Algorithms for NLP

Tagging / Parsing

Taylor Berg-Kirkpatrick – CMU

Slides: Dan Klein – UC Berkeley
So How Well Does It Work?

- Choose the most common tag
  - 90.3% with a bad unknown word model
  - 93.7% with a good one

- TnT (Brants, 2000):
  - A carefully smoothed trigram tagger
  - Suffix trees for emissions
  - 96.7% on WSJ text (SOTA is 97+%)  

- Noise in the data
  - Many errors in the training and test corpora
  - The average of interbank offered rates plummeted …
  - Probably about 2% guaranteed error from noise (on this data)
Overview: Accuracies

Roadmap of (known / unknown) accuracies:

- Most freq tag: ~90% / ~50%
- Trigram HMM: ~95% / ~55%
- TnT (HMM++): 96.2% / 86.0%
- Maxent P(t|w): 93.7% / 82.6%
- MEMM tagger: 96.9% / 86.9%
- State-of-the-art: 97+% / 89+%  
- Upper bound: ~98%

Most errors on unknown words
Common Errors

- Common errors [from Toutanovna & Manning 00]

<table>
<thead>
<tr>
<th></th>
<th>JJ</th>
<th>NN</th>
<th>NNP</th>
<th>NNPS</th>
<th>RB</th>
<th>RP</th>
<th>IN</th>
<th>VB</th>
<th>VBD</th>
<th>VBN</th>
<th>VBP</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>JJ</td>
<td>0</td>
<td>177</td>
<td>56</td>
<td>0</td>
<td>61</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>108</td>
<td>0</td>
<td>488</td>
</tr>
<tr>
<td>NN</td>
<td>244</td>
<td>0</td>
<td>103</td>
<td>0</td>
<td>12</td>
<td>1</td>
<td>1</td>
<td>29</td>
<td>5</td>
<td>6</td>
<td>19</td>
<td>525</td>
</tr>
<tr>
<td>NNP</td>
<td>107</td>
<td>106</td>
<td>0</td>
<td>132</td>
<td>5</td>
<td>0</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>427</td>
</tr>
<tr>
<td>NNPS</td>
<td>1</td>
<td>0</td>
<td>110</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>142</td>
</tr>
<tr>
<td>RB</td>
<td>72</td>
<td>21</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>138</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>295</td>
</tr>
<tr>
<td>RP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>39</td>
<td>0</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>104</td>
</tr>
<tr>
<td>IN</td>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>169</td>
<td>103</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>323</td>
</tr>
<tr>
<td>VB</td>
<td>17</td>
<td>64</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>85</td>
<td>189</td>
</tr>
<tr>
<td>VBD</td>
<td>10</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>143</td>
<td>2</td>
<td>166</td>
</tr>
<tr>
<td>VBN</td>
<td>101</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>108</td>
<td>0</td>
<td>1</td>
<td>221</td>
</tr>
<tr>
<td>VBP</td>
<td>5</td>
<td>34</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>49</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>104</td>
</tr>
<tr>
<td>Total</td>
<td>626</td>
<td>536</td>
<td>348</td>
<td>144</td>
<td>317</td>
<td>122</td>
<td>279</td>
<td>102</td>
<td>140</td>
<td>269</td>
<td>108</td>
<td>3651</td>
</tr>
</tbody>
</table>

**official knowledge**
made up the story
recently sold shares
Richer Features
Better Features

- Can do surprisingly well just looking at a word by itself:
  - **Word** the: the $\rightarrow$ **DT**
  - **Lowercased word** Importantly: importantly $\rightarrow$ **RB**
  - **Prefixes** unfathomable: un- $\rightarrow$ **JJ**
  - **Suffixes** Surprisingly: -ly $\rightarrow$ **RB**
  - **Capitalization** Meridian: CAP $\rightarrow$ **NNP**
  - **Word shapes** 35-year: d-x $\rightarrow$ **JJ**

- Then build a maxent (or whatever) model to predict tag
  - Maxent $P(t \mid w)$: 93.7% / 82.6%
Why Local Context is Useful

- Lots of rich local information!

- We could fix this with a feature that looked at the next word

- We could fix this by linking capitalized words to their lowercase versions

- Solution: discriminative sequence models (MEMMs, CRFs)

- Reality check:
  - Taggers are already pretty good on newswire text...
  - What the world needs is taggers that work on other text!
Sequence-Free Tagging?

- What about looking at a word and its environment, but no sequence information?
  - Add in previous / next word
  - Previous / next word shapes
  - Crude entity detection
  - Phrasal verb in sentence?
  - Conjunctions of these things

- All features except sequence: 96.6% / 86.8%
- Uses lots of features: > 200K
Other sequence tasks use similar models

Example: name entity recognition (NER)

Tim Boon has signed a contract extension with Leicestershire which will keep him at Grace Road.
MEMM Taggers

- Idea: left-to-right local decisions, condition on previous tags and also entire input

\[ P(t|w) = \prod_i P_{\text{ME}}(t_i|w, t_{i-1}, t_{i-2}) \]

- Train up \( P(t_i|w, t_{i-1}, t_{i-2}) \) as a normal maxent model, then use to score sequences
- This is referred to as an MEMM tagger [Ratnaparkhi 96]
- Beam search effective! (Why?)
- What about beam size 1?

- Subtle issues with local normalization (cf. Lafferty et al 01)
Because of regularization term, the more common prefixes have larger weights even though entire-word features are more specific.

