Deep Grammar Error Detection and Automated Lexical Acquisition

Steps towards Wide-Coverage Open Texts Processing

Yi Zhang
yzhang@coli.uni-sb.de

Department of Computational Linguistics
Saarland University

IGK Colloquium
17^{th} Nov. 2005

Outline

1. Background and Motivation
   - Deep Processing: State-of-the-Art
   - Coverage of Deep Processing
2. Grammar Error Detection
   - Previous Work on Grammar Error Detection
   - Error Mining
3. Automated Lexical Acquisition
   - Previous Work on Lexical Acquisition
   - Statistical Lexical Type Predictor

Outline

1. Background and Motivation
   - Deep Processing: State-of-the-Art
   - Coverage of Deep Processing
2. Grammar Error Detection
   - Previous Work on Grammar Error Detection
   - Error Mining
3. Automated Lexical Acquisition
   - Previous Work on Lexical Acquisition
   - Statistical Lexical Type Predictor

What is deep processing?

- Deep processing means to maximally exploit grammatical knowledge for language processing.
- Focus on linguistic precision and semantic modelling
- Grammar-centric approach
- The opposite of deep is not statistical but shallow.

Why we need deep processing?

- Explicit model of grammaticality
- Ability to capture subtle linguistic interactions
- Semantics

Problems with deep processing

Efficiency

- Detailed language modelling creates large search space.
- Alleviated by efficient parsing algorithms and better hardware

Specificity

- Linguistic sound vs. application interesting
- Ranking of the results is necessary.

Robustness/Coverage

- Strict grammaticality metric
- Insufficient coverage of the grammar
- Dynamic nature of language

Problems with deep processing

Efficiency

- Detailed language modelling creates large search space.
- Alleviated by efficient parsing algorithms and better hardware

Specificity

- Linguistic sound vs. application interesting
- Ranking of the results is necessary.

Robustness/Coverage

- Strict grammaticality metric
- Insufficient coverage of the grammar
- Dynamic nature of language

Problems with deep processing

Efficiency

- Detailed language modelling creates large search space.
- Alleviated by efficient parsing algorithms and better hardware

Specificity

- Linguistic sound vs. application interesting
- Ranking of the results is necessary.

Robustness/Coverage

- Strict grammaticality metric
- Insufficient coverage of the grammar
- Dynamic nature of language

Robustness and specificity

Robustness and specificity are a pair of dual problems.

Grammar Engineering

- Overgeneration $\asymp$ specificity
- Undergeneration $\asymp$ robustness

Application

- Ranked output
- High coverag over noisy inputs

- For deep grammars, a balance point should be achieved to maximize linguistic accuracy.
- Robustness and specificity should come with extra mechanism.

Robustness and specificity

Robustness and specificity are a pair of dual problems.

Grammar Engineering

- Overgeneration $\asymp$ specificity
- Undergeneration $\asymp$ robustness

Application

- Ranked output
- High coverag over noisy inputs

- For deep grammars, a balance point should be achieved to maximize linguistic accuracy.
- Robustness and specificity should come with extra mechanism.

Robustness and specificity

Robustness and specificity are a pair of dual problems.

Grammar Engineering

- Overgeneration $\asymp$ specificity
- Undergeneration $\asymp$ robustness

Application

- Ranked output
- High coverag over noisy inputs

- For deep grammars, a balance point should be achieved to maximize linguistic accuracy.
- Robustness and specificity should come with extra mechanism.

Robustness and specificity

Robustness and specificity are a pair of dual problems.

Grammar Engineering

- Overgeneration $\asymp$ specificity
- Undergeneration $\asymp$ robustness

Application

- Ranked output
- High coverag over noisy inputs

- For deep grammars, a balance point should be achieved to maximize linguistic accuracy.
- Robustness and specificity should come with extra mechanism.

