Lecture 5
Part-of-Speech Tagging

CS 6320
Outline

- English word classes
- Penn Treebank tagset
- Rule-Based POS Tagging
- HMM POS Tagging
- Transformation-Based POS Tagging
- Probabilistic Models of Spelling
  - Noisy Channel Model
  - Applying Bayes Noisy Channel to Spelling
Closed Word Classes

- Prepositions: usually before a NP. 
  (on, under, over, at, from, etc.)
- Determiners: usually precede a NP. 
  (a, an, the, etc.)
- Pronouns: can replace an NP. 
  (she, who, I, others, etc.)
- Conjunctions: join two phrases, clauses or sentences. 
  (and, but, or, etc.)
- Auxiliary verbs 
  (can, may, should, are)
- Particles: appear after verbs. 
  (up, down, on, off, in, out)
- Numerals 
  (one, two, first, second)
Open Word Classes

- Nouns
- Verbs
- Adjectives
- Adverbs
## Prepositions from CELEX

<table>
<thead>
<tr>
<th>Preposition</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>of</td>
<td>540,085</td>
</tr>
<tr>
<td>in</td>
<td>331,235</td>
</tr>
<tr>
<td>for</td>
<td>142,421</td>
</tr>
<tr>
<td>to</td>
<td>125,691</td>
</tr>
<tr>
<td>with</td>
<td>124,965</td>
</tr>
<tr>
<td>on</td>
<td>109,129</td>
</tr>
<tr>
<td>at</td>
<td>100,169</td>
</tr>
<tr>
<td>by</td>
<td>77,794</td>
</tr>
<tr>
<td>from</td>
<td>74,843</td>
</tr>
<tr>
<td>about</td>
<td>38,428</td>
</tr>
<tr>
<td>than</td>
<td>20,210</td>
</tr>
<tr>
<td>over</td>
<td>18,071</td>
</tr>
<tr>
<td>through</td>
<td>14,964</td>
</tr>
<tr>
<td>after</td>
<td>13,670</td>
</tr>
<tr>
<td>between</td>
<td>13,275</td>
</tr>
<tr>
<td>under</td>
<td>9,525</td>
</tr>
<tr>
<td>per</td>
<td>6,515</td>
</tr>
<tr>
<td>among</td>
<td>6,090</td>
</tr>
<tr>
<td>within</td>
<td>5,030</td>
</tr>
<tr>
<td>towards</td>
<td>4,700</td>
</tr>
<tr>
<td>above</td>
<td>3,056</td>
</tr>
<tr>
<td>near</td>
<td>2,026</td>
</tr>
<tr>
<td>off</td>
<td>1,695</td>
</tr>
<tr>
<td>past</td>
<td>1,575</td>
</tr>
<tr>
<td>worth</td>
<td>1,563</td>
</tr>
<tr>
<td>toward</td>
<td>1,390</td>
</tr>
<tr>
<td>plus</td>
<td>750</td>
</tr>
<tr>
<td>till</td>
<td>686</td>
</tr>
<tr>
<td>amongst</td>
<td>525</td>
</tr>
<tr>
<td>via</td>
<td>351</td>
</tr>
<tr>
<td>amid</td>
<td>222</td>
</tr>
<tr>
<td>underneath</td>
<td>164</td>
</tr>
<tr>
<td>versus</td>
<td>113</td>
</tr>
<tr>
<td>amidst</td>
<td>67</td>
</tr>
<tr>
<td>sans</td>
<td>20</td>
</tr>
<tr>
<td>circa</td>
<td>14</td>
</tr>
<tr>
<td>pace</td>
<td>12</td>
</tr>
<tr>
<td>nigh</td>
<td>9</td>
</tr>
<tr>
<td>re</td>
<td>4</td>
</tr>
<tr>
<td>mid</td>
<td>3</td>
</tr>
<tr>
<td>o’er</td>
<td>2</td>
</tr>
<tr>
<td>but</td>
<td>0</td>
</tr>
<tr>
<td>ere</td>
<td>0</td>
</tr>
<tr>
<td>less</td>
<td>0</td>
</tr>
<tr>
<td>midst</td>
<td>0</td>
</tr>
<tr>
<td>o’</td>
<td>0</td>
</tr>
<tr>
<td>thru</td>
<td>0</td>
</tr>
<tr>
<td>vice</td>
<td>0</td>
</tr>
</tbody>
</table>
## English Particles

