Parts-of-Speech (English)

- One basic kind of linguistic structure: syntactic word classes

<table>
<thead>
<tr>
<th>Open class (lexical) words</th>
<th>Closed class (functional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nouns</td>
<td>Determiners</td>
</tr>
<tr>
<td>Proper IBM, Italy</td>
<td>the some</td>
</tr>
<tr>
<td>Common cat / cats snow</td>
<td>Conjunctions and or</td>
</tr>
<tr>
<td>Verbs</td>
<td>Pronouns</td>
</tr>
<tr>
<td>Main see registered</td>
<td>he / its</td>
</tr>
<tr>
<td>Adjectives</td>
<td></td>
</tr>
<tr>
<td>yellow</td>
<td></td>
</tr>
<tr>
<td>Adverbs</td>
<td></td>
</tr>
<tr>
<td>slowly</td>
<td></td>
</tr>
<tr>
<td>Numbers</td>
<td></td>
</tr>
<tr>
<td>122,312</td>
<td></td>
</tr>
<tr>
<td>one</td>
<td></td>
</tr>
<tr>
<td>... more</td>
<td></td>
</tr>
<tr>
<td>Prepositions</td>
<td></td>
</tr>
<tr>
<td>to with</td>
<td></td>
</tr>
<tr>
<td>Particles</td>
<td></td>
</tr>
<tr>
<td>off up</td>
<td></td>
</tr>
<tr>
<td>... more</td>
<td></td>
</tr>
</tbody>
</table>
Part-of-Speech Ambiguity

**Example**

<table>
<thead>
<tr>
<th>VBD</th>
<th>VB</th>
<th>VBN</th>
<th>VBZ</th>
<th>VBP</th>
<th>VBZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fed raises interest rates 0.5 percent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mrs./NNP Shafer/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

- **Two basic sources of constraint:**
  - Grammatical environment
  - Identity of the current word
- **Many more possible features:**
  - … but we won’t be able to use them for a while
Part-of-Speech Tagging

Republicans warned Sunday that the Obama administration’s $800 billion economic stimulus effort will lead to what one called a “financial disaster.”

The administration is also readying a second phase of the financial bailout program launched by the Bush administration last fall.

Why POS Tagging?

- Useful in and of itself
  - 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

```
IN
DT NNP NN VBD VBN RP NN NNS
```

The Georgia branch had taken on loan commitments …

```
VDN
DT NN IN NN VBD NNS VBD
```

The average of interbank offered rates plummeted …
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?

Transitions

- Transitions \( P(s|s') \) encode well-formed tag sequences
  - In a bigram tagger, states = tags

\[
<\diamond> \quad <t_1> \quad <t_2> \quad <t_n>
\]

- In a trigram tagger, states = tag pairs

\[
<\diamond, \diamond> \quad <\diamond, t_1> \quad <t_1, t_2> \quad <t_n, t_n>
\]
Estimating Transitions

- Use standard smoothing methods to estimate transitions:
  \[ P(t_i \mid t_{i-1}, t_{i-2}) = \lambda_2 \hat{P}(t_i \mid t_{i-1}, t_{i-2}) + \lambda_i \hat{P}(t_i \mid t_{i-1}) + (1 - \lambda_i - \lambda_2) \hat{P}(t_i) \]
- 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)

Estimating Emissions

- 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 Good-Turning smoothing
  - Issue: unknown words aren’t black boxes:
    343,127.23  11-year  Minteria  reintroducibly
  - Solution: unknown words classes (affixes or shapes)
    \( D^+, D^- \cdot, D^-, x^+ \cdot X\cdot, x^* \)
- [Brants 00] used a suffix trie as its emission model
Disambiguation

- Given these two multinomials, we can score any word / tag sequence pair

\[
\begin{align*}
&<\star,\star> <\star,\text{NNP}> <\text{NNP},\text{VBZ}> <\text{VBZ},\text{NN}> <\text{NN},\text{NNS}> <\text{NNS},\text{CD}> <\text{CD},\text{NN}> <\text{STOP}> \\
&\text{Fed} \ \text{raises} \ \text{interest} \ \text{rates} \ 0.5 \ \text{percent} \ .
\end{align*}
\]

\[
P(\text{NNP}|<\star,\star>) \ P(\text{Fed}|\text{NNP}) \ P(\text{VBZ}|<\text{NNP},\star>) \ P(\text{raises}|\text{VBZ}) \ P(\text{NN}|\text{VBZ},\text{NNP}) \ldots.
\]