Outline

1. Background and Motivation
   - Deep Processing: State-of-the-Art
   - Coverage of Deep Processing
2. Grammar Error Detection
   - Previous Work on Grammar Error Detection
   - Error Mining
3. Automated Lexical Acquisition
   - Previous Work on Lexical Acquisition
   - Statistical Lexical Type Predictor

Coverage problem of deep processing

Road-testing ERG over BNC [Baldwin et al., 2004]

- Test on 20,000 strings from BNC
- Full lexical span for only 32%
- Among these
  - 57% are parsed (overall coverage 57% $\times$ 32% $\approx$ 18%)
  - 83% of the parses are correct
  - 40% parsing failures are caused by missing lexical entries
  - 39% parsing failures are caused by missing constructions

The focus of this talk

- Deep grammar error detection
  The lexical coverage is a major problem for deep processing.
- Automated deep lexical acquisition

Outline

1. Background and Motivation
   - Deep Processing: State-of-the-Art
   - Coverage of Deep Processing
2. Grammar Error Detection
   - Previous Work on Grammar Error Detection
   - Error Mining
3. Automated Lexical Acquisition
   - Previous Work on Lexical Acquisition
   - Statistical Lexical Type Predictor

Symbolic approach

Inductive Logic Programming

$$\text{Background} \wedge \text{Hypothesis} \models \text{Evidence}$$

ILP based grammar extension [Cussens and Pulman, 2000]

After a failed parse, abduction is used to find needed edges, which, if they existed, would allow a complete parse of the sentence. Linguistic constraints are applied to restrict the generation of implausible edges.

Problems

The generated rules are too general or too specific.

Symbolic approach

Inductive Logic Programming

$$\text{Background} \wedge \text{Hypothesis} \models \text{Evidence}$$

ILP based grammar extension [Cussens and Pulman, 2000]

After a failed parse, abduction is used to find needed edges, which, if they existed, would allow a complete parse of the sentence. Linguistic constraints are applied to restrict the generation of implausible edges.

Problems

The generated rules are too general or too specific.

Symbolic approach

Inductive Logic Programming

$$\text{Background} \wedge \text{Hypothesis} \models \text{Evidence}$$

ILP based grammar extension [Cussens and Pulman, 2000]

After a failed parse, abduction is used to find needed edges, which, if they existed, would allow a complete parse of the sentence. Linguistic constraints are applied to restrict the generation of implausible edges.

Problems

The generated rules are too general or too specific.

Outline

1. Background and Motivation
   - Deep Processing: State-of-the-Art
   - Coverage of Deep Processing
2. Grammar Error Detection
   - Previous Work on Grammar Error Detection
   - Error Mining
3. Automated Lexical Acquisition
   - Previous Work on Lexical Acquisition
   - Statistical Lexical Type Predictor

Error Mining

[van Noord, 2004]

- Large hand-crafted grammars are error-prone.
- Manual detection of errors is time consuming.
- Small test suite based validations are not reliable.
- Parsing failures are good indication of (under-generating) errors.

Parsability

Definition

$$R(w_i \dots w_j) = \frac{C(w_i \dots w_j | OK)}{C(w_i \dots w_j)}$$

- If the parsability of a particular word sequence is (much) lower, it indicates that something is wrong.
- Parsabilities can be calculated efficiently for large corpus with suffix arrays and perfect hashing.

Error mining experiment of ERG with BNC

- 1.8M sentences (21.2M words) with only ASCII characters and no more than 20 words each
- Running *best-only* parsing with PET took less 2 days on *elf*

Status	Num. of Sentence	Percentage
Parsed	301,503	16.74%
No lexical span	1,260,404	69.97%
No parse found	239,272	13.28%
Edge limit reached	96	0.01%

Error mining experiment of ERG with BNC

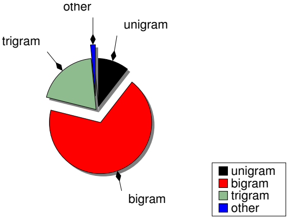
- 1.8M sentences (21.2M words) with only ASCII characters and no more than 20 words each
- Running *best-only* parsing with PET took less 2 days on *elf*

Status	Num. of Sentence	Percentage
Parsed	301,503	16.74%
No lexical span	1,260,404	69.97%
No parse found	239,272	13.28%
Edge limit reached	96	0.01%

Error analysis

	Number	Percentage
uni-gram	2,336	10.52%
bi-gram	15,183	68.36%
tri-gram	4,349	19.58%