**Local Context**

<table>
<thead>
<tr>
<th>State</th>
<th>Prev</th>
<th>Cur</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>at</td>
<td>Grace</td>
<td>Road</td>
</tr>
<tr>
<td>Tag</td>
<td>IN</td>
<td>NNP</td>
<td>NNP</td>
</tr>
<tr>
<td>Sig</td>
<td>x</td>
<td>Xx</td>
<td>Xx</td>
</tr>
</tbody>
</table>

**Feature Weights**

<table>
<thead>
<tr>
<th>Feature Type</th>
<th>Feature</th>
<th>PERS</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous word</td>
<td>at</td>
<td>-0.73</td>
<td>0.94</td>
</tr>
<tr>
<td>Current word</td>
<td>Grace</td>
<td>0.03</td>
<td>0.00</td>
</tr>
<tr>
<td>First char of word</td>
<td>G</td>
<td>0.45</td>
<td>-0.04</td>
</tr>
<tr>
<td>Current POS tag</td>
<td>NNP</td>
<td>0.47</td>
<td>0.45</td>
</tr>
<tr>
<td>Prev and cur tags</td>
<td>IN NNP</td>
<td>-0.10</td>
<td>0.14</td>
</tr>
<tr>
<td>Previous state</td>
<td>Other</td>
<td>-0.70</td>
<td>-0.92</td>
</tr>
<tr>
<td>Current signature</td>
<td>Xx</td>
<td>0.80</td>
<td>0.46</td>
</tr>
<tr>
<td>Prev state, cur sig</td>
<td>O-Xx</td>
<td>0.68</td>
<td>0.37</td>
</tr>
<tr>
<td>Prev-cur-next sig</td>
<td>x-Xx-Xx</td>
<td>-0.69</td>
<td>0.37</td>
</tr>
<tr>
<td>P. state - p-cur sig</td>
<td>O-x-Xx</td>
<td>-0.20</td>
<td>0.82</td>
</tr>
<tr>
<td>Total:</td>
<td></td>
<td>-0.58</td>
<td>2.68</td>
</tr>
</tbody>
</table>
Decoding

- Decoding MEMM taggers:
  - Just like decoding HMMs, different local scores
  - Viterbi, beam search, posterior decoding

- Viterbi algorithm (HMMs):
  \[ \delta_i(s) = \arg \max_{s'} P(s|s') P(w_{i-1}|s') \delta_{i-1}(s') \]

- Viterbi algorithm (MEMMs):
  \[ \delta_i(s) = \arg \max_{s'} P(s|s', w) \delta_{i-1}(s') \]

- General:
  \[ \delta_i(s) = \arg \max_{s'} \phi_i(s', s) \delta_{i-1}(s') \]
Conditional Random Fields (and Friends)
Maximum Entropy II

- Remember: maximum entropy objective

\[ L(w) = \sum_i \left( w^\top f_i(y^i) - \log \sum_y \exp(w^\top f_i(y)) \right) \]

- Problem: lots of features allow perfect fit to training set
- Regularization (compare to smoothing)

\[ \max_w \sum_i \left( w^\top f_i(y^i) - \log \sum_y \exp(w^\top f_i(y)) \right) - k\|w\|^2 \]
Derivative for Maximum Entropy

\[ L(w) = -k||w||^2 + \sum_i \left( w^\top f_i(y^i) - \log \sum_y \exp(w^\top f_i(y)) \right) \]

\[ \frac{\partial L(w)}{\partial w_n} = -2kw_n + \sum_i \left( f_i(y^i)_n - \sum_y P(y|x_i)f_i(y)_n \right) \]

Big weights are bad

Expected count of feature \( n \) in predicted candidates

Total count of feature \( n \) in correct candidates
Perceptron

- Linear model:

\[
\text{score}(t|w) = \lambda^T f(t, w)
\]

- ... that decompose along the sequence

\[
= \lambda^T \sum_i f(t_i, t_{i-1}, w, i)
\]

- ... allow us to predict with the Viterbi algorithm

\[
t^* = \arg\max_t \text{score}(t|w)
\]

- ... which means we can train with the perceptron algorithm (or related updates, like MIRA)
Conditional Random Fields

- Make a maxent model over entire taggings
  - MEMM

\[ P(t|w) = \prod_i \frac{1}{Z(i)} \exp \left( \lambda^\top f(t_i, t_{i-1}, w, i) \right) \]

- CRF

\[ P(t|w) = \frac{1}{Z(w)} \exp \left( \lambda^\top f(t, w) \right) \]

\[ = \frac{1}{Z(w)} \exp \left( \lambda^\top \sum_i f(t_i, t_{i-1}, w, i) \right) \]

\[ = \frac{1}{Z(w)} \prod_i \phi_i(t_i, t_{i-1}) \]
CRFs

- Like any maxent model, derivative is:

\[
\frac{\partial L(\lambda)}{\partial \lambda} = \sum_k \left( f_k(t^k) - \sum_t P(t|w_k)f_k(t) \right)
\]

- So all we need is to be able to compute the expectation of each feature (for example the number of times the label pair DT-NN occurs, or the number of times NN-interest occurs) **under the model distribution**

- Critical quantity: counts of posterior marginals:

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

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

- How many (expected) times is word w tagged with s?

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

- How to compute that marginal?

\[
\alpha_i(s) = \sum_{s'} \phi_i(s', s) \alpha_{i-1}(s')
\]
\[
\beta_i(s) = \sum_{s'} \phi_{i+1}(s, s') \beta_{i+1}(s')
\]
\[
P(t_i = s | w) = \frac{\alpha_i(s) \beta_i(s)}{\alpha_N(\text{END})}
\]
Global Discriminative Taggers

- **Newer, higher-powered discriminative sequence models**
  - CRFs (also perceptrons, M3Ns)
  - Do not decompose training into independent local regions
  - Can be deathly slow to train – require repeated inference on training set
- **Differences tend not to be too important for POS tagging**
- **Differences more substantial on other sequence tasks**
- **However: one issue worth knowing about in local models**
  - “Label bias” and other explaining away effects
  - MEMM taggers’ local scores can be near one without having both good “transitions” and “emissions”
  - This means that often evidence doesn’t flow properly
  - Why isn’t this a big deal for POS tagging?
  - Also: in decoding, condition on predicted, not gold, histories
[Brill 95] presents a transformation-based tagger

- Label the training set with most frequent tags

```
DT  MD  VBD  VBD .
The  can  was  rusted .
```

- Add transformation rules which reduce training mistakes
  
  - MD → NN : DT __
  - VBD → VBN : VBD __ .

- Stop when no transformations do sufficient good
- Does this remind anyone of anything?