<table>
<thead>
<tr>
<th>aboard</th>
<th>aside</th>
<th>besides</th>
<th>forward(s)</th>
<th>opposite</th>
<th>through</th>
</tr>
</thead>
<tbody>
<tr>
<td>about</td>
<td>astray</td>
<td>between</td>
<td>home</td>
<td>out</td>
<td>throughout</td>
</tr>
<tr>
<td>above</td>
<td>away</td>
<td>beyond</td>
<td>in</td>
<td>outside</td>
<td>together</td>
</tr>
<tr>
<td>across</td>
<td>back</td>
<td>by</td>
<td>inside</td>
<td>over</td>
<td>under</td>
</tr>
<tr>
<td>ahead</td>
<td>before</td>
<td>close</td>
<td>instead</td>
<td>overhead</td>
<td>underneath</td>
</tr>
<tr>
<td>alongside</td>
<td>behind</td>
<td>down</td>
<td>near</td>
<td>past</td>
<td>up</td>
</tr>
<tr>
<td>apart</td>
<td>below</td>
<td>east, etc.</td>
<td>off</td>
<td>round</td>
<td>within</td>
</tr>
<tr>
<td>around</td>
<td>beneath</td>
<td>eastward(s),etc.</td>
<td>on</td>
<td>since</td>
<td>without</td>
</tr>
</tbody>
</table>
## Conjunctions

<table>
<thead>
<tr>
<th>conjunction</th>
<th>occurrences</th>
<th>occurrences</th>
<th>occurrences</th>
<th>occurrences</th>
<th>occurrences</th>
<th>occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>514,946</td>
<td>5,040</td>
<td>considering</td>
<td>174</td>
<td>forasmuch as</td>
<td>0</td>
</tr>
<tr>
<td>that</td>
<td>134,773</td>
<td>4,843</td>
<td>lest</td>
<td>131</td>
<td>however</td>
<td>0</td>
</tr>
<tr>
<td>but</td>
<td>96,889</td>
<td>3,952</td>
<td>albeit</td>
<td>104</td>
<td>immediately</td>
<td>0</td>
</tr>
<tr>
<td>or</td>
<td>76,563</td>
<td>3,078</td>
<td>providing</td>
<td>96</td>
<td>in as far as</td>
<td>0</td>
</tr>
<tr>
<td>as</td>
<td>54,608</td>
<td>2,826</td>
<td>whereupon</td>
<td>85</td>
<td>in so far as</td>
<td>0</td>
</tr>
<tr>
<td>if</td>
<td>53,917</td>
<td>2,205</td>
<td>seeing</td>
<td>63</td>
<td>inasmuch as</td>
<td>0</td>
</tr>
<tr>
<td>when</td>
<td>37,975</td>
<td>1,333</td>
<td>directly</td>
<td>26</td>
<td>insomuch as</td>
<td>0</td>
</tr>
<tr>
<td>because</td>
<td>23,626</td>
<td>1,290</td>
<td>ere</td>
<td>12</td>
<td>insomuch that</td>
<td>0</td>
</tr>
<tr>
<td>so</td>
<td>12,933</td>
<td>1,120</td>
<td>notwithstanding</td>
<td>3</td>
<td>like</td>
<td>0</td>
</tr>
<tr>
<td>before</td>
<td>10,720</td>
<td>913</td>
<td>according as</td>
<td>0</td>
<td>neither nor</td>
<td>0</td>
</tr>
<tr>
<td>though</td>
<td>10,329</td>
<td>867</td>
<td>as if</td>
<td>0</td>
<td>now that</td>
<td>0</td>
</tr>
<tr>
<td>than</td>
<td>9,511</td>
<td>864</td>
<td>as long as</td>
<td>0</td>
<td>only</td>
<td>0</td>
</tr>
<tr>
<td>while</td>
<td>8,144</td>
<td>686</td>
<td>as though</td>
<td>0</td>
<td>provided that</td>
<td>0</td>
</tr>
<tr>
<td>after</td>
<td>7,042</td>
<td>594</td>
<td>both and</td>
<td>0</td>
<td>providing that</td>
<td>0</td>
</tr>
<tr>
<td>whether</td>
<td>5,978</td>
<td>351</td>
<td>but that</td>
<td>0</td>
<td>seeing as</td>
<td>0</td>
</tr>
<tr>
<td>for</td>
<td>5,935</td>
<td>281</td>
<td>but then</td>
<td>0</td>
<td>seeing as how</td>
<td>0</td>
</tr>
<tr>
<td>although</td>
<td>5,424</td>
<td>188</td>
<td>but then again</td>
<td>0</td>
<td>seeing that</td>
<td>0</td>
</tr>
<tr>
<td>until</td>
<td>5,072</td>
<td>185</td>
<td>either or</td>
<td>0</td>
<td>without</td>
<td>0</td>
</tr>
</tbody>
</table>
Tags

