Generative and Discriminative Models in Statistical Parsing

Michael Collins

MIT

December 11, 2009
Canadian Utilities had 1988 revenue of C$ 1.16 billion, mainly from its natural gas and electric utility businesses in Alberta, where the company serves about 800,000 customers.
Generative and discriminative models for parsing:
  - SPATTER
  - 5 lexicalized models

Two hybrid generative/discriminative models
Discriminative Model 1: SPATTER

(Magerman 1995; Jelinek et al 1994)

- Input sentence $= x$, parse tree $y$ represented as a sequence of decisions, $d_1d_2 \ldots d_n$.

\[
P(y|x) = \prod_{i=1}^{n} P(d_i|d_1 \ldots d_{i-1}, x)
\]

$P(d_i|d_1 \ldots d_{i-1}, x)$ estimated using decision trees
The Label-Bias Problem

\[ P(y|x) = \prod_{i=1}^{n} P(d_i|d_1 \ldots d_{i-1}, x) \]

- If you think the label-bias problem is bad for MEMMs, you should try parsing...

- bill VP NP CC N V N
  
  bill VB N and jane likes bill

- bill likes mary
Discriminative Model 2: Lexical Dependencies (C, 1996)

The “probability” for this parse tree:

\[
P(S-VP-NP | \text{Mary saw Bill}) \times P(VP-V-NP | \text{Mary saw Bill}) \times P(\text{ROOT} | \text{Mary saw Bill})
\]
Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D1: SPATTER</strong></td>
<td><strong>84.1</strong></td>
</tr>
<tr>
<td>D2</td>
<td><strong>85.5</strong></td>
</tr>
<tr>
<td>G1</td>
<td>87.8</td>
</tr>
<tr>
<td>G2</td>
<td>89.6</td>
</tr>
<tr>
<td>D4</td>
<td>91.1</td>
</tr>
</tbody>
</table>

- **D1**: \( P(y|x) = \prod_{i=1}^{n} P(d_i|d_1 \ldots d_{i-1}, x) \)

- **D2**: \( P(\text{S-VP-NP}|\text{Mary saw Bill}) \)

- **D2** gives some improvements, and is considerably simpler, but it’s pretty suspect as a probabilistic model
A parse tree is represented as a set of *spines* and *adjunctions*:
Markov Grammars (continued)

Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case S, VP)

The automata generate sequences of modifier spines at each level of the tree
Markov Grammars (continued)

$P(\text{NP-cake}|\text{VP-v-eats, RIGHT, ADJACENT})$

- Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case $S, \ VP$)

- The automata generate sequences of modifier spines at each level of the tree
Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case $S, VP$)

The automata generate sequences of modifier spines at each level of the tree
Markov Grammars (continued)

Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case $S, VP$)

The automata generate sequences of modifier spines at each level of the tree

$$P(\text{STOP}|\text{VP-}v\text{-eats, RIGHT, !ADJACENT})$$
Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case $S$, $VP$)

The automata generate sequences of modifier spines at each level of the tree
Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case $S$, $VP$).

The automata generate sequences of modifier spines at each level of the tree.
Markov Grammars (continued)

Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case $S$, $VP$).

The automata generate sequences of modifier spines at each level of the tree.
Each spine has a separate left/right weighted finite-state automaton (HMM) at each level of the tree (in this case S, VP).

The automata generate sequences of modifier spines at each level of the tree.
### Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1: SPATTER</td>
<td>84.1</td>
</tr>
<tr>
<td>D2</td>
<td>85.5</td>
</tr>
<tr>
<td>G1</td>
<td>87.8</td>
</tr>
<tr>
<td>G2</td>
<td>89.6</td>
</tr>
<tr>
<td>D4</td>
<td>91.1</td>
</tr>
</tbody>
</table>

- **D2**: $P(S-VP-NP|\text{Mary saw Bill})$

- **G1/G2**: $P(NP-cake|VP-v-eats, \text{RIGHT, ADJACENT, ...})$

- Markov grammars are coherent probabilistic models, and give improvements, but there are many details...
A discriminative model for dependency parsing:

\[ y^* = \arg \max_y \sum_{r \in y} w \cdot f(x, r) \]

where each \( r \) is a tuple \( \langle h, m \rangle \) representing a dependency from modifier \( m \) to head \( h \)

\( f(x, r) \) is a feature vector associated with dependency \( r \), \( w \) is a parameter vector (trained using MIRA, averaged perceptron, etc.)

A simple, direct model, allows easy incorporation of features. Very easy to replicate
A parse tree is represented as a set of spines and adjunctions:
Discriminative Model 4: a TAG-Based Model
(Carreras, C, and Koo, 2008)

Feature vectors $f(x, h, m, \sigma_h, \sigma_m, \text{POS})$ where
- $x$ is the sentence
- $h = 3$ (index of head word), $m = 5$ (index of modifier word)
- $\sigma_h$ and $\sigma_m$ are the head and modifier spines
- POS is the position being adjoined into (e.g., VP)
Discriminative Model 4: a TAG-Based Model
(Carreras, C, and Koo, 2008)

Feature vectors $f(x, h, m, \sigma_h, \sigma_m, \text{POS})$ where
- $x$ is the sentence
- $h = 3$ (index of head word), $m = 5$ (index of modifier word)
- $\sigma_h$ and $\sigma_m$ are the head and modifier spines
- POS is the position being adjoined into (e.g., VP)
Discriminative Model 4: a TAG-Based Model

(Carreras, C, and Koo, 2008)

Feature vectors $f(x, h, m, \sigma_h, \sigma_m, \text{POS})$ where

- $x$ is the sentence
- $h = 3$ (index of head word), $m = 5$ (index of modifier word)
- $\sigma_h$ and $\sigma_m$ are the head and modifier spines
- POS is the position being adjoined into (e.g., VP)
Discriminative Model 4: a TAG-Based Model

