CSCI 5832
Natural Language Processing

Jim Martin
Lecture 8

Today 2/7

- Finish remaining LM issues
  - Smoothing
  - Backoff and Interpolation
- Parts of Speech
- POS Tagging
- HMMs and Viterbi

Laplace smoothing

- Also called add-one smoothing
- Just add one to all the counts!
- Very simple

- MLE estimate: \( P(w_i) = \frac{c_i}{N} \)
- Laplace estimate: \( P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V} \)
- Reconstructed counts: \( c'_i = (c_i + 1) \frac{N}{N + V} \)
### Laplace smoothed bigram counts

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>828</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>want</td>
<td>3</td>
<td>1</td>
<td>609</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>to</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>687</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>212</td>
</tr>
<tr>
<td>eat</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>17</td>
<td>3</td>
<td>1</td>
<td>43</td>
</tr>
<tr>
<td>chinese</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>83</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>food</td>
<td>16</td>
<td>1</td>
<td>16</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>lunch</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>spend</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Laplace-smoothed bigrams

\[
P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}
\]

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0015</td>
<td>0.21</td>
<td>0.0025</td>
<td>0.0025</td>
<td>0.0025</td>
<td>0.0025</td>
<td>0.0025</td>
<td>0.0025</td>
</tr>
<tr>
<td>want</td>
<td>0.0015</td>
<td>0.0026</td>
<td>0.26</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
</tr>
<tr>
<td>to</td>
<td>0.00079</td>
<td>0.0026</td>
<td>0.013</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
</tr>
<tr>
<td>eat</td>
<td>0.0014</td>
<td>0.0046</td>
<td>0.014</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
</tr>
<tr>
<td>chinese</td>
<td>0.0012</td>
<td>0.0026</td>
<td>0.0002</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
</tr>
<tr>
<td>food</td>
<td>0.0063</td>
<td>0.0026</td>
<td>0.0063</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
</tr>
<tr>
<td>lunch</td>
<td>0.0017</td>
<td>0.0056</td>
<td>0.0006</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
</tr>
<tr>
<td>spend</td>
<td>0.0012</td>
<td>0.0056</td>
<td>0.0012</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
<td>0.0029</td>
</tr>
</tbody>
</table>

### Reconstituted counts

\[
c^*(w_{n-1}w_n) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}
\]

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>eat</th>
<th>chinese</th>
<th>food</th>
<th>lunch</th>
<th>spend</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.8</td>
<td>527</td>
<td>0.64</td>
<td>0.64</td>
<td>0.64</td>
<td>0.64</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td>want</td>
<td>1.2</td>
<td>0.39</td>
<td>2.38</td>
<td>0.78</td>
<td>2.7</td>
<td>2.7</td>
<td>0.78</td>
<td></td>
</tr>
<tr>
<td>to</td>
<td>1.9</td>
<td>0.63</td>
<td>4.99</td>
<td>1.9</td>
<td>0.63</td>
<td>1.9</td>
<td>0.63</td>
<td>1.9</td>
</tr>
<tr>
<td>eat</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
<td>0.34</td>
</tr>
<tr>
<td>chinese</td>
<td>0.2</td>
<td>0.988</td>
<td>0.098</td>
<td>0.098</td>
<td>0.098</td>
<td>0.2</td>
<td>0.2</td>
<td>0.098</td>
</tr>
<tr>
<td>food</td>
<td>0.43</td>
<td>0.43</td>
<td>0.86</td>
<td>0.86</td>
<td>0.86</td>
<td>0.43</td>
<td>0.43</td>
<td>0.43</td>
</tr>
<tr>
<td>lunch</td>
<td>0.57</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
<td>0.38</td>
<td>0.19</td>
<td>0.19</td>
</tr>
<tr>
<td>spend</td>
<td>0.32</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
<td>0.16</td>
</tr>
</tbody>
</table>
Big Changes to Counts

- C(count to) went from 608 to 238!
- P(to|want) from .66 to .26!
- Discount $d = \frac{c^*}{c}$
  - $d$ for “chinese food” = .10!!! A 10x reduction
  - So in general, Laplace is a blunt instrument
  - Could use more fine-grained method (add-k)
- Despite its flaws Laplace (add-k) is however still used to smooth other probabilistic models in NLP, especially
  - For pilot studies
  - In domains where the number of zeros isn’t so huge.

