Improving knowledge-based parsing with corpus-based methods

Gertjan van Noord et al.
University of Groningen
Full knowledge-based parsing is

- fragile
- slow
- inaccurate
Parsing: state of the art

- Full knowledge-based parsing is
  - fragile
  - slow
  - inaccurate

  This is no longer true!

- Improvements:
  - robustness
  - efficiency
  - disambiguation
Parsing: state of the art

- Full knowledge-based parsing is
  - fragile
  - slow
  - inaccurate

This is no longer true!

- Improvements:
  - robustness
  - efficiency
  - disambiguation

- And most importantly:
Corpora

- Corpora, corpora, corpora
Corpora

- Corpora, corpora, corpora
  - annotated corpora
  - unannotated corpora
  - corpora annotated by the parser itself
Overview

- Very little background:
  - Alpino parser for Dutch

- Use of annotated corpora:
  - Maxent disambiguation model

- Use of unannotated corpora:
  - POS-tagger to reduce lexical ambiguities
  - Error Mining to increase coverage

- Corpora annotated by the parser:
  - Question Answering
  - Similar Words
  - Improved disambiguation model!
Alpino

- Stochastic Attribute Value Grammar for Dutch
- Attribute Value Grammar: HPSG
  
  *Linguistic Sophistication*

- Stochastic: Maximum Entropy Disambiguation Model
  
  *Principled Account of Disambiguation*

- Output: CGN Dependency Structures
## Extrinsic Motivation

<table>
<thead>
<tr>
<th>corpus</th>
<th>sentences</th>
<th>length</th>
<th>accuracy%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eindhoven corpus (cdbl)</td>
<td>7136</td>
<td>20</td>
<td>88.9</td>
</tr>
<tr>
<td>Trouw 2001</td>
<td>1400</td>
<td>17</td>
<td>91.0</td>
</tr>
<tr>
<td>CLEF questions (tuned for questions)</td>
<td>1545</td>
<td>11</td>
<td>96.3</td>
</tr>
</tbody>
</table>

Accuracy: proportion of correct named dependencies
Disambiguation Model

• Identify *features* for disambiguation

• Training the model: assign a *weight* to each feature, by
  ✓ increase weights of features in the correct parse
  ✓ decrease weights of features in incorrect parses

• Applying the model:
  ✓ For each parse, sum weights of features occurring in it
  ✓ Select parse with highest sum
Features

- Describe arbitrary properties of parses
- Need not be independent of each other
- Can encode a variety of linguistic (and other) preferences
- Latest models: 500,000 features; 50,000 after frequency cut-off
Problem: Efficiency

- Need access to *all* parses of a sentence
  - training the model
  - applying the model

- Number of parses can be exponential

- In practice, number of parses can be Really Big
Number of parses

![Graph showing the relationship between sentence length (words) and the number of parses. The x-axis represents sentence length in words, ranging from 5 to 20. The y-axis represents the number of parses, ranging from 0 to 4000. The graph shows a steep increase in the number of parses as the sentence length increases.]
Solution 1: Use Parse Forest

- Geman and Johnson (2002)
- Miyao and Tsujii (2002)
- Train model directly from forest
- Best parse can be computed efficiently from forest
Drawbacks

- Strong Locality Requirement on Features
- Features are no longer arbitrary characteristics of parses
- Non-local features can be locally *encoded* in grammar, but
  - Complicate grammar dramatically
  - Reduce parser efficiency
Some non-local features

- In coordinated structure, the conjuncts are parallel or not
- In extraction structure, the extraction is local or not
- In extraction structure, the extracted element is a subject
- Constituent ordering
  - pronoun precedes full np
  - accusative pronoun precedes dative pronoun
  - dative full np precedes accusative full np
Solution 2: Use Sample for training

- Osborne (2000): representative small sample of parses
- Take into account relative quality of parses during training
- Provides solution for cases where treebank structures are of different nature than parses
- Training material consists of parser output (annotated with quality score)
Solution 2: Use Sample for training