- There are various tagsets to choose from.
- POS tagging is the assignment of correct POS tags for each word in an input sentence.
- Penn Treebank part-of-speech tags.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Coordin. Conjunction</td>
<td>and, but, or</td>
<td>SYM</td>
<td>Symbol</td>
<td>+, %, &amp;</td>
</tr>
<tr>
<td>CD</td>
<td>Cardinal number</td>
<td>one, two, three</td>
<td>TO</td>
<td>“to”</td>
<td>to</td>
</tr>
<tr>
<td>DT</td>
<td>Determiner</td>
<td>a, the</td>
<td>UH</td>
<td>Interjection</td>
<td>ah, oops</td>
</tr>
<tr>
<td>EX</td>
<td>Existential ‘there’</td>
<td>there</td>
<td>VB</td>
<td>Verb, base form</td>
<td>eat</td>
</tr>
<tr>
<td>FW</td>
<td>Foreign word</td>
<td>mea culpa</td>
<td>VBD</td>
<td>Verb, past tense</td>
<td>ate</td>
</tr>
<tr>
<td>IN</td>
<td>Preposition/sub-conj</td>
<td>of, in, by</td>
<td>VBG</td>
<td>Verb, gerund</td>
<td>eating</td>
</tr>
<tr>
<td>JJ</td>
<td>Adjective</td>
<td>yellow</td>
<td>VBN</td>
<td>Verb, past participle</td>
<td>eaten</td>
</tr>
<tr>
<td>JJR</td>
<td>Adj., comparative</td>
<td>bigger</td>
<td>VBP</td>
<td>Verb, non-3sg pres</td>
<td>eat</td>
</tr>
<tr>
<td>JJS</td>
<td>Adj., superlative</td>
<td>wildest</td>
<td>VBZ</td>
<td>Verb, 3sg pres</td>
<td>eats</td>
</tr>
<tr>
<td>LS</td>
<td>List item marker</td>
<td>1, 2, One</td>
<td>WDT</td>
<td>Wh-determiner</td>
<td>which, that</td>
</tr>
<tr>
<td>MD</td>
<td>Modal</td>
<td>can, should</td>
<td>WP</td>
<td>Wh-pronoun</td>
<td>what, who</td>
</tr>
<tr>
<td>NN</td>
<td>Noun, sing. or mass</td>
<td>llama</td>
<td>WP$</td>
<td>Possessive who-</td>
<td>whose</td>
</tr>
<tr>
<td>NNS</td>
<td>Noun, plural</td>
<td>llamas</td>
<td>WRB</td>
<td>Wh-adverb</td>
<td>how, where</td>
</tr>
<tr>
<td>NNP</td>
<td>Proper noun, singular</td>
<td>IBM</td>
<td>$</td>
<td>Dollar sign</td>
<td>$</td>
</tr>
<tr>
<td>NNPS</td>
<td>Proper noun, plural</td>
<td>Carolinas</td>
<td>#</td>
<td>Pound sign</td>
<td>#</td>
</tr>
<tr>
<td>PDT</td>
<td>Predeterminer</td>
<td>all, both</td>
<td>“</td>
<td>Left quote</td>
<td>(&quot; or &quot;&quot;)</td>
</tr>
<tr>
<td>POS</td>
<td>Possessive ending</td>
<td>’s</td>
<td>”</td>
<td>Right quote</td>
<td>(’ or ”)</td>
</tr>
<tr>
<td>PP</td>
<td>Personal pronoun</td>
<td>I, you, he</td>
<td>(</td>
<td>Left parenthesis</td>
<td>[ ], (, {, &lt;)</td>
</tr>
<tr>
<td>PP$</td>
<td>Possessive pronoun</td>
<td>your, one’s</td>
<td>)</td>
<td>Right parenthesis</td>
<td>[ ], ), }, &gt;)</td>
</tr>
<tr>
<td>RB</td>
<td>Adverb</td>
<td>quickly, never</td>
<td>,</td>
<td>Comma</td>
<td>.</td>
</tr>
<tr>
<td>RBR</td>
<td>Adverb, comparative</td>
<td>faster</td>
<td>;</td>
<td>Sentence-final punc</td>
<td>( . ! ?)</td>
</tr>
<tr>
<td>RBS</td>
<td>Adverb, superlative</td>
<td>fastest</td>
<td>:</td>
<td>Mid-sentence punc</td>
<td>( ; ... - -)</td>
</tr>
</tbody>
</table>
Why is POS Tagging Useful?

