Lecture 6: Parts-of-Speech

Dan Klein – UC Berkeley

Parts-of-Speech (English)

- One basic kind of linguistic structure: syntactic word classes

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

- **Words can have multiple parts of speech**
  
  ```plaintext
  Fed raises interest rates 0.5 percent
  ```

  Mrs. NNP Shafer/NPP 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:**
  - Suffixes, capitalization, name databases (gazetteers), etc…
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

\[
\begin{array}{cccccccc}
\text{DT} & \text{NNP} & \text{NN} & \text{VBD} & \text{VBN} & \text{RP} & \text{NN} & \text{NNS} \\
\text{IN} & \text{NN} & \text{NN} & \text{VBD} & \text{NNS} & \text{VBD} \\
\end{array}
\]

The Georgia branch had taken on loan commitments …

\[
\begin{array}{cccccccc}
\text{DT} & \text{NN} & \text{IN} & \text{NN} & \text{VBD} & \text{NNS} & \text{VBD} & \text{VDN} \\
\end{array}
\]

The average of interbank offered rates plummeted …

---

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:
  \[
P(t_i | t_{i-1}, t_{i-2}) = \lambda_2 \hat{P}(t_i | t_{i-1}, t_{i-2}) + \hat{P}(t_i | t_{i-1}) + (1 - \lambda_1 - \lambda_2) \hat{P}(t_i)
  \]
- 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 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 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* D+ D*-x* Xx* x*-“ly”

- [Brants 00] used a suffix trie as its 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

| < , , > | < , , NNP > | < , , NNP, VBZ > | < VBZ, NN > | < NN, NNS > | < NNS, CD > | < CD, NN > | < STOP > |
| NNP | VBZ | NN | NNS | CD | NN | .
| Fed raises interest rates 0.5 percent . |

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

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

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

Beam search works ok in practice
- … but sometimes you want the optimal answer
- … and you need optimal answers to validate your beam search
- … and there’s usually a better option than naïve beams

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
```
The State Lattice / Trellis

The Viterbi Algorithm

- Dynamic program for computing
  \[ \delta_i(s) = \max_{s_0 \ldots s_{i-1} s} P(s_0 \ldots s_{i-1} s, w_i \ldots w_{i-1}) \]
  - The score of a best path up to position \( i \) ending in state \( s \)

  \[ \delta_0(s) = \begin{cases} 
    1 & \text{if } s = <\bullet,\bullet> \\
    0 & \text{otherwise} 
  \end{cases} \]

  \[ \delta_i(s) = \max_{s'} P(s \mid s') P(w \mid s') \delta_{i-1}(s') \]

- Also store a backtrace
  \[ \psi_i(s) = \arg \max_{s'} P(s \mid s') P(w \mid s') \delta_{i-1}(s') \]

- 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 ~97.5%)

- 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%
  - 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>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>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>14</td>
<td>29</td>
<td>5</td>
<td>6</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>7</td>
<td>1</td>
<td>2</td>
<td>0</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>142</td>
</tr>
<tr>
<td>RB</td>
<td>72</td>
<td>21</td>
<td>7</td>
<td>0</td>
<td>16</td>
<td>38</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>0</td>
<td>39</td>
<td>165</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>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>
</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>3</td>
<td>0</td>
<td>143</td>
</tr>
<tr>
<td>VBN</td>
<td>101</td>
<td>3</td>
<td>3</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>1</td>
<td>2</td>
<td>49</td>
<td>6</td>
<td>3</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>
</tr>
</tbody>
</table>

- NN/JJ
- NN
- VBD RP/IN
- DT NN
- RB
- VBD/VBN
- NNS
- Official knowledge
- Made up the story
- Recently sold shares

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%
Why Linear 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 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
  
  - Previous / next word shapes
  
  - Occurrence pattern features
  
  - Crude entity detection
  
  - Phrasal verb in sentence?
  
  - Conjunctions of these things

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

- One step up: also condition on previous tags

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

- 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'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
    
    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 %

TBL Tagger II

- What gets learned? [from Brill 95]
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

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
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)

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 \textit{DT-NN} occurs, or the number of times \textit{NN-interest} occurs in a sentence.

- How many times does, say, \textit{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 to compute these marginals
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 \(O(s^2 n)\) time (why?)

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

- 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>Iter</th>
<th>Correct tags (% words) after ML on 1M words</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
</tbody>
</table>

Distributional Clustering

- the president said that the downturn was over

<table>
<thead>
<tr>
<th></th>
<th>president</th>
<th>the __ of</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>president</td>
<td>the __ said</td>
</tr>
<tr>
<td></td>
<td>governor</td>
<td>the __ appointed</td>
</tr>
<tr>
<td>said</td>
<td>sources __ ♦</td>
<td></td>
</tr>
<tr>
<td>said</td>
<td>president __ that</td>
<td></td>
</tr>
<tr>
<td>reported</td>
<td>sources __ ♦</td>
<td></td>
</tr>
</tbody>
</table>

[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 nearly formally fully quite officially just nearly only less</td>
</tr>
<tr>
<td>casing</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 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 sax rogue 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>