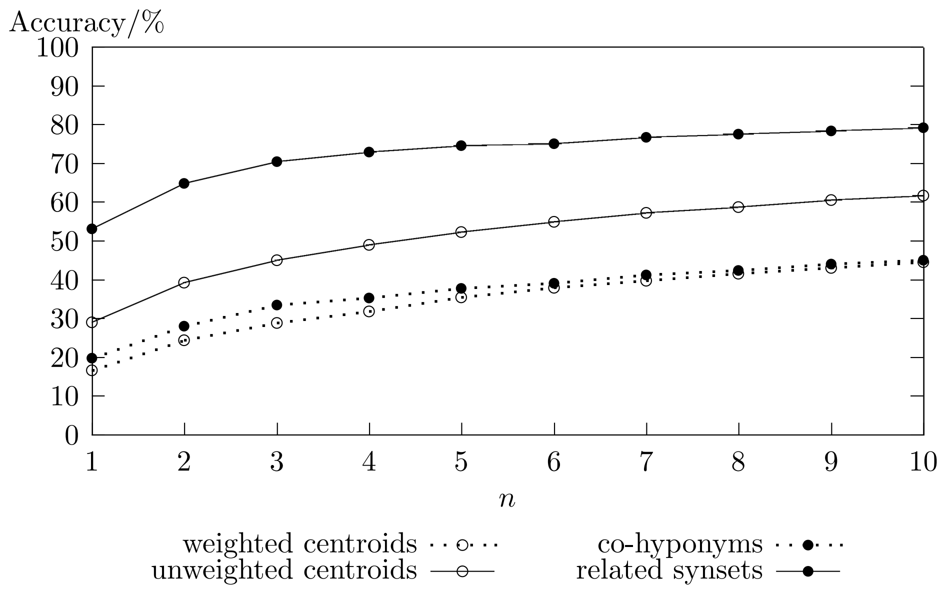


Figure 8.1: Accuracy out of n for different frame candidate methods. Unbroken lines show our methods, and solid dots indicate use of WordNet information.

more frequent predicates to exert a stronger influence on the frame vector. As a variant of this definition, we also consider *unweighted centroids*, for which we set all $w_{v,f} = 1$.

To derive frame candidates for an unknown predicate v_0, represented by a vector $\vec{v}_0$, all FrameNet frames f are ordered by their cosine similarity $\cos(\vec{v}_0, \vec{f})$ to v_0. The most similar n frames are considered as frame candidates.

WordNet-based Methods. As an alternative to their vector-based method, Pennacchiotti et al. (2008) also propose a method based on WordNet. It treats nouns, verbs, and adjectives differently, but here we are only interested in verbal predicates. Instead of using WordNet similarity measures, which did not show convincing performance in Johansson and Nugues (2007b), their method measures similarity between an unknown predicate v_0 and a frame f by counting the number of co-hyponyms of v_0 which may evoke f. (The co-hyponyms of a WordNet synset are the other hyponyms of its hypernyms, i.e., the “sister nodes” in the WordNet graph.) It then considers f a frame candidate for v_0, if this number is greater than a parameter τ, set to $\tau = 2$ in their experiments.[1]

Following up on their basic idea, we propose an extension based on counts

[1]personal communication

of synonyms, hypernyms, hyponyms and co-hyponyms in WordNet. We define the WordNet-based similarity of the unknown predicate v_0 to the frame f as:

$$\mathrm{sim}_W(v_0, f) = \sum_{v \in f} r(v_0, v) \tag{8.2}$$

where $r(v, v')$ is 1 if v and v' are synonyms , 0.5 if one is a hypernym of the other, 0.25 if they are co-hyponyms, and 0 otherwise. These numbers were chosen heuristically to represent different degrees of relatedness according to WordNet. Again, the most similar n frames are considered as frame candidates. We found that giving positive scores to pairs related more distantly than via co-hyponymy did not improve performance, probably because the verb hierarchy in WordNet is rather shallow and relatively unrelated concepts are included in this case. It therefore seems unlikely that much could be gained from refining the measure r, e.g., by incorporating one of the WordNet similarity measures discussed in Section 4.2.

Evaluation. We evaluate the four methods in a leave-one-out procedure over the verbal FEEs of FrameNet (excluding the 20% of the verbs which will be considered "unknown" in the experiments of the following sections, so as not to bias the evaluation of those experiments). Leaving out one of those predicates in turn, we predict frame candidates for it based on information about the evokable frames of all other predicates. The resulting candidate set is then compared with the set of true evokable frames according to the frame lexicon.[2]

Figure 8.1 shows evaluation results for the two vector-based methods "weighted centroids" and "unweighted centroids", as well as the WordNet-based methods "co-hyponyms" (for the original method) and "related synsets" (for our extension considering a wider range of related synsets). It shows the proportion of the tested predicates for which at least one of the frame candidates is among the true evokable frames, considering sets of $n = 1, \ldots, 10$ frame candidates. We can see that the "weighted centroids" method performs worst, ranging from 16.6% for 1 candidate to to 44.5% for sets of 10 candidates per predicate. Interestingly, performance increases by a large margin when *unweighted centroids* are considered instead of weighted ones. Accuracy increases by about 15%, ranging from 29.1% for 1 candidate to 61.6% for 10 candidates. Apparently, the stabilizing effect of the centroid computation, which allows common meaning aspects of the predicates to reinforce each other and reduces the effect of spurious word senses, is more pronounced when all predicates are weighted equally. The figure further shows that the original WordNet-based method "co-hyponyms" performs

[2]This evaluation is very similar to the one in Pennacchiotti et al. (2008). Their higher numerical results are due to a different test set, which includes FEEs of all parts of speech, not only verbs. Moreover, they exclude infrequent predicates.

poorly in our experiments. It only slightly outperforms even the "weighted centroids" method, and does not reach the level of performance shown by the "unweighted centroids" method, which does not have access to WordNet information. This result suggests that the reported improvements of the WordNet-based method in Pennacchiotti et al. (2008) are due to their more refined treatment of nouns, which are not considered in our experiments. Finally, our WordNet-based method "related synsets" shows best performance, leading to large improvements over the three other methods. Accuracy grows from 53.1% for 1 candidate to 79.2% for 10 candidates. In the experiments of the following sections, we will therefore consider the vector-based method using unweighted centroids and our extended WordNet-based method to derive frame candidates for unknown predicates.

8.2 Evaluation Procedure

To simulate frame and role labeling for unknown predicates, we divide all annotated predicates into two sets, one of which is considered "known" and the other "unknown". Annotated sentences of unknown predicates serve as testing data, while annotation instances of known predicates are used as the seed corpus. We apply our expansion algorithm to generate novel instances for the unknown predicates, thus complementing the seed corpus. To assess the effect of our semi-supervised approach, we again compare the performance of the SRL system from Chapter 6 when trained on this augmented training set to its performance when trained on the original seed corpus alone. In the first case, the SRL system will be able to profit from specific training instances for the predicates in the test set, while in the second case performance will depend solely upon its ability to generalize from known to unknown predicates. We will use the same choices of lexical similarity measure (lex_{cos}) and weight parameter ($\alpha = e^{-0.6}$) that were found to be optimal for the experiments in Chapter 7.

To divide the predicates of the FrameNet and SALSA corpora into "known" and "unknown", we first sort them by their number of annotation instances. We then mark every fifth predicate in that list (i.e., 20% of the predicates) as "unknown". The remaining predicates constitute the "known" set. For FrameNet, we thus obtain 400 unknown predicates and for SALSA 98 unknown predicates. While this form of sampling is quasi-random, our procedure additionally ensures that the sets of unknown predicates show a balanced distribution over annotation frequencies.

The major challenge in labeling instances of unknown predicates is the identification of correct *frames* for them. We will therefore focus on the evaluation measure of frame labeling accuracy. Nonetheless, we have to assess whether the SRL system is also able to accurately identify the roles of correctly predicted frames. As discussed in Section 6.4, the most appropriate

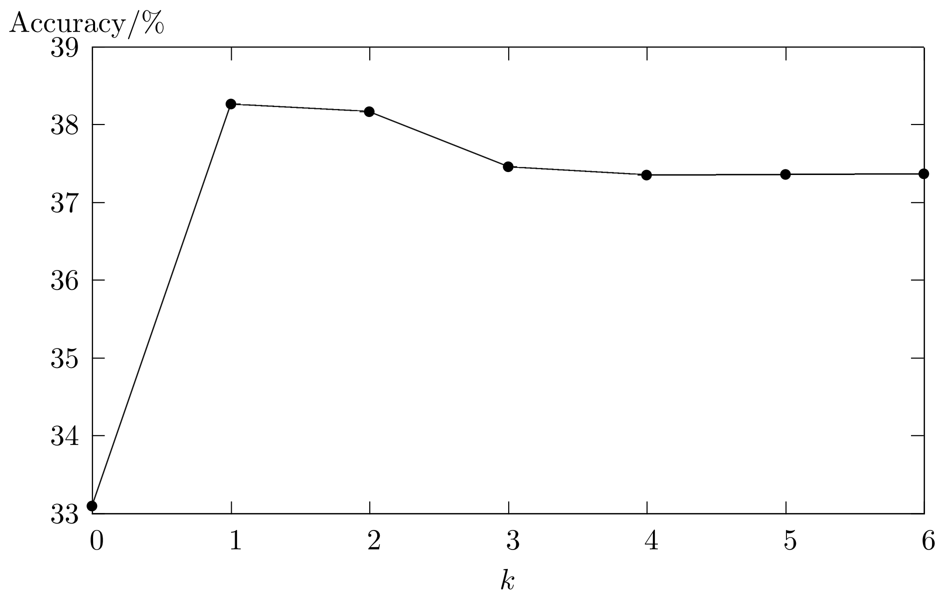


Figure 8.2: Frame labeling accuracy resulting from different values for the parameter k, which determines the expansion size

evaluation measures for this task are labeled precision, labeled recall, and labeled F_1 score, factoring in both frame and role labeling performance. We will therefore report role labeling performance in terms of these measures.