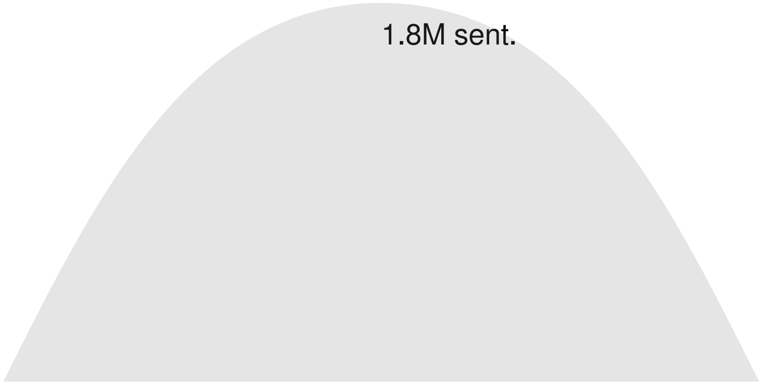
Table: N-grams with $R < 0.1$



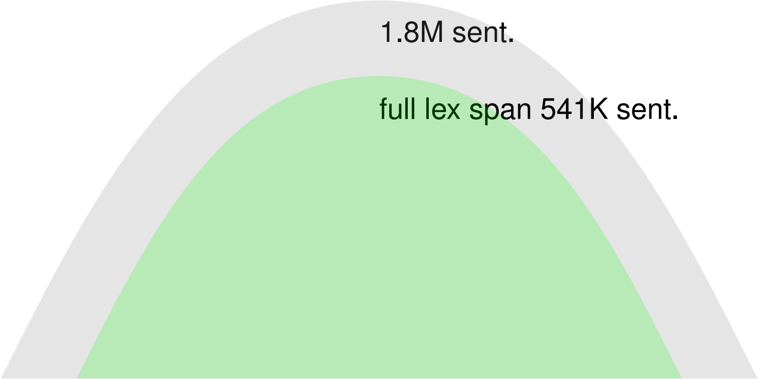
N-gram	Count
weed	59
the poor	49
a fight	113
in connection	85
as always	84
peered at	28
the World Cup	57