- Probably the most widely used tagger (esp. outside NLP)
- ... but definitely not the most accurate: 96.6% / 82.0 %
## Learned Transformations

### What gets learned? [from Brill 95]

<table>
<thead>
<tr>
<th>#</th>
<th>From</th>
<th>To</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NN</td>
<td>VB</td>
<td>Previous tag is TO</td>
</tr>
<tr>
<td>2</td>
<td>VBP</td>
<td>VB</td>
<td>One of the previous three tags is MD</td>
</tr>
<tr>
<td>3</td>
<td>NN</td>
<td>VB</td>
<td>One of the previous two tags is MD</td>
</tr>
<tr>
<td>4</td>
<td>VB</td>
<td>NN</td>
<td>One of the previous two tags is DT</td>
</tr>
<tr>
<td>5</td>
<td>VBD</td>
<td>VBN</td>
<td>One of the previous three tags is VBZ</td>
</tr>
<tr>
<td>6</td>
<td>VBN</td>
<td>VBD</td>
<td>Previous tag is PRP</td>
</tr>
<tr>
<td>7</td>
<td>VBN</td>
<td>VBD</td>
<td>Previous tag is NNP</td>
</tr>
<tr>
<td>8</td>
<td>VBD</td>
<td>VBN</td>
<td>Previous tag is VBD</td>
</tr>
<tr>
<td>9</td>
<td>VBP</td>
<td>VB</td>
<td>Previous tag is TO</td>
</tr>
<tr>
<td>10</td>
<td>POS</td>
<td>VBZ</td>
<td>Previous tag is PRP</td>
</tr>
<tr>
<td>11</td>
<td>VB</td>
<td>VBP</td>
<td>Previous tag is NNS</td>
</tr>
<tr>
<td>12</td>
<td>VBD</td>
<td>VBN</td>
<td>One of the previous two tags is VBP</td>
</tr>
<tr>
<td>13</td>
<td>IN</td>
<td>WDT</td>
<td>One of next two tags is VB</td>
</tr>
<tr>
<td>14</td>
<td>VBD</td>
<td>VBN</td>
<td>One of previous two tags is VB</td>
</tr>
<tr>
<td>15</td>
<td>VB</td>
<td>VBP</td>
<td>Previous tag is PRP</td>
</tr>
<tr>
<td>16</td>
<td>IN</td>
<td>WDT</td>
<td>Next tag is VBZ</td>
</tr>
<tr>
<td>17</td>
<td>IN</td>
<td>DT</td>
<td>Next tag is NN</td>
</tr>
<tr>
<td>18</td>
<td>JJ</td>
<td>NNP</td>
<td>Next tag is NNP</td>
</tr>
<tr>
<td>19</td>
<td>IN</td>
<td>WDT</td>
<td>Next tag is VBD</td>
</tr>
<tr>
<td>20</td>
<td>JJR</td>
<td>RBR</td>
<td>Next tag is JJ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>From</th>
<th>To</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NN</td>
<td>NNS</td>
<td>Has suffix -s</td>
</tr>
<tr>
<td>2</td>
<td>NN</td>
<td>CD</td>
<td>Has character .</td>
</tr>
<tr>
<td>3</td>
<td>NN</td>
<td>JJ</td>
<td>Has character -</td>
</tr>
<tr>
<td>4</td>
<td>NN</td>
<td>VBN</td>
<td>Has suffix -ed</td>
</tr>
<tr>
<td>5</td>
<td>NN</td>
<td>VBG</td>
<td>Has suffix -ing</td>
</tr>
<tr>
<td>6</td>
<td>??</td>
<td>RB</td>
<td>Has suffix -ly</td>
</tr>
<tr>
<td>7</td>
<td>??</td>
<td>JJ</td>
<td>Adding suffix -ly results in a word.</td>
</tr>
<tr>
<td>8</td>
<td>NN</td>
<td>CD</td>
<td>The word $ can appear to the left.</td>
</tr>
<tr>
<td>9</td>
<td>NN</td>
<td>JJ</td>
<td>Has suffix -al</td>
</tr>
<tr>
<td>10</td>
<td>NN</td>
<td>VB</td>
<td>The word would can appear to the left.</td>
</tr>
<tr>
<td>11</td>
<td>NN</td>
<td>CD</td>
<td>Has character 0</td>
</tr>
<tr>
<td>12</td>
<td>NN</td>
<td>JJ</td>
<td>The word be can appear to the left.</td>
</tr>
<tr>
<td>13</td>
<td>NNS</td>
<td>JJ</td>
<td>Has suffix -us</td>
</tr>
<tr>
<td>14</td>
<td>NNS</td>
<td>VBZ</td>
<td>The word it can appear to the left.</td>
</tr>
<tr>
<td>15</td>
<td>NN</td>
<td>JJ</td>
<td>Has suffix -ble</td>
</tr>
<tr>
<td>16</td>
<td>NN</td>
<td>JJ</td>
<td>Has suffix -ic</td>
</tr>
<tr>
<td>17</td>
<td>NN</td>
<td>CD</td>
<td>Has character 1</td>
</tr>
<tr>
<td>18</td>
<td>NNS</td>
<td>NN</td>
<td>Has suffix -ss</td>
</tr>
<tr>
<td>19</td>
<td>??</td>
<td>JJ</td>
<td>Deleting the prefix un- results in a word</td>
</tr>
<tr>
<td>20</td>
<td>NN</td>
<td>JJ</td>
<td>Has suffix -ive</td>
</tr>
</tbody>
</table>
EngCG Tagger

- **English constraint grammar tagger**
  - [Tapanainen and Voutilainen 94]
  - Something else you should know about
  - Hand-written and knowledge driven
  - “Don’t guess if you know” (general point about modeling more structure!)
  - Tag set doesn’t make all of the hard distinctions as the standard tag set (e.g. JJ/NN)
  - They get stellar accuracies: 99% on their tag set
  - Linguistic representation matters...
  - ... but it’s easier to win when you make up the rules

```plaintext
walk <SV> <SVO> V SUBJUNCTIVE VFIN
walk <SV> <SVO> V IMP VFIN
walk <SV> <SVO> V INF
walk <SV> <SVO> V PRES -SG3 VFIN
walk N NOM SG

walk V-SUBJUNCTIVE V-IMP V-INF
     V-PRES-BASE N-NOM-SG
```
Domain Effects

- Accuracies degrade outside of domain
  - Up to triple error rate
  - Usually make the most errors on the things you care about in the domain (e.g. protein names)

- Open questions
  - How to effectively exploit unlabeled data from a new domain (what could we gain?)
  - How to best incorporate domain lexica in a principled way (e.g. UMLS specialist lexicon, ontologies)
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 parameters:

