Natural Language Processing

Learning PCFGs

Treebank PCFGs [Charniak 96]
- Use PCFGs for broad coverage parsing
- Can take a grammar right off the trees (doesn't work well):

```
ROOT \rightarrow S 1
S \rightarrow NP VP 1
NP \rightarrow PRP 1
VP \rightarrow VBD ADJP 1
....
```

Model | F1  
---|---
Baseline | 72.0

Conditional Independence?
- Not every NP expansion can fill every NP slot
  - A grammar with symbols like "NP" won't be context-free
  - Statistically, conditional independence too strong

Non-Independence
- Independence assumptions are often too strong.
  - Example: the expansion of an NP is highly dependent on the parent of the NP (i.e., subjects vs. objects).
  - Also: the subject and object expansions are correlated!

Grammar Refinement
- Example: PP attachment

They raised a point of order
Grammar Refinement

Structure Annotation

- Structure Annotation [Johnson ’98, Klein&Manning ’03]
- Lexicalization [Collins ’99, Charniak ’00]
- Latent Variables [Matsuzaki et al. 05, Petrov et al. ’06]

The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
- Structural annotation

Typical Experimental Setup

- Corpus: Penn Treebank, WSI
- Training: sections 02-21
- Development: section 22 (here, first 20 files)
- Test: section 23

- Accuracy – F1: harmonic mean of per-node labeled precision and recall.
- Here: also size – number of symbols in grammar.

Vertical Markovization

- Vertical Markov order: rewrites depend on past $k$ ancestor nodes. (cf. parent annotation)

Horizontal Markovization

- Order 1
- Order $\infty$
Unary Splits

- **Problem:** unary rewrites used to transmute categories so a high-probability rule can be used.

- **Solution:** Mark unary rewrite sites with -U

<table>
<thead>
<tr>
<th>Annotation</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>77.8</td>
<td>7.5K</td>
</tr>
<tr>
<td>UNARY</td>
<td>78.3</td>
<td>8.0K</td>
</tr>
</tbody>
</table>

Tag Splits

- **Problem:** Treebank tags are too coarse.

- **Example:** Sentential, PP, and other prepositions are all marked IN.

- **Partial Solution:** Subdivide the IN tag.

<table>
<thead>
<tr>
<th>Annotation</th>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>78.3</td>
<td>8.0K</td>
</tr>
<tr>
<td>SPLIT-IN</td>
<td>80.3</td>
<td>8.1K</td>
</tr>
</tbody>
</table>

Some Test Set Results

- Beats “first generation” lexicalized parsers.
- Lots of room to improve – more complex models next.

<table>
<thead>
<tr>
<th>Parser</th>
<th>LP</th>
<th>LR</th>
<th>F1</th>
<th>CB</th>
<th>0 CB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magerman 95</td>
<td>84.9</td>
<td>84.6</td>
<td>84.7</td>
<td>1.26</td>
<td>56.6</td>
</tr>
<tr>
<td>Collins 96</td>
<td>86.3</td>
<td>85.8</td>
<td>86.0</td>
<td>1.14</td>
<td>59.9</td>
</tr>
<tr>
<td>Unlexicalized</td>
<td>86.9</td>
<td>85.7</td>
<td>86.3</td>
<td>1.10</td>
<td>60.3</td>
</tr>
<tr>
<td>Charniak 97</td>
<td>87.4</td>
<td>87.5</td>
<td>87.4</td>
<td>1.00</td>
<td>62.1</td>
</tr>
<tr>
<td>Collins 99</td>
<td>88.7</td>
<td>88.6</td>
<td>88.6</td>
<td>0.90</td>
<td>67.1</td>
</tr>
</tbody>
</table>

Efficient Parsing for Structural Annotation

Grammar Projections

Coarse Grammar

<table>
<thead>
<tr>
<th>NP</th>
<th>VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>VP</td>
</tr>
</tbody>
</table>

Fine Grammar

<table>
<thead>
<tr>
<th>NP</th>
<th>DT 'N'</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>DT 'NP' N' DT 'NP'</td>
</tr>
</tbody>
</table>

Note: X-bar Grammars are projections with rules like NP → Y 'X' or NP → X 'Y'.
Coarse-to-Fine Pruning

For each coarse chart item $X[i,j]$, compute posterior probability:

$$\frac{P_{\text{de}}(X[i,j]) - P_{\text{det}}(X[i,j])}{P_{\text{root}}(0, n)} < \text{threshold}$$

E.g. consider the span 5 to 12:

Coarse:          \[ \begin{array}{c}
                                \vspace{1cm}
                            \end{array} \]

Refined: \[ \begin{array}{c}
                                \vspace{1cm}
                            \end{array} \]

Computing (Max-)Marginals

Inside and Outside Scores

Pruning with A*

- You can also speed up the search without sacrificing optimality
- For agenda-based parsers:
  - Can select which items to process first
  - Can do with any "figure of merit" [Charniak 98]
  - If your figure-of-merit is a valid A* heuristic, no loss of optimality [Klein and Manning 03]

