Alpino system

- Constructional HPSG grammar of Dutch based on OVIS grammar (van Noord, Bouma, Koeling, and Nederhof 1999)
- Robustness provided by unknown word heuristics and the possibility to skip parts of the input string.
- Lexicon (semi-)automatically constructed from Celex and Parole (Bouma 2001)
- Part-of-speech tagger filters out unlikely lexical entries before parsing (Prins and van Noord 2001)
- Stochastic Attribute Value Grammars (van Noord and Malouf 2005)

Alpino produces detailed syntactic parses, but no semantics

Suffers from some of the challenges typically facing deep analysis systems:
- Efficiency
- Robustness
- Ambiguity
- Coverage

Aan Charles Masterman, een collega uit de eerste ministeriele jaren, was die neiging al eerder opgevallen.

‘This tendency was already conspicuous to Charles Masterman, a colleague from the first ministerial years.’ ⇒ 521,472 readings

guessing: ‘Aan’ is just ‘aan’
guessing: ‘Aan’ is a name
guessing: ‘Aan Charles’ is a name
guessing: ‘Aan Charles Masterman’ is a name
guessing: ‘Aan Charles’ is the adjective form of a name
guessing: ‘Charles’ is a name
guessing: ‘Charles Masterman’ is a name
guessing: ‘Charles’ is the adjective form of a name
guessing: ‘Charles’ is a noun
guessing: ‘Masterman’ is a name
guessing: ‘Masterman’ is a noun
guessing: ‘ministeriele’ is ‘ministerile’ without diacritics
guessing: ‘ministeriele’ is a compound with ‘iele’
guessing: ‘ministeriele’ is a compound with ‘steriele’
Stochastic Attribute Value Grammars

- Suppose we have a set of sentences \( W \), a set of parses \( X \), and a function \( Y(w) \), the set of parses whose yield is \( w \in W \).
- And, suppose the parses can be characterized by a set of relevant features \( f_i(x) \).
- We want a distribution \( p(x|w) \) which gives us the conditional probability of a parse \( x \) given a sentence \( w \):

\[
p(x|w) = \frac{\prod_i \theta_i^{f_i(x)}}{\sum_y \prod_i \theta_i^{f_i(y)}}
\]

Stochastic Attribute Value Grammars

- For a context free grammar, the feature \( f_i(x) \) is the number of times rule \( i \) is used in the derivation of \( x \).
- The parameter \( \theta_i \) is the probability of rule \( i \), chosen to maximize the likelihood of a training corpus.
- The model then reduces to:

\[
p(x|w) = \frac{\prod_{r \in x} p(r)}{\sum_{y \in Y(w)} \prod_{t \in y} p(t)}
\]

Stochastic Attribute Value Grammars

- Nice and simple, but assumes statistical independence of \( r \)'s:

\[
p(r_1, r_2) = p(r_1) \times p(r_2)
\]

In general, independence of features does not hold

- Unification produces dependencies between rule applications
- Taking into account lexical and constructional properties of parses can create sets of overlapping features
- MaxEnt models provide a more general and flexible principle to guide the choice of parameter values

Stochastic Attribute Value Grammars

- We want a model \( q \) that satisfies the constraints imposed by the empirical distribution \( \hat{p} \):

\[
E_{\hat{p}}[f_i] = E_q[f_i]
\]

- Maximum Entropy Principle (Jaynes 1957)
In the absence of additional information, we should assume that all events have equal probability. In other words, we should choose the distribution which maximizes the entropy:

\[
H(p) = -\sum_x p(x) \log p(x)
\]
Stochastic Attribute Value Grammars

- So, we want to find the distribution which minimizes $D(\tilde{p} || p)$ and also maximizes $H(p)$.

- The solution to this constrained optimization problem yields the Conditional log-linear distributions (Abney, 1997; Berger, et al. 1997; Johnson et al. 1999):

For a parse $x$ of sentence $w$:

$$p(x|w) = \frac{\exp(\sum_i \lambda_i f_i(x, w))}{\sum_y \exp(\sum_i \lambda_i f_i(x, y))}$$

Parameter Estimation

- We select weights $\lambda_i$ to minimize the KL divergence, or, equivalently, maximize the log likelihood of the training data:

$$L(\lambda) = \sum_w \tilde{p}(w) \sum_{x \in Y(w)} p(x|w) \log p(x|w; \lambda)$$

- Improved Iterative Scaling (Della Pietra, Della Pietra, Lafferty 1997) or general optimization methods (conjugate gradient, quasi-Newton) can be used to maximize $L$ with respect to $\lambda$

Treebank

- Training and evaluating models requires a set of correct parses

- The Alpino Treebank contains detailed dependency graphs for approximately 7,100 sentences (145,000 words) of Dutch newspaper text

- Use of dependency relations provides compatibility with Corpus Gesproken Nederlands

- Also abstracts away from some of the details of the grammar

Dependency relations

*Cathy zag hen wild zwaaien.*
Evaluation

• Concept accuracy of a dependency graph (by analogy with word accuracy):

\[
CA^i = 1 - \frac{D_i^f}{\max(D_g^i, D_p^i)}
\]

where:

– \(D_p^i\) is the number of relations produced by the parser for sentence \(i\)
– \(D_g^i\) is the number of relations in the treebank parse
– \(D_i^f\) is the number of incorrect and missing relations produced by the parser