8.3 Experiments on English

For our experiments on the English FrameNet data, we split all verbal FEEs occuring in FrameNet into "known" and "unknown" sets according to the procedure described in Section 8.2. We exclude all sentences of the development corpora used in Sections 7.2 and 7.3. The remaining sentences of unknown predicates constitute our test set (8,415 sentences), while those of known predicates make up the seed set (34,051 sentences). As expansion corpus, we again use the entire BNC, parsed with RASP. In the following, we separately consider the two frame candidate methods which have been found to perform well in Section 8.1, one based on a distributional vector space and one relying on WordNet information.

Vector-based Frame Candidates. We first examine the influence of the parameter k of the expansion algorithm, which determines the number of generated training instances per seed sentence. We consider a simple set-up in which the expansion algorithm is provided with 2 frame candidates per predicate, derived by the vector-based method. We then vary the parameter k and add the resulting generated instances to the seed corpus, producing training sets of different sizes for the SRL system. The resulting performance of the SRL system on the test set, choosing between the given 2 frame candidates for each predicate, is shown in Figure 8.2. Here, $k = 0$ indicates the performance when training on the seed corpus alone, while for $k \geq 1$ the performance resulting from the expanded training sets is shown. We can see that the additional training data leads to substantial improvements in all cases. The following example shows how a seed sentence with the verb *zigzag* gives rise to annotations for the verbs *snake*, *scuttle*, and *tumble*, which are considered unknown in our experiment:

(seed) [Flies]$_{\textit{Theme}}$ [**zigzagged**]$_{\textsc{Motion}}$ [across the room]$_{\textit{Path}}$, speeding about their business like bees in a swarm.

(1) [A rubber hose-pipe]$_{\textit{Theme}}$ [**snaked**]$_{\textsc{Motion}}$ [across the yard from the kitchen window]$_{\textit{Path}}$, bringing hot water from the tap in the big sink.

(2) [He]$_{\textit{Theme}}$ [**scuttled**]$_{\textsc{Motion}}$ [across the rich thick carpet like a toddler]$_{\textit{Path}}$, making her laugh.

(3) Still locked together [they]$_{\textit{Theme}}$ [**tumbled**]$_{\textsc{Motion}}$ [across the room]$_{\textit{Path}}$.

Here, the frame annotation of the first neighbour is incorrect: the correct frame would have been PATH_SHAPE. The other two neighbours, however, provide accurate novel annotation data for the unknown verbs *scuttle* and *tumble* (except for the phrase *"like a toddler"* which should not be part of the role *Path*).

Of course, the neighbours of a seed need not feature different verbs. In the following example, the three nearest neighbours all exemplify the unknown verb *fall*, for which the two frame candidates CHANGE_POSITION_ON_A_SCALE and MOTION_DIRECTIONAL were determined:

(seed) [...] [his glass]$_{\textit{Theme}}$ [**toppled**]$_{\textsc{Motion\_directional}}$ [off the table]$_{\textit{Source}}$ and shattered.

(1) But it really made yet more [hair]$_{\textit{Theme}}$ [**fall**]$_{\textsc{Motion\_directional}}$ [off the top of his bald head]$_{\textit{Source}}$.

(2) Not only had £250,000 [worth of equipment]$_{\textit{Theme}}$ [**fallen**]$_{\textsc{Motion\_directional}}$ [off the back of a submarine]$_{\textit{Source}}$, [...]

(3) [...] even if [the VW badge]$_{\textit{Theme}}$ has [**fallen**]$_{\textsc{Motion\_directional}}$ [off the car]$_{\textit{Source}}$ [...]

The seed sentence here gives rise to three correct instances of the frame MOTION_DIRECTIONAL as training data for the unknown verb *fall*. The role annotation only contains minor errors (leaving out *"yet more"* and *"£250,000"*).

While Figure 8.2 shows improvements for all values of $k \geq 1$ relative to the unexpanded training set, the largest gain is obtained for $k = 1$. This indicates that the optimal trade-off between quantity and accuracy of the new instances is slightly shifted compared to the experiment on known predicates, which showed optimal performance for values of k between 2 and 4 (see Figure 7.6 on page 106). This is not surprising, considering that now labeled and unlabeled sentences are in general less similar to each other by virtue of featuring different predicates. A high level of accuracy can then be restored by resorting only to new instances with high similarity values, corresponding to high confidence in their annotation.

With the optimal value of $k = 1$, we now compare the performance of the *unexpanded classifier*, i.e., the SRL system trained only on the seed corpus, and the *expanded classifier*, i.e., the same system trained on the augmented training set, for various numbers of frame candidates. In each case, the expansion algorithm is provided with a number of candidates per predicate, and the SRL system is then employed to choose between the *same frame candidates*. These candidates thus replace the frame lexicon information, which was available to the SRL system in our earlier experiments on known predicates.

Figure 8.3(a) shows frame labeling accuracy on the test set for different numbers of frame candidates. Performance of the expanded classifier is compared to that of the unexpanded classifier. In addition, we show a baseline, which randomly chooses one of the frame candidates for each test sentence, and the upper bound, which quantifies the proportion of the test sentences where the correct frame can be found among the frame candidates at all. We can see that the unexpanded classifiers outperforms the random baseline by a wide margin. This shows that the SRL system is indeed able to generalize to unknown predicates, even without specific training data. However, the expanded classifier in turn performs significantly better than the unexpanded one for all numbers of frame candidates ($p < 0.001$)[3]. The special case of 1 candidate shows the performance of type-based frame labeling, which always assigns the first frame candidate, independent of sentential context. Of course, both classifiers coincide in this case. However, while the unexpanded classifier does not improve over this baseline, the expanded one outperforms it for 2, 3, and 4 candidates. For 2 and 3 candidates the improvement is statistically significant ($p < 0.001$ and $p < 0.05$, respectively).

[3]Throughout this chapter, statistical significance of improvements in frame labeling accuracy is determined by McNemar's test, while for labeled F_1 scores stratified shuffling (Noreen, 1989) is employed.

