Natural Language Processing

Part-of-Speech Tagging

Dan Klein – UC Berkeley
Parts of Speech
### Parts-of-Speech (English)

- **One basic kind of linguistic structure: syntactic word classes**

#### Open class (lexical) words

<table>
<thead>
<tr>
<th>Category</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nouns</td>
<td></td>
</tr>
<tr>
<td>Proper</td>
<td>IBM</td>
</tr>
<tr>
<td></td>
<td>Italy</td>
</tr>
<tr>
<td>Common</td>
<td>cat / cats</td>
</tr>
<tr>
<td></td>
<td>snow</td>
</tr>
<tr>
<td>Verbs</td>
<td></td>
</tr>
<tr>
<td>Main</td>
<td>see</td>
</tr>
<tr>
<td></td>
<td>registered</td>
</tr>
<tr>
<td>Auxiliary</td>
<td>can</td>
</tr>
<tr>
<td></td>
<td>had</td>
</tr>
<tr>
<td>Adjectives</td>
<td>yellow</td>
</tr>
<tr>
<td>Adverbs</td>
<td>slowly</td>
</tr>
<tr>
<td>Numbers</td>
<td></td>
</tr>
<tr>
<td></td>
<td>122,312</td>
</tr>
<tr>
<td></td>
<td>one</td>
</tr>
<tr>
<td>Prepositions</td>
<td>to with</td>
</tr>
<tr>
<td>Particles</td>
<td>off up</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Determiners</td>
<td>the some</td>
</tr>
<tr>
<td>Conjunctions</td>
<td>and or</td>
</tr>
<tr>
<td>Pronouns</td>
<td>he its</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Note:** More examples and categories might be included in the document. The table and graphic provide an overview of the types of words found in English.
Part-of-Speech Ambiguity

- Words can have multiple parts of speech

- Two basic sources of constraint:
  - Grammatical environment
  - Identity of the current word

- Many more possible features:
  - Suffixes, capitalization, name databases (gazetteers), etc...

Fed raises interest rates 0.5 percent

Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG
All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around-IN the/DT corner/NN
Chateau/NNP Petrus/NNP costs/VBZ around/RB 250/CD
Why POS Tagging?

- Useful in and of itself (more than you’d think)
  - Text-to-speech: record, lead
  - Lemmatization: saw[v] → see, saw[n] → saw
  - Quick-and-dirty NP-chunk detection: grep {JJ | NN}* {NN | NNS}

- Useful as a pre-processing step for parsing
  - Less tag ambiguity means fewer parses
  - However, some tag choices are better decided by parsers

The average of interbank offered rates plummeted ...

The Georgia branch had taken on loan commitments ...

The average of interbank offered rates plummeted ...
Part-of-Speech Tagging
Classic Solution: HMMs

- We want a model of sequences $s$ and observations $w$

\[ P(s, w) = \prod_i P(s_i|s_{i-1}) P(w_i|s_i) \]

- Assumptions:
  - States are tag $n$-grams
  - Usually a dedicated start and end state / word
  - Tag/state sequence is generated by a markov model
  - Words are chosen independently, conditioned only on the tag/state
  - These are totally broken assumptions: why?
States

- States encode what is relevant about the past
- Transitions $P(s|s')$ encode well-formed tag sequences
  - In a bigram tagger, states = tags
  
  - In a trigram tagger, states = tag pairs
Estimating Transitions

- Use standard smoothing methods to estimate transitions:

\[
P(t_i | t_{i-1}, t_{i-2}) = \lambda_2 \hat{P}(t_i | t_{i-1}, t_{i-2}) + \lambda_1 \hat{P}(t_i | t_{i-1}) + (1 - \lambda_1 - \lambda_2) \hat{P}(t_i)
\]

- Can get a lot fancier (e.g. KN smoothing) or use higher orders, but in this case it doesn’t buy much

- One option: encode more into the state, e.g. whether the previous word was capitalized (Brants 00)

- BIG IDEA: The basic approach of state-splitting / refinement turns out to be very important in a range of tasks
Estimating Emissions

\[ P(s, w) = \prod_{i} P(s_i|s_{i-1}) P(w_i|s_i) \]

- **Emissions are trickier:**
  - Words we’ve never seen before
  - Words which occur with tags we’ve never seen them with
  - One option: break out the fancy smoothing (e.g. KN, Good-Turing)
  - Issue: unknown words aren’t black boxes:
    - 343,127.23  11-year  Minteria  reintroducibly
  - Basic solution: unknown words classes (affixes or shapes)
    - \( D^+, D^+.D^+ \) \( D^+-x^+ \) \( Xx^+ \) \( x^+ \)-“ly”
  - Common approach: Estimate \( P(t|w) \) and invert
  - [Brants 00] used a suffix trie as its (inverted) emission model
Disambiguation (Inference)

- Problem: find the most likely (Viterbi) sequence under the model
  \[ t^* = \arg \max_t P(t|w) \]

- Given model parameters, we can score any tag sequence

\[
\begin{align*}
<\bullet, \bullet > & \quad <\bullet, \text{NNP}> & \quad <\text{NNP}, \text{VBZ}> & \quad <\text{VBZ}, \text{NN}> & \quad <\text{NN}, \text{NNS}> & \quad <\text{NNS}, \text{CD}> & \quad <\text{CD}, \text{NN}> & \quad <\text{STOP}>
\end{align*}
\]

\[
\begin{align*}
\text{NNP} & \quad \text{VBZ} & \quad \text{NN} & \quad \text{NNS} & \quad \text{CD} & \quad \text{NN} & \quad .
\end{align*}
\]

Fed raises interest rates 0.5 percent.