- Osborne (2000): representative small sample of parses
- Take into account relative quality of parses during training
- Provides solution for cases where treebank structures are of different nature than parses
- Training material consists of parser output (annotated with quality score)
Construct Training Material

- Construct the first 1,000 parses of each sentence from the corpus
- For each parse, count the frequency of all features
- Compare each parse with the gold standard, and assign corresponding score
- Each parse is represented by a vector of feature frequencies and a quality score
Results

- cdbl-part of Alpino treebank (145,000 words annotated with dependency structures)
- ten-fold cross-validation
- Model should select best parse for each sentence out of maximally 1000 parses per sentence

<table>
<thead>
<tr>
<th>Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
</tr>
<tr>
<td>oracle</td>
</tr>
<tr>
<td>model</td>
</tr>
<tr>
<td>rate</td>
</tr>
<tr>
<td>exact</td>
</tr>
</tbody>
</table>
Use disambiguation model

- parser constructs parse forest
- best-first beam-search algorithm selects best parse from forest
HMM lexical analysis filter

- large lexicon → lexical ambiguity is source of inefficiency
- many lexical categories are non-sensical in a given context
- **ik ben in Haifa**
  * I am in Haifa
  - *in*: adjective, particle, preposition, verb
- Some lexical ambiguity is allowed
  - *in*: particle, preposition
HMM Tagger

- Goal: filter unlikely tags
- Tags: simplified lexical categories (ignore subcategorization information)
- Train on output of parser
- Learn tag-sequences preferred by parser
HMM Tagger

- Goal: filter unlikely tags
- Tags: simplified lexical categories (ignore subcategorization information)
- Train on output of parser
- Learn tag-sequences preferred by parser
HMM Tagger

- **Goal**: filter unlikely tags
- **Tags**: simplified lexical categories (ignore subcategorization information)
- **Train on output of parser**
- **Learn tag-sequences preferred by parser**
- **No need for human-annotated data**
- **Large amounts of training data available**
**Tagger: Training**

- Run Alpino (without the tagger) on large corpus
- Ignore *difficult* sentences
- Remember the tag sequence associated with best parse
- Latest models trained on millions of parsed sentences
Trigram HMM Tagger

- $P(t_i|t_{i-2}t_{i-1})$
- $P(w_i|t_i)$
- Forward-backward algorithm to compute \textit{a posteriori} probabilities
- For each position, compare best tag with competing tag and remove if difference above threshold
Tagger: Results

- over 95% stand-alone on raw data
- many of the “mistakes” are correct
- improves Alpino significantly
Frequency of CPU–time (6805 sentences)
Results

- Up to 20 times faster (depending on threshold)
- Small accuracy *improvement*
- Prevents parser from hallucinating full parses for fragmentary input
  - *in Haifa vanavond In(prep)/Cash(v) Haifa tonight*
Next: Error Mining
Robustness can be enhanced:

- Treat unknown words
- Use partial parse results
- Learn from your mistakes
Goal: improve coverage of the parser

- sets of hand-crafted examples $\implies$ problems must be anticipated
- treebanks $\implies$ much too small
- Instead: use unannotated material
Goal: improve coverage of the parser

- Run the parser on many sentences
- Analyse sentences with missing parses
- Find words and word sequences that occur (much) more often in these sentences
Error Mining

- Error Mining Metric

- Results:
  - Linguistic Examples

- Technical details: compute Ngram frequencies for large N and large corpora
  - Suffix Arrays
  - Perfect Hash Finite Automata
Corpora

- Various newspapers 1994-2002 (Trouw, NRC, AD, Volskrant, Parool)
- Sentences up to 20 (25, 30) words (with time-out)
- $\geq 2M$ sentences, $\geq 40M$ words, $\geq 200M$ chars
Metric (1)

- **full** parse: a parse spanning the whole sentence
- \( C(w) \): frequency of word \( w \)
- \( C(w|OK) \): frequency of word \( w \) in sentences with a full parse
- compute for all words \( w \):

\[
R(w) = \frac{C(w|OK)}{C(w)}
\]
Coverage

- For this material: 91–96%
- An R-value significantly below .91 is interesting

```
0.000  7  l’d
daangroeit
0.000  9  aanzoek
0.000  7  adoreert
0.000  8  afkeur
0.000 21  afroep
0.000  7  après
0.000  7  berge
0.000  7  einmal
```
Metric (2)