A* Parsing

<table>
<thead>
<tr>
<th>Estimator</th>
<th>SK</th>
<th>INZ</th>
<th>OBLA</th>
<th>TRIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun</td>
<td>1.8/76</td>
<td>1.6/69</td>
<td>1.8/77</td>
<td>1.6/73</td>
</tr>
<tr>
<td>Verb</td>
<td>1.7/85</td>
<td>1.5/78</td>
<td>1.7/84</td>
<td>1.5/72</td>
</tr>
<tr>
<td>Adjective</td>
<td>1.6/90</td>
<td>1.4/83</td>
<td>1.6/88</td>
<td>1.4/80</td>
</tr>
<tr>
<td>Prepos</td>
<td>1.5/95</td>
<td>1.3/88</td>
<td>1.5/91</td>
<td>1.3/84</td>
</tr>
</tbody>
</table>

Lexicalization
The Game of Designing a Grammar

- Annotation refines base treebank symbols to improve statistical fit of the grammar
  - Structural annotation [Johnson '98, Klein and Manning 03]
  - Head lexicalization [Collins '99, Charniak '00]

Problems with PCFGs

- If we do no annotation, these trees differ only in one rule:
  - VP \(\rightarrow\) VP PP
  - NP \(\rightarrow\) NP PP
  - Parse will go one way or the other, regardless of words
  - We addressed this in one way with unlexicalized grammars (how?)
  - Lexicalization allows us to be sensitive to specific words

Lexicalized Trees

- Add “head words” to each phrasal node
  - Syntactic vs. semantic heads
  - Headship not in (most) treebanks
  - Usually use head rules, e.g.:
    - NP:
      - Take leftmost NP
      - Take rightmost N*
      - Take right child
    - VP:
      - Take leftmost VB*
      - Take leftmost VP
      - Take left child

Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like
  \[ VP(\text{eat}) \rightarrow VP(\text{eat}) NP(\text{food}) \text{ and NP} (\text{today}) \]
  - Never going to get these atomically off of a treebank
  - Solution: break up derivation into smaller steps

Lexical Derivation Steps

- A derivation of a local tree [Collins 99]
  - Choose a head tag and word
  - Choose a complement tag
  - Generate children (incl. adjuncts)
  - Recursively derive children
Lexicalized CKY

\[
\begin{align*}
\text{bestScore}(X, i, j, h) \\
\quad \text{if } (j = i+1) \\
\quad \quad \text{return } \text{tagScore}(X, s[i]) \\
\quad \text{else} \\
\quad \quad \text{return } \max \max \text{score}(X[h] \rightarrow Y[h] Z[h']) \cdot \text{bestScore}(Y, i, k, h) \cdot \text{bestScore}(Z, k, j, h') \\
\quad \quad \quad \quad \max \text{score}(X[h] \rightarrow Y[h'] Z[h]) \cdot \text{bestScore}(Y, i, k, h') \cdot \text{bestScore}(Z, k, j, h) 
\end{align*}
\]

Efficient Parsing for Lexical Grammars

Quartic Parsing

- Turns out, you can do (a little) better [Eisner 99]

- Gives an \(O(n^4)\) algorithm
- Still prohibitive in practice if not pruned

Pruning with Beams

- The Collins parser prunes with per-cell beams [Collins 99]
  - Essentially, run the \(O(n^4)\) CKY
  - Remember only a few hypotheses for each span \(<i,j>\).
  - If we keep \(K\) hypotheses at each span, then we do at most \(O(nK^2)\) work per span (why?)
  - Keeps things more or less cubic (and in practice is more like linear!)
- Also: certain spans are forbidden entirely on the basis of punctuation (crucial for speed)

Pruning with a PCFG

- The Charniak parser prunes using a two-pass, coarse-to-fine approach [Charniak 97+]
  - First, parse with the base grammar
  - For each \(X[i,j]\) calculate \(P(X[i,j])\)
    - This isn’t trivial, and there are clever speed ups
  - Second, do the full \(O(n^4)\) CKY
    - Skip any \(X[i,j]\) which had low (say, < 0.0001) posterior
    - Avoids almost all work in the second phase!

- Charniak et al 06: can use more passes
- Petrov et al 07: can use many more passes

Results

- Some results
  - Collins 99 – 88.6 F1 (generative lexical)
  - Charniak and Johnson 05 – 89.7 / 91.3 F1 (generative lexical / reranked)
  - Petrov et al 06 – 90.7 F1 (generative unlexical)
  - McClosky et al 06 – 92.1 F1 (gen + rerank + self-train)

- However
  - Bilexical counts rarely make a difference (why?)
  - Gildea 01 – Removing bilexical counts costs < 0.5 F1
Latent Variable PCFGs

- Annotation refines base treebank symbols to improve statistical fit of the grammar:
  - Parent annotation [Johnson '98]
  - Head lexicalization [Collins '99, Charniak '00]
  - Automatic clustering?