Better Discounting Methods

- Intuition used by many smoothing algorithms
  - Good-Turing
  - Kneser-Ney
  - Witten-Bell
- Is to use the count of things we’ve seen once to help estimate the count of things we’ve never seen

Good-Turing

- Imagine you are fishing
  - There are 8 species: carp, perch, whitefish, trout, salmon, eel, catfish, bass
- You have caught
  - 10 carp, 3 perch, 2 whitefish, 1 trout, 1 salmon, 1 eel
- Is to use the count of things we’ve seen once to help estimate the count of things we’ve never seen
  - 18 fish (tokens)
  - 6 species (types)
- How likely is it that you’ll next see another trout?
Good-Turing

- Now how likely is it that next species is new (i.e. catfish or bass)
  - There were 18 distinct events... 3 of those represent singleton species
  - \( \frac{3}{18} \)

Good-Turing

- But that 3/18 isn't represented in our probability mass. Certainly not the one we used for estimating another trout.

Good-Turing Intuition

- Notation: \( N_x \) is the frequency-of-frequency-x
  - So \( N_{x=1}, N_{x=3}, \) etc
  - To estimate total number of unseen species
  - Use number of species (words) we've seen once
    - \( c_0 = c_1 \), \( p_0 = \frac{N_v}{N} \)
  - All other estimates are adjusted (down) to give probabilities for unseen
    - \( c^* = (c + 1) \frac{N_x + 1}{N_c} \)

Slide from Josh Goodman
Good-Turing Intuition

- Notation: \( N_x \) is the frequency-of-frequency-x
- So \( N_{10} = 1 \), \( N_1 = 3 \), etc
- To estimate total number of unseen species
- Use number of species (words) we’ve seen once
- \( c_0 = c_1 \) \quad \( p_0 = N_{x}/N \) \quad \( p_0 \_N_{x}/N=3/18 \)

\[ P_{GT} (\text{things with frequency zero in training}) = \frac{N_1}{N} \]

- All other estimates are adjusted (down) to give probabilities for unseen

\[ c^* = (c+1) \frac{N_{c+1}}{N_c} \]

Slide from Josh Goodman

Bigram frequencies of frequencies and GT re-estimates

<table>
<thead>
<tr>
<th>c (MLE)</th>
<th>( N_c )</th>
<th>( c^* ) (GT)</th>
<th>c (MLE)</th>
<th>( N_c )</th>
<th>( c^* ) (GT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>74,671,100,900</td>
<td>0.00000270</td>
<td>0</td>
<td>2,081,496</td>
<td>0.002553</td>
</tr>
<tr>
<td>1</td>
<td>2,018,046</td>
<td>0.0146</td>
<td>1</td>
<td>531</td>
<td>0.533960</td>
</tr>
<tr>
<td>2</td>
<td>449,721</td>
<td>0.126</td>
<td>2</td>
<td>1,149</td>
<td>1.357284</td>
</tr>
<tr>
<td>3</td>
<td>188,933</td>
<td>0.224</td>
<td>3</td>
<td>642</td>
<td>2.373832</td>
</tr>
<tr>
<td>4</td>
<td>105,668</td>
<td>0.324</td>
<td>4</td>
<td>381</td>
<td>4.081365</td>
</tr>
<tr>
<td>5</td>
<td>68,379</td>
<td>0.422</td>
<td>5</td>
<td>311</td>
<td>3.781350</td>
</tr>
<tr>
<td>6</td>
<td>48,190</td>
<td>0.519</td>
<td>6</td>
<td>196</td>
<td>4.500000</td>
</tr>
</tbody>
</table>