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

\[
\begin{align*}
&\text{NNP \ VBZ \ NN \ NNS \ CD \ NN} \ \rightarrow \ \log P = -23 \\
&\text{NNP \ NNS \ NN \ NNS \ CD \ NN} \ \rightarrow \ \log P = -29 \\
&\text{NNP \ VBZ \ VB \ NNS \ CD \ NN} \ \rightarrow \ \log P = -27
\end{align*}
\]

Finding the Best Trajectory

- Too many trajectories (state sequences) to list
- Option 1: Beam Search

- A beam is a set of partial hypotheses
- Start with just the single empty trajectory
- At each derivation step:
  - Consider all continuations of previous hypotheses
  - Discard most, keep top k, or those within a factor of the best, (or some combination)

- Beam search works relatively well in practice
  - … but sometimes you want the optimal answer
  - … and you need optimal answers to validate your beam search
The State Lattice / Trellis

The State Lattice / Trellis

START Fed raises interest rates END
The Viterbi Algorithm

- Dynamic program for computing
  \[ \delta_i(s) = \max_{s_0 \cdots s_{i-1}} P(s_0 \cdots s_{i-1}, w_1 \cdots w_{i-1}) \]
  - The score of a best path up to position i ending in state s
  \[ \delta_0(s) = \begin{cases} 1 & \text{if } s = \bullet, \bullet \\ 0 & \text{otherwise} \end{cases} \]
  \[ \delta_i(s) = \max_{s'} P(s | s') P(w | s') \delta_{i-1}(s') \]
  - Also store a backtrace
  \[ \psi_i(s) = \arg \max_{s'} P(s | s') P(w | s') \delta_{i-1}(s') \]
- Memoized solution
- Iterative solution

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 ~97.2%)
- 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%
  - Maxent P(t|w): 93.7% / 82.6%
  - TnT (HMM++): 96.2% / 86.0%
  - MEMM tagger: 96.9% / 86.9%
  - Cyclic tagger: 97.2% / 89.0%
  - Upper bound: ~98%

Most errors on unknown words

Common Errors

- Common errors [from Toutanova & Manning 00]

<table>
<thead>
<tr>
<th>NN/JJ</th>
<th>JJ</th>
<th>RN</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>
<td></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>
<td></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>
<td></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>
<td></td>
</tr>
<tr>
<td>RB</td>
<td>72</td>
<td>21</td>
<td>7</td>
<td>0</td>
<td>16</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>295</td>
<td></td>
</tr>
<tr>
<td>RP</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>0</td>
<td>104</td>
<td></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>
<td></td>
</tr>
<tr>
<td>VB</td>
<td>17</td>
<td>64</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>7</td>
<td>85</td>
<td>189</td>
<td></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>0</td>
<td>143</td>
<td>2</td>
<td>166</td>
<td></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>
<td></td>
</tr>
<tr>
<td>VBP</td>
<td>5</td>
<td>34</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>49</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>104</td>
<td>3651</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JJ</th>
<th>VBD</th>
<th>RP</th>
<th>IN</th>
<th>DT</th>
<th>NN</th>
<th>official knowledge</th>
<th>made up the story</th>
<th>RB</th>
<th>VBD</th>
<th>VBN</th>
<th>NNS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>34</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>49</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>104</td>
</tr>
</tbody>
</table>
Better Features

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

- Then build a maxent (or whatever) model to predict tag
- Maxent P(t|w): 93.7% / 82.6%

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?
Why Linear Context is Useful

- Lots of 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: maximum entropy 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!

Maxent Taggers

- One step up: also condition on previous tags

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

- Train up \( P(t_i|w, t_{i-1}, t_{i-2}) \) as a normal maxent problem, then use to score sequences

- This is referred to as a maxent tagger [Ratnaparkhi 96]

- Beam search effective! (Why?)

