Discriminative Log-Linear Grammars with Latent Variables

by Slav Petrov and Dan Klein (Berkeley)
NIPS 2007

Presented by:
Eugene Weinstein, NYU
February 19th, 2008
PCFG for parsing

- Probabilistic/Stochastic Context Free Grammar
- Just a CFG with likelihoods/frequencies
- Example [Manning and Schütze ‘99]

A simple PCFG (in CNF)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>1.0</td>
</tr>
<tr>
<td>PP → P NP</td>
<td>1.0</td>
</tr>
<tr>
<td>VP → V NP</td>
<td>0.7</td>
</tr>
<tr>
<td>VP → VP PP</td>
<td>0.3</td>
</tr>
<tr>
<td>P → with</td>
<td>1.0</td>
</tr>
<tr>
<td>V → saw</td>
<td>1.0</td>
</tr>
<tr>
<td>NP → astronomers</td>
<td>0.1</td>
</tr>
<tr>
<td>NP → ears</td>
<td>0.18</td>
</tr>
<tr>
<td>NP → saw</td>
<td>0.04</td>
</tr>
<tr>
<td>NP → stars</td>
<td>0.18</td>
</tr>
<tr>
<td>NP → telescopes</td>
<td>0.1</td>
</tr>
</tbody>
</table>
PCFG: Two Parses

\[ P(t_1) = 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \times 0.18 \times 1.0 \times 0.18 = 0.0009072 \]

\[ P(t_2) = 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \times 0.18 \times 1.0 \times 1.0 \times 0.18 = 0.0006804 \]

\[ P(w_{15}) = P(t_1) + P(t_2) = 0.0015876 \]
Training PCFGs

• Generative training is fast (via Expectation-Maximization)
• Discriminative training generally achieves better results
• Objective function computation requires re-parsing data set with each training iteration: slow

• In this paper: a discriminative formulation for learning log-linear PCFGs
• Introduce latent variables for more refined categories
• Heuristics and approximations for speeding up training
**Data**

- Penn Treebank: 40,000 sentences, 1M words, hand parsed
- Data format: (sentence, parse tree) pairs \((w_i, T_i)\)

- Problem: classes are too broad, want to refine them
- e.g., NP under VP, etc.
Splitting Categories

• Introduce a subcategory for each category node

• Train new refined grammar using the same data

• Previously, tree labels are observed, now model these new subcategories as latent variables

Parse Tree $T$
Sentence $w$

Parse Derivations $t_i : T$

Model Parameters $\theta$

Grammar $G$

$S_1 \rightarrow NP_{10} \ VP_{11}$
$S_2 \rightarrow NP_{17} \ VP_{23}$
$NP_{10} \rightarrow PRP_2$
$NP_{17} \rightarrow PRP_2$

...
Log-Linear Model

- Linear model with parameter set $\theta$: $P_{\theta}(t|w_i) = \sum_{X \rightarrow \gamma \in t} \theta_{X \rightarrow \gamma}$

- Features: occurrence count of rule $X \rightarrow \gamma$ in $t$: $f_{X \rightarrow \gamma}(t)$

$$P_{\theta}(t|w_i) = \sum_{X \rightarrow \gamma \in t} \theta_{X \rightarrow \gamma} \cdot f_{X \rightarrow \gamma}(t) = \langle \theta, f(t) \rangle$$

- Log-linear model:

$$P_{\theta}(t|w_i) = \frac{\exp \sum_{X \rightarrow \gamma \in t} \theta_{X \rightarrow \gamma}}{Z(\theta, w_i)} = \frac{\exp \langle \theta, f(t) \rangle}{Z(\theta, w_i)}$$

- Partition function: $Z(\theta, w_i) = \sum_{t:T_i} \exp \langle \theta, f(t) \rangle$
Generative Training

- Maximize joint log-likelihood of data, labels (trees):
  \[ L_{\text{joint}}(\theta) = \log \prod_i P_\theta(T_i, w_i) = \log \prod_i \sum_{t:T_i} P_\theta(t, w_i) \]
  \[ \theta^* = \arg\max_{\theta} \left( \log \prod_i \sum_{t:T_i} P_\theta(t, w_i) \right) \]

- Apply EM algorithm
  - E step: for each production \( X \rightarrow \gamma \), compute expected count in each \( t \) consistent with \( T \)
  - Inside/outside: efficient recursive algorithm
  - M step: update
    \[ \theta_{X \rightarrow \gamma} = \log \frac{\sum_T \mathbb{E}_\theta[f_{X \rightarrow \gamma}(t)|T]}{\sum_{\gamma'} \sum_T \mathbb{E}_\theta[f_{X \rightarrow \gamma'}(t)|T]} \]
Discriminative Training

• Maximize conditional log-likelihood

\[ \mathcal{L}_{\text{cond}}(\theta) = \log \prod_i P_{\theta}(T_i|w_i) = \log \prod_i \sum_{t:T_i} P_{\theta}(t|w_i) \]

\[ \theta^* = \arg\max_{\theta} \left( \log \prod_i \sum_{t:T_i} P_{\theta}(t|w_i) \right) \]

• Optimization solved via gradient ascent, differential

\[ \frac{\partial \mathcal{L}_{\text{cond}}(\theta)}{\partial \theta_{X \rightarrow \gamma}} = \sum_i \left( \mathbb{E}_\theta [f_{X \rightarrow \gamma}(t)|T_i] - \mathbb{E}_\theta [f_{X \rightarrow \gamma}(t)|w_i] \right) \]

• Each differential computed over training corpus - slow!
Speeding Up Training

- Hierarchical splitting: at each training iteration, split one category in two
- Only derivations with the split category re-calculated
Pruning approach: start with fully refined tree, prune unlikely splits of categories
Training Efficiency

- Drastic improvement in training time when starting out with full subcategory tree and pruning unlikely nodes
Experiments

- Train grammars on WSJ corpus in the Penn Treebank
- 1M words, 40K sentences

![Graph showing parsing performance (F1-score)]

- **F1-score**
  - Discriminative: 91%
  - Generative: 89%
  - State-of-the-art (Petrov et al. '06): 87%

![Bar chart showing F1-score and Exact Match]

- **F1-score**
  - Discriminative: 90%
  - Generative: 88%
  - State-of-the-art (Petrov et al. '06): 87%

- **Exact Match**
  - Discriminative: 40%
  - Generative: 37%
  - State-of-the-art (Petrov et al. '06): 31%
Regularization

- Introduce L1 or L2 regularization term into objective:

\[ \mathcal{L}'_{\text{cond}}(\theta) = \mathcal{L}_{\text{cond}}(\theta) - \frac{1}{2} \sum_{X \rightarrow \gamma} \frac{|\theta_{X \rightarrow \gamma}|}{\sigma} \]

\[ \mathcal{L}''_{\text{cond}}(\theta) = \mathcal{L}_{\text{cond}}(\theta) - \sum_{X \rightarrow \gamma} \left( \frac{\theta_{X \rightarrow \gamma}}{\sigma} \right)^2 \]

- L1 regularization results in sparser solution
Model Observations

- Generative models learn subcategories like {New, San, Wall} and {York, Francisco, Street}; but not discriminative
- However, this doesn’t correlate with higher accuracy
- Generative: can merge 80% of categories without changing accuracy, discriminative: only 50%