- Often, words are problematic only in certain contexts

- $C(w_i \ldots w_j)$: frequency

- $C(w_i \ldots w_j|OK)$: frequency in full parse

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

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.902</td>
<td>716</td>
<td>via</td>
<td>via</td>
<td></td>
</tr>
<tr>
<td>0.167</td>
<td>15</td>
<td>via</td>
<td>via</td>
<td>indirectly</td>
</tr>
<tr>
<td>0.916</td>
<td>165</td>
<td>waard</td>
<td>worth</td>
<td></td>
</tr>
<tr>
<td>0.000</td>
<td>10</td>
<td>de waard</td>
<td>the host</td>
<td></td>
</tr>
</tbody>
</table>
Metric (3)

- **Threshold:** only consider sequences where
  \[ C(w_i \ldots w_j) - C(w_i \ldots w_j | \text{OK}) > 5 \]

- **Consider longer sequences only if worse than corresponding shorter ones:**
  \[ R(w_h w_i \ldots w_j w_k) < R(w_h w_i \ldots w_j) \text{ and } \]
  \[ R(w_h w_i \ldots w_j w_k) < R(w_i \ldots w_j w_k) \]

- *if the score is the same, the longer Ngram is typically more informative*
Sort according to R value (1)

0.000 82 ! enz. chess
0.000 8 ! gevolgd chess
0.000 7 , zo 12-17u announcement
0.000 15 , zo 13-17u
0.000 316 - fl. new books
0.000 12 ; 127 blz.
0.000 10 ; 142 blz.
0.000 14 ; 143 blz.
0.000 19 16x27 checkers
0.000 7 2Klaver pas bridge
0.000 8 4 t/m 12 jaar announcement (theater, ..)
0.000 17 I have foreign language
### Sort according to R value (2)

<table>
<thead>
<tr>
<th>R Value</th>
<th>Rank</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>7</td>
<td>de huisraad</td>
</tr>
<tr>
<td>0.000</td>
<td>7</td>
<td>Maar eerlijk is eerlijk</td>
</tr>
<tr>
<td>0.000</td>
<td>9</td>
<td>en noem maar</td>
</tr>
<tr>
<td>0.000</td>
<td>18</td>
<td>is daar een voorbeeld</td>
</tr>
<tr>
<td>0.000</td>
<td>7</td>
<td>par excellence</td>
</tr>
<tr>
<td>0.000</td>
<td>7</td>
<td>In vroeger tijden</td>
</tr>
<tr>
<td>0.000</td>
<td>7</td>
<td>dan ten hele</td>
</tr>
<tr>
<td>0.000</td>
<td>7</td>
<td>hele gedwaald</td>
</tr>
<tr>
<td>0.000</td>
<td>7</td>
<td>het libido</td>
</tr>
<tr>
<td>0.000</td>
<td>9</td>
<td>kinds af</td>
</tr>
<tr>
<td>0.000</td>
<td>8</td>
<td>tenzij .</td>
</tr>
</tbody>
</table>

*unless* .
List problematic examples

@ Vroeger was het nee, tenzij.
@ Nou ja, het is een Nee, tenzij . . .
@ De Nederlandse wetgever staat een 'nee, tenzij.
@ Orgaandonatie tenzij . . . . ik de nagel van mijn rechter
@ Officeel is het: ja, tenzij.
@ Anderen: nee, tenzij.
@ Gebied tussen ja, mits en nee, tenzij.
@ Geen jacht tenzij.
@ U zult niet doden, tenzij.
Metric (4)

- $R(w) = \frac{C(w|OK)}{C(w)}$

- But $\frac{1}{4}$ less interesting, reliable than $\frac{10}{40}$

- Construct binomial confidence interval of $\frac{C(w|OK)}{C(w)}$

- Use maximum value of this interval for sorting

- Slower

- Better results
Spots variety of errors

- tokenization
- mistakes in lexicon
- incomplete lexical descriptions
- frozen expressions (with idiosyncratic syntax)
- incomplete grammatical descriptions
- ...
Error Mining: Conclusion