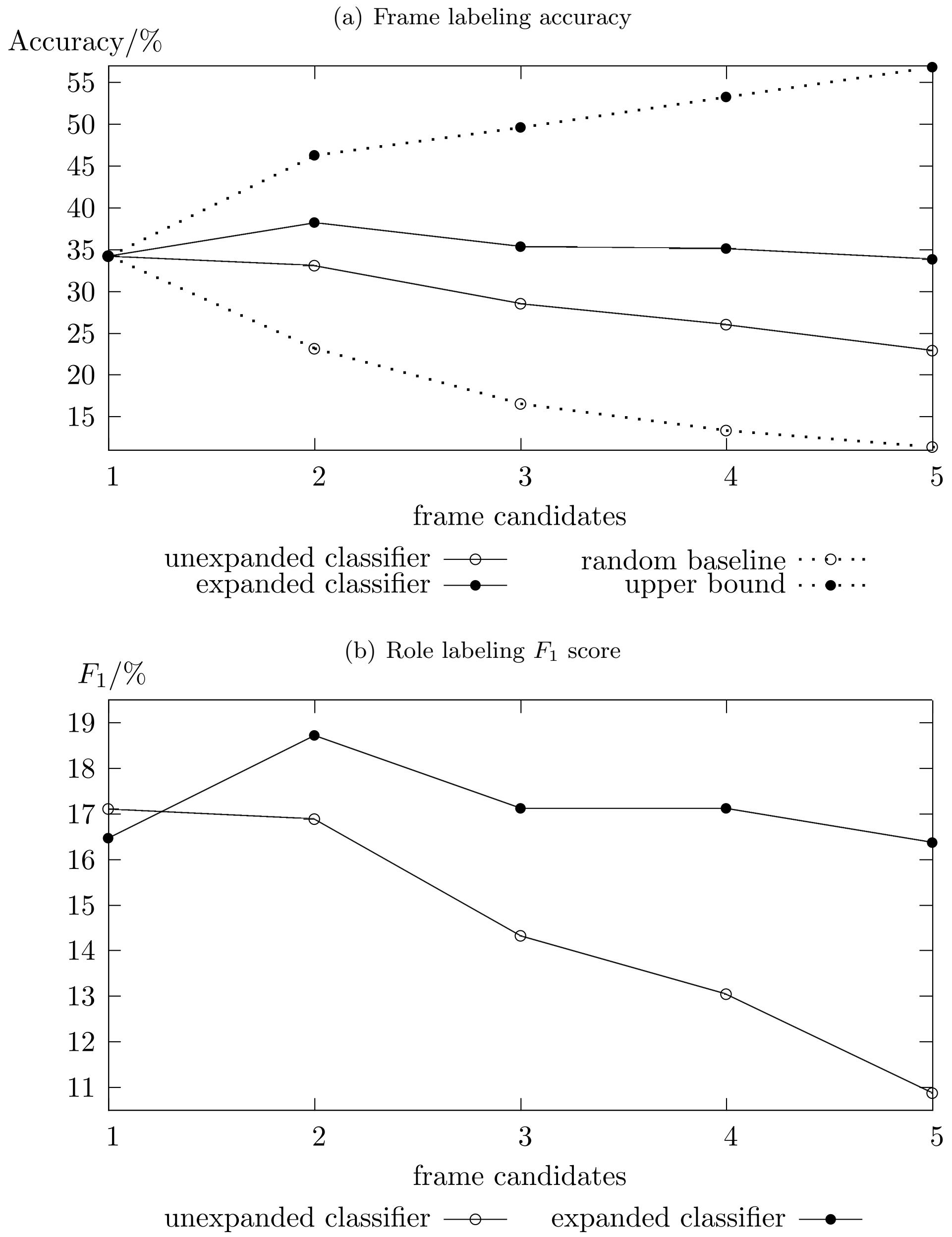


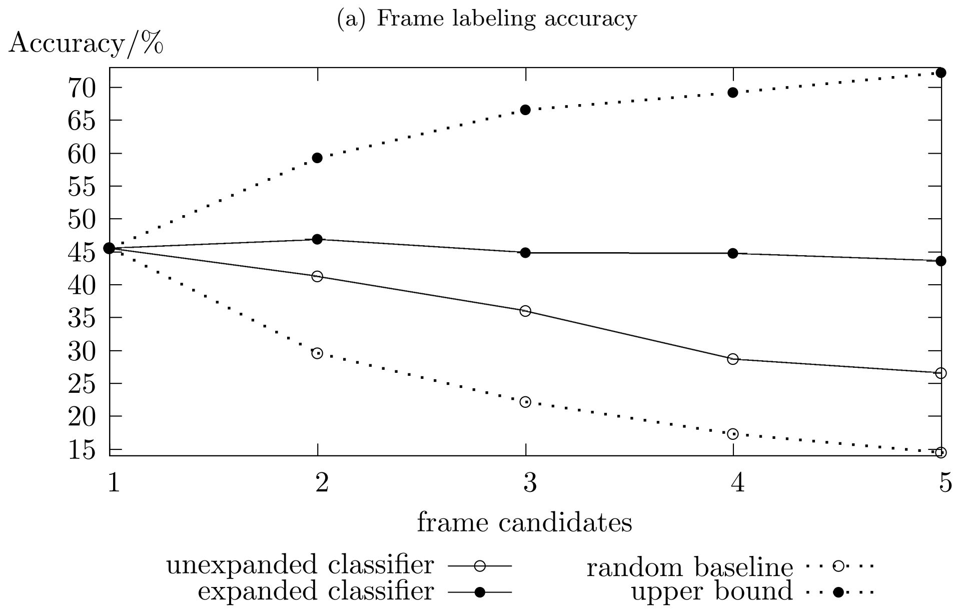
Figure 8.3: Results of expanded vs. unexpanded classifiers on FrameNet data, choosing among different numbers of frame candidates produced by the *vector-based method*. For frame labeling, random baseline and upper bound performance are shown.

This means that the additional training material enables the classifier to successfully favour lower scoring candidates over higher scoring ones based on sentential context.

Figure 8.3(b) shows role labeling performance in terms of labeled F_1 scores for the expanded and unexpanded classifiers. (There is no meaningful random baseline for the task of predicting role spans and labels. Any textual span could be annotated with any role, so that the probability of a correct random guess is typically close to 0.) It can be seen that the expanded classifier again outperforms the unexpanded one. Only in the artificial case of 1 candidate it produces slightly lower results. Moreover, for 2 candidates it also significantly outperforms the first-candidate baseline ($p < 0.001$). This shows that the expanded classifier is not only able to correctly choose lower scoring frame candidates for unknown verbs, but also to accurately label their roles.

The full set of numerical results of this experiment can be found in the appendix. The overall scale of the labeled F_1 scores may seem rather low. This is due to both the difficulty of the task of predicting fine-grained sense distinctions for unknown predicates, and the comprehensive evaluation measure, which takes into account all three stages of the SRL system: frame labeling, role recognition and role classification.

WordNet-based Frame Candidates. To assess the level of performance attainable when WordNet information is available, we conducted a similar set of experiments with frame candidates produced by our WordNet-based method. The expansion parameter k was left at the value 1 determined previously. Results in terms of frame labeling accuracy and labeled F_1 scores for the role labeling task are shown in Figure 8.4. They confirm the expectation that the more accurate frame candidates taking into account WordNet information also improve the final frame and role labeling performance. While substantially higher in absolute terms, the evaluation qualitatively shows the same picture as in the case of the vector-based method. In terms of frame labeling accuracy, the unexpanded classifier again outperforms random choice of a frame candidate, and is in turn significantly outperformed by the expanded classifier for all numbers of candidates ($p < 0.001$), except in the trivial case of 1 candidate. The type-based first-candidate baseline is significantly exceeded by the expanded classifier working on 2 frame candidates per predicate ($p < 0.05$). Results in terms of labeled F_1 scores also confirm the earlier results, showing significant improvements of the expanded classifier over both the unexpanded one ($p < 0.001$ for all numbers of candidates ≥ 2) and the first-candidate baseline ($p < 0.05$ for 2 frame candidates). These results show that, where available, information from a taxonomy like WordNet can substantially increase labeling performance of unknown predicates in our semi-supervised approach.



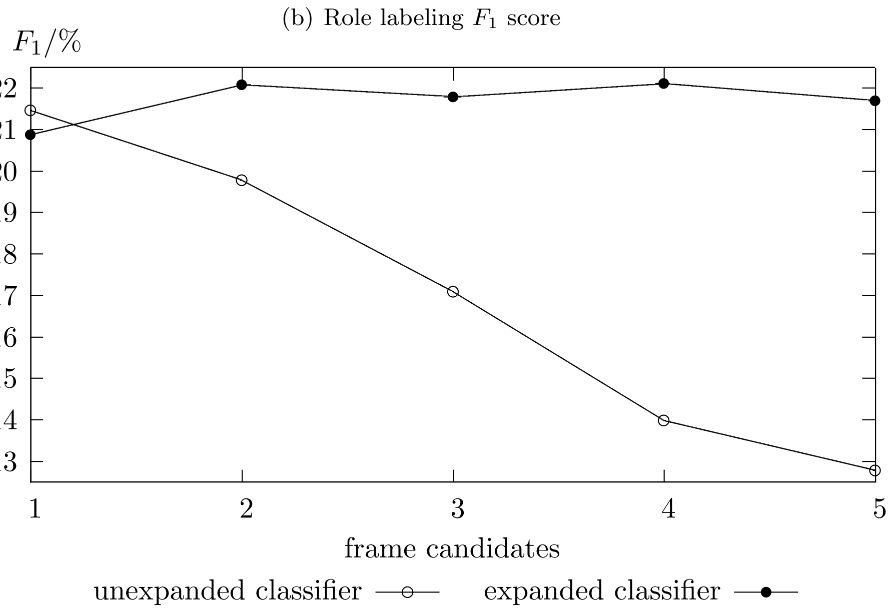


Figure 8.4: Results of expanded vs. unexpanded classifiers on FrameNet data, choosing among different numbers of frame candidates produced by the *WordNet-based method*. For frame labeling, random baseline and upper bound performance are shown.

8.4 Experiments on German

In this section we conduct a similar suite of experiments on the German SALSA corpus. Dividing the annotation instances of the SALSA corpus into known and unknown predicates according to the procedure described in Section 8.2, we obtain a test set with annotations of unknown predicates (4,532 sentences) and a seed corpus with those of known predicates (14,962 sentences). As unlabeled expansion corpus we again use the "Süddeutsche Zeitung" newspaper corpus.

We first discuss the specific problems of *proto-frames* and then show results of expanded and unexpanded classifiers choosing between frame candidates produced by the vector-based method. We also experimented with our WordNet-based frame candidate method, making use of GermaNet (Hamp and Feldweg, 1997), which provides a taxonomy like that of WordNet for German words. However, results did not significantly improve relative to the vector-based method.

Proto-frames. In contrast to FrameNet, the SALSA project followed a more data-driven annotation policy and did not choose representative example sentences, but rather exhaustively annotated *each occurence* of a number of verbs in the TIGER treebank. As a consequence of this decision, a large number of word senses was encountered that are not yet covered by the frame lexicon defined by FrameNet. For example, the German verb *kümmern* can evoke the FrameNet frame EXPERIENCER_OBJ, which according to FrameNet describes a situation in which "some phenomenon (the *Stimulus*) provokes a particular emotion in an *Experiencer*". This is exemplified in:

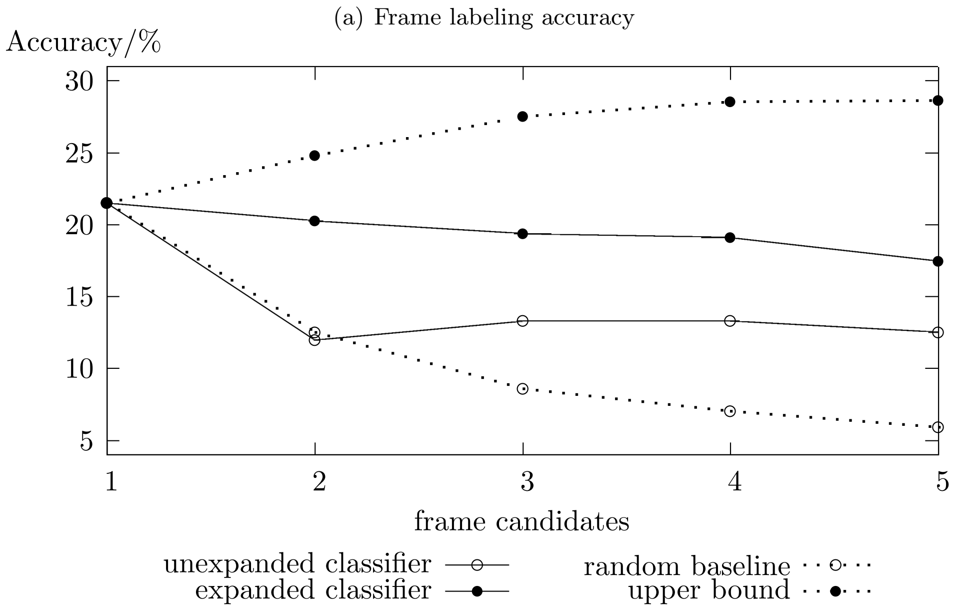
(8.3) [Die Zweifler]$_{Stimulus}$ [**kümmerten**]$_{\text{EXPERIENCER\_OBJ}}$ [uns]$_{Experiencer}$ nicht mehr.

[The doubters]$_{Stimulus}$ [**worried**]$_{\text{EXPERIENCER\_OBJ}}$ [us]$_{Experiencer}$ no more.

