Natural Language Processing

Parsing III
Dan Klein – UC Berkeley
Unsupervised Tagging
Unsupervised Tagging?

- AKA part-of-speech induction
- Task:
  - Raw sentences in
  - Tagged sentences out
- Obvious thing to do:
  - Start with a (mostly) uniform HMM
  - Run EM
  - Inspect results
EM for HMMs: Process

- Alternate between recomputing distributions over hidden variables (the tags) and reestimating parameters
- Crucial step: we want to tally up how many (fractional) counts of each kind of transition and emission we have under current params:

\[
\text{count}(w, s) = \sum_{i: w_i = w} P(t_i = s | w)
\]

\[
\text{count}(s \rightarrow s') = \sum_i P(t_{i-1} = s, t_i = s' | w)
\]

- Same quantities we needed to train a CRF!
**Merialdo: Setup**

- Some (discouraging) experiments [Merialdo 94]

- Setup:
  - You know the set of allowable tags for each word
  - Fix k training examples to their true labels
    - Learn $P(w|t)$ on these examples
    - Learn $P(t|t_{-1}, t_{-2})$ on these examples
  - On n examples, re-estimate with EM

- Note: we know allowed tags but not frequencies
### Merialdo: Results

<table>
<thead>
<tr>
<th>Iter</th>
<th>Correct tags (% words) after ML on 1M words</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>77.0 90.0 95.4 96.2 96.6 96.9 97.0</td>
</tr>
<tr>
<td>1</td>
<td>80.5 92.6 95.8 96.3 96.6 96.7 96.8</td>
</tr>
<tr>
<td>2</td>
<td>81.8 93.0 95.7 96.1 96.3 96.4 96.4</td>
</tr>
<tr>
<td>3</td>
<td>83.0 93.1 95.4 95.8 96.1 96.2 96.2</td>
</tr>
<tr>
<td>4</td>
<td>84.0 93.0 95.2 95.5 95.8 96.0 96.0</td>
</tr>
<tr>
<td>5</td>
<td>84.8 92.9 95.1 95.4 95.6 95.8 95.8</td>
</tr>
<tr>
<td>6</td>
<td>85.3 92.8 94.9 95.2 95.5 95.6 95.7</td>
</tr>
<tr>
<td>7</td>
<td>85.8 92.8 94.7 95.1 95.3 95.5 95.5</td>
</tr>
<tr>
<td>8</td>
<td>86.1 92.7 94.6 95.0 95.2 95.4 95.4</td>
</tr>
<tr>
<td>9</td>
<td>86.3 92.6 94.5 94.9 95.1 95.3 95.3</td>
</tr>
<tr>
<td>10</td>
<td>86.6 92.6 94.4 94.8 95.0 95.2 95.2</td>
</tr>
</tbody>
</table>
Latent Variable PCFGs
Annotation refines base treebank symbols to improve statistical fit of the grammar

- Parent annotation [Johnson ’98]
The Game of Designing a Grammar

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

- Parent annotation [Johnson ’98]
- Head lexicalization [Collins ’99, Charniak ’00]
- Automatic clustering?
Latent Variable Grammars

Parse Tree

Sentence $w$

Derivations $t : T$

Parameters $\theta$

Grammar $G$

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0 \rightarrow NP_0 \ VP_0$</td>
<td>?</td>
</tr>
<tr>
<td>$S_0 \rightarrow NP_1 \ VP_0$</td>
<td>?</td>
</tr>
<tr>
<td>$S_0 \rightarrow NP_0 \ VP_1$</td>
<td>?</td>
</tr>
<tr>
<td>$S_0 \rightarrow NP_1 \ VP_1$</td>
<td>?</td>
</tr>
<tr>
<td>$S_1 \rightarrow NP_0 \ VP_0$</td>
<td>?</td>
</tr>
<tr>
<td>$S_1 \rightarrow NP_1 \ VP_1$</td>
<td>?</td>
</tr>
<tr>
<td>$NP_0 \rightarrow PRP_0$</td>
<td>?</td>
</tr>
<tr>
<td>$NP_0 \rightarrow PRP_1$</td>
<td>?</td>
</tr>
<tr>
<td>$NP_0 \rightarrow VBD_0$</td>
<td>?</td>
</tr>
<tr>
<td>$NP_0 \rightarrow VBD_1$</td>
<td>?</td>
</tr>
</tbody>
</table>

Lexicon

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PRP_0 \rightarrow She$</td>
<td>?</td>
</tr>
<tr>
<td>$PRP_1 \rightarrow She$</td>
<td>?</td>
</tr>
<tr>
<td>$VBD_0 \rightarrow was$</td>
<td>?</td>
</tr>
<tr>
<td>$VBD_1 \rightarrow was$</td>
<td>?</td>
</tr>
<tr>
<td>$VBD_2 \rightarrow was$</td>
<td>?</td>
</tr>
</tbody>
</table>
EM algorithm:

- Brackets are known
- Base categories are known
- Only induce subcategories

Just like Forward-Backward for HMMs.
Refinement of the DT tag

DT

- the (0.50)
- a (0.24)
- The (0.08)

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

- the (0.80)
- The (0.15)
- a (0.01)

- this (0.39)
- that (0.28)
- That (0.11)

- some (0.20)
- all (0.19)
- those (0.12)

DT-1  DT-2  DT-3  DT-4
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)
```

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

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