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

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

- Same quantities we needed to train a CRF!
EM for HMMs: Quantities

- **Total path values (correspond to probabilities here):**

  \[
  \alpha_i(s) = P(w_0 \ldots w_i, s_i) = \sum_{s_{i-1}} P(s_i|s_{i-1})P(w_i|s_i)\alpha_{i-1}(s_{i-1})
  \]

  \[
  \beta_i(s) = P(w_i + 1 \ldots w_n|s_i) = \sum_{s_{i+1}} P(s_{i+1}|s_i)P(w_{i+1}|s_{i+1})\beta_{i+1}(s_{i+1})
  \]
The State Lattice / Trellis

```
^  ^  ^  ^  ^  ^  ^
N  N  N  N  N  N  N
V  V  V  V  V  V  V
J  J  J  J  J  J  J
D  D  D  D  D  D  D
$  $  $  $  $  $  $
```

START       Fed           raises       interest       rates       END
EM for HMMs: Process

- From these quantities, can compute expected transitions:

\[
\text{count}(s \to s') = \frac{\sum_i \alpha_i(s) P(s'|s) P(w_i|s) \beta_{i+1}(s')}{P(w)}
\]

- And emissions:

\[
\text{count}(w, s) = \frac{\sum_{i:w_i=w} \alpha_i(s) \beta_{i+1}(s)}{P(w)}
\]
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></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>77.0</td>
</tr>
<tr>
<td>1</td>
<td>80.5</td>
</tr>
<tr>
<td>2</td>
<td>81.8</td>
</tr>
<tr>
<td>3</td>
<td>83.0</td>
</tr>
<tr>
<td>4</td>
<td>84.0</td>
</tr>
<tr>
<td>5</td>
<td>84.8</td>
</tr>
<tr>
<td>6</td>
<td>85.3</td>
</tr>
<tr>
<td>7</td>
<td>85.8</td>
</tr>
<tr>
<td>8</td>
<td>86.1</td>
</tr>
<tr>
<td>9</td>
<td>86.3</td>
</tr>
<tr>
<td>10</td>
<td>86.6</td>
</tr>
</tbody>
</table>
The president said that the downturn was over.

| president | the __ of |
| president | the __ said |
| governor  | the __ of |
| governor  | the __ appointed |
| said      | sources __ ♦ |
| said      | president __ that |
| reported  | sources __ ♦ |

[Finch and Chater 92, Shuetze 93, many others]
Three main variants on the same idea:

- Pairwise similarities and heuristic clustering
  - E.g. [Finch and Chater 92]
  - Produces dendrograms

- Vector space methods
  - E.g. [Shuetze 93]
  - Models of ambiguity

- Probabilistic methods
  - Various formulations, e.g. [Lee and Pereira 99]
# Nearest Neighbors

<table>
<thead>
<tr>
<th>word</th>
<th>nearest neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>accompanied</td>
<td>submitted banned financed developed authorized headed canceled awarded barred</td>
</tr>
<tr>
<td>almost</td>
<td>virtually merely formally fully quite officially just nearly only less</td>
</tr>
<tr>
<td>causing</td>
<td>reflecting forcing providing creating producing becoming carrying particularly</td>
</tr>
<tr>
<td>classes</td>
<td>elections courses payments losses computers performances violations levels pictures</td>
</tr>
<tr>
<td>directors</td>
<td>professionals investigations materials competitors agreements papers transactions</td>
</tr>
<tr>
<td>goal</td>
<td>mood roof eye image tool song pool scene gap voice</td>
</tr>
<tr>
<td>japanese</td>
<td>chinese iraqi american western arab foreign european federal soviet indian</td>
</tr>
<tr>
<td>represent</td>
<td>reveal attend deliver reflect choose contain impose manage establish retain</td>
</tr>
<tr>
<td>think</td>
<td>believe wish know realize wonder assume feel say mean bet</td>
</tr>
<tr>
<td>york</td>
<td>angeles francisco sox rouge kong diego zone vegas inning layer</td>
</tr>
<tr>
<td>on</td>
<td>through in at over into with from for by across</td>
</tr>
<tr>
<td>must</td>
<td>might would could cannot will should can may does helps</td>
</tr>
<tr>
<td>they</td>
<td>we you i he she nobody who it everybody there</td>
</tr>
</tbody>
</table>
Dendrograms

Pronouns: Object
Auxiliary Verbs
Adverbs
WH words
Verb: “to be”
Determiners
Pronouns: Object/Possess.
Prepositions
Interjections
Nouns: Proper (names)
Adjectives: Colour,
Number
Adjectives
Nouns
Nouns: Proper (names)
Verbs
Verbs: -ing form
Verbs

- go
- come
- sit
- stay
- stand
- start
- put
- take
- get
- bring
- give
- keep
- hold
- pick
- leave
- throw
- turn
- move
- pull
- cut
- try
- finish
- tell
- show
- ask
- be
- eat
- read
- play
- use
- find
- buy
- hear
- wear
- call
- say
- talk
- sing
- write
- draw
- help
- fix
- hit
- break
- feel
Dendrograms
[Shuetze 93] clusters words as points in $\mathbb{R}^n$

Vectors too sparse, use SVD to reduce

Cluster these 50-200 dim vectors instead.
A Probabilistic Version?

\[ P(S, C) = \prod_i P(c_i)P(w_i \mid c_i)P(w_{i-1}, w_{i+1} \mid c_i) . \]

the president said that the downturn was over

the president said that the downturn was over
What Else?

- Various newer ideas:
  - Context distributional clustering [Clark 00]
  - Morphology-driven models [Clark 03]
  - Contrastive estimation [Smith and Eisner 05]
  - Feature-rich induction [Haghighi and Klein 06]

- Also:
  - What about ambiguous words?
  - Using wider context signatures has been used for learning synonyms (what’s wrong with this approach?)
  - Can extend these ideas for grammar induction (later)
Computing Marginals

\[ P(s_t|x) = \frac{P(s_t, x)}{P(x)} \]

= sum of all paths through s at t

sum of all paths
Forward Scores

\[ v_t(s_t) = \max_{s_{t-1}} v_{t-1}(s_{t-1}) \phi_t(s_{t-1}, s_t) \]

\[ \alpha_t(s_t) = \sum_{s_{t-1}} \alpha_{t-1}(s_{t-1}) \phi_t(s_{t-1}, s_t) \]
Backward Scores

\[ \beta_t(s_t) = \sum_{s_{t+1}} \beta_{t+1}(s_{t+1}) \phi_t(s_t, s_{t+1}) \]
Total Scores

\[ P(s_t, x) = \alpha_t(s_t) \beta_t(s_t) \]

\[ P(x) = \sum_{s_t} \alpha_t(s_t) \beta_t(s_t) \]

\[ = \alpha_T(\text{stop}) \]

\[ = \beta_0(\text{start}) \]
Syntax
The move followed a round of similar increases by other lenders, reflecting a continuing decline in that market.
Phrase Structure Parsing

- Phrase structure parsing organizes syntax into *constituents* or *brackets*.