Table: Examples

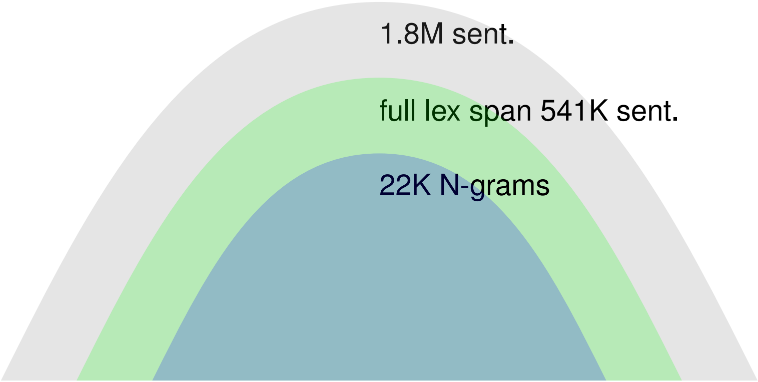
Pin down the errors



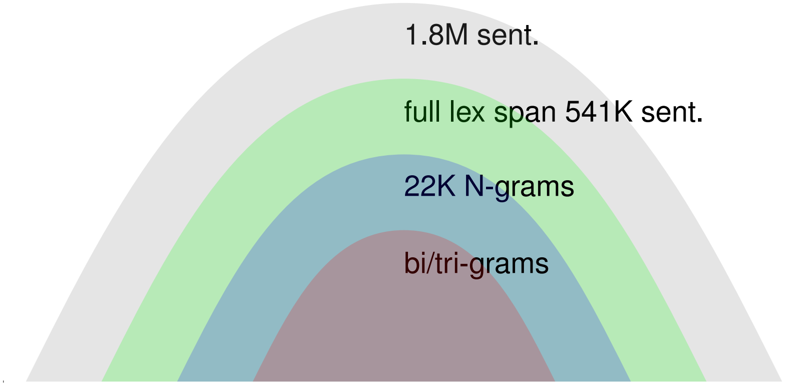
Pin down the errors



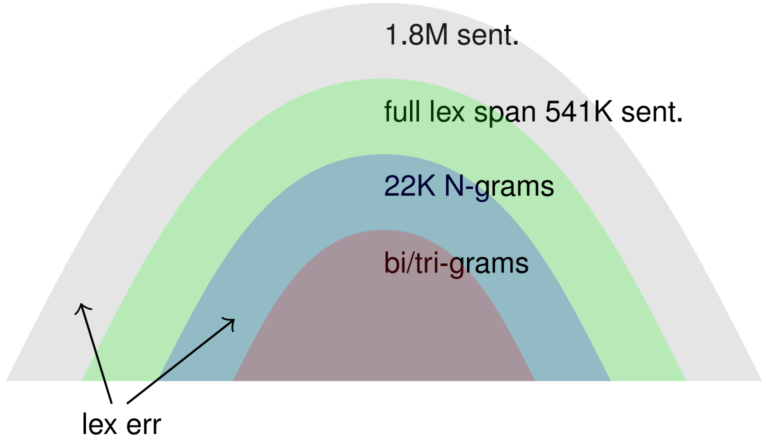
Pin down the errors



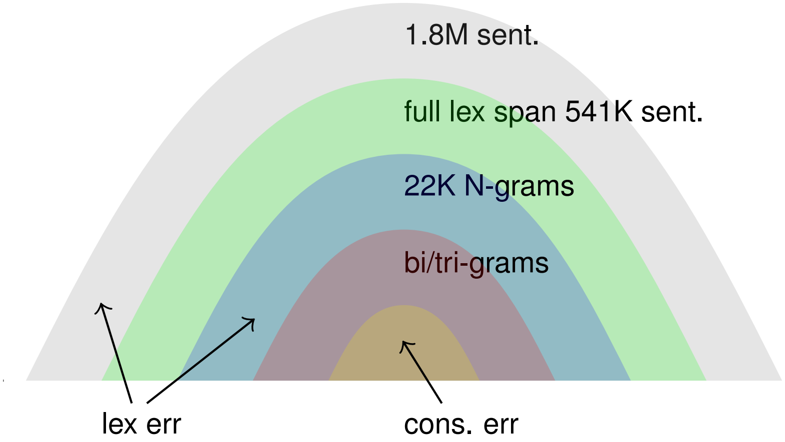
Pin down the errors



Pin down the errors



Pin down the errors



Detecting lexical error

- Missing lexical span
- Low parsability unigrams
- Language dependent heuristics:
  i.e. low parsability bigrams started with determiner like *"the poor"*, *"a fight"*

Outline

Unification-based approach

[Erbach, 1990, Barg and Walther, 1998, Fouvry, 2003]

- Use underspecified lexical entries to parse the whole sentence
- Generate lexical entries afterwards by collecting information from the full parse

An example of how unification-based approach works

the kangaroo jumps

An example of how unification-based approach works

$$\begin{bmatrix} \text{STEM} & \langle \text{"THE"} \rangle \\ \text{HEAD} & \textit{det} \end{bmatrix}$$

$$\begin{bmatrix} \text{STEM} & \langle \text{"KANGAROO"} \rangle \end{bmatrix}$$

$$\begin{bmatrix} \text{STEM} & \langle \text{"JUMPS"} \rangle \\ \text{HEAD} & {}_{\textit{verb}}\begin{bmatrix} \text{PERSON} & \boxed{2}\ \textit{3rd} \\ \text{NUM} & \boxed{3}\ \textit{sg} \end{bmatrix} \\ \text{SUBJ} & \left\langle \begin{bmatrix} \text{HEAD} & {}_{\textit{noun}}\begin{bmatrix} \text{PERSON} & \boxed{2} \\ \text{NUM} & \boxed{3} \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix}$$

the kangaroo jumps

An example of how unification-based approach works

$$\begin{bmatrix} \text{HEAD} & \boxed{5} \\ \text{SPR} & \langle\rangle \\ \text{HEAD-DTR} & \begin{bmatrix}\text{HEAD} & \boxed{5} \\ \text{SPR} & \left\langle \boxed{1} \right\rangle\end{bmatrix} \\ \text{NON-HEAD-DTR} & \boxed{1}\end{bmatrix}$$

$$\begin{bmatrix} \text{STEM} & \left\langle \text{"THE"} \right\rangle \\ \text{HEAD} & \textit{det} \end{bmatrix} \qquad \begin{bmatrix} \text{STEM} & \left\langle \text{"KANGAROO"} \right\rangle \end{bmatrix} \qquad \begin{bmatrix} \text{STEM} & \left\langle \text{"JUMPS"} \right\rangle \\ \text{HEAD} & \textit{verb}\begin{bmatrix}\text{PERSON} & \boxed{2}\ \textit{3rd} \\ \text{NUM} & \boxed{3}\ \textit{sg}\end{bmatrix} \\ \text{SUBJ} & \left\langle \begin{bmatrix} \text{HEAD} & \textit{noun}\begin{bmatrix}\text{PERSON} & \boxed{2} \\ \text{NUM} & \boxed{3}\end{bmatrix}\end{bmatrix} \right\rangle \end{bmatrix}$$