- Error mining metric spots many errors
- Coverage can be increased dramatically
- But
  - only those errors that lead to parsing failure
  - coverage is not the same as accuracy
Next: Applications

- Question Answering
- Semantic Similarity
- Parsing!
Question Answering

- Joost: Dutch QA system based on Alpino
- participates in CLEF (Evaluation platform for QA)
- four years of newspaper articles (NRC, AD 1994, 1995)
- Syntactic Analysis of Full CLEF Corpus
• computes overlap of dependency structure of question and of corpus sentence

• information extraction from CLEF corpus to create tables for frequently occurring questions
  ★ What is the capital of . . . ?
  ★ Who was . . . ?
  ★ How did . . . die?
  ★ . . .

• Many other tricks . . .
CLEF 2005 results

- Joost: about 50% of the questions answered correctly
- best result for Dutch
- third result overall (more than 40 submissions)
- side-effect: huge annotated corpus is very interesting resource...
**CLEF data**

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentences</td>
<td>4,150,858</td>
</tr>
<tr>
<td>No parse</td>
<td>0.3%</td>
</tr>
<tr>
<td>Fragment parse</td>
<td>8.9%</td>
</tr>
<tr>
<td>Full parse</td>
<td>90.8%</td>
</tr>
<tr>
<td>CPU-hours</td>
<td>20,000</td>
</tr>
</tbody>
</table>
4 million dependency structures

- Compressed archives of XML files
- Pseudo random access
- dictd gzip
- Total: 2.4 Gb (rather than 25 Gb)
Example

Dat wekte de woede van Turkse inwoners van de wijk.
Treebank Tools

- DtView
- DtEdit
- DtSearch
Dat wkte de woede van Turkse inwoners van de wijk.
DtSearch

- XPATH standard
- interface with mg

- Search queries
  - hierarchical relations
  - grammatical relations
  - syntactic category
  - surface order
  - lemma

- Matches:
  - display sentence
  - display sentence with brackets
  - display matching part of sentence
  - your own style-sheets
DtSearch Example

dtsearch -s -q '//*[@node[../@cat="smain" and @rel="obj2" and not(@cat="pp") and ./@begin = ../@begin]']

[Haar] ging het goed af.
"[Ons] staat helemaal geen Big Brother-scenario voor ogen.
[Ook hun] past enige schroom.
[Zelfs de bloeddorstigste tegenstander] adviseerde hij nog zijn gedrag wat aan te passen.
[Die] geef ik voor de wedstrijd een zoen...
Discover Ontological Information

- Similar words occur in similar contexts

- Dependency relations: more fine-grained notion of context
  - subject-verb
  - verb-object
  - adjective-noun
  - coordination
  - apposition
  - prepositional complement
Vectors describing contexts

- Every word is represented by an $n$-dimensional vector
- Every dimension is a context characteristic
- Every cell is a (normalized, weighted) frequency

<table>
<thead>
<tr>
<th></th>
<th>zie.obj</th>
<th>verf.obj</th>
<th>verzorg.obj</th>
<th>laat_uît.obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>bus</td>
<td>50</td>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>hond</td>
<td>56</td>
<td>1</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>...</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
Comparing Vectors

- **Dice:**
  \[ \sum_{i} 2 \cdot \frac{\min(v_i, w_i)}{v_i + w_i} \]

- **Nearest Neighbours (here: 20)**
Results for BMW
Results for BMW

Volkswagen, Mercedes, Honda, Chrysler, Audi, Volvo,
Ford, Toyota, Fiat, Peugeot, Opel, Mitsubishi, Renault,
Mazda, Jaguar, General Motors, Rover, Nissan, VW, Porsche
Results for Sony

Matsushita, Toshiba, Time Warner, JVC, Hitachi, Nokia, Samsung, Motorola, Philips, Siemens, Apple, Canon, IBM, PolyGram, Thomson, Mitsubishi, Kodak, Pioneer, AT&T, Sharp
Hinault