GT smoothed bigram probs

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>want</th>
<th>to</th>
<th>it</th>
<th>the</th>
<th>other</th>
<th>food</th>
<th>hands</th>
<th>speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>0.00014</td>
<td>0.0014</td>
<td>0.000048</td>
<td>0.000035</td>
<td>0.000038</td>
<td>0.00017</td>
<td>0.000032</td>
<td>0.000049</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>0.00134</td>
<td>0.00132</td>
<td>0.000069</td>
<td>0.000045</td>
<td>0.000051</td>
<td>0.000043</td>
<td>0.000033</td>
<td>0.000040</td>
<td></td>
</tr>
<tr>
<td>to</td>
<td>0.000032</td>
<td>0.00152</td>
<td>0.000065</td>
<td>0.000051</td>
<td>0.000018</td>
<td>0.000038</td>
<td>0.000017</td>
<td>0.000017</td>
<td></td>
</tr>
<tr>
<td>it</td>
<td>0.000031</td>
<td>0.00132</td>
<td>0.000046</td>
<td>0.000039</td>
<td>0.000018</td>
<td>0.000016</td>
<td>0.000017</td>
<td>0.000018</td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>0.000205</td>
<td>0.00132</td>
<td>0.000069</td>
<td>0.000038</td>
<td>0.000018</td>
<td>0.000020</td>
<td>0.000017</td>
<td>0.000018</td>
<td></td>
</tr>
<tr>
<td>other</td>
<td>0.000187</td>
<td>0.00132</td>
<td>0.000069</td>
<td>0.000049</td>
<td>0.000038</td>
<td>0.000038</td>
<td>0.000017</td>
<td>0.000025</td>
<td></td>
</tr>
<tr>
<td>food</td>
<td>0.000368</td>
<td>0.00132</td>
<td>0.000069</td>
<td>0.000049</td>
<td>0.000020</td>
<td>0.000020</td>
<td>0.000018</td>
<td>0.000025</td>
<td></td>
</tr>
<tr>
<td>hands</td>
<td>0.000164</td>
<td>0.00132</td>
<td>0.000069</td>
<td>0.000039</td>
<td>0.000018</td>
<td>0.000038</td>
<td>0.000017</td>
<td>0.000018</td>
<td></td>
</tr>
</tbody>
</table>
Backoff and Interpolation

- Another really useful source of knowledge
- If we are estimating:
  - trigram $P(z|xy)$
  - but $c(xyz)$ is zero
- Use info from:
  - Bigram $P(z|y)$
- Or even:
  - Unigram $P(z)$
- How to combine the trigram/bigram/unigram info?

Backoff versus Interpolation

- **Backoff**: use trigram if you have it, otherwise bigram, otherwise unigram
- **Interpolation**: mix all three

Interpolation

- Simple interpolation
  $$P(w_t|x_{t-1}, x_{t-2}) = \lambda_0 P(w_t|x_{t-1}) + \lambda_1 P(w_t|x_{t-2})$$

- Lambdas conditional on context:
  $$P(w_t|x_{t-1}, x_{t-2}) = \lambda_0 P(w_t|x_{t-1}) + \lambda_1 P(w_t|x_{t-2}) + \lambda_2 P(w_t|x_{t-3})$$
  $$\sum \lambda_i = 1$$
How to set the lambdas?

- Use a **held-out** corpus
- Choose lambdas which maximize the probability of some held-out data
  - i.e. fix the N-gram probabilities
  - Then search for lambda values
  - That when plugged into previous equation
  - Give largest probability for held-out set
  - Can use EM to do this search

Practical Issues

- We do everything in log space
  - Avoid underflow
  - (also adding is faster than multiplying)

\[ p_1 \times p_2 \times p_3 \times p_4 = \exp(\log p_1 + \log p_2 + \log p_3 + \log p_4) \]

Language Modeling Toolkits

- SRILM
- CMU-Cambridge LM Toolkit
Google N-Gram Release

All Our N-gram are Belong to You
By Peter Norvig - 03/05/08 11:26:50 AM
Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of NLP projects, such as statistical machine translation, speech recognition, spelling correction, entity detection, information extraction, and others. While such models have usually been estimated from training data, another avenue is to share the enormous dataset with everyone. We processed 1,024,006,297,229 words of running text and are publishing the counts for all 1,170,476,033 five-word sequences that appear at least 40 times. There are 12,849,391 unique words, after discarding words that appear less than 200 times.