- In general, this involves nested trees.

- Linguists can, and do, argue about details.

- Lots of ambiguity.

- Not the only kind of syntax...

new art critics write reviews with computers
Constituency Tests

- How do we know what nodes go in the tree?

- **Classic constituency tests:**
  - Substitution by *proform*
  - Question answers
  - Semantic grounds
    - Coherence
    - Reference
    - Idioms
  - Dislocation
  - Conjunction

- Cross-linguistic arguments, too
Conflicting Tests

- Constituency isn’t always clear
  - Units of transfer:
    - think about ~ penser à
    - talk about ~ hablar de
  - Phonological reduction:
    - I will go → I’ll go
    - I want to go → I wanna go
    - a le centre → au centre

- Coordination
  - He went to and came from the store.
Classical NLP: Parsing

- Write symbolic or logical rules:

  Grammar (CFG)  |  Lexicon
  --- | ---
  ROOT → S  |  NN → interest
  S → NP VP  |  NNS → raises
  NP → DT NN  |  VBP → interest
  NP → NN NNS  |  VBZ → raises
  VP → VBP NP PP  |  ...

- Use deduction systems to prove parses from words
  - Minimal grammar on “Fed raises” sentence: 36 parses
  - Simple 10-rule grammar: 592 parses
  - Real-size grammar: many millions of parses

- This scaled very badly, didn’t yield broad-coverage tools
Ambiguities
The board approved [its acquisition] [by Royal Trustco Ltd.] [of Toronto] [for $27 a share] [at its monthly meeting].
Attachments

- I cleaned the dishes from dinner
- I cleaned the dishes with detergent
- I cleaned the dishes in my pajamas
- I cleaned the dishes in the sink
Prepositional phrases:
They cooked the beans in the pot on the stove with handles.

Particle vs. preposition:
The puppy tore up the staircase.

Complement structures
The tourists objected to the guide that they couldn’t hear. She knows you like the back of her hand.

Gerund vs. participial adjective
Visiting relatives can be boring. Changing schedules frequently confused passengers.
Syntactic Ambiguities II

- Modifier scope within NPs
  *impractical design requirements*
  *plastic cup holder*

- Multiple gap constructions
  *The chicken is ready to eat.*
  *The contractors are rich enough to sue.*

- Coordination scope:
  *Small rats and mice can squeeze into holes or cracks in the wall.*
Dark Ambiguities

- **Dark ambiguities:** most analyses are shockingly bad (meaning, they don’t have an interpretation you can get your mind around)

  This analysis corresponds to the correct parse of

  “This will panic buyers!”

- **Unknown words and new usages**

- **Solution:** We need mechanisms to focus attention on the best ones, probabilistic techniques do this
Ambiguities as Trees

(a) S
   / \  
  NP  VP
     / \    
    PP VP

   NP
   / \
  $30 billion  from debt ...

(b) NP
   / \  
  QP  NP
       / \ 
  PDTDTPDT PDT
  ... half  a  dozen newspapers

(c) VP
   / \  
  VBZ ADVP ADJP ADJP
       / \  / \ 
      RB  just fine
PCFGs
A context-free grammar is a tuple \(<N, T, S, R>\)

- \(N\) : the set of non-terminals
  - Phrasal categories: \(S, \text{NP}, \text{VP}, \text{ADJP}\), etc.
  - Parts-of-speech (pre-terminals): \(\text{NN}, \text{JJ}, \text{DT}, \text{VB}\)
- \(T\) : the set of terminals (the words)
- \(S\) : the start symbol
  - Often written as \(\text{ROOT}\) or \(\text{TOP}\)
  - \(Not\) usually the sentence non-terminal \(S\)
- \(R\) : the set of rules
  - Of the form \(X \rightarrow Y_1 \ Y_2 \ ... \ Y_k\), with \(X, Y_i \in N\)
  - Examples: \(S \rightarrow \text{NP} \ \text{VP}\), \(\text{VP} \rightarrow \text{VP} \ \text{CC} \ \text{VP}\)
  - Also called rewrites, productions, or local trees

A PCFG adds:
- A top-down production probability per rule \(P(Y_1 \ Y_2 \ ... \ Y_k \mid X)\)
(S (NP-SBJ The move)
  (VP followed)
    (NP (NP a round)
      (PP of
        (NP (NP similar increases)
          (PP by
            (NP other lenders))
          (PP against
            (NP Arizona real estate loans))))))

(S-ADV (NP-SBJ *)
  (VP reflecting)
    (NP (NP a continuing decline)
      (PP-LOC in
        (NP that market))))

.))
Treebank Grammars

- Need a PCFG for broad coverage parsing.
- Can take a grammar right off the trees (doesn’t work well):

Better results by enriching the grammar (e.g., lexicalization).
Can also get state-of-the-art parsers without lexicalization.
Treebank Grammars can be enormous

- As FSAs, the raw grammar has ~10K states, excluding the lexicon
- Better parsers usually make the grammars larger, not smaller
Chomsky Normal Form

- **Chomsky normal form:**
  - All rules of the form $X \rightarrow YZ$ or $X \rightarrow w$
  - In principle, this is no limitation on the space of (P)CFGs
    - N-ary rules introduce new non-terminals

- Unaries / empties are “promoted”
- In practice it’s kind of a pain:
  - Reconstructing n-aries is easy
  - Reconstructing unaries is trickier
  - The straightforward transformations don’t preserve tree scores
- Makes parsing algorithms simpler!
CKY Parsing
A Recursive Parser

bestScore(X,i,j,s)
    if (j = i+1)
        return tagScore(X,s[i])
    else
        return max score(X->YZ) *
            bestScore(Y,i,k) *
            bestScore(Z,k,j)