\[
\begin{align*}
P(\text{NNP}|<\bullet, \bullet>) & \quad P(\text{Fed}|\text{NNP}) & \quad P(\text{VBZ}|<\text{NNP}, \bullet>) & \quad P(\text{raises}|\text{VBZ}) & \quad P(\text{NN}|\text{VBZ}, \text{NNP})& \quad .
\end{align*}
\]

- In principle, we’re done – list all possible tag sequences, score each one, pick the best one (the Viterbi state sequence)

\[
\begin{align*}
\text{NNP} & \quad \text{VBZ} & \quad \text{NN} & \quad \text{NNS} & \quad \text{CD} & \quad \text{NN} & \quad \rightarrow & \quad \log P = -23
\end{align*}
\]
\[
\begin{align*}
\text{NNP} & \quad \text{NNS} & \quad \text{NN} & \quad \text{NNS} & \quad \text{CD} & \quad \text{NN} & \quad \rightarrow & \quad \log P = -29
\end{align*}
\]
\[
\begin{align*}
\text{NNP} & \quad \text{VBZ} & \quad \text{VB} & \quad \text{NNS} & \quad \text{CD} & \quad \text{NN} & \quad \rightarrow & \quad \log P = -27
\end{align*}
\]
START       Fed           raises       interest         rates         END
The State Lattice / Trellis

<table>
<thead>
<tr>
<th>START</th>
<th>Fed</th>
<th>raises</th>
<th>interest</th>
<th>rates</th>
<th>END</th>
</tr>
</thead>
</table>

P(Fed|N)
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 (SOA is ~97.5%)

- **Noise in the data**
  - Many errors in the training and test corpora

```
DT NN IN NN VBD NNS VBD
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 Toutanova & Manning 00]

<table>
<thead>
<tr>
<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></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>
</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>
</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>
</tr>
<tr>
<td>NNPS</td>
<td>0</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>
</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>
</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>
</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>
</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>
</tr>
<tr>
<td>VBD</td>
<td>10</td>
<td>5</td>
<td>3</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>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>
</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>
</tr>
</tbody>
</table>

NN/JJ  NN  VBD RP/IN DT NN  RB  VBD/VBN  NNS
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
  - Lowercased word
  - Prefixes
  - Suffixes
  - Capitalization
  - Word shapes

- Then build a maxent (or whatever) model to predict tag

- Maxent $P(t|w): 93.7\% / 82.6\%$
Why Linear Context is Useful

- Lots of rich local information!
  
  RB
  PRP VBD IN RB IN PRP VBD .
  They left as soon as he arrived .

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

  JJ
  NNP NNS VBD VBN .
  Intrinsic flaws remained undetected .

  - 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 WSJ journal text...
  - What the world needs is taggers that work on other text!
  - Though: other tasks like IE have used the same methods to good effect
Sequence-Free Tagging?

- What about looking at a word and its environment, but no sequence information?
  - Add in previous / next word the __
  - Previous / next word shapes X __ X
  - Occurrence pattern features [X: x X occurs]
  - Crude entity detection __ ..... (Inc.|Co.)
  - Phrasal verb in sentence? put ...... __
  - Conjunctions of these things

- All features except sequence: 96.6% / 86.8%
- Uses lots of features: > 200K
- Why isn’t this the standard approach?
Feature-Rich Sequence Models

- Problem: HMMs make it hard to work with arbitrary features of a sentence

- Example: name entity recognition (NER)

```
PER PER O   O   O   O   O   O   ORG   O   O   O   O   O   LOC   LOC   O
```

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

Local Context

<table>
<thead>
<tr>
<th>State</th>
<th>Prev</th>
<th>Cur</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other</td>
<td></td>
<td>???</td>
<td>???</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word</th>
<th>Prev</th>
<th>Cur</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>at</td>
<td></td>
<td>Grace</td>
<td>Road</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>Prev</th>
<th>Cur</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td>NNP</td>
<td>NNP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sig</th>
<th>Prev</th>
<th>Cur</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
<td>Xx</td>
<td>Xx</td>
</tr>
</tbody>
</table>
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?
**NER Features**

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>Beginning bigram</td>
<td>&lt;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>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td></td>
<td><strong>-0.58</strong></td>
<td><strong>2.68</strong></td>
</tr>
</tbody>
</table>
Conditional Random Fields (and Friends)
Perceptron Taggers

- Linear models:

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

- ... 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)

[Collins 01]
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?

\[
\begin{align*}
\alpha_i(s) &= \sum_{s'} \phi_i(s', s) \alpha_{i-1}(s') \\
\beta_i(s) &= \sum_{s'} \phi_{i+1}(s, s') \beta_{i+1}(s') \\
P(t_i = s | w) &= \frac{\alpha_i(s) \beta_i(s)}{\alpha_N(\text{END})}
\end{align*}
\]
Transformation-Based Learning

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

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

- 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

water
milk
paper
juice
money
food
stuff
coffee
tea
cheese
cream
butter
cake
soup
meat
bread
fish
egg
chicken
orange
apple
cookie
sandwich
dinner
lunch
breakfast
supper
tv
candy

in
on
over
for
with
at
from
by
into
of
than
about
as
after
under
behind
next
last
every
up
down
out
back
off
away
home
around
together
outside
inside
through
round
upstairs, downstairs
along
somewhere
straight
either
anymore
A Probabilistic Version?

\[ P(S, C) = \prod_{i} P(c_i) P(w_i | c_i) P(w_{i-1}, w_{i+1} | c_i) \]

- the president said that the downturn was over

\[ P(S, C) = \prod_{i} P(c_i) P(w_i | c_i) P(w_{i-1}, w_{i+1} | c_i) \]

- the president said that the downturn was over