Google N-Gram Release

- serve as the incoming 92
- serve as the incubator 99
- serve as the independent 794
- serve as the index 223
- serve as the indication 72
- serve as the indicator 120
- serve as the indicators 45
- serve as the indispensable 111
- serve as the indispensable 40
- serve as the individual 234

LM Summary

- Probability
  - Basic probability
  - Conditional probability
  - Bayes Rule
- Language Modeling (N-grams)
  - N-gram Intro
  - The Chain Rule
  - Perplexity
  - Smoothing:
    - Add-1
    - Good-Turing
Break

• Moving quiz to Thursday (2/14)
• Readings
  • Chapter 2: All
  • Chapter 3:
    • Skip 3.4.1 and 3.12
  • Chapter 4
    • Skip 4.7, 4.9, 4.10 and 4.11
  • Chapter 5
    • Read 5.1 through 5.5

Outline

• Probability
• Part of speech tagging
  • Parts of speech
  • Tag sets
  • Rule-based tagging
  • Statistical tagging
    • Simple most-frequent-tag baseline
  • Important Ideas
    • Training sets and test sets
    • Unknown words
    • Error analysis
    • HMM tagging

Part of Speech tagging

• Part of speech tagging
  • Parts of speech
  • What’s POS tagging good for anyhow?
  • Tag sets
  • Rule-based tagging
  • Statistical tagging
    • Simple most-frequent-tag baseline
  • Important Ideas
    • Training sets and test sets
    • Unknown words
    • HMM tagging
Parts of Speech

- 8 (ish) traditional parts of speech
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc
  - Called: parts-of-speech, lexical category, word classes, morphological classes, lexical tags, POS
  - Lots of debate in linguistics about the number, nature, and universality of these
    - We’ll completely ignore this debate.

POS examples

- N noun chair, bandwidth, pacing
- V verb study, debate, munch
- ADJ adjective purple, tall, ridiculous
- ADV adverb unfortunately, slowly
- P preposition of, by, to
- PRO pronoun I, me, mine
- DET determiner the, a, that, those

POS Tagging: Definition

- The process of assigning a part-of-speech or lexical class marker to each word in a corpus:
POS Tagging example

<table>
<thead>
<tr>
<th>WORD</th>
<th>tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>DET</td>
</tr>
<tr>
<td>koala</td>
<td>N</td>
</tr>
<tr>
<td>put</td>
<td>V</td>
</tr>
<tr>
<td>the</td>
<td>DET</td>
</tr>
<tr>
<td>keys</td>
<td>N</td>
</tr>
<tr>
<td>on</td>
<td>P</td>
</tr>
<tr>
<td>the</td>
<td>DET</td>
</tr>
<tr>
<td>table</td>
<td>N</td>
</tr>
</tbody>
</table>

What is POS tagging good for?

• First step of a vast number of practical tasks
• Speech synthesis
  • How to pronounce "lead"?
  • Insult inSULT
  • Object obJECT
  • OveRflow oveRFLOW
  • DIScount disCOUNT
  • CONTENT conTENT
• Parsing
  • Need to know if a word is an N or V before you can parse
• Information extraction
  • Finding names, relations, etc.
• Machine Translation

Open and Closed Classes

• Closed class: a relatively fixed membership
  • Prepositions: of, in, by, ...
  • Auxiliaries: may, can, will had, been, ...
  • Pronouns: I, you, she, mine, his, them, ...
  • Usually function words (short common words which play a role in grammar)
• Open class: new ones can be created all the time
  • English has 4: Nouns, Verbs, Adjectives, Adverbs
  • Many languages have these 4, but not all!
Open class words

• Nouns
  • Proper nouns (Boulder, Granby, Eli Manning)
  • Common nouns (the rest)
  • Count nouns and mass nouns
  • Mass: don’t get counted (snow, salt, communism) (two snows)
• Adverbs: tend to modify things
  • Unfortunately, John walked home extremely slowly yesterday
  • Directional/locative adverbs (here, home, downhill)
  • Degree adverbs (extremely, very, somewhat)
  • Manner adverbs (slowly, slickly, delicately)
• Verbs:
  • In English, have morphological affixes (inflectional)

Closed Class Words

• Idiosyncratic
• Examples:
  • prepositions: on, under, over, ...
  • particles: up, down, on, off, ...
  • determiners: a, an, the, ...
  • pronouns: she, who, I, ...
  • conjunctions: and, but, or, ...
  • auxiliary verbs: can, may should, ...
  • numerals: one, two, three, third, ...