However, the verb *kümmern* can also have a sense not covered by any existing FrameNet frame. In the annotation of the SALSA corpus, a proto-frame was created for this and preliminarily called KUEMMERN-SALSA1. The following sentence shows an example:

(8.4) [Die Friedensvermittler]$_{Agent}$ müssen sich mehr [um die großen menschlichen Probleme]$_{Theme}$ [**kümmern**]$_{\text{KUEMMERN-SALSA1}}$.

[The peace mediators]$_{Agent}$ must more [of the big humanitarian problems]$_{Theme}$ [**take care**]$_{\text{KUEMMERN-SALSA1}}$.



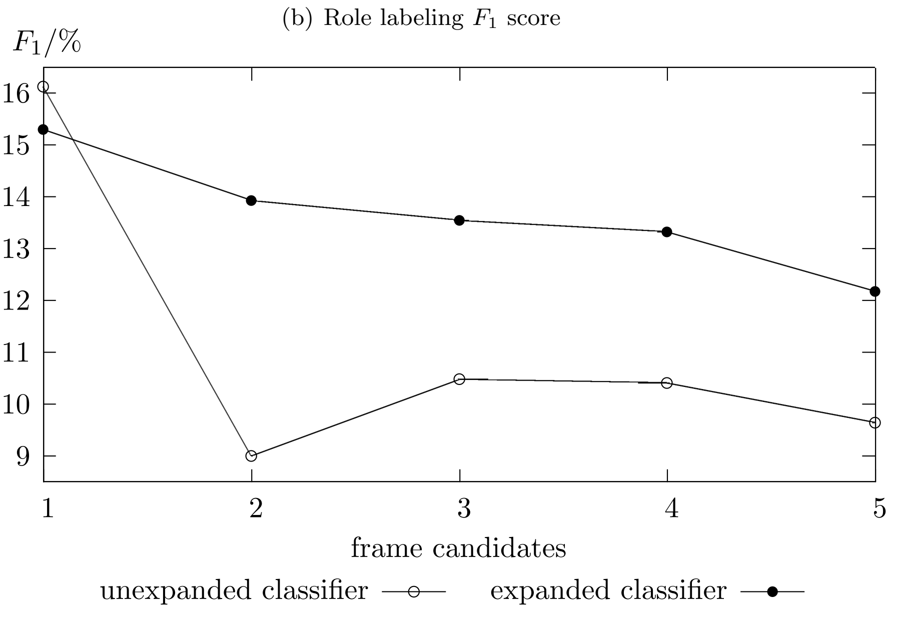


Figure 8.5: Results of expanded vs. unexpanded classifiers on SALSA data, choosing among different numbers of frame candidates produced by the vector-based method. For frame labeling, random baseline and upper bound performance are shown.

While representing general concepts, such proto-frames have not yet been generalized across predicates. In our example this means that *kümmern* is the only predicate evoking the KUEMMERN-SALSA1 frame. This shortcoming of proto-frames, which can be evoked by 237 of the 492 verb types in the SALSA corpus and make up 38% of the annotated sentences, renders them inappropriate for the experiments in this chapter. Our method has no chance of acquiring annotation instances for proto-frames evoked by an unknown predicate, since these are never exemplified on any other predicate, and in particular not on any known predicate. In deriving frame candidates, we therefore do not consider any proto-frames, as they can never lead to correct annotations in our evaluation.

Results. Figure 8.5 shows results of our approach on the SALSA test set. We can see that the unexpanded classifier here performs relatively poorly, not even exceeding the random baseline on 2 candidates. The expanded classifier performs significantly better for all numbers of candidates ($p < 0.001$), except for the trivial case of 1 candidate. However, it is not able to improve over type-based classification, which always assigns the first frame candidate. As the upper bound shows, considering the first two frame candidates, instead of only the first one, increases the proportion of test sentences for which the classifier could possibly find the correct frame by about 6%. On the FrameNet corpus, the corresponding increase in the upper bound was about 12%. For larger numbers of frame candidates, the upper bound also levels off quickly, showing that further candidates are to a large extent incorrect.

The difference in the quality of frame candidates can be explained by the different structure of FrameNet and SALSA. In FrameNet, 2,113 different verbs evoke a total of 362 frames, i.e., each frame is associated with an average of 5.8 verbs. In SALSA, on the other hand, there are only 503 annotated verbs, whose senses distribute over 678 frames. This means that each frame on average is associated with only 0.7 verbs. Even disregarding the 444 proto-frames, this ratio is still only 2.1 and thus much lower than for FrameNet. Accordingly, for more than 23% of the frames evoked by unknown verbs (excluding proto-frames) there is no known verb exemplifying this frame, compared with only about 8% for FrameNet. On the other hand, more than 25% of the frames evoked by unknown FrameNet verbs have 10 or more known verbs exemplifying them, but for SALSA this is only the case for about 13% of such frames. This shows that due to the different annotation procedure, the potential for generalization to unknown predicates is much lower in the SALSA corpus than in the FrameNet corpus.

8.5 Summary

In this chapter, we have applied our semi-supervised SRL approach to the challenging task of labeling the frames and roles of unknown predicates. We addressed the problem of predicting frame candidates for unknown predicate types by adaptation and extension of existing methods from the literature. A method based on distributional similarity and one based on WordNet-similarity were chosen for the subsequent experiments. We then applied the expansion algorithm from Chapter 3 to automatically generate annotation instances for a number of unknown predicates from the FrameNet and SALSA corpora. Evaluation showed that those instances significantly improve the ability of an SRL system to choose between frame candidates for predicates which otherwise are not exemplified by any training instances. Similar improvements were observed in both frame and role labeling performance. In the case of FrameNet, our semi-supervised SRL method also achieved significant improvements over a type-based approach, which always chooses the first frame candidate independently of sentential context. On the SALSA corpus, however, we could not improve upon this dominant word sense. We analyzed the difference in the experimental outcomes and concluded that the success of our semi-supervised approach on unknown predicates depends on the diversity of the predicates in the seed corpus. This casts doubt on the annotation strategy of exhaustively labeling all instances of a relatively small number of predicates. As we have shown in the previous chapter, additional instances for known predicates can be successfully obtained by semi-supervised learning. Annotation effort is therefore much better spent on a small number of instances spread over as wide a range of predicates as possible.

Chapter 9

Conclusions

In this final chapter, we summarize the theoretical and practical contributions of the thesis. We then discuss a few directions for further research, some of which have already been mentioned in the previous chapters.

9.1 Contributions

In this thesis, we have presented a novel semi-supervised approach to semantic role labeling (SRL). Many state-of-the-art SRL systems, following the supervised learning paradigm, depend on large corpora with manual role annotations for good performance. The creation of such corpora requires substantial annotation efforts, which moreover have to be repeated whenever the system is applied to a new language or domain. Our semi-supervised approach, on the other hand, improves the performance of supervised SRL systems by automatically inferring additional training data from an unlabeled corpus. Consequently, it reduces the annotation effort required to attain satisfactory levels of performance.

We have described an algorithm that annotates selected sentences from a large unlabeled corpus based on their similarity to sentences from a small manually labeled seed corpus. Empirical evaluation showed that these novel annotation instances improve SRL performance on predicates for which only a few or no manually labeled example sentences are available.

Similarity-based Expansion Algorithm. The expansion algorithm we presented is based on a similarity measure between predicate-argument structures, represented by syntactic dependency graphs. We have derived a general form for this measure in terms of optimal graph alignments and shown how to formulate and efficiently solve an integer linear program for the optimization problem. Our framework features a lexical similarity measure, quantifying the similarity of words, and a syntactic similarity measure, taking into account structural similarity. This formulation easily allows further research to investigate how the individual similarity measures can be improved. For our experiments, we have shown how to instantiate the framework with measures following common approaches from the literature. We have also addressed the computational complexity of these similarity measures. Specifically, we presented efficient algorithms on sparse vector representations of distributional and WordNet-based similarity.

Improvements in SRL Performance. We have evaluated our semi-supervised approach in a number of different experimental settings. For predicates with a few existing labeled example sentences, we have shown that additional instances generated by the expansion algorithm significantly improve labeling performance of an SRL system. Improvements have been shown to correspond to the effect of 1 or 2 manually labeled instances per predicate for the relatively large English FrameNet corpus, and up to 10 and more labeled instances per predicate for the smaller German SALSA corpus. These results highlight that our approach is especially effective on small resources. We have also addressed the problem of predicates missing from the annotation corpus, which pose a large problem to supervised SRL systems. Our semi-supervised method significantly improves labeling performance for such predicates over a classifier that has to generalize solely from instances of other predicates. On the FrameNet corpus, we also observed significant improvements over a type-based approach, which always labels predicates with their dominant sense. For SALSA, the potential for generalization across predicates was found to be lower, and choosing the dominant sense could not be improved upon. We concluded that annotation effort is best spread over a large number of different predicates, each exemplified by a small number of instances.

9.2 Future Work

Future directions to the research presented in this thesis are many and varied. One straightforward extension would be to replace the measures of lexical and syntactic similarity. However, considering the comparatively poor performance of the WordNet-based lexical similarity measure discussed in Section 7.2, it cannot be assumed that measures performing well on other tasks will automatically be suitable for use in the framework presented here. We have already mentioned that a more advanced model of selectional preference might be able to better capture the kind of information required for measuring lexical similarity in our approach. This raises the research question of how selectional preferences can be translated into argument similarity.

Likewise, the syntactic similarity measure offers ample room for future research. As discussed in Section 4.3, it is unlikely that a syntactic representation based on a small number of dependency labels will allow the definition of a measure that is significantly better than our simple binary definition. However, since the expansion framework is neutral as to the type and number of edge labels in the dependency graphs, syntactic formalisms with more fine-grained distinctions, such as head-driven phrase structure grammar or combinatory categorial grammar, could be employed. It should be interesting to investigate whether such syntactic categories can be leveraged for a more advanced measure of syntactic similarity.