- Will this parser work?
- Why or why not?
- Memory requirements?
One small change:

```python
bestScore(X,i,j,s)
    if (scores[X][i][j] == null)
        if (j = i+1)
            score = tagScore(X,s[i])
        else
            score = max score(X->YZ) *
            bestScore(Y,i,k) *
            bestScore(Z,k,j)
    scores[X][i][j] = score
return scores[X][i][j]
```
Can also organize things bottom-up

```plaintext
bestScore(s)
    for (i : [0, n-1])
        for (X : tags[s[i]])
            score[X][i][i+1] =
                tagScore(X, s[i])
        for (diff : [2, n])
            for (i : [0, n-diff])
                j = i + diff
            for (X->YZ : rule)
                for (k : [i+1, j-1])
                    score[X][i][j] = max(score[X][i][j],
                                          score(X->YZ) * score[Y][i][k] * score[Z][k][j])
```
Unary Rules

- Unary rules?

```python
bestScore(X, i, j, s)
  if (j = i+1)
    return tagScore(X, s[i])
  else
    return max
    max score(X->YZ) * 
    bestScore(Y, i, k) * 
    bestScore(Z, k, j) 
    max score(X->Y) * 
    bestScore(Y, i, j)
```
We need unaries to be non-cyclic
- Can address by pre-calculating the *unary closure*
- Rather than having zero or more unaries, always have exactly one

Alternate unary and binary layers
- Reconstruct unary chains afterwards
Alternating Layers

\[
\text{bestScoreB}(X,i,j,s) \\
\quad \text{return } \max \max \text{ score}(X \rightarrow YZ) \times \\
\qquad \text{bestScoreU}(Y,i,k) \times \\
\qquad \text{bestScoreU}(Z,k,j)
\]

\[
\text{bestScoreU}(X,i,j,s) \\
\quad \text{if } (j = i+1) \\
\qquad \text{return } \text{tagScore}(X,s[i]) \\
\quad \text{else} \\
\qquad \text{return } \max \max \text{ score}(X \rightarrow Y) \times \\
\qquad \quad \text{bestScoreB}(Y,i,j)
\]
Analysis
Memory

- How much memory does this require?
  - Have to store the score cache
  - Cache size: $|\text{symbols}| \times n^2$ doubles
  - For the plain treebank grammar:
    - $X \sim 20K$, $n = 40$, double $\sim 8$ bytes $= \sim 256$MB
    - Big, but workable.

- Pruning: Beams
  - $\text{score}[X][i][j]$ can get too large (when?)
  - Can keep beams (truncated maps $\text{score}[i][j]$) which only store the best few scores for the span $[i,j]$

- Pruning: Coarse-to-Fine
  - Use a smaller grammar to rule out most $X[i,j]$
  - Much more on this later...
How much time will it take to parse?

- For each diff (<= n)
  - For each i (<= n)
    - For each rule X → Y Z
      - For each split point k
        Do constant work

- Total time: |rules| * n³
- Something like 5 sec for an unoptimized parse of a 20-word sentence
Parsing with the vanilla treebank grammar:

Why’s it worse in practice?
- Longer sentences “unlock” more of the grammar
- All kinds of systems issues don’t scale

~ 20K Rules (not an optimized parser!)

Observed exponent: 3.6
Same-Span Reachability

ADJP ADVP FRAG INTJ NP PP PRN QP S SBAR UCP VP WHNP

NX SQ X RRC

SINV SBARQ PRT WHPP

WHADJP WHADVP

LST CONJP NAC
Many states are more likely to match larger spans!
Lots of tricks to make CKY efficient

- Some of them are little engineering details:
  - E.g., first choose k, then enumerate through the Y:\[i,k\] which are non-zero, then loop through rules by left child.
  - Optimal layout of the dynamic program depends on grammar, input, even system details.

- Another kind is more important (and interesting):
  - Many X[i,j] can be suppressed on the basis of the input string
  - We’ll see this next class as figures-of-merit, A* heuristics, coarse-to-fine, etc
Agenda-Based Parsing
Agenda-Based Parsing

- Agenda-based parsing is like graph search (but over a hypergraph)

- Concepts:
  - Numbering: we number fenceposts between words
  - “Edges” or items: spans with labels, e.g. PP[3,5], represent the sets of trees over those words rooted at that label (cf. search states)
  - A chart: records edges we’ve expanded (cf. closed set)
  - An agenda: a queue which holds edges (cf. a fringe or open set)

```
0 critics 1 write 2 reviews 3 with 4 computers 5
```

PP
Building an item for the first time is called discovery. Items go into the agenda on discovery.

To initialize, we discover all word items (with score 1.0).

AGENDA

critics[0,1], write[1,2], reviews[2,3], with[3,4], computers[4,5]

CHART [EMPTY]
Unary Projection

- When we pop a word item, the lexicon tells us the tag item successors (and scores) which go on the agenda.
Item Successors

- When we pop items off of the agenda:
  - Graph successors: unary projections (NNS → critics, NP → NNS)
    
    \[ Y[i,j] \text{ with } X \rightarrow Y \text{ forms } X[i,j] \]

  - Hypergraph successors: combine with items already in our chart
    
    \[ Y[i,j] \text{ and } Z[j,k] \text{ with } X \rightarrow Y Z \text{ form } X[i,k] \]

  - Enqueue / promote resulting items (if not in chart already)
  - Record backtraces as appropriate
  - Stick the popped edge in the chart (closed set)

- Queries a chart must support:
  - Is edge \( X[i,j] \) in the chart? (What score?)
  - What edges with label \( Y \) end at position \( j \)?
  - What edges with label \( Z \) start at position \( i \)?
An Example

critics write reviews with computers
Empty Elements

- Sometimes we want to posit nodes in a parse tree that don’t contain any pronounced words:
  
  I want you to parse this sentence
  
  I want [   ] to parse this sentence

- These are easy to add to a agenda-based parser!
  - For each position $i$, add the “word” edge $\varepsilon[i,i]$
  - Add rules like $NP \rightarrow \varepsilon$ to the grammar
  - That’s it!

```
I like to parse empties
```
UCS / A*

- With weighted edges, order matters
  - Must expand optimal parse from bottom up (subparses first)
  - CKY does this by processing smaller spans before larger ones
  - UCS pops items off the agenda in order of decreasing Viterbi score
  - A* search also well defined

- You can also speed up the search without sacrificing optimality
  - 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 optimiality [Klein and Manning 03]
There was nothing magical about words spanning exactly one position.

When working with speech, we generally don’t know how many words there are, or where they break.