Prepositions from CELEX

<table>
<thead>
<tr>
<th>of</th>
<th>5460083</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>3312315</td>
</tr>
<tr>
<td>for</td>
<td>142421</td>
</tr>
<tr>
<td>to</td>
<td>125691</td>
</tr>
<tr>
<td>with</td>
<td>124965</td>
</tr>
<tr>
<td>on</td>
<td>109129</td>
</tr>
<tr>
<td>at</td>
<td>100169</td>
</tr>
<tr>
<td>by</td>
<td>77704</td>
</tr>
<tr>
<td>from</td>
<td>74848</td>
</tr>
<tr>
<td>about</td>
<td>38426</td>
</tr>
<tr>
<td>than</td>
<td>202940</td>
</tr>
<tr>
<td>over</td>
<td>18077</td>
</tr>
</tbody>
</table>

| through | 14064 |
|         |       |
| between | 13273 |
| under   | 9525  |
| per     | 6515  |
| among   | 5096  |
| within  | 5030  |
| towards | 4700  |
| above   | 3056  |
| near    | 2026  |
| off     | 1695  |
| past    | 1575  |

| worth | 1563 |
| 12    |      |
| erase | 750  |
| till  | 686  |
| amongst| 525 |
| via   | 351  |
| midst | 164  |
| versus| 115  |
| amidst| 87  |
| sans  | 20   |
| circa | 14   |

| pace | 107 |
|      |     |
| negh | 9   |
| re   | 4   |
| mid  | 3   |
| o’er | 2   |
| but  | 0   |
| ere  | 0   |
| less | 0   |
| midot| 0   |
| o’   | 0   |
| thru | 0   |
| vice | 0   |
English particles

Conjunctions

POS tagging: Choosing a tagset

- There are so many parts of speech, potential distinctions we can draw
- To do POS tagging, need to choose a standard set of tags to work with
- Could pick very coarse tagets
  - N, V, Adj, Adv
- More commonly used set is finer grained, the "UPenn TreeBank tagset", 45 tags
  - PRP$, WRB, WP$, VBG
- Even more fine-grained tagsets exist
Using the UPenn tagset

• The grand jury commented on a number of other topics.

• Prepositions and subordinating conjunctions marked IN ("although/IN I/PRP...")

• Except the preposition/complementizer "to" is just marked "TO".

POS Tagging

• Words often have more than one POS:
  back
• The back door = JJ
• On my back = NN
• Win the voters back = RB
• Promised to back the bill = VB

• The POS tagging problem is to determine the POS tag for a particular instance of a word.

These examples from Dekang Lin
How hard is POS tagging?

Measuring ambiguity

<table>
<thead>
<tr>
<th></th>
<th>Original 47-tag corpus</th>
<th>Treebank 45-tag corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unambiguous (1 tag)</td>
<td>44,019</td>
<td>38,857</td>
</tr>
<tr>
<td>Ambiguos (2-7 tags)</td>
<td>5,490</td>
<td>8,944</td>
</tr>
<tr>
<td>Details: 2 tags</td>
<td>4,967</td>
<td>6,731</td>
</tr>
<tr>
<td>3 tags</td>
<td>411</td>
<td>1621</td>
</tr>
<tr>
<td>4 tags</td>
<td>91</td>
<td>557</td>
</tr>
<tr>
<td>5 tags</td>
<td>17</td>
<td>90</td>
</tr>
<tr>
<td>6 tags</td>
<td>2 (well, beat)</td>
<td>32</td>
</tr>
<tr>
<td>7 tags</td>
<td>2 (still, down)</td>
<td>6 (well, set, round, open, fit, down)</td>
</tr>
<tr>
<td>8 tags</td>
<td>4 (‘s, half, back, a)</td>
<td></td>
</tr>
<tr>
<td>9 tags</td>
<td>3 (that, more, in)</td>
<td></td>
</tr>
</tbody>
</table>

2 methods for POS tagging

1. Rule-based tagging
   • (ENGTWOL)
2. Stochastic (=Probabilistic) tagging
   • HMM (Hidden Markov Model) tagging

Rule-based tagging

• Start with a dictionary
• Assign all possible tags to words from the dictionary
• Write rules by hand to selectively remove tags
• Leaving the correct tag for each word.
Start with a dictionary