- 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
Using the Penn Tagset

- The grand jury commented on a number of other topics.
- Prepositions and subordinating conjunctions marked IN ("although I")
- 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>87-tag Original Brown</th>
<th>45-tag Treebank Brown</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unambiguous (1 tag)</strong></td>
<td>44,019</td>
<td>38,857</td>
</tr>
<tr>
<td><strong>Ambiguous (2–7 tags)</strong></td>
<td>5,490</td>
<td>8,844</td>
</tr>
<tr>
<td><strong>Details:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 tags</td>
<td>4,967</td>
<td>6,731</td>
</tr>
<tr>
<td>3 tags</td>
<td>411</td>
<td>1,621</td>
</tr>
<tr>
<td>4 tags</td>
<td>91</td>
<td>357</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></td>
<td>4 (’s, half, back, a)</td>
</tr>
<tr>
<td>9 tags</td>
<td></td>
<td>3 (that, more, in)</td>
</tr>
</tbody>
</table>
POS - Tagging

- We use:
  - A set of tags
  - A dictionary that indicates all possible tags for each word
  - Input text
  - General purpose vs. special purpose

- Methods:
  - Rule-based tagging
  - Statistical tagging
  - Transformation--based 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 with more than 1 tag
Assign Every Possible Tag

She promised to back the bill

NN
RB
VBN JJ VB
PRP VBD TO VB DT NN

© 2016 Dan I. Moldovan, Human Language Technology Research Institute, The University of Texas at Dallas
Write Rules to Eliminate Tags

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

<table>
<thead>
<tr>
<th>PRP</th>
<th>VBD</th>
<th>TO</th>
<th>VB</th>
<th>DT</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>She</td>
<td>promised</td>
<td>to</td>
<td>back</td>
<td>the</td>
<td>bill</td>
</tr>
</tbody>
</table>

She promised 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 ...*

<table>
<thead>
<tr>
<th>Word</th>
<th>Tagging</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pavlov</td>
<td>PAVLOV N NOM SG PROPER</td>
</tr>
<tr>
<td>had</td>
<td>HAVE V PAST VFIN SVO</td>
</tr>
<tr>
<td>shown</td>
<td>SHOW PCP2 SVOO SVO SV</td>
</tr>
<tr>
<td>that</td>
<td>ADV</td>
</tr>
<tr>
<td>salivation</td>
<td>N NOM SG</td>
</tr>
</tbody>
</table>
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 A/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 verb like “consider” which ; allows adjective complements

*Then* eliminate non-ADV tags
*Else* eliminate ADV
Transformation-based Tagging

- The pure rule-based approach is too expensive, slow, tedious.
- Brill's Transformation-Based Learning (TLB)
- Basic idea is to do a poor job first, and then use learned rules to improve things.
- Example: tag ``race"
  Step 1:
  \[ P(\text{race}|\text{NN}) = 0.98 \]
  \[ P(\text{race}|\text{VB}) = 0.02 \]
  - Tag all uses of race as nouns
    is/VBZ expected/VBN to/TO race/NN tomorrow/NN
    the/DT race/NN for/IN outer/JJ space/NN
Step 2:
- Rule: Change NN to VB when the previous tag is TO.
- Assume some tagged training corpus.
- Make the assumption that the word depends only on its tag.
How are the rules learned?