Another avenue for future research would be the combination of our approach with methods of cross-lingual annotation projection, which infer annotations for a target language by projection from semantically annotated corpora of a source language. For languages without any role semantic resource, initial annotations could be created by cross-lingual projection. As the size of the generated corpus would be limited by the availability of suitable bi-texts, our semi-supervised method could be applied to infer additional annotation instances from an unlabeled corpus in the target language, and thus improve the utility of the resource to an SRL system.

As our method produces novel annotated sentences, it is also suitable for combination with active learning approaches. In this context, annotation effort could be reduced by offering automatically pre-annotated sentences to humans, who then only have to check and possibly correct them. The selection of sentences to be presented to the annotators could be based on annotation confidence as quantified by our similarity measure.

The general formulation of our expansion framework allows its application to other tasks. Deschacht and Moens (2009) employ a simpler version of it to augment subsets of the PropBank corpus, observing improvements over a supervised system for a small seed corpus. They also show that defining the lexical similarity measure in terms of Jensen-Shannon divergence instead of cosine similarity can improve performance. Another possibility would be to employ our framework in paraphrase acquistion, e.g., extending the multiple sequence alignment approach of Barzilay and Lee (2003) with our notion of graph alignments.

Finally, for the application of our method to resource-poor languages, it would be interesting to investigate how to reduce the dependence on full syntactic analyses, e.g., by employing shallow parsers or chunkers and defining an appropriate syntactic similarity measure. Here, the resulting impact on labeling performance will certainly also depend on the specific language.

Appendix: Complete Results

In this appendix, we give complete results for the expansion experiments of Chapters 7 and 8. Significance tests were performed for frame labeling accuracy, exact match (both by McNemar's test), and labeled F_1 score (by stratified shuffling according to Noreen (1989), using the `sigf` tool (Padó, 2006)). Additionally, we report labeled precision (P) and labeled recall (R). For simplicity, we only indicate two levels of significance, $p < 0.05$ with a single asterisk (*) and $p < 0.001$ with double asterisks (**).

Known Predicates, FrameNet

This table shows experimental results of adding from 1 to 6 automatically generated nearest neighbours (NN) to different seed corpora, containing from 1 to 10 manually labeled sentences per predicate.

Training set	Size	P/%	R/%	F_1/%	Ex. match/%
1 seeds/pred.	2,092	40.74	23.69	29.96	6.38
+ 1-NN	3,297	40.52	24.23	30.33	6.81 *
+ 2-NN	4,481	40.29	24.99	30.85 *	6.97 *
+ 3-NN	5,649	39.52	25.02	30.64 *	7.35 **
+ 4-NN	6,803	39.52	25.39	30.92 *	7.30 **
+ 5-NN	7,947	39.04	25.34	30.73 *	7.12 *
+ 6-NN	9,076	38.40	25.16	30.40	6.89
2 seeds/pred.	4,105	45.22	29.81	35.94	9.40
+ 1-NN	6,500	45.09	30.84	36.63 *	10.19 **
+ 2-NN	8,850	44.82	31.50	37.00 **	10.32 **
+ 3-NN	11,157	44.65	31.85	37.18 **	10.32 **
+ 4-NN	13,423	43.99	31.94	37.01 **	10.15 *
+ 5-NN	15,652	42.64	31.23	36.05	9.73
+ 6-NN	17,846	42.57	31.36	36.11	9.63

Known Predicates, FrameNet (continued)

Training set	Size	P/%	R/%	F_1/%	Ex. match/%
3 seeds/pred.	6,021	45.03	31.29	36.92	9.81
+ 1-NN	9,492	44.78	32.45	37.63 *	10.35 *
+ 2-NN	12,874	44.15	32.69	37.57 *	10.37 *
+ 3-NN	16,179	43.90	33.00	37.68 *	10.68 *
+ 4-NN	19,424	43.60	33.36	37.80 *	10.35
+ 5-NN	22,609	43.15	33.26	37.56 *	10.50 *
+ 6-NN	25,734	42.72	33.17	37.34	10.45 *
4 seeds/pred.	7,823	44.42	32.21	37.35	9.48
+ 1-NN	12,321	44.45	33.31	38.09 *	10.20 **
+ 2-NN	16,688	44.26	34.13	38.54 **	10.40 **
+ 3-NN	20,944	43.71	34.20	38.37 **	10.72 **
+ 4-NN	25,098	43.37	34.35	38.34 **	10.57 **
+ 5-NN	29,166	43.25	34.45	38.35 *	10.67 **
+ 6-NN	33,142	42.48	34.24	37.92	10.40 *
5 seeds/pred.	9,515	45.45	33.81	38.78	10.35
+ 1-NN	15,026	45.47	34.90	39.49 *	10.95 *
+ 2-NN	20,363	45.03	35.39	39.63 *	11.42 **
+ 3-NN	25,533	44.56	35.51	39.53 *	11.56 **
+ 4-NN	30,576	44.44	35.78	39.64 *	11.70 **
+ 5-NN	35,494	44.22	35.94	39.65 *	11.72 **
+ 6-NN	40,286	43.74	35.83	39.39 *	11.49 **
6 seeds/pred.	11,105	46.50	35.44	40.22	10.95
+ 1-NN	17,553	46.05	36.11	40.48	11.56 *
+ 2-NN	23,779	45.71	36.67	40.70	12.07 **
+ 3-NN	29,787	45.16	36.83	40.57	11.92 **
+ 4-NN	35,623	44.82	36.92	40.49	11.80 *
+ 5-NN	41,310	44.60	36.91	40.40	12.13 **
+ 6-NN	46,851	44.07	36.86	40.14	12.02 **
8 seeds/pred.	13,999	47.60	37.29	41.82	12.25
+ 1-NN	22,115	47.08	37.71	41.88	12.48
+ 2-NN	29,907	46.45	38.01	41.81	12.64
+ 3-NN	37,400	46.01	38.11	41.69	12.69
+ 4-NN	44,656	45.55	38.12	41.51	12.78
+ 5-NN	51,705	45.53	38.38	41.65	13.22 *
+ 6-NN	58,562	45.00	38.24	41.34	13.34 **
10 seeds/pred.	16,595	48.97	39.02	43.43	13.73
+ 1-NN	26,180	48.24	39.55	43.47	14.01
+ 2-NN	35,336	47.11	39.32	42.86	13.80
+ 3-NN	44,113	46.69	39.45	42.77	13.85
+ 4-NN	52,602	46.18	39.31	42.47	13.63
+ 5-NN	60,827	46.22	39.76	42.75	13.68
+ 6-NN	68,791	45.69	39.58	42.42	13.95

Known Predicates, SALSA

Here, we show experimental results of adding from 1 to 20 automatically generated nearest neighbours (NN) to different seed corpora, containing from 1 to 20 manually labeled sentences per predicate. Additionally, performance on the complete SALSA seed corpus is shown ("all seeds").

Training set	Size	$P/\%$	$R/\%$	$F_1/\%$	Ex. match/%
1 seed/pred.	485	47.30	13.64	21.18	4.93
+ 1-NN	881	52.91	19.34	28.33 **	7.80 **
+ 2-NN	1,277	55.59	22.58	32.11 **	10.31 **
+ 3-NN	1,671	55.81	24.53	34.08 **	12.01 **
+ 4-NN	2,065	56.79	26.09	35.76 **	13.08 **
+ 5-NN	2,458	55.88	26.80	36.23 **	13.55 **
+ 6-NN	2,851	56.20	27.77	37.17 **	13.96 **
+ 7-NN	3,244	57.13	28.08	37.65 **	14.21 **
+ 8-NN	3,637	56.65	28.50	37.92 **	14.57 **
+ 9-NN	4,030	56.45	28.67	38.03 **	14.73 **
+ 10-NN	4,423	55.74	28.76	37.94 **	14.52 **
+ 11-NN	4,816	55.60	28.56	37.74 **	14.42 **
+ 12-NN	5,208	55.86	28.93	38.12 **	14.83 **
+ 13-NN	5,600	55.14	28.76	37.80 **	14.83 **
+ 14-NN	5,992	55.34	29.10	38.14 **	15.24 **
+ 15-NN	6,384	55.48	29.55	38.56 **	15.19 **
+ 16-NN	6,776	55.34	29.55	38.53 **	15.50 **
+ 17-NN	7,167	55.61	29.67	38.69 **	15.55 **
+ 18-NN	7,557	55.56	29.92	38.89 **	15.85 **
+ 19-NN	7,947	55.56	30.03	38.99 **	15.75 **
+ 20-NN	8,337	55.99	30.09	39.14 **	15.70 **
5 seeds/pred.	2,008	56.63	32.81	41.55	15.96
+ 1-NN	3,623	59.95	39.14	47.36 **	20.57 **
+ 2-NN	5,225	59.84	41.32	48.88 **	21.86 **
+ 3-NN	6,816	59.99	42.82	49.98 **	23.40 **
+ 4-NN	8,401	60.40	44.38	51.17 **	24.88 **
+ 5-NN	9,981	60.14	44.75	51.32 **	25.19 **
+ 6-NN	11,556	60.49	45.55	51.97 **	25.24 **
+ 7-NN	13,129	60.21	46.00	52.15 **	25.40 **
+ 8-NN	14,699	60.51	46.77	52.76 **	25.65 **
+ 9-NN	16,267	60.30	46.65	52.61 **	25.71 **
+ 10-NN	17,831	60.50	47.08	52.95 **	26.37 **
+ 11-NN	19,388	60.70	47.45	53.26 **	26.83 **
+ 12-NN	20,944	60.72	47.79	53.48 **	26.78 **
+ 13-NN	22,496	60.18	48.01	53.42 **	26.63 **
+ 14-NN	24,045	60.60	48.07	53.61 **	26.78 **

Known Predicates, SALSA (continued)