- What’s the advantage of beam size 1?
Decoding

- Decoding maxent taggers:
  - Just like decoding HMMs
  - 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 (Maxent):
  \[ \delta_i(s) = \arg\max_{s'} P(s|s',w) \delta_{i-1}(s') \]

TBL Tagger

- [Brill 95] presents a transformation-based tagger
  - Label the training set with most frequent tags

<table>
<thead>
<tr>
<th>DT</th>
<th>MD</th>
<th>VBD</th>
<th>VBD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>can</td>
<td>was</td>
<td>rusted</td>
</tr>
</tbody>
</table>

  - 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 not the most accurate: 96.6% / 82.0 %
### TBL Tagger II

- **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 DT</td>
</tr>
<tr>
<td>4</td>
<td>VB</td>
<td>NN</td>
<td>One of the previous three tags is VBD</td>
</tr>
<tr>
<td>5</td>
<td>VBD</td>
<td>VBN</td>
<td>One of the previous three tags is VBE</td>
</tr>
<tr>
<td>6</td>
<td>VB</td>
<td>VBD</td>
<td>Previous tag is PRP</td>
</tr>
<tr>
<td>7</td>
<td>VBN</td>
<td>VB</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>VBP</td>
<td>Previous tag is PRP</td>
</tr>
<tr>
<td>11</td>
<td>VB</td>
<td>VBP</td>
<td>Previous tag in NNS</td>
</tr>
<tr>
<td>12</td>
<td>VBD</td>
<td>VBN</td>
<td>One of previous three tags in VBD</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 in VBP</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>JHR</td>
<td>RBH</td>
<td>Next tag is JJ</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: 98.5% on *their* tag set
  - Linguistic representation matters…
  - … but it’s easier to win when you make up the rules
CRF 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

CRFs

- Make a maxent model over entire taggings
  - MEMM
    \[ P(t|w) = \prod_i \frac{1}{Z(i)} \exp\left( \lambda^T f(t_i, t_{i-1}, w, i) \right) \]
  - CRF
    \[ P(t|w) = \frac{1}{Z(w)} \exp\left( \lambda^T f(t, w) \right) \]
    \[ = \frac{1}{Z(w)} \exp\left( \lambda^T \sum_i f(t_i, t_{i-1}, w, i) \right) \]
    \[ = \frac{1}{Z(w)} \prod_i \phi_i(t_i, t_{i-1}) \]
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 each feature, for example the number of times the label pair DT-NN occurs, or the number of times NN-interest occurs in a sentence.

- How many times does, say, DT-NN occur at position 10? The ratio of the scores of trajectories with that configuration to the score of all.

- This requires exactly the same forward-backward score ratios as for EM, but using the local potentials phi instead of the local probabilities.

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?

- 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:

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

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

- But we need a dynamic program to help, because there are too many sequences to sum over
EM for HMMs: Quantities

- Cache total path values:

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

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

- Can calculate in \(O(s^2n)\) time (why?)

The State Lattice / Trellis

```
START       Fed           raises       interest         rates         END
```

```
\[ \begin{array}{ccccccc}
\& \& \& \& \& \& \\
\& \& \& \& \& \& \\
N \& N \& N \& N \& N \& N \& N \\
V \& V \& V \& V \& V \& V \& V \\
J \& J \& J \& J \& J \& J \& J \\
D \& D \& D \& D \& D \& D \& D \\
\$ \& \$ \& \$ \& \$ \& \$ \& \$ \& \$
\end{array} \]```
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:

\[ \text{count}(w, s) = \frac{\sum_{i:w_i=w} \alpha_i(s) \beta_{i+1}(s)}{P(w)} \]

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 \( P(t|t_{i-1}, t_{i-2}) \) on these examples
  - On n examples, re-estimate with EM

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

<table>
<thead>
<tr>
<th>Number of tagged sentences used for the initial model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iter</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

Distributional Clustering

* The president said that the downturn was over *


[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 loans 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 eeupean 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 roose long 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>
Vector Space Version

- [Shuetze 93] clusters words as points in $\mathbb{R}^n$

- Vectors too sparse, use SVD to reduce

\[
\begin{align*}
\prod_{i=1}^{n} & \mathbb{P}(c_i, w_i) \\
= & \mathbb{P}(w_i | c_i) \mathbb{P}(w_{i-1}, w_{i+1} | c_i)
\end{align*}
\]

Cluster these 50-200 dim vectors instead.

A Probabilistic Version?

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

- Various newer ideas:
  - Context distributional clustering [Clark 00]
  - Morphology-driven models [Clark 03]
  - Contrastive estimation [Smith and Eisner 05]

- 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)

Forward Recurrence

\[ a(i) = \sum_{l=1}^{L} a_{l,i} \cdot b_l(a_i) \]
Backward Recurrence

Fractional Transitions