(Carreras, C, and Koo, 2008)

Feature vectors $f(x, h, m, \sigma_h, \sigma_m, \text{POS})$ where

- $x$ is the sentence
- $h = 3$ (index of head word), $m = 5$ (index of modifier word)
- $\sigma_h$ and $\sigma_m$ are the head and modifier spines
- POS is the position being adjoined into (e.g., VP)
Discriminative Model 4: a TAG-Based Model
(Carreras, C, and Koo, 2008)

Feature vectors $f(x, h, m, \sigma_h, \sigma_m, \text{POS})$ where

- $x$ is the sentence
- $h = 3$ (index of head word), $m = 5$ (index of modifier word)
- $\sigma_h$ and $\sigma_m$ are the head and modifier spines
- POS is the position being adjoined into (e.g., VP)
Discriminative Model 4: a TAG-Based Model

(Carreras, C, and Koo, 2008)

Feature vectors $f(x, h, m, \sigma_h, \sigma_m, \text{POS})$ where

- $x$ is the sentence
- $h = 3$ (index of head word), $m = 5$ (index of modifier word)
- $\sigma_h$ and $\sigma_m$ are the head and modifier spines
- POS is the position being adjoined into (e.g., VP)
Discriminative Model 4: a TAG-Based Model
(Carreras, C, and Koo, 2008)

Feature vectors \( f(x, h, m, \sigma_h, \sigma_m, \text{POS}) \) where
- \( x \) is the sentence
- \( h = 3 \) (index of head word), \( m = 5 \) (index of modifier word)
- \( \sigma_h \) and \( \sigma_m \) are the head and modifier spines
- POS is the position being adjoined into (e.g., VP)
Discriminative Model 4: a TAG-Based Model
(Carreras, C, and Koo, 2008)

Trigram dependency features:

\[
\begin{array}{ccc}
S & \rightarrow & VP \\
V & \rightarrow & v \\
NP & \rightarrow & n \\
PP & \rightarrow & p \\
boys & \rightarrow & eat \\
a & \rightarrow & cake \\
a & \rightarrow & fork \\
with & \rightarrow & \\
\end{array}
\]
Discriminative Model 4: a TAG-Based Model
(Carreras, C, and Koo, 2008)

More trigram dependency features:

```
S -> VP -> PP -> NP
```

```
boys, eat with a cake, a fork
```
## Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D1: SPATTER</strong></td>
<td><strong>84.1</strong></td>
</tr>
<tr>
<td>D2</td>
<td>85.5</td>
</tr>
<tr>
<td>G1</td>
<td>87.8</td>
</tr>
<tr>
<td>G2</td>
<td>89.6</td>
</tr>
<tr>
<td>D4</td>
<td>91.1</td>
</tr>
</tbody>
</table>

- **D1:**

  \[
  y^* = \arg \max_y \sum_{i=1}^{n} \log P(d_i|d_1 \ldots d_{i-1}, x)
  \]

- **D4:**

  \[
  y^* = \arg \max_y \sum_{r \in y} w \cdot f(x, r)
  \]
"Hybrid" Discriminative/Generative Model 1: Word Clusters (Koo, C, Carreras, 2008)

Feature vectors $f(x, h, m)$ depend heavily on lexical items, which are sparse

A semi-supervised method: use unlabeled data to induce hierarchical word clusters, then use these within features
### Results

Dependency accuracy for a 2nd order parser:

<table>
<thead>
<tr>
<th>Training size</th>
<th>Baseline</th>
<th>Clusters</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1k</td>
<td>81.95</td>
<td>85.33</td>
<td>3.38</td>
</tr>
<tr>
<td>2k</td>
<td>85.02</td>
<td>87.54</td>
<td>2.52</td>
</tr>
<tr>
<td>4k</td>
<td>87.88</td>
<td>89.67</td>
<td>1.79</td>
</tr>
<tr>
<td>8k</td>
<td>89.71</td>
<td>91.37</td>
<td>1.66</td>
</tr>
<tr>
<td>16k</td>
<td>91.14</td>
<td>92.22</td>
<td>1.08</td>
</tr>
<tr>
<td>32k</td>
<td>92.09</td>
<td>93.21</td>
<td>1.12</td>
</tr>
<tr>
<td>All</td>
<td>92.42</td>
<td>93.30</td>
<td>0.88</td>
</tr>
</tbody>
</table>
"Hybrid" Discriminative/Generative Model 2
(Suzuki et al, 2009)

Step 1 Train a CRF-style dependency model on the labeled examples

\[ y^* = \arg\max_y \sum_{r \in y} w \cdot f(x, r) \]

Step 2 Use the model from step 1 to produce parse trees on unlabeled data, and estimate generative models

\[ P(y, x; \theta_i) \text{ for } i = 1 \ldots k \]

(typically \( k \approx 100 \))

Step 3 Add new features \( \log P(y, x; \theta_i) \) for \( i = 1 \ldots k \) to the supervised model, and retrain
The $k$ generative models are derived directly from the original feature vectors $f(x, r)$!

First partition the feature vector into $k$ sets of disjoint features (typically by feature type)

Next, define a naive-bayes model for each partition
Final Thoughts

- Advantages of generative models:
  - Very fast to train
  - Very useful in semi-supervised approaches
  - Invaluable as language models in speech recognition, machine translation
  - Better than discriminative models with small amounts of training data? (I’m skeptical about this...)

- Advantages of discriminative models:
  - Very easy to incorporate new features (including features induced from unlabeled data)
  - Easy to implement and replicate (no issues of smoothing, independence assumptions etc. — all you need is the feature definitions)