Training set	Size	P/%	R/%	F_1/%	Ex. match/%
5 seeds/pred.					
+ 15-NN	25,588	59.87	47.90	53.22 **	26.73 **
+ 16-NN	27,129	59.74	47.82	53.12 **	26.42 **
+ 17-NN	28,667	59.55	47.73	52.99 **	26.58 **
+ 18-NN	30,199	59.58	47.87	53.09 **	26.42 **
+ 19-NN	31,728	59.87	47.99	53.27 **	26.73 **
+ 20-NN	33,254	59.83	48.07	53.31 **	26.89 **
10 seeds/pred.	3,517	58.39	40.87	48.08	21.04
+ 1-NN	6,270	62.19	46.23	53.03 **	25.60
+ 2-NN	8,993	62.95	49.55	55.45 **	28.12 **
+ 3-NN	11,688	62.59	50.68	56.01 **	29.25 **
+ 4-NN	14,358	62.26	51.19	56.19 **	29.60 **
+ 5-NN	17,011	62.34	51.64	56.49 **	29.50 **
+ 6-NN	19,651	63.30	52.69	57.51 **	30.48 **
+ 7-NN	22,282	63.11	52.98	57.60 **	30.48 **
+ 8-NN	24,900	62.43	52.78	57.20 **	30.27 **
+ 9-NN	27,510	62.76	53.06	57.51 **	30.63 **
+ 10-NN	30,109	62.69	53.23	57.58 **	30.73 **
+ 11-NN	32,696	62.65	53.52	57.72 **	31.14 **
+ 12-NN	35,276	62.57	53.57	57.72 **	31.04 **
+ 13-NN	37,849	62.33	53.55	57.60 **	31.35 **
+ 14-NN	40,413	62.08	53.20	57.30 **	30.94 **
+ 15-NN	42,968	61.72	53.01	57.03 **	30.73 **
+ 16-NN	45,514	61.77	52.84	56.96 **	30.63 **
+ 17-NN	48,053	61.75	52.75	56.90 **	30.43 **
+ 18-NN	50,582	61.64	52.86	56.92 **	30.43 **
+ 19-NN	53,103	61.63	52.84	56.89 **	30.27 **
+ 20-NN	55,612	61.60	52.92	56.93 **	30.58 **
20 seeds/pred.	6,041	63.14	51.05	56.45	29.60
+ 1-NN	10,648	64.23	54.40	58.91 **	32.27 **
+ 2-NN	15,156	63.98	56.01	59.73 **	33.56 **
+ 3-NN	19,599	63.52	56.35	59.72 **	33.45 **
+ 4-NN	24,002	63.40	56.41	59.70 **	33.50 **
+ 5-NN	28,367	63.38	56.44	59.71 **	33.30 **
+ 6-NN	32,699	63.49	56.81	59.96 **	33.61 **
+ 7-NN	37,005	63.26	56.75	59.83 **	33.45 **
+ 8-NN	41,278	63.38	56.98	60.01 **	33.56 **
+ 9-NN	45,517	63.37	57.20	60.13 **	33.50 **
+ 10-NN	49,729	63.60	57.37	60.33 **	33.56 **
+ 11-NN	53,908	64.05	57.66	60.69 **	34.02 **
+ 12-NN	58,060	63.97	57.35	60.48 **	34.22 **

Known Predicates, SALSA (continued)

Training set	Size	P/%	R/%	F_1/%	Ex. match/%
20 seeds/pred.					
+ 13-NN	62,190	63.93	57.66	60.63 **	33.97 **
+ 14-NN	66,292	63.54	57.49	60.36 **	33.91 **
+ 15-NN	70,368	63.50	57.37	60.28 **	33.81 **
+ 16-NN	74,419	63.11	57.26	60.04 **	33.76 **
+ 17-NN	78,452	63.60	57.29	60.28 **	34.12 **
+ 18-NN	82,459	63.50	57.43	60.31 **	34.12 **
+ 19-NN	86,450	63.18	57.18	60.03 **	34.12 **
+ 20-NN	90,426	62.73	56.89	59.67 **	33.71 **
all seeds	17,545	65.21	59.90	62.44	35.35
+ 1-NN	29,499	65.79	61.03	63.32 *	36.58 *
+ 2-NN	40,912	65.92	62.05	63.93 *	37.51 **
+ 3-NN	51,995	65.71	61.51	63.54 *	37.51 **
+ 4-NN	62,811	65.52	61.49	63.44	37.40 *
+ 5-NN	73,427	65.94	62.25	64.04 *	38.12 **
+ 6-NN	83,861	65.73	62.34	63.99 *	38.22 **
+ 7-NN	94,130	66.04	62.65	64.30 **	38.33 **
+ 8-NN	104,253	66.16	62.71	64.39 **	38.48 **
+ 9-NN	114,247	65.64	62.54	64.05 *	38.17 **
+ 10-NN	124,117	65.90	62.71	64.26 **	38.33 **
+ 11-NN	133,875	65.53	62.34	63.90 *	38.07 **
+ 12-NN	143,542	65.68	62.37	63.98 *	37.92 **
+ 13-NN	153,128	65.29	62.05	63.63 *	37.71 **
+ 14-NN	162,620	65.74	62.37	64.01 *	37.92 **
+ 15-NN	172,017	65.55	62.28	63.87 *	37.66 *
+ 16-NN	181,339	65.27	62.05	63.62 *	37.56 *
+ 17-NN	190,585	65.43	61.94	63.64 *	37.61 *
+ 18-NN	199,747	65.47	61.94	63.65 *	37.46 *
+ 19-NN	208,830	65.37	61.88	63.58 *	37.66 *
+ 20-NN	217,841	65.56	62.14	63.80 *	37.97 **

Unknown Predicates, FrameNet

Here, we show results of the unexpanded and the expanded classifier choosing between 1 to 5 frame candidates. Results under the vector-based and the WordNet-based frame candidate method are shown separately. Frame labeling accuracy, role labeling performance, and exact match scores are shown, with significance over the unexpanded classifier choosing between the same number of candidates indicated by asterisks. For frame labeling accuracy, we additionally provide the results of the random baseline and the upper bound.

Vector-based Frame Candidates

	Frame labeling accuracy / %			
Candidates	Random	Unexpanded	Expanded	Upper bound
1	34.22	34.22	34.22	34.22
2	23.16	33.10	38.27 **	46.31
3	16.54	28.57	35.35 **	49.63
4	13.33	26.02	35.14 **	53.31
5	11.37	22.94	33.84 **	56.83

	Role labeling performance / %					
	Unexpanded			Expanded		
Candidates	P	R	F_1	P	R	F_1
1	21.14	14.37	17.11	19.99	14.01	16.47
2	19.77	14.73	16.88	21.43	16.62	18.72 **
3	16.50	12.67	14.33	19.36	15.35	17.12 **
4	15.00	11.54	13.05	19.30	15.39	17.12 **
5	12.59	9.58	10.88	18.44	14.72	16.37 **

	Exact match / %	
Candidates	Unexpanded	Expanded
1	5.09	4.88
2	5.63	6.27 *
3	4.79	5.67 **
4	4.22	5.68 **
5	3.02	5.28 **

For 2 candidates, the expanded classifier also performs significantly better than the best unexpanded classifier in terms of frame labeling accuracy and F_1 score ($p < 0.001$), as well as exact match ($p < 0.05$). For 3 candidates it performs significantly better in terms of frame labeling accuracy ($p < 0.05$).

WordNet-based Frame Candidates

	Frame labeling accuracy / %			
Candidates	Random	Unexpanded	Expanded	Upper bound
1	45.50	45.50	45.50	45.50
2	29.61	41.24	46.89 **	59.23
3	22.20	36.02	44.82 **	66.60
4	17.31	28.75	44.75 **	69.23
5	14.45	26.56	43.58 **	72.25

	Role labeling performance / %					
	Unexpanded			Expanded		
Candidates	P	R	F_1	P	R	F_1
1	24.77	18.94	21.47	23.61	18.72	20.88
2	22.52	17.63	19.78	24.60	20.05	22.09 **
3	19.52	15.20	17.09	24.23	19.79	21.79 **
4	16.18	12.31	13.98	24.59	20.09	22.11 **
5	14.78	11.27	12.78	24.12	19.70	21.69 **

	Exact match / %	
Candidates	Unexpanded	Expanded
1	6.54	6.56
2	5.87	7.02 **
3	5.04	7.24 **
4	4.02	7.26 **
5	3.77	7.44 **

For 2 candidates, the expanded classifier also performs significantly better than the best unexpanded classifier in terms of frame labeling accuracy, F_1 score, and exact match ($p < 0.05$). In terms of exact match, it also performs significantly better for 3 candidates ($p < 0.05$), 4 candidates ($p < 0.05$), and 5 candidates ($p < 0.001$).

Unknown Predicates, SALSA

These tables show results of the unexpanded and the expanded classifier choosing between 1 to 5 frame candidates generated by the vector-based frame candidate method. Frame labeling accuracy, role labeling performance, and exact match scores are shown, with significance over the unexpanded classifier choosing between the same number of candidates indicated by asterisks. For frame labeling accuracy, we additionally provide the results of the random baseline and the upper bound.

	Frame labeling accuracy / %			
Candidates	Random	Unexpanded	Expanded	Upper bound
1	21.50	21.50	21.50	21.50
2	12.51	11.98	20.28 **	24.82
3	8.58	13.29	19.39 **	27.52
4	7.02	13.29	19.11 **	28.55
5	5.91	12.53	17.47 **	28.64

	Role labeling performance / %					
	Unexpanded			Expanded		
Candidates	P	R	F_1	P	R	F_1
1	19.23	13.84	16.13	17.89	13.37	15.30
2	10.02	8.15	8.99	15.21	12.85	13.93 **
3	11.58	9.56	10.48	14.59	12.65	13.55 **
4	11.49	9.52	10.41	14.36	12.42	13.32 **
5	10.61	8.83	9.64	13.08	11.39	12.17 **

	Exact match / %	
Candidates	Unexpanded	Expanded
1	8.44	7.94
2	5.29	7.41 **
3	6.46	7.55 *
4	6.60	7.41 *
5	6.24	6.85

Index

Bibliography

ABEND, Omri, Roi REICHART, and Ari RAPPOPORT. 2009. Unsupervised argument identification for semantic role labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 28–36. Singapore.