We can represent the possibilities as a lattice and parse these just as easily.
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
  \|\nS
  \|\nNP VP
  \|\nPRP VBD ADJP
   \|\nHe was JJ
       \|\n       right
```

```
ROOT → S 1
S → NP VP . 1
NP → PRP 1
VP → VBD ADJP 1
.....
```

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>72.0</td>
</tr>
</tbody>
</table>
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!
Example: PP attachment

They raised a point of order
Grammar Refinement

- Structure Annotation [Johnson ’98, Klein&Manning ’03]
- Lexicalization [Collins ’99, Charniak ’00]
- Latent Variables [Matsuzaki et al. 05, Petrov et al. ’06]
Structural Annotation
Annotation refines base treebank symbols to improve statistical fit of the grammar

- Structural annotation
Typical Experimental Setup

- **Corpus**: Penn Treebank, WSJ

  - **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 $\infty$

Symbols

Horizontal Markov Order

Horizontal Markov Order

Symbols

Horizontal Markov Order

Symbols

Horizontal Markov Order
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>
A Fully Annotated (Unlex) Tree

```
ROOT
   \— S\^ROOT-v
     \— "S
         \— NP\^S-B
             \— "S
                 \— DT-U\^NP
                     \— "This
                 \— VBZ\^BE\^VP
                     \— "is
                 \— NP\^VP-B
                     \— "^S
                         \— !
                     \— "^S
                         \— "buying
                         \— "panic
```
Some Test Set Results

<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>

- Beats “first generation” lexicalized parsers.
- Lots of room to improve – more complex models next.
Efficient Parsing for Structural Annotation
Grammar Projections

Coarse Grammar

NP → DT N’

Fine Grammar

NP^S → DT^NP N’[...DT]^NP

Note: X-Bar Grammars are projections with rules like XP → Y X’ or XP → X’ Y or X’ → X
Coarse-to-Fine Pruning

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

$$\frac{P_{IN}(X, i, j) \cdot P_{OUT}(X, i, j)}{P_{IN}(root, 0, n)} < \text{threshold}$$

E.g. consider the span 5 to 12:
Computing (Max-)Marginals

\[
\beta(x, i, j) = \sum_{y, k} \beta(y, i, k) \cdot \beta(y, j, k).
\]
Inside and Outside Scores

\[ \alpha(x, j, i) = \sum_{y \in \mathcal{B}^k} \beta(B, j, i) \]
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 optimiality [Klein and Manning 03]
# A* Parsing

<table>
<thead>
<tr>
<th>Estimate</th>
<th>SX</th>
<th>SXL</th>
<th>SXLR</th>
<th>TRUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summary</td>
<td>(1,6,NP)</td>
<td>(1,6,NP,VBZ)</td>
<td>(1,6,NP,VBZ,“,”)</td>
<td>(entire context)</td>
</tr>
<tr>
<td>Best Tree</td>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
<td><img src="image3.png" alt="Diagram" /></td>
<td><img src="image4.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Score</td>
<td>-11.3</td>
<td>-13.9</td>
<td>-15.1</td>
<td>-18.1</td>
</tr>
</tbody>
</table>
Lexicalization
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]
If we do no annotation, these trees differ only in one rule:
- VP → VP PP
- NP → 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
Problems with PCFGs

- What’s different between basic PCFG scores here?
- What (lexical) correlations need to be scored?
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 rightmost JJ
      - Take right child
    - VP:
      - Take leftmost VB*
      - Take leftmost VP
      - Take left child
Lexicalized PCFGs?

- Problem: we now have to estimate probabilities like

\[
\text{VP(saw)} \rightarrow \text{VBD(saw)} \text{ NP-C(her)} \text{ NP(today)}
\]

- 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 bag

Generate children (incl. adjuncts)

Recursively derive children
bestScore(X,i,j,h)
if (j = i+1)
    return tagScore(X,s[i])
else
    return max max score(X[h]->Y[h] Z[h']) * bestScore(Y,i,k,h) * bestScore(Z,k,j,h')
          k,h',X->YZ
max max score(X[h]->Y[h'] Z[h]) * bestScore(Y,i,k,h') * bestScore(Z,k,j,h)
          k,h',X->YZ
Efficient Parsing for Lexical Grammars
- 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 $O(n^5)$ 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 $P(X|i,j,s)$
    - This isn’t trivial, and there are clever speed ups
  - Second, do the full $O(n^5)$ 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
The Game of Designing a Grammar

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

Parse Tree
Sentence $T$

Derivations $t : T$

Parameters $\theta$

Grammar $G$

<table>
<thead>
<tr>
<th>Rule</th>
<th>Symbol</th>
<th>CONDITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0 \rightarrow NP_0 VP_0$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
<tr>
<td>$S_0 \rightarrow NP_1 VP_0$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
<tr>
<td>$S_0 \rightarrow NP_0 VP_1$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
<tr>
<td>$S_0 \rightarrow NP_1 VP_1$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
<tr>
<td>$S_1 \rightarrow NP_0 VP_0$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
<tr>
<td>$NP_0 \rightarrow PRP_0$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
<tr>
<td>$NP_0 \rightarrow PRP_1$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
</tbody>
</table>

Lexicon

<table>
<thead>
<tr>
<th>Rule</th>
<th>Symbol</th>
<th>CONDITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PRP_0 \rightarrow She$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
<tr>
<td>$PRP_1 \rightarrow She$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
<tr>
<td>$VBD_0 \rightarrow was$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
<tr>
<td>$VBD_1 \rightarrow was$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
<tr>
<td>$VBD_2 \rightarrow was$</td>
<td>$\cdot$</td>
<td>?</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\cdot$</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)
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:
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 Lexical 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. E.</td>
<td>L.</td>
<td></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>it</th>
<th>He</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRP-1</td>
<td>it</td>
<td>he</td>
<td>they</td>
</tr>
<tr>
<td>PRP-2</td>
<td>it</td>
<td>them</td>
<td>him</td>
</tr>
</tbody>
</table>
Learned Splits

- **Relative adverbs (RBR):**

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

- **Cardinal Numbers (CD):**

<table>
<thead>
<tr>
<th>CD-7</th>
<th>one</th>
<th>two</th>
<th>Three</th>
</tr>
</thead>
<tbody>
<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>
<tr>
<td>Language</td>
<td>Method</td>
<td>≤ 40 words F1</td>
<td>all F1</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------</td>
<td>--------------</td>
<td>--------</td>
</tr>
<tr>
<td>ENG</td>
<td>Charniak&amp;Johnson ‘05 (generative)</td>
<td>90.1</td>
<td>89.6</td>
</tr>
<tr>
<td></td>
<td>Split / Merge</td>
<td>90.6</td>
<td>90.1</td>
</tr>
<tr>
<td>GER</td>
<td>Dubey ‘05</td>
<td>76.3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Split / Merge</td>
<td>80.8</td>
<td>80.1</td>
</tr>
<tr>
<td>CHN</td>
<td>Chiang et al. ‘02</td>
<td>80.0</td>
<td>76.6</td>
</tr>
<tr>
<td></td>
<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

```
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: ...

