A generative re-ranking model for dependency parsing

Federico Sangati, Willem Zuidema and Rens Bod

Institute for Logic, Language and Computation
University of Amsterdam

November 9, 2009
Discriminative vs. Generative models

**Discriminative Models**

- Parsing as a classification task.
- Transition-based parsers. (Nivre and Hall, 2005)
- Graph-based parsers. (McDonald, 2006)
- STATE-OF-THE-ART! (Buchholz et al., 2006; Nivre et al., 2007)

**Probabilistic Generative Models**

- Define probabilities over structures. (Eisner, 1996)
- Perform more poorly... although not much represented in the last evaluation challenges.
- Very important for many NLP tasks (SR, MT, NLG, ...): need probabilities.
The idea

Is there a principled way of combining the two?

- Discriminative model provides the k-best candidates.
- Generative model computes the prob. of each candidate.
- Selects the one with max. probability (re-ranking).
- Generative model trained on the training corpus but NOT on the output of the discriminative model.

Motivation

- Implement and compare different generative models...
- without implementing different parsers (we actually don’t need any parser).
- ‘Parser simulator’\textsuperscript{a} methodology.

\textsuperscript{a}Reut Tsarfaty terminology
Decomposition

Reverse the process: we can decompose any given structure into **events** and corresponding **conditioning contexts**.

Example

A generative model chooses each dependent $D$ of a node $N$ conditioned on $N$ and their relative position (left, right).

$$P(D | N \text{ direction})$$

Event: $D$ is a right dependent of $N$. ($D \, N \, R$)

Conditioning context: $N$ has a right dependent. ($N \, R$)
Decomposition

Decompose each dependency structure in the training corpus, and keep track of the frequency of each event and conditioning context.

Training corpus

\[
P(D | N \text{ direction})
\]

<table>
<thead>
<tr>
<th>Events</th>
<th>Freq.</th>
<th>Cond. Contexts</th>
<th>Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>won EOS L</td>
<td>1</td>
<td>EOS L</td>
<td>4</td>
</tr>
<tr>
<td>lost EOS L</td>
<td>1</td>
<td>EOS L</td>
<td>4</td>
</tr>
<tr>
<td>STOP EOS L</td>
<td>2</td>
<td>EOS L</td>
<td>4</td>
</tr>
<tr>
<td>STOP EOS R</td>
<td>2</td>
<td>EOS R</td>
<td>2</td>
</tr>
<tr>
<td>Obama won L</td>
<td>1</td>
<td>won L</td>
<td>2</td>
</tr>
<tr>
<td>STOP won L</td>
<td>1</td>
<td>won L</td>
<td>2</td>
</tr>
<tr>
<td>STOP Obama L</td>
<td>1</td>
<td>Obama L</td>
<td>1</td>
</tr>
<tr>
<td>STOP Obama R</td>
<td>1</td>
<td>Obama R</td>
<td>1</td>
</tr>
</tbody>
</table>
Re-ranking phase

Decomposition

A given candidate structure can be decomposed into:

- events \((e_1, e_2, \ldots, e_n)\)
- conditioning contexts \((c_1, c_2, \ldots, c_n)\).

The probability of the structure:

\[
\prod_{i=1}^{n} \frac{f(e_i)}{f(c_i)}
\]

Test structure

\[
P( D | N \text{ direction } )
\]

<table>
<thead>
<tr>
<th>Events</th>
<th>Freq.</th>
<th>Cond. Contexts</th>
<th>Freq.</th>
<th>(f(e_i)/f(c_i))</th>
</tr>
</thead>
<tbody>
<tr>
<td>won EOS L</td>
<td>1</td>
<td>EOS L</td>
<td>4</td>
<td>1/4</td>
</tr>
<tr>
<td>STOP EOS L</td>
<td>2</td>
<td>EOS L</td>
<td>4</td>
<td>1/2</td>
</tr>
<tr>
<td>STOP EOS R</td>
<td>2</td>
<td>EOS R</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Obama won L</td>
<td>1</td>
<td>won L</td>
<td>2</td>
<td>1/2</td>
</tr>
<tr>
<td>STOP won L</td>
<td>1</td>
<td>won L</td>
<td>2</td>
<td>1/2</td>
</tr>
<tr>
<td>STOP Obama L</td>
<td>1</td>
<td>Obama L</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>STOP Obama R</td>
<td>1</td>
<td>Obama R</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1/32
Important

The only thing to define: how a generative model decomposed a structure into events.

Provided

- a way of decomposing a given structure into events,
- a consistent way of representing them

both training and re-ranking phases can be performed identically for many different generative models.
Eisner model

Generative model inspired by the work of Eisner, 1996.