ANDERSEN, Øistein E., Julien NIOCHE, Ted BRISCOE, and John CARROLL. 2008. The BNC parsed with RASP4UIMA. In *Proceedings of the 6th International Language Resources and Evaluation Conference*, 865–869. Marrakech, Morocco.

ANDO, Rie K., and Tong ZHANG. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 1–9. Ann Arbor, MI, USA.

BAKER, Collin F., Michael ELLSWORTH, and Katrin ERK. 2007. SemEval-2007 Task 19: Frame semantic structure extraction. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*, 99–104. Prague, Czech Republic.

BAKER, Collin F., Charles J. FILLMORE, and John B. LOWE. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, 86–90. Montreal, Canada.

BARZILAY, Regina, and Lillian LEE. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 16–23. Edmonton, Canada.

BISHOP, Christopher M. 2007. *Pattern Recognition and Machine Learning.* Berlin: Springer.

BLACK, E., S. ABNEY, D. FLICKINGER, C. GDANIEC, R. GRISHMAN, P. HARRISON, D. HINDLE, R. INGRIA, F. JELINEK, J. KLAVANS,

M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the DARPA Speech and Natural Language Workshop*, 306–311. San Mateo, CA, USA: Morgan Kaufman.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Blum, Avrim, and Shuchi Chawla. 2001. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th International Conference on Machine Learning*, 19–26. Williamstown, MA, USA.

Blum, Avrim, and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 92–100. New York, NY, USA.

Borin, Lars, Dana Dannélls, Markus Forsberg, Maria Toporowska Gronostaj, and Dimitrios Kokkinakis. 2009. Thinking green: Toward Swedish FrameNet++. In *FrameNet Masterclass and Workshop at the Eighth International Workshop on Treebanks and Linguistic Theories*. Milan, Italy.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER Treebank. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories*, 24–41. Sozopol, Bulgaria.

Briscoe, Ted, and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, 41–48. Sydney, Australia.

Briscoe, Ted, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, 77–80. Sydney, Australia.

Budanitsky, Alexander, and Graeme Hirst. 2006. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics* 32(1): 13–47.

Burchardt, Aljoscha, Katrin Erk, and Anette Frank. 2005. A WordNet detour to FrameNet. In *Proceedings of the GLDV 2005 Workshop GermaNet II*. Bonn, Germany.

Burchardt, Aljoscha, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2006. The SALSA Corpus: A

German corpus resource for lexical semantics. In *Proceedings of the 5th International Language Resources and Evaluation Conference*, 969–974. Genoa, Italy.

Burchardt, Aljoscha, Marco Pennacchiotti, Stefan Thater, and Manfred Pinkal. 2009. Assessing the impact of frame semantics on textual entailment. *Journal of Natural Language Engineering* 15(4):527–550.

Cardona, George. 1976. *Panini: A Survey of Research.* Mouton.

Carreras, Xavier, and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 Shared Task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning*, 89–97. Boston, MA, USA.

———. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, 152–164. Ann Arbor, MI, USA.

Carroll, John, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: A survey and a new proposal. In *Proceedings of the 1st International Language Resources and Evaluation Conference*, 447–454. Granada, Spain.

Chan, Yee Seng, and Hwee Tou Ng. 2007. Domain adaptation with active learning for word sense disambiguation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 49–56. Prague, Czech Republic.

Chen, Jinying, Andrew Schein, Lyle Ungar, and Martha Palmer. 2006. An empirical study of the behavior of active learning for word sense disambiguation. In *Proceedings of the 2006 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 120–127. New York, NY, USA.

Chomsky, Noam. 1957. *Syntactic Structures.* Mouton.

Cohn, Trevor, and Philip Blunsom. 2005. Semantic role labelling with tree conditional random fields. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, 169–172. Ann Arbor, MI, USA.

Collins, Michael. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics* 29(4):589–637.

Cook, Stephen A. 1971. The complexity of theorem proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, 151–158. Shaker Heights, OH, USA.

DANTZIG, George B. 1963. *Linear Programming and Extensions.* Princeton University Press.

DEMPSTER, Arthur P., Nan M. LAIRD, and Donald B. RUBIN. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39(1):1–38.

DESCHACHT, Koen, and Marie-Francine MOENS. 2009. Semi-supervised semantic role labeling using the latent words language model. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 21–29. Singapore.

DIPPER, Stefanie. 2003. *Implementing and Documenting Large-Scale Grammars – German LFG*, vol. 9(1) of *Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung (AIMS).* University of Stuttgart, Germany.

DOWTY, David. 1991. Thematic proto-roles and argument selection. *Language* 67(3):547–619.

DRIDAN, Rebecca. 2010. *Using Lexical Statistics to Improve HPSG Parsing*, vol. 30 of *Saarbrücken Dissertations in Computational Linguistics and Language Technology.* Saarbrücken, Germany: German Research Center for Artificial Intelligence and Saarland University.

ERK, Katrin, and Sebastian PADÓ. 2006. Shalmaneser – a toolchain for shallow semantic parsing. In *Proceedings of the 5th International Language Resources and Evaluation Conference*, 527–532. Genoa, Italy.

———. 2008. A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing.* Honolulu, HI, USA.

FAN, Rong-En, Kai-Wei CHANG, Cho-Jui HSIEH, Xiang-Rui WANG, and Chih-Jen LIN. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research* 9:1871–1874.

FELLBAUM, Christiane, ed. 1998. *WordNet: An Electronic Lexical Database.* MIT Press.

FILLMORE, Charles J. 1968. The case for case. In *Universals in Linguistic Theory*, ed. Emmon Bach and Robert T. Harms, 1–88. New York, NY, USA: Holt, Rinehart and Winston.

———. 1976. Frame semantics and the nature of language. *Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech* 280:20–32.

FIRTH, John R. 1957. A synopsis of linguistic theory 1930-1955. In *Studies in Linguistic Analysis*, 1–32. Oxford, UK: Philological Society.

FLEISCHMAN, Michael, Namhee KWON, and Eduard HOVY. 2003. Maximum entropy models for FrameNet classification. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, 49–56. Sapporo, Japan.

FORST, Martin. 2007. Filling statistics with linguistics – property design for the disambiguation of German LFG parses. In *Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing*. Prague, Czech Republic.

FORST, Martin, Núria BERTOMEU, Berthold CRYSMANN, Frederik FOUVRY, Silvia HANSEN-SCHIRRA, and Valia KORDONI. 2004. Towards a dependency-based gold standard for German parsers – the TiGer Dependency Bank. In *Proceedings of the 5th International Workshop on Linguistically Interpreted Corpora*. Geneva, Switzerland.

FRIEDMAN, Jerome H. 1996. Another approach to polychotomous classification. Department of Statistics, Stanford University.

FUNG, Pascale, and Benfeng CHEN. 2004. BiFrameNet: Bilingual frame semantics resource construction by cross-lingual induction. In *Proceedings of the 20th International Conference on Computational Linguistics*, 931–937. Geneva, Switzerland.

GILDEA, Daniel, and Daniel JURAFSKY. 2002. Automatic labeling of semantic roles. *Computational Linguistics* 28(3):245–288.

GORDON, Andrew S., and Reid SWANSON. 2007. Generalizing semantic role annotations across syntactically similar verbs. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 192–199. Prague, Czech Republic.

GRENAGER, Trond, and Christopher D. MANNING. 2006. Unsupervised discovery of a statistical verb lexicon. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, 1–8. Sydney, Australia.

HAMP, Birgit, and Helmut FELDWEG. 1997. GermaNet – a lexical-semantic net for German. In *Proceedings of the Workshop on Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, 9–15. Madrid, Spain.

HARRIS, Zellig S. 1968. *Mathematical Structures of Language*. New York: Interscience.

He, Shan, and Daniel Gildea. 2006. Self-training and co-training for semantic role labeling: Primary report. Tech. Rep. 891, Computer Science Department, University of Rochester.

Jiang, Jay J., and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the 10th Research on Computational Linguistics International Conference*, 19–33. Taipei, Taiwan.

Joachims, Thorsten. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, 200–209. Bled, Slovenia.

Johansson, Richard. 2008. Dependency-based Semantic Analysis of Natural-language Text. Ph.D. thesis, Department of Computer Science, Lund University.

Johansson, Richard, and Pierre Nugues. 2006. A FrameNet-based semantic role labeler for Swedish. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, 436–443. Sydney, Australia.

———. 2007a. Syntactic representations considered for frame-semantic analysis. In *Proceedings of the Sixth International Workshop on Treebanks and Linguistic Theories*. Bergen, Norway.

———. 2007b. Using WordNet to extend FrameNet coverage. In *Proceedings of the Workshop on Building Frame-semantic Resources for Scandinavian and Baltic Languages*, 27–30. Tartu, Estonia.

Karmarkar, Narendra. 1984. A new polynomial time algorithm for linear programming. *Combinatorica* 4(4):373–395.

Khachiyan, Leonid G. 1980. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics* 20(1):53–72.

King, Tracy H., Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 dependency bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora at EACL 2003*, 1–8. Budapest, Hungary.

Kipper, Karin, Hoa Trang Dang, and Martha Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, 691–696. Austin, TX, USA.

Klau, Gunnar W. 2009. A new graph-based method for pairwise global network alignment. *BMC Bioinformatics* 10(Suppl 1):S59.

KLEE, Victor L., and George J. MINTY. 1972. How good is the simplex algorithm? In *Inequalities III*, ed. Oved Shisha, 159–175. Academic Press.

KUHN, Harold W. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2(1-2):83–97.

LAND, Ailsa H., and Alison G. DOIG. 1960. An automatic method for solving discrete programming problems. *Econometrica* 28(3):497–520.

LANDAUER, Thomas K., and Susan T. DUMAIS. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review* 104(2):211–240.

LANG, Joel, and Mirella LAPATA. 2010. Unsupervised induction of semantic roles. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 939–947. Los Angeles, CA, USA.