```
, (1.00)   , (1.00)
```
Adaptive Splitting

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

- **Proper Nouns (NNP):**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NNP-12</td>
<td>John</td>
<td>Robert</td>
<td>James</td>
</tr>
<tr>
<td>NNP-2</td>
<td>J.</td>
<td>E.</td>
<td>L.</td>
</tr>
<tr>
<td>NNP-1</td>
<td>Bush</td>
<td>Noriega</td>
<td>Peters</td>
</tr>
<tr>
<td>NNP-15</td>
<td>New</td>
<td>San</td>
<td>Wall</td>
</tr>
<tr>
<td>NNP-3</td>
<td>York</td>
<td>Francisco</td>
<td>Street</td>
</tr>
</tbody>
</table>

- **Personal pronouns (PRP):**

<table>
<thead>
<tr>
<th>PRP-0</th>
<th>He</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRP-1</td>
<td>it</td>
<td>they</td>
</tr>
<tr>
<td>PRP-2</td>
<td>it</td>
<td>them</td>
</tr>
</tbody>
</table>
Learned Splits

- **Relative adverbs (RBR):**

<table>
<thead>
<tr>
<th></th>
<th>further</th>
<th>lower</th>
<th>higher</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBR-0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RBR-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RBR-2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Cardinal Numbers (CD):**

<table>
<thead>
<tr>
<th></th>
<th>one</th>
<th>two</th>
<th>Three</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD-7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD-4</td>
<td>1989</td>
<td>1990</td>
<td>1988</td>
</tr>
<tr>
<td>CD-11</td>
<td>million</td>
<td>billion</td>
<td>trillion</td>
</tr>
<tr>
<td>CD-0</td>
<td>1</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>CD-3</td>
<td>1</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>CD-9</td>
<td>78</td>
<td>58</td>
<td>34</td>
</tr>
</tbody>
</table>
## Final Results (Accuracy)

<table>
<thead>
<tr>
<th>Language</th>
<th>Method</th>
<th>≤ 40 words F1</th>
<th>all F1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ENG</strong></td>
<td>Charniak&amp;Johnson ‘05 (generative)</td>
<td>90.1</td>
<td>89.6</td>
</tr>
<tr>
<td></td>
<td><em>Split / Merge</em></td>
<td><strong>90.6</strong></td>
<td><strong>90.1</strong></td>
</tr>
<tr>
<td><strong>GER</strong></td>
<td>Dubey ‘05</td>
<td>76.3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td><em>Split / Merge</em></td>
<td>80.8</td>
<td>80.1</td>
</tr>
<tr>
<td><strong>CHN</strong></td>
<td>Chiang et al. ‘02</td>
<td>80.0</td>
<td>76.6</td>
</tr>
<tr>
<td></td>
<td><em>Split / Merge</em></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

```
S
   NP
     PRP
       They

   VP
     "?????????

   V
     raised

   NP
     DT
       a
     NN
       point

   PP
     IN
       of
     NP
       order
```
Hierarchical Pruning

coarse:...

split in two...

split in four...

split in eight...
1621 min
111 min
35 min
15 min
(no search error)
Other Syntactic Models
Parse Reranking

- Assume the number of parses is very small
- We can represent each parse $T$ as an arbitrary feature vector $\varphi(T)$
  - Typically, all local rules are features
  - Also non-local features, like how right-branching the overall tree is
  - [Charniak and Johnson 05] gives a rich set of features
K-Best Parsing

[Huang and Chiang 05, Pauls, Klein, Quirk 10]
Dependency Parsing

- Lexicalized parsers can be seen as producing *dependency trees*

- Each local binary tree corresponds to an attachment in the dependency graph
Dependency Parsing

- Pure dependency parsing is only cubic [Eisner 99]

Some work on *non-projective* dependencies
- Common in, e.g. Czech parsing
- Can do with MST algorithms [McDonald and Pereira 05]
Shift-Reduce Parsers

- Another way to derive a tree:

  Parsing
  - No useful dynamic programming search
  - Can still use beam search [Ratnaparkhi 97]
Data-oriented parsing:

- Rewrite large (possibly lexicalized) subtrees in a single step

Formally, a tree-insertion grammar
- Derivational ambiguity whether subtrees were generated atomically or compositionally
- Most probable parse is NP-complete
TIG: Insertion

$\phi$

$\psi$

$\phi'$

$\psi'$

$\phi$

$\psi$

$\phi'$

$\psi'$

S

NP↓

VP

V

NP↓
saw

NP

D↓

N

man

S

NP

VP

D↓

N

man

V

NP↓
saw
Tree-adjoining grammars

- Start with *local trees*
- Can insert structure with *adjunction* operators
- Mildly context-sensitive
- Models long-distance dependencies naturally
- ... as well as other weird stuff that CFGs don’t capture well (e.g. cross-serial dependencies)
**CCG Parsing**

- **Combinatory Categorial Grammar**
  - Fully (mono-) lexicalized grammar
  - Categories encode argument sequences
  - Very closely related to the lambda calculus (more later)
  - Can have spurious ambiguities (why?)

```
John ⊨ NP
shares ⊨ NP
buys ⊨ (S\NP)/NP
sleeps ⊨ S\NP
well ⊨ (S\NP)\(S\NP)
```

```
S
   /\
  S\NP John
     \   /\
      \buys shares \NP
```