Evaluation

• To compute the accuracy of the parser on a corpus, we can compute mean \(CA^i\)

• Given that shorter sentences are typically much easier, a more informative measure is the total \(CA\) score:

\[
CA = 1 - \frac{\sum_i D_i^f}{\max(\sum_i D_g^i, \sum_i D_p^i)}
\]

• To emphasize the performance of the stochastic parse selection component, we also define error reduction:

\[
CA_\kappa = \frac{CA - \text{baseline } CA}{\text{best } CA - \text{baseline } CA}
\]

Features

- \(r1\) (Rule) 324
- \(r2\) (RuleM, Int, RuleD) 2679
- \(mf\) (Cat1, Cat2) 727
- \(f1\) (Pos) 451
- \(f2\) (Word, Pos) 2553
- \(\text{dep23}\) (ArgPos, Rel, Pos) 478
- \(\text{dep34}\) (ArgWord, ArgPos, Rel, Pos) 6241
- \(\text{dep35}\) (ArgWord, ArgPos, Rel, Word, Pos) 6880
- \(h1\) (Ident) 44
- misc boolean features 12

Training

• To construct the training distribution, each sentence in the training set was parsed using the Alpino grammar

• Each parse was scored by comparison with the correct parse for the sentence in the treebank

• We give each parse a weight proportional to its \(CA\)

• Rather than consider all parses, we generate (up to) 1,000 parses per sentence, and add a random selection of (up to) 250 parses to the training set (Osborne 2000)
Overtraining

- MaxEnt models with many features are capable of representing very complex decision boundaries
- If we don’t have enough training data (and we never do...), the model will ‘learn’ accidental properties of the particular training sample we selected
- One way to control this overtraining is by limiting the capacity of the model to represent complex concepts

Regularization

- Capacity control may be achieved by smoothing the parameter estimates via a ‘roughness’ penalty:
  \[ L'(\lambda) = L(\lambda) - \frac{1}{2\sigma^2} \sum \lambda_i^2 \]
- This bounds the values that the parameters can range over, creating a preference for models which are (in some sense) simpler
- This is equivalent to assuming a Gaussian prior distribution for parameter values

<table>
<thead>
<tr>
<th>(\sigma^2)</th>
<th>F-score</th>
<th>CA%</th>
<th>CA_%</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>73.41</td>
<td>72.07</td>
<td>57.46</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>76.48</td>
<td>75.26</td>
<td>69.52</td>
<td>12</td>
</tr>
<tr>
<td>100</td>
<td>78.01</td>
<td>76.89</td>
<td>75.65</td>
<td>30</td>
</tr>
<tr>
<td>1000</td>
<td>78.65</td>
<td>77.53</td>
<td>78.14</td>
<td>80</td>
</tr>
<tr>
<td>10000</td>
<td>78.42</td>
<td>77.32</td>
<td>77.34</td>
<td>160</td>
</tr>
<tr>
<td>100000</td>
<td>77.81</td>
<td>76.72</td>
<td>75.05</td>
<td>250</td>
</tr>
<tr>
<td>none</td>
<td>77.52</td>
<td>76.43</td>
<td>73.97</td>
<td>275</td>
</tr>
</tbody>
</table>

Feature Selection

- A crude but effective method of capacity control is dimensionality reduction through feature selection
- Models with very large feature sets are also more difficult to construct and use, often with little benefit
- We use a simple frequency cutoff, deleting all features which fail to distinguish fewer than \(n\) parses
- A cutoff of 2 gives a good compromise between model size and performance
### Feature Selection

<table>
<thead>
<tr>
<th>cutoff</th>
<th># features</th>
<th>F-score</th>
<th>CA%</th>
<th>CA_%</th>
<th>κ%</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>285,497</td>
<td>78.80</td>
<td>77.69</td>
<td>78.75</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>35,850</td>
<td>78.70</td>
<td>77.59</td>
<td>78.35</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20,391</td>
<td>78.65</td>
<td>77.53</td>
<td>78.14</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15,546</td>
<td>78.60</td>
<td>77.49</td>
<td>77.97</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9,619</td>
<td>78.58</td>
<td>77.47</td>
<td>77.90</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5,748</td>
<td>78.53</td>
<td>77.40</td>
<td>77.65</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>3,658</td>
<td>78.33</td>
<td>77.21</td>
<td>76.92</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>2,120</td>
<td>78.05</td>
<td>76.94</td>
<td>75.88</td>
<td></td>
</tr>
</tbody>
</table>

### Performance

- Complete Alpino system uses a beam search to extract the best (?) parse from a complete packed forest
- Also evaluated on newly annotated treebank entries

<table>
<thead>
<tr>
<th>corpus</th>
<th>sents</th>
<th>length</th>
<th>F-score</th>
<th>CA%</th>
<th>CA_%</th>
<th>κ%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpino</td>
<td>7136</td>
<td>19.7</td>
<td>85.78</td>
<td>84.66</td>
<td>72.30</td>
<td></td>
</tr>
<tr>
<td>CLEF</td>
<td>446</td>
<td>11.3</td>
<td>90.99</td>
<td>88.68</td>
<td>74.04</td>
<td></td>
</tr>
<tr>
<td>Trouw</td>
<td>500</td>
<td>17.0</td>
<td>88.94</td>
<td>87.86</td>
<td>80.51</td>
<td></td>
</tr>
</tbody>
</table>