- Nodes are generated recursively in a **top-down** manner.
- Left and right children are generated as two separate **Markov sequences** of nodes, each conditioned on sibling and ancestral information (**context**).

$$P(T(N)) = \prod_{l=1}^{L} P(D_{\downarrow l}|\text{context}) \cdot P(T(D_{\downarrow l})) \times \prod_{r=1}^{R} P(D_{\uparrow r}|\text{context}) \cdot P(T(D_{\uparrow r}))$$
The feature space

\[ P(D|\text{context}) = \]

\[ P(\text{dist}(N, D), \text{term}(D), \text{word}(D), \text{tag}(D)|N, S, G, \text{dir}) \]

distance \quad \text{is terminal?}

\[ \text{grand parent} \quad \downarrow \]

\[ \text{previous sister} \quad \uparrow \]

Breaking down

\[ P(\text{dist}(N, D), \text{term}(D), \text{word}(D), \text{tag}(D)|N, S, G, \text{dir}) = \]

\[ P(\text{tag}(D)|H, S, G, \text{dir}) \times \]

\[ P(\text{word}(D)|\text{tag}(D), H, S, G, \text{dir}) \times \]

\[ P(\text{term}(D)|\text{word}(D), \text{tag}(D), H, S, G, \text{dir}) \times \]

\[ P(\text{dist}(P, D)|\text{term}(D), \text{word}(D), \text{tag}(D), H, S, G, \text{dir}) \]

Backoff

\[ P(\text{tag}(D)|H, S, G, \text{dir}) \]

reduction list:

\[ \text{wt}(H), \text{wt}(S), \text{wt}(G), \text{dir} \]
\[ \text{wt}(H), \text{wt}(S), \text{t}(G), \text{dir} \]
\[ \text{wt}(H), \text{t}(S), \text{t}(G), \text{dir} \]
\[ \text{t}(H), \text{wt}(S), \text{t}(G), \text{dir} \]
\[ \text{t}(H), \text{t}(S), \text{t}(G), \text{dir} \]

\[ P(\text{word}(D)|\text{tag}(D), H, S, G, \text{dir}) \]

reduction list:

\[ \text{wt}(H), \text{t}(S), \text{dir} \]
\[ \text{t}(H), \text{t}(S), \text{dir} \]

\[ P(\text{term}(D)|\text{word}(D), \text{tag}(D), H, S, G, \text{dir}) \]

reduction list:

\[ \text{tag}(D), \text{wt}(H), \text{t}(S), \text{dir} \]
\[ \text{tag}(D), \text{t}(H), \text{t}(S), \text{dir} \]

\[ P(\text{dist}(P, D)|\text{term}(D), \text{word}(D), \text{tag}(D), H, S, G, \text{dir}) \]

reduction list:

\[ \text{word}(D), \text{tag}(D), \text{t}(H), \text{t}(S), \text{dir} \]
\[ \text{tag}(D), \text{t}(H), \text{t}(S), \text{dir} \]
Results

Unlabeled Parsing

- Corpus: **Penn WSJ-40** converted to dependency structure according to Collins (1999).
- Training/Test: sec 02-21 / sec 22 (gold pos-tags)
- **UAS**: Unlabeled attachment score
- Discriminative model: **MST parser**, 2\(^{nd}\) order (McDonald, 2006)

![Graph showing UAS for different values of k-best with Oracle-Best, Oracle-worst, and Reranked models.](image)

<table>
<thead>
<tr>
<th>k-best</th>
<th>Oracle best</th>
<th>Oracle worst</th>
<th>Reranked</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>92.58</td>
<td>92.58</td>
<td>92.58</td>
</tr>
<tr>
<td>2</td>
<td>94.22</td>
<td>88.66</td>
<td>92.89</td>
</tr>
<tr>
<td>3</td>
<td>95.05</td>
<td>87.04</td>
<td>93.02</td>
</tr>
<tr>
<td>4</td>
<td>95.51</td>
<td>85.82</td>
<td>93.02</td>
</tr>
<tr>
<td>5</td>
<td>95.78</td>
<td>84.96</td>
<td>93.02</td>
</tr>
<tr>
<td>6</td>
<td>96.02</td>
<td>84.20</td>
<td>93.06</td>
</tr>
<tr>
<td>7</td>
<td>96.23</td>
<td>83.62</td>
<td>93.09</td>
</tr>
<tr>
<td>8</td>
<td>96.40</td>
<td>83.06</td>
<td>93.02</td>
</tr>
<tr>
<td>9</td>
<td>96.54</td>
<td>82.57</td>
<td>92.97</td>
</tr>
<tr>
<td>10</td>
<td>96.64</td>
<td>82.21</td>
<td>92.96</td>
</tr>
<tr>
<td>100</td>
<td>98.48</td>
<td>73.30</td>
<td>92.32</td>
</tr>
<tr>
<td>1000</td>
<td>99.34</td>
<td>64.86</td>
<td>91.47</td>
</tr>
</tbody>
</table>

Federico Sangati, Willem Zuidema and Rens Bod

A generative re-ranking model for dependency parsing
Conclusions

- Combining discriminative and generative models: improvements over state-of-the-art results.
- Open question: can we come up with a better generative model?
- Efficiency:
  - MST parser: training + parse 1-best test $\rightarrow$ 6 h.
  - Our method: training + re-ranking 100-best $\rightarrow$ 5 min!
- ‘Parser simulator’: efficient framework to evaluate many different generative models.
- Explore different feature spaces.
Thank you!

http://staff.science.uva.nl/~fsangati

{f.sangati,zuidema,rens.bod}@uva.nl
UAS improvement of the reranked 7-best over the MST 1-best

Improvement UAS