the | kangaroo | jumps

An example of how unification-based approach works

$$\begin{bmatrix} \text{SUBJ} & \langle\rangle \\ \text{HEAD-DTR} & \begin{bmatrix}\text{SUBJ} & \left\langle \boxed{4} \right\rangle\end{bmatrix} \\ \text{NON-HEAD-DTR} & \boxed{4}\end{bmatrix}$$

$$\begin{bmatrix} \text{HEAD} & \boxed{5} \\ \text{SPR} & \langle\rangle \\ \text{HEAD-DTR} & \begin{bmatrix}\text{HEAD} & \boxed{5} \\ \text{SPR} & \left\langle \boxed{1} \right\rangle\end{bmatrix} \\ \text{NON-HEAD-DTR} & \boxed{1}\end{bmatrix}$$

$$\begin{bmatrix} \text{STEM} & \left\langle \text{"THE"} \right\rangle \\ \text{HEAD} & \textit{det}\end{bmatrix} \qquad \begin{bmatrix} \text{STEM} & \left\langle \text{"KANGAROO"} \right\rangle \end{bmatrix} \qquad \begin{bmatrix} \text{STEM} & \left\langle \text{"JUMPS"} \right\rangle \\ \text{HEAD} & \begin{bmatrix}\text{PERSON} & \boxed{2}\ \textit{3rd} \\ \text{NUM} & \boxed{3}\ \textit{sg}\end{bmatrix}_{\textit{verb}} \\ \text{SUBJ} & \left\langle \begin{bmatrix}\text{HEAD} & \begin{bmatrix}\text{PERSON} & \boxed{2} \\ \text{NUM} & \boxed{3}\end{bmatrix}_{\textit{noun}}\end{bmatrix} \right\rangle \end{bmatrix}$$

the kangaroo jumps

Problems with unification-based approaches

- Generated lexical entries might be:
  - *too general*: overgeneration
  - *too specific*: undergeneration
- Computational complexity increased significantly with underspecified lexical entries, especially when two unknowns occur next to each other.

Outline

Statistical approach

[Baldwin, 2005]

- Based on a set of lexical types
- Treat lexical acquisition as a classification task
- Generalize the acquisition model over various sencondary language resources
  - POS tagger
  - Chunker
  - Treebank
  - Dependency parser
  - Lexical ontology

DELPHIN

Importing lexicon from a semantic lexical ontology

Assumption

There is a strong correlation between the semantic and syntactic similarity of words. [Levin, 1993]

Fact

Above 90% of the synsets in WordNet (2.0) share at least one lexical type among all included words.

Importing lexicon from a semantic lexical ontology

Assumption

There is a strong correlation between the semantic and syntactic similarity of words. [Levin, 1993]

Fact

Above 90% of the synsets in WordNet (2.0) share at least one lexical type among all included words.

Importing lexicon from WordNet

[Baldwin, 2005]

- Construct semantic neighbours (all synonyms, direct hyponyms, direct hypernyms)
- Take a majority vote across the lexical types of the semantic neighbours

Improvement

Voting is weighted and must exceed a threshold.

Importing lexicon from WordNet

[Baldwin, 2005]

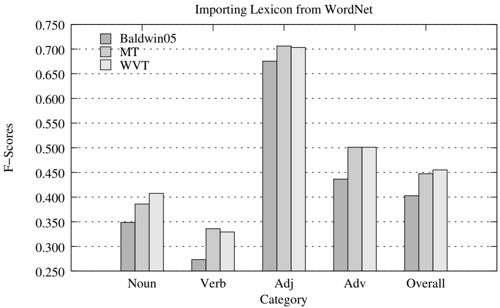
- Construct semantic neighbours (all synonyms, direct hyponyms, direct hypernyms)
- Take a majority vote across the lexical types of the semantic neighbours

Improvement

Voting is weighted and must exceed a threshold.

Importing lexicon from WordNet

Results



- The sparse ERG lexicon (as compared to WordNet) makes the voting less reliable.