LEE, Lillian. 1999. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 25–32. College Park, MD, USA.

LEVIN, Beth. 1993. *English Verb Classes and Alternations: A Preliminary Investigation.* University of Chicago Press.

LI, Hang, and Naoki ABE. 1998. Generalizing case frames using a thesaurus and the MDL principle. *Computational Linguistics* 24(2):239–248.

LIDSTONE, George J. 1920. Note on the general case of the Bayes-Laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries* 8:182–192.

LITKOWSKI, Kenneth C. 2004. SENSEVAL-3 Task: Automatic labeling of semantic roles. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, 9–12. Barcelona, Spain.

LITKOWSKI, Kenneth C., and Orin HARGRAVES. 2005. The preposition project. In *Proceedings of the Second ACL-SIGSEM Workshop on The Linguistic Dimensions of Prepositions and Their Use in Computational Linguistic Formalisms and Applications*, 171–179. Colchester, UK.

———. 2007. SemEval-2007 Task 06: Word-sense disambiguation of prepositions. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*, 24–29. Prague, Czech Republic.

LUND, Kevin, and Curt BURGESS. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods* 28(2): 203–208.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330.

Màrquez, Lluís, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. 2008. Semantic role labeling: An introduction to the special issue. *Computational Linguistics* 34(2):145–159.

Màrquez, Lluís, Pere Comas, Jesús Giménez, and Neus Catalià. 2005a. Semantic role labeling as sequential tagging. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, 193–196. Ann Arbor, MI, USA.

Màrquez, Lluís, Mihai Surdeanu, Pere Comas, and Jordi Turmo. 2005b. A robust combination strategy for semantic role labeling. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 644–651. Vancouver, Canada.

Màrquez, Lluís, Luis Villarejo, M. A. Martí, and Mariona Taulé. 2007. SemEval-2007 Task 09: Multilevel semantic annotation of Catalan and Spanish. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*, 42–47. Prague, Czech Republic.

McClosky, David, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the 2006 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 152–159. New York, NY, USA.

Mehrotra, Sanjay. 1992. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization* 2:575–601.

Mel'čuk, Igor. 2003. Levels of dependency in linguistic description: Concepts and problems. In *Dependency and Valency. An International Handbook of Contemporary Research*, ed. Vilmos Ágel, Ludwig M. Eichinger, Hans-Werner Eroms, Peter Hellwig, Hand Jürgen Heringer, and Henning Lobin, 188–229. De Gruyter.

Meyers, Adam, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank Project: An interim report. In *Proceedings of the Workshop Frontiers in Corpus Annotation at HLT-NAACL 2004*, 24–31. Boston, MA, USA.

Mihalcea, Rada. 2004. Co-training and self-training for word sense disambiguation. In *Proceedings of the Eighth Conference on Computational Natural Language Learning*, 33–40. Boston, MA, USA.

Mitchell, Jeff, and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics: Human Language Technologies*, 236–244. Columbus, OH, USA.

———. 2011. Composition in distributional models of semantics. *Cognitive Science* (to appear).

Montague, Richard. 1973. The proper treatment of quantification in ordinary English. In *Approaches to Natural Language*, ed. Jaakko Hintikka, Julius Moravcsik, and Patrick Suppes, 221–242. Dordrecht, Netherlands: Reidel.

Moschitti, Alessandro, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics* 34(2):193–224.

Navigli, Roberto. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys* 41(2):1–69.

Nilsson, Jens, Joakim Nivre, and Johan Hall. 2006. Graph transformations in data-driven dependency parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 257–264. Sydney, Australia.

Noreen, Eric W. 1989. *Computer-intensive methods for testing hypotheses: An introduction.* Wiley-Interscience.

Ohara, Kyoko Hirose, Seiko Fujii, Toshio Ohori, Ryoko Suzuki, Hiroaki Saito, and Shun Ishizaki. 2004. The Japanese FrameNet Project: An introduction. In *Proceedings of the LREC Workshop on Building Lexical Resources from Semantically Annotated Corpora*, 9–11. Lisbon, Portugal.

Padó, Sebastian. 2006. *User's guide to* `sigf`*: Significance testing by approximate randomisation.*

———. 2007. *Cross-Lingual Annotation Projection Models for Role-Semantic Information*, vol. 21 of *Saarbrücken Dissertations in Computational Linguistics and Language Technology.* Saarbrücken, Germany: German Research Center for Artificial Intelligence and Saarland University.

Padó, Sebastian, and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics* 33(2):161–199.

Padó, Sebastian, Marco Pennacchiotti, and Caroline Sporleder. 2008. Semantic role assignment for event nominalisations by leveraging verbal data. In *Proceedings of the 22nd International Conference on Computational Linguistics*, 665–672. Manchester, UK.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics* 31(1):71–106.

Palmer, Martha, Daniel Gildea, and Nianwen Xue. 2010. *Semantic Role Labeling*. Morgan and Claypool.

Pennacchiotti, Marco, Diego De Cao, Roberto Basili, Danilo Croce, and Michael Roth. 2008. Automatic induction of FrameNet lexical units. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 457–465. Honolulu, HI, USA.

Pollard, Carl, and Ivan A. Sag. 1994. *Head-driven Phrase Structure Grammar*. University of Chicago Press.

Powell, Michael J. D. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal* 7(2):155–162.

Pradhan, Sameer, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. SemEval-2007 Task-17: English lexical sample, SRL and all words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations*, 87–92. Prague, Czech Republic.

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James H. Martin, and Daniel Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 581–588. Ann Arbor, MI, USA.

Punyakanok, Vasin, Dan Roth, and Wen tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics* 34(2):257–287.

Resnik, Philip. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 448–453. Montreal, Canada.

———. 1996. Selectional constraints: An information-theoretic model and its computational realization. *Cognition* 61(1-2):127–159.

Riezler, Stefan, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 118–125. Edmonton, Canada.

RIJSBERGEN, C. J. VAN. 1979. *Information Retrieval.* Butterworth-Heinemann.

ROHRER, Christian, and Martin FORST. 2006. Improving coverage and parsing quality of a large-scale LFG for German. In *Proceedings of the 5th International Language Resources and Evaluation Conference*, 2206–2211. Genoa, Italy.

RUPPENHOFER, Josef, Caroline SPORLEDER, Roser MORANTE, Collin F. BAKER, and Martha PALMER. 2010a. SemEval-2010 Task 10: Linking events and their participants in discourse. In *Proceedings of the Fifth International Workshop on Semantic Evaluations*, 45–50. Uppsala, Sweden.

RUPPENHOFER, Josef, Jonas SUNDE, and Manfred PINKAL. 2010b. Generating FrameNets of various granularities: The FrameNet Transformer. In *Proceedings of the 7th International Language Resources andi Evaluation Conference*, 2736–2743. Valletta, Malta.

SAINT-DIZIER, Patrick. 2006. *Syntax and Semantics of Prepositions.* Springer.

SHANNON, Claude E. 1948. A mathematical theory of communication. *Bell System Technical Journal* 27(3):379–423.

SHEN, Dan, and Mirella LAPATA. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 12–21. Prague, Czech Republic.

SIERKSMA, Gerard. 2001. *Linear and Integer Programming: Theory and Practice.* CRC Press.

STEEDMAN, Mark. 1987. Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory* 5(3):403–439.

SUBIRATS, Carlos, and Miriam PETRUCK. 2003. Surprise: Spanish FrameNet. In *Proceedings of the Workshop on Frame Semantics, XVII International Congress of Linguists.* Prague, Czech Republic.

SURDEANU, Mihai, Sanda HARABAGIU, John WILLIAMS, and Paul AARSETH. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 8–15. Sapporo, Japan.

SURDEANU, Mihai, Richard JOHANSSON, Adam MEYERS, Lluís MÀRQUEZ, and Joakim NIVRE. 2008. The CoNLL-2008 Shared Task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, 159–177. Manchester, UK.

SWIER, Robert S., and Suzanne STEVENSON. 2004. Unsupervised semantic role labelling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 95–102. Barcelona, Spain.

———. 2005. Exploiting a verb lexicon in automatic semantic role labelling. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 883–890. Vancouver, Canada.

TESNIÈRE, Lucien. 1959. *Éléments de Syntaxe Structurale.* Klincksieck. (German translation: ENGEL, Ulrich. 1999. *Grundzüge der Strukturalen Syntax.* Klett-Cotta).

THATER, Stefan, Hagen FÜRSTENAU, and Manfred PINKAL. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 948–957. Uppsala, Sweden.

THOMPSON, Cynthia A. 2004. Semi-supervised semantic role labeling. AAAI Spring Symposium.

THOMPSON, Cynthia A., Roger LEVY, and Christopher D. MANNING. 2003. A generative model for semantic role labeling. In *Proceedings of the 14th European Conference on Machine Learning*, 397–408. Cavtat, Croatia.

TOUTANOVA, Kristina, Aria HAGHIGHI, and Christopher D. MANNING. 2008. A global joint model for semantic role labeling. *Computational Linguistics* 34(2):161–191.

WEAVER, Warren. 1955. Translation. In *Machine Translation of Languages*, ed. William N. Locke and A. Donald Booth, 15–23. MIT Press.

WEEDS, Julie, David WEIR, and Diana MCCARTHY. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics*, 1015–1021. Geneva, Switzerland.

WU, Dekai, and Pascale FUNG. 2009. Semantic roles for SMT: A hybrid two-pass model. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, 13–16. Boulder, CO, USA.

WU, Zhibiao, and Martha PALMER. 1994. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, 133–138. Las Cruces, NM, USA.

XUE, Nianwen, and Martha PALMER. 2004. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 88–94. Barcelona, Spain.

ZHU, Jingbo, and Eduard HOVY. 2007. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 783–790. Prague, Czech Republic.

ZHU, Jingbo, Huizhen WANG, Tianshun YAO, and Benjamin K. TSOU. 2008. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22nd International Conference on Computational Linguistics*, 1137–1144. Manchester, UK.