- she: PRP
- promised: VBN, VBD
- to: TO
- back: VB, JJ, RB, NN
- the: DT
- bill: NN, VB

- Etc… for the ~100,000 words of English

Use the dictionary to assign every possible tag

NN
RB
VBN
JJ
VB
PRP VBD
TO
VB
DT
NN
She promised
to
to
back
the
bill

Write rules to eliminate tags

Eliminate VBN if VBD is an option when VBN|VBD follows "<start> PRP"

VBN
NN
RB
JJ
VB
PRP VBD
TO
VB
DT
NN
She promised
to
to
back
the
bill
Stage 1 of ENGTWOL Tagging

- First Stage: Run words through FST morphological analyzer to get all parts of speech.
- Example: Pavlov had shown that salivation ...
  
Pavlov  PAVLOV N NOM SG PROPER
  had  HAVE V PAST VFIN SVO
  shown  HAVE PCP2 SVO
  that  ADV
  salivation  N NOM SG

Stage 2 of ENGTWOL Tagging

- Second Stage: Apply NEGATIVE constraints.
- Example: Adverbial “that” rule
  - Eliminates all readings of “that” except the one in “It isn’t that odd”

Given input: “that”
  If (+1 ADV/QUANT) ;if next word is adj/adv/quantifier
  (+2 SENT-LIM) ;following which is E-O-S
  (NOT -1 SVOC/A) ; and the previous word is not a CS
  ; verb like “consider” which
  ; allows adjective complements
  ; in “I consider that odd”
  Then eliminate non-ADV tags
  Else eliminate ADV

Hidden Markov Model Tagging

- Using an HMM to do POS tagging
- Is a special case of Bayesian inference
  - Foundational work in computational linguistics
  - Bledsoe 1959: OCR
  - Mosteller and Wallace 1964: authorship identification
- It is also related to the “noisy channel” model that’s the basis for ASR, OCR and MT
POS tagging as a sequence classification task

- We are given a sentence (an "observation" or "sequence of observations")
  - Secretariat is expected to race tomorrow
- What is the best sequence of tags which corresponds to this sequence of observations?
- Probabilistic view:
  - Consider all possible sequences of tags
  - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words w1...wn.

Getting to HMM

- We want, out of all sequences of n tags t1...tn, the single tag sequence such that P(t1...tn|w1...wn) is highest.
  \[ \hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n|w_1^n) \]
- Hat ^ means "our estimate of the best one"
- Argmax, f(x) means "the x such that f(x) is maximized"

Getting to HMM

- This equation is guaranteed to give us the best tag sequence
  \[ \hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n|w_1^n) \]
- But how to make it operational? How to compute this value?
- Intuition of Bayesian classification:
  - Use Bayes rule to transform into a set of other probabilities that are easier to compute
Using Bayes Rule

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

\[ \hat{t}_1^n = \arg\max_{t_1^n} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)} \]

\[ \hat{t}_1^n = \arg\max_{t_1^n} P(w_1^n|t_1^n)P(t_1^n) \]

Likelihood and Prior

\[ \hat{t}_1^n = \arg\max_{t_1^n} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(t_1^n)} \]

\[ P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i) \]

\[ P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1}) \]

\[ \hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n|w_1^n) \approx \arg\max_{t_1^n} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}) \]

Two Kinds of probabilities (1)

- Tag transition probabilities \( p(t_i|t_{i-1}) \)
  - Determiners likely to precede adjs and nouns
    - That/DT flight/NN
    - The/DT yellow/JJ hat/NN
    - So we expect \( P(\text{NN|DT}) \) and \( P(\text{JJ|DT}) \) to be high
    - But \( P(\text{DT|JJ}) \) to be:
  - Compute \( P(\text{NN|DT}) \) by counting in a labeled corpus:

\[ P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})} \]

\[ P(\text{NN|DT}) = \frac{C(\text{DT, NN})}{C(\text{DT})} = \frac{56,509}{116,454} = .49 \]
Two kinds of probabilities (2)

- Word likelihood probabilities \( p(w_i|t_i) \)
- VBZ (3sg Pres verb) likely to be “is”
- Compute \( P(\text{is}|\text{VBZ}) \) by counting in a labeled corpus:

\[
P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}
\]

\[
P(\text{is}|\text{VBZ}) = \frac{C(\text{VBZ}, \text{is})}{C(\text{VBZ})} = \frac{10,073}{21,627} = .47
\]

An Example: the verb “race”

- Secretariat/NNP is/VBZ expected/VBN to/TO race/VB tomorrow/NR
- People/NNS continue/VB to/TO inquire/VB the/DN reason/NN for/IN the/DN race/NN for/IN outer/JJ space/NN
- How do we pick the right tag?