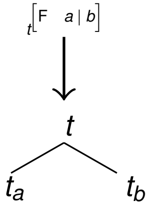
Maximum entropy model based lexical type predictor

$$p(t,c) = \frac{exp(\sum_i \theta_i f_i(t,c))}{\sum_{t' \in T} exp(\sum_i \theta_i f_i(t',c))}$$

- A statistical classifier that predicts for each occurrence of unknown word or missing lexical entry
- Input: features from the context
- Output: atomic lexical types

Atomic lexical types

- The lexical information is encoded in atomic lexical types.
- Attribute-value structures can be decomposed into atomic lexical types.



Baseline models

- Select the majority lexical type for each POS

POS	Majority Lexical Type
noun	n_intr_le
verb	v_np_trans_le
adj.	adj_intrans_le
adv.	adv_int_vp_le

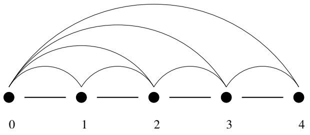
- General purpose POS tagger trained with lexical types: *TnT*, *MXPOST*

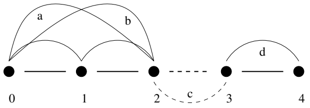
Basic features

- Prefix/suffix of the word
- Context words and their lexical types

Model	Precision
Baseline	30.7%
TnT	40.4%
MXPOST	40.2%
ME basic	50.0%

Partial parsing results





Model	Precision
Baseline	30.7%
TnT	40.4%
MXPOST	40.2%
ME basic	50.0%
ME +pp	50.5%

Turning to the disambiguation model

- Generate top *n* candidate entries for the unknown word
- Parse the sentence with candidate entries
- Use disambiguation model to select the best parse
- Pick the corresponding entry

Turning to the disambiguation model

- Generate top *n* candidate entries for the unknown word
- Parse the sentence with candidate entries
- Use disambiguation model to select the best parse
- Pick the corresponding entry

Turning to the disambiguation model

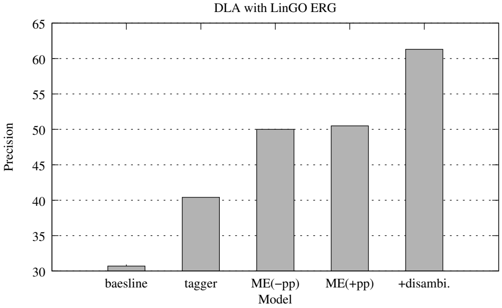
- Generate top *n* candidate entries for the unknown word
- Parse the sentence with candidate entries
- Use disambiguation model to select the best parse
- Pick the corresponding entry

Turning to the disambiguation model

- Generate top *n* candidate entries for the unknown word
- Parse the sentence with candidate entries
- Use disambiguation model to select the best parse
- Pick the corresponding entry

Experiment

Results



What has been done?

- Error mining based lexical error detection
  - Experiment with ERG and BNC shows a major part of parsing failure is due to missing lexical entries.
  - Some rules are used to discover missing lexical entries.
- Statistical lexical acquisition
  - A maximum entropy based lexical type prediction model is designed and evaluated with various feature templates.
  - Lexical ontology based acquisition method is tried.
  - Disambiguation model is incorporated to enhance robustness.

Baldwin, T. (2005).
Bootstrapping deep lexical resources: Resources for courses.
In *Proceedings of the ACL-SIGLEX Workshop on Deep Lexical Acquisition*, pages 67–76, Ann Arbor, Michigan. Association for Computational Linguistics.

Baldwin, T., Bender, E. M., Flickinger, D., Kim, A., and Oepen, S. (2004).
Road-testing the English Resource Grammar over the British National Corpus.
In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, Lisbon, Portugal.

Barg, P. and Walther, M. (1998).
Processing unkonwn words in HPSG.
In *Proceedings of the 36th Conference of the ACL and the 17th International Conference on Computational Linguistics*, Montreal, Quebec, Canada.

Cussens, J. and Pulman, S. (2000).
Incorporating Linguistics Constraints into Inductive Logic Programming.
In *Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop*.

Erbach, G. (1990).
Syntactic processing of unknown words.
IWBS Report 131, IBM, Stuttgart.

Fouvry, F. (2003).
Lexicon acquisition with a large-coverage unification-based grammar.
In *Companion to the 10th of EACL*, pages 87–90, ACL, Budapest, Hungary.

Levin, B. (1993).
English verb classes and alternations.
University of Chicago Press, Chicago, USA.

van Noord, G. (2004).

Error mining for wide-coverage grammar engineering.
In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 446–453, Barcelona, Spain.