QP | NP | VP
Bracket Posteriors
1621 min
111 min
35 min
15 min
(no search error)
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
EM for HMMs: Quantities

- Total path values (correspond to probabilities here):

\[ \alpha_i(s) = P(w_0 \ldots w_i, s_i) = \sum_{s_{i-1}} P(s_i|s_{i-1})P(w_i|s_i)\alpha_{i-1}(s_{i-1}) \]

\[ \beta_i(s) = P(w_i + 1 \ldots w_n|s_i) = \sum_{s_{i+1}} P(s_{i+1}|s_i)P(w_{i+1}|s_{i+1})\beta_{i+1}(s_{i+1}) \]
The State Lattice / Trellis

START       Fed           raises       interest         rates         END
EM for HMMs: Process

- From these quantities, can compute expected transitions:

\[
\text{count}(s \rightarrow s') = \frac{\sum_i \alpha_i(s) P(s'|s) P(w_i|s) \beta_{i+1}(s')}{P(w)}
\]

- And emissions:

\[
\text{count}(w, s) = \frac{\sum_{i: w_i = w} \alpha_i(s) \beta_{i+1}(s)}{P(w)}
\]
<table>
<thead>
<tr>
<th>Iter</th>
<th>0</th>
<th>100</th>
<th>2000</th>
<th>5000</th>
<th>10000</th>
<th>20000</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>77.0</td>
<td>90.0</td>
<td>95.4</td>
<td>96.2</td>
<td>96.6</td>
<td>96.9</td>
<td>97.0</td>
</tr>
<tr>
<td>1</td>
<td>80.5</td>
<td>92.6</td>
<td>95.8</td>
<td>96.3</td>
<td>96.6</td>
<td>96.7</td>
<td>96.8</td>
</tr>
<tr>
<td>2</td>
<td>81.8</td>
<td>93.0</td>
<td>95.7</td>
<td>96.1</td>
<td>96.3</td>
<td>96.4</td>
<td>96.4</td>
</tr>
<tr>
<td>3</td>
<td>83.0</td>
<td>93.1</td>
<td>95.4</td>
<td>95.8</td>
<td>96.1</td>
<td>96.2</td>
<td>96.2</td>
</tr>
<tr>
<td>4</td>
<td>84.0</td>
<td>93.0</td>
<td>95.2</td>
<td>95.5</td>
<td>95.8</td>
<td>96.0</td>
<td>96.0</td>
</tr>
<tr>
<td>5</td>
<td>84.8</td>
<td>92.9</td>
<td>95.1</td>
<td>95.4</td>
<td>95.6</td>
<td>95.8</td>
<td>95.8</td>
</tr>
<tr>
<td>6</td>
<td>85.3</td>
<td>92.8</td>
<td>94.9</td>
<td>95.2</td>
<td>95.5</td>
<td>95.6</td>
<td>95.7</td>
</tr>
<tr>
<td>7</td>
<td>85.8</td>
<td>92.8</td>
<td>94.7</td>
<td>95.1</td>
<td>95.3</td>
<td>95.5</td>
<td>95.5</td>
</tr>
<tr>
<td>8</td>
<td>86.1</td>
<td>92.7</td>
<td>94.6</td>
<td>95.0</td>
<td>95.2</td>
<td>95.4</td>
<td>95.4</td>
</tr>
<tr>
<td>9</td>
<td>86.3</td>
<td>92.6</td>
<td>94.5</td>
<td>94.9</td>
<td>95.1</td>
<td>95.3</td>
<td>95.3</td>
</tr>
<tr>
<td>10</td>
<td>86.6</td>
<td>92.6</td>
<td>94.4</td>
<td>94.8</td>
<td>95.0</td>
<td>95.2</td>
<td>95.2</td>
</tr>
</tbody>
</table>
Factory payrolls fell in Sept.
- Total time dominated by calculation of A* tables in each projection... $O(n^3)$
We can relax independence assumptions by encoding dependencies into the PCFG symbols:

- Parent annotation [Johnson 98]
  - S\(^{\text{ROOT}}\)
    - NP\(^{\text{S}}\)
      - PRP
      - VBD
      - ADVP\(^{\text{VP}}\)
    - VBD
    - ADVP\(^{\text{VP}}\)
    - He
    - was
    - right

- Marking possessive NPs
  - NP\(^{\text{POS}}\)
    - NNP
    - POS
    - JJ
    - NN
    - new
    - ad

What are the most useful “features” to encode?
Other Tag Splits

- **UNARY-DT**: mark demonstratives as $\text{DT}^\U$ (“the X” vs. “those”)
- **UNARY-RB**: mark phrasal adverbs as $\text{RB}^\U$ (“quickly” vs. “very”)
- **TAG-PA**: mark tags with non-canonical parents (“not” is an $\text{RB}^\text{VP}$)
- **SPLIT-AUX**: mark auxiliary verbs with –$\text{AUX}$ [cf. Charniak 97]
- **SPLIT-CC**: separate “but” and “&” from other conjunctions
- **SPLIT-%**: “%” gets its own tag.

<table>
<thead>
<tr>
<th>F1</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>80.4</td>
<td>8.1K</td>
</tr>
<tr>
<td>80.5</td>
<td>8.1K</td>
</tr>
<tr>
<td>81.2</td>
<td>8.5K</td>
</tr>
<tr>
<td>81.6</td>
<td>9.0K</td>
</tr>
<tr>
<td>81.7</td>
<td>9.1K</td>
</tr>
<tr>
<td>81.8</td>
<td>9.3K</td>
</tr>
</tbody>
</table>