Disambiguating “race”
Example

- \( P(\text{NN}|\text{TO}) = .00047 \)
- \( P(\text{VB}|\text{TO}) = .83 \)
- \( P(\text{race}|\text{NN}) = .00057 \)
- \( P(\text{race}|\text{VB}) = .00012 \)
- \( P(\text{NR}|\text{VB}) = .0027 \)
- \( P(\text{NR}|\text{NN}) = .0012 \)
- \( P(\text{VB}|\text{TO})P(\text{NR}|\text{VB})P(\text{race}|\text{VB}) = .00000027 \)
- \( P(\text{NN}|\text{TO})P(\text{NR}|\text{NN})P(\text{race}|\text{NN}) = .00000000032 \)

So we (correctly) choose the verb reading.

Hidden Markov Models

- What we’ve described with these two kinds of probabilities is a Hidden Markov Model
- Let’s just spend a bit of time tying this into the model
- First some definitions.

Definitions

- A weighted finite-state automaton adds probabilities to the arcs
  - The sum of the probabilities leaving any arc must sum to one
- A Markov chain is a special case of a WFST in which the input sequence uniquely determines which states the automaton will go through
- Markov chains can’t represent inherently ambiguous problems
  - Useful for assigning probabilities to unambiguous sequences
Markov chain for weather

Markov chain for words

Markov chain = “First-order observable Markov Model”

• A set of states
  • \( Q = q_1, q_2, \ldots, q_N \); the state at time \( t \) is \( q_t \)
• Transition probabilities:
  • a set of probabilities \( A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn} \)
  • Each \( a_{ij} \) represents the probability of transitioning from state \( i \) to state \( j \)
  • The set of these is the transition probability matrix \( A \)
• Current state only depends on previous state
  \[ p(q_t | q_{t-1}, q_{t-2}, \ldots, q_1) = p(q_t | q_{t-1}) \]
**Markov chain for weather**

- What is the probability of 4 consecutive rainy days?
- Sequence is rainy-rainy-rainy-rainy
- I.e., state sequence is 3-3-3-3
- \( P(3,3,3,3) = \pi_3 a_{11} a_{11} a_{11} = 0.2 \times (0.6)^3 = 0.0432 \)

**HMM for Ice Cream**

- You are a climatologist in the year 2799
- Studying global warming
- You can’t find any records of the weather in Baltimore, MA for summer of 2007
- But you find Jason Eisner’s diary
- Which lists how many ice-creams Jason ate every date that summer
- Our job: figure out how hot it was

**Hidden Markov Model**

- For Markov chains, the output symbols are the same as the states.
  - See hot weather, we’re in state hot
- But in part-of-speech tagging (and other things)
  - The output symbols are words
  - But the hidden states are part-of-speech tags
- So we need an extension!
- A Hidden Markov Model is an extension of a Markov chain in which the input symbols are not the same as the states.
- This means we don’t know which state we are in.
Hidden Markov Models

- States $Q = q_1, q_2, ..., q_N$
- Observations $O = o_1, o_2, ..., o_N$
  - Each observation is a symbol from a vocabulary $V = \{v_1, v_2, ..., v_N\}$
- Transition probabilities
  - Transition probability matrix $A = \{a_{ij}\}$
- Observation likelihoods
  - Output probability matrix $B = \{b_i(k)\}$
- Special initial probability vector $\pi$

Eisner task

- Given
  - Ice Cream Observation Sequence: 1,2,3,2,2,2,3...
- Produce:

HMM for ice cream
Transitions between the hidden states of HMM, showing A probs

B observation likelihoods for POS HMM