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
17th 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 $\propto$ specificity
- Undergeneration $\propto$ robustness

Application

- Ranked output
- High coverage 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 are a pair of **dual problems**.

**Grammar Engineering**
- Overgeneration $\approx$ specificity
- Undergeneration $\approx$ robustness

**Application**
- Ranked output
- High coverage 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 are a pair of dual problems.

Grammar Engineering
- Overgeneration $\propto$ specificity
- Undergeneration $\propto$ robustness

Application
- Ranked output
- High coverage 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 are a pair of dual problems.

**Grammar Engineering**
- Overgeneration $\propto$ specificity
- Undergeneration $\propto$ robustness

**Application**
- Ranked output
- High coverage 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

\[ \text{Background} \land \text{Hypothesis} \models \text{Evidence} \]

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

\[ \text{Background} \land \text{Hypothesis} \models \text{Evidence} \]

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

Background $\wedge$ Hypothesis $\models$ 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.
<table>
<thead>
<tr>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Background and Motivation</td>
</tr>
<tr>
<td>- Deep Processing: State-of-the-Art</td>
</tr>
<tr>
<td>- Coverage of Deep Processing</td>
</tr>
<tr>
<td>2. Grammar Error Detection</td>
</tr>
<tr>
<td>- Previous Work on Grammar Error Detection</td>
</tr>
<tr>
<td>- Error Mining</td>
</tr>
<tr>
<td>3. Automated Lexical Acquisition</td>
</tr>
<tr>
<td>- Previous Work on Lexical Acquisition</td>
</tr>
<tr>
<td>- Statistical Lexical Type Predictor</td>
</tr>
</tbody>
</table>
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

\[ R(w_i \ldots w_j) = \frac{C(w_i \ldots w_j | OK)}{C(w_i \ldots w_j)} \]

- 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*

<table>
<thead>
<tr>
<th>Status</th>
<th>Num. of Sentence</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parsed</td>
<td>301,503</td>
<td>16.74%</td>
</tr>
<tr>
<td>No lexical span</td>
<td>1,260,404</td>
<td>69.97%</td>
</tr>
<tr>
<td>No parse found</td>
<td>239,272</td>
<td>13.28%</td>
</tr>
<tr>
<td>Edge limit reached</td>
<td>96</td>
<td>0.01%</td>
</tr>
</tbody>
</table>
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*

<table>
<thead>
<tr>
<th>Status</th>
<th>Num. of Sentence</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parsed</td>
<td>301,503</td>
<td>16.74%</td>
</tr>
<tr>
<td>No lexical span</td>
<td>1,260,404</td>
<td>69.97%</td>
</tr>
<tr>
<td>No parse found</td>
<td>239,272</td>
<td>13.28%</td>
</tr>
<tr>
<td>Edge limit reached</td>
<td>96</td>
<td>0.01%</td>
</tr>
</tbody>
</table>
## Error analysis

<table>
<thead>
<tr>
<th>N-gram</th>
<th>Number</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>uni-gram</td>
<td>2,336</td>
<td>10.52%</td>
</tr>
<tr>
<td>bi-gram</td>
<td>15,183</td>
<td>68.36%</td>
</tr>
<tr>
<td>tri-gram</td>
<td>4,349</td>
<td>19.58%</td>
</tr>
</tbody>
</table>

**Table: N-grams with \( R < 0.1 \)**

<table>
<thead>
<tr>
<th>N-gram</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>weed</td>
<td>59</td>
</tr>
<tr>
<td>the poor</td>
<td>49</td>
</tr>
<tr>
<td>a fight</td>
<td>113</td>
</tr>
<tr>
<td>in connection</td>
<td>85</td>
</tr>
<tr>
<td>as always</td>
<td>84</td>
</tr>
<tr>
<td>peered at</td>
<td>28</td>
</tr>
<tr>
<td>the World Cup</td>
<td>57</td>
</tr>
</tbody>
</table>

**Table: Examples**
Pin down the errors

1.8M sent.
Pin down the errors

1.8M sent.

full lex span 541K sent.
Pin down the errors

1.8M sent.

full lex span 541K sent.

22K N-grams
Pin down the errors

1.8M sent.

full lex span 541K sent.

22K N-grams

bi/tri-grams
Pin down the errors

- 1.8M sentences
- Full lexical span: 541K sentences
- 22K N-grams
- Bi/tri-grams
- Lex error
Pin down the errors

1.8M sent.
full lex span 541K sent.
22K N-grams
bi/tri-grams
lex err
cons. err
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

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
Unification-based approach


- 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

the kangaroo jumps
An example of how unification-based approach works

the kangaroo jumps
An example of how unification-based approach works

The kangaroo jumps

```
[STEM ⟨"THE"⟩]  [STEM ⟨"KANGAROO"⟩]
HEAD det

[HEAD SPR]  [HEAD "JUMPS"]
5

[HEAD-DTR]  [HEAD noun PERSON 2 3rd]

[NON-HEAD-DTR 1]  [NUM 3 sg]

[SUBJ 4]  [PERSON 2]

[HEAD-DTR]  [HEAD noun PERSON 3]

[NON-HEAD-DTR]  [SUBJ ⟨⟩]
```

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

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
Statistical approach

[Baldwin, 2005]
- Based on a set of lexical types
- Treat lexical acquisition as a classification task
- Generalize the acquisition model over various secondary language resources
  - POS tagger
  - Chunker
  - Treebank
  - Dependency parser
  - Lexical ontology
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.
The sparse ERG lexicon (as compared to WordNet) makes the voting less reliable.
Maximum entropy model based lexical type predictor

\[ p(t, c) = \frac{\exp(\sum \theta_i f_i(t, c))}{\sum_{t' \in T} \exp(\sum \theta_i f_i(t', c))} \]

- 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

<table>
<thead>
<tr>
<th>POS</th>
<th>Majority Lexical Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>noun</td>
<td>n_intr_le</td>
</tr>
<tr>
<td>verb</td>
<td>v_np_trans_le</td>
</tr>
<tr>
<td>adj.</td>
<td>adj_intrans_le</td>
</tr>
<tr>
<td>adv.</td>
<td>adv_int_vp_le</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>30.7%</td>
</tr>
<tr>
<td>TnT</td>
<td>40.4%</td>
</tr>
<tr>
<td>MXPOST</td>
<td>40.2%</td>
</tr>
<tr>
<td>ME basic</td>
<td>50.0%</td>
</tr>
</tbody>
</table>
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

![Graph showing precision for different models: baseline, tagger, ME(-pp) Model, ME(+pp), and +disambi. under the heading DLA with LinGO ERG. The graph indicates varying precision levels across these models.]
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.
Bootstrapping deep lexical resources: Resources for courses.

Road-testing the English Resource Grammar over the British National Corpus.

Processing unknown words in HPSG.
In Proceedings of the 36th Conference of the ACL and the 17th International Conference on Computational Linguistics, Montreal, Quebec, Canada.

Incorporating Linguistics Constraints into Inductive Logic Programming.
In Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop.

Syntactic processing of unknown words.
IWBS Report 131, IBM, Stuttgart.

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

English verb classes and alternations.
University of Chicago Press, Chicago, USA.

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.