Kübler, Vermandel, Bruyère, Depredomme, Mottiat, Merckx, Depoorter, De Bruyne, Argentin, Schepers, Criquielion, Dierickx, Van Steenbergen, Kint, Bartali, Ockers, Coppi, Fignon, Kelly, De Vlaeminck
Paris

Londen, Brussel, Moskou, Washington, Berlijn, New York,
Rome, Madrid, Bonn, Wenen, Peking, Frankfurt, Athene, Tokio,
München, Barcelona, Praag, Antwerpen, Stockholm, Tokyo
Grenoble

Rouen, Saint Etienne, Pau, Saint-Etienne, Rennes,
Marne-la-Vallée, Aix, Orléans, Toulouse, Montpellier, Amiens,
Strasbourg, Lyon, Lens, Avignon, Clermont-Ferrand, Straatsburg,
Caen, Bayonne, Limoges
huis (house)

woning, gebouw, pand, auto, straat, kantoor, kamer,
boerderij, tuin, winkel, kerk, brug, huisje, appartement,
hotel, flat, muur, boom, paleis, villa

house, building, house, car, street, office, room,
farm, garden, shop, church, bridge, small house, appartment,
hotel, flat, wall, tree, palace, villa
verliefdheid (enamour, love)

jaloezie, verraad, afgunst, weerzin, romance, hartstocht, overspel,
passie, erotik, vriendschap, obsessie, schuldgevoelen, fascinatie,
vergankelijkheid, seksualiteit, animositeit, seks, lust, verlangen, zeeroof

jealousy, treason, envy, dislike, romance, passion, adultery,
passion, erotics, friendship, obsession, feelings of guilt, fascination,
transiency, sexuality, animosity, sex, lust, desire, piracy
broccoli, prei, spruitje, knolselderij, andijvie, courgette, sperzieboon, zuurkool, worteltje, bleekselderij, bloemkool, snijboon, aubergine, peen, zilveruitje, ijsbergsla, koolsoort, winterpeen, doperwtjes, komkommer

broccoli, leek, sprout, celeriac, endive, zucchini, butter bean, sauerkraut, carrot, blanched celery, cauliflower, haricot, aubergine, carrot, onion, iceberg lettuce, cabbage, carrot, peas, cucumber
Another application: Parsing!

- Use automatically parsed corpus to learn *selection restrictions*

- *Bier drinkt de vrouw* *Beer, the woman drinks*

- Lexical features:
  - $d(\text{woman}, \text{obj1}, \text{drink})$ $d(\text{beer}, \text{su}, \text{drink})$
  - $d(\text{woman}, \text{su}, \text{drink})$ $d(\text{beer}, \text{obj1}, \text{drink})$

- Such features are too infrequent to be useful; the training corpus is too small to estimate weights for those features
Estimate such features from parsed corpus

- \( zd(\text{obj1}, \text{verb}, \text{fn}(N)) \) instead of \( d(\text{woman}, \text{obj1}, \text{drink}) \)

- This parse contains a direct object of a verb, and the frequency of \( d(\text{woman}, \text{obj1}, \text{drink}) \) in a large corpus is \( N \)

- \( \text{fn:} \)
  - 0: has been seen less than 10 times
  - 1: has been seen 10-100 times
  - . . .
  - 4: has been seen \( > \) 10,000 times
## Experiment

<table>
<thead>
<tr>
<th>Accuracy previous</th>
<th>rate previous</th>
<th>Accuracy new</th>
<th>rate new</th>
</tr>
</thead>
<tbody>
<tr>
<td>83.6</td>
<td>79.2</td>
<td>83.9</td>
<td>80.3</td>
</tr>
</tbody>
</table>
Future work

- tune parameters
- incorporate semantic similarity results
It’s Free!

http://www.let.rug.nl/~vannoord/alp/Alpino/

http://www.let.rug.nl/~vannoord/trees/

http://www.let.rug.nl/~gosse/Sets/