Learning Latent Annotations

- EM algorithm:
  - Brackets are known
  - Base categories are known
  - Only induce subcategories

Refinement of the DT tag

Hierarchical refinement
### Hierarchical Estimation Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flat Training</td>
<td>87.3</td>
</tr>
<tr>
<td>Hierarchical Training</td>
<td>88.4</td>
</tr>
</tbody>
</table>

### Refinement of the , tag

- Splitting all categories equally is wasteful:

```
, (1.00)
, (1.00)
, (1.00)
```

### Adaptive Splitting

- Want to split complex categories more
- Idea: split everything, roll back splits which were least useful

```
the (0.54)
a (0.25)
The (0.09)

a (0.61)
the (0.19)
an (0.11)
```

```
the (0.80)
the (0.18)
```

```
the (0.96)
a (0.01)
The (0.93)
No (0.01)
```

### Adaptive Splitting Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>88.4</td>
</tr>
<tr>
<td>With 50% Merging</td>
<td>89.5</td>
</tr>
</tbody>
</table>

### Number of Phrasal Subcategories

![Number of Phrasal Subcategories](image)

### Number of Lexical Subcategories

![Number of Lexical Subcategories](image)
Learned Splits

- Proper Nouns (NNP):
  
<table>
<thead>
<tr>
<th>NNP-14</th>
<th>NNP-12</th>
<th>NNP-2</th>
<th>NNP-1</th>
<th>NNP-15</th>
<th>NNP-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>J.</td>
<td>E.</td>
<td>L.</td>
<td>Bush</td>
<td>Noriega</td>
<td>Peters</td>
</tr>
<tr>
<td>New</td>
<td>San</td>
<td>Wall</td>
<td>York</td>
<td>Francisco</td>
<td>Street</td>
</tr>
</tbody>
</table>

- Personal pronouns (PRP):
  
<table>
<thead>
<tr>
<th>PRP-0</th>
<th>PRP-1</th>
<th>PRP-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>It</td>
<td>He</td>
<td>I</td>
</tr>
<tr>
<td>it</td>
<td>he</td>
<td>them</td>
</tr>
<tr>
<td>them</td>
<td>him</td>
<td></td>
</tr>
</tbody>
</table>

Relative adverbs (RBR):

<table>
<thead>
<tr>
<th>RBR-0</th>
<th>RBR-1</th>
<th>RBR-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>further</td>
<td>lower</td>
<td>higher</td>
</tr>
<tr>
<td>more</td>
<td>less</td>
<td>More</td>
</tr>
<tr>
<td>earlier</td>
<td>Earlier</td>
<td>later</td>
</tr>
</tbody>
</table>

Cardinal Numbers (CD):

<table>
<thead>
<tr>
<th>CD-0</th>
<th>CD-1</th>
<th>CD-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>78</td>
<td>58</td>
<td>34</td>
</tr>
<tr>
<td>1989</td>
<td>1990</td>
<td>1988</td>
</tr>
<tr>
<td>million</td>
<td>billion</td>
<td>trillion</td>
</tr>
<tr>
<td>1</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>78</td>
<td>58</td>
<td>34</td>
</tr>
</tbody>
</table>

Final Results (Accuracy)

<table>
<thead>
<tr>
<th></th>
<th>≤ 40 words</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Charniak&amp;Johnson '05 (generative)</td>
<td>90.1</td>
<td>89.6</td>
</tr>
<tr>
<td>Split / Merge</td>
<td>90.6</td>
<td>90.1</td>
</tr>
<tr>
<td>GER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dubey '05</td>
<td>76.3</td>
<td>-</td>
</tr>
<tr>
<td>Split / Merge</td>
<td>80.8</td>
<td>80.1</td>
</tr>
<tr>
<td>CHN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chiang et al. '02</td>
<td>80.0</td>
<td>76.6</td>
</tr>
<tr>
<td>Split / Merge</td>
<td>86.3</td>
<td>83.4</td>
</tr>
</tbody>
</table>

Still higher numbers from reranking / self-training methods

Efficient Parsing for Hierarchical Grammars

Coarse-to-Fine Inference

- Example: PP attachment

Hierarchical Pruning

- coarse:

  \[ \ldots \]

  \[ \ldots \]

  \[ \ldots \]

  \[ \ldots \]

- split in two:

  \[ \ldots \]

  \[ \ldots \]

  \[ \ldots \]

  \[ \ldots \]

- split in four:

  \[ \ldots \]

  \[ \ldots \]

  \[ \ldots \]

  \[ \ldots \]

- split in eight:

  \[ \ldots \]

  \[ \ldots \]

  \[ \ldots \]

  \[ \ldots \]
Bracket Posteriors

1621 min
111 min
35 min
15 min
(no search error)