1. Tag the corpus with the most likely tag for each word (unigram model)
2. Choose a transformation that deterministically replaces an existing tag with a new tag such that the resulting tagged training corpus has the lowest error rate out of all transformations
3. Apply that transformation to the training set
4. Iterate
5. Return as your tagger one that:
   - First tags using unigrams and then
   - Applies the learned transformations in order.
Brill’s Tagging 3/4

- Transformations
- Change Tag a to Tag b when:

<table>
<thead>
<tr>
<th>#</th>
<th>Change tags</th>
<th>From</th>
<th>To</th>
<th>Condition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NN</td>
<td>VB</td>
<td></td>
<td>Previous tag is TO</td>
<td>to/TO race/NN → VB</td>
</tr>
<tr>
<td>2</td>
<td>VBP</td>
<td>VB</td>
<td></td>
<td>One of the previous 3 tags is MD</td>
<td>might/MD vanish/VBP → VB</td>
</tr>
<tr>
<td>3</td>
<td>NN</td>
<td>VB</td>
<td></td>
<td>One of the previous 2 tags is MD</td>
<td>might/MD not reply/NN → VB</td>
</tr>
<tr>
<td>4</td>
<td>VB</td>
<td>NN</td>
<td></td>
<td>One of the previous 2 tags is DT</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>VBD</td>
<td>VBN</td>
<td></td>
<td>One of the previous 3 tags is VBZ</td>
<td></td>
</tr>
</tbody>
</table>
Brill’s Tagging 4/4

```python
function TBL(corpus) returns transforms-queue
    INITIALIZE WITH MOST LIKELY TAGS(corpus)
    until end condition is met do
        templates ← GENERATE POTENTIAL RELEVANT TEMPLATES
        best-transform ← GET-BEST-TRANSFORM(corpus, templates)
        APPLY-TRANSFORM(best-transform, corpus)
        ENQUEUE(best-transform-rule, transforms-queue)
    end
    return(transforms-queue)
```

```python
function GET-BEST-TRANSFORM(corpus, templates) returns transform
    for each template in templates
    (instance, score) ← GET-BEST-INSTANCE(corpus, template)
    if (score > best-transform.score) then best-transform ← (instance, score)
    return(best-transform)
```

```python
function GET-BEST-INSTANCE(corpus, template) returns transform
    for from-tag ← from tag - 1 to tag - n do
        for to-tag ← from tag - 1 to tag - n do
            for pos ← from 1 to corpus-size do
                if (correct-tag(pos) == to-tag && current-tag(pos) == from-tag)
                    num-good-transforms(current-tag(pos - 1))++
                elseif (correct-tag(pos) == from-tag && current-tag(pos) == from-tag)
                    num-bad-transforms(current-tag(pos - 1))++
                end
            end
            best-Z ← ARGMAX(num-good-transforms(1) + num-bad-transforms(1))
            if (num-good-transforms(best-Z) - num-bad-transforms(best-Z) > best-instance.Z) then
                best-instance ← “Change tag from from-tag to to-tag"
                if previous tag is best-Z
            end
        end
    end
    return(best-instance)
```

```python
procedure APPLY-TRANSFORM(transform, corpus)
    for pos ← from 1 to corpus-size do
        if (current-tag(pos) == best-rule-from)
            && (current-tag(pos - 1) == best-rule-prev)
            current-tag(pos) = best-rule-to
```

© 2016 Dan I. Moldovan, Human Language Technology Research Institute, The University of Texas at Dallas
POS Tagging as Sequence Classification

- We are given a sentence (an “observation” or “sequence of observations”)
  - *Secretariat is expected to race tomorrow*
- What is the best sequence of tags that 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 $w_1...w_n$. 
Using Hidden Markov Models (HMMs)

- We want, out of all sequences of \( n \) tags \( t_1 \ldots t_n \) the single tag sequence such that \( P(t_1 \ldots t_n | w_1 \ldots w_n) \) is highest.

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

- Hat \( \hat{\cdot} \) means “our estimate of the best one”
- \( \arg \max_x f(x) \) means “the \( x \) such that \( f(x) \) is maximized”
Getting to HMMs

- This equation is guaranteed to give us the best tag sequence
  \[ \hat{t}_1^n = \arg \max_{t_1^n} P(t_1^n \mid w_1^n) \]

- But how to make it operational? How to compute this value?

- Intuition of Bayesian classification:
  - Use Bayes rule to transform this equation 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

\[ \hat{t}_1^n = \arg\max_{t_1^n} \left\{ \underbrace{P(w_1^n|t_1^n)}_{\text{likelihood}} \right\} \underbrace{P(t_1^n)}_{\text{prior}} \]

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

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

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

- 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 \( P(\text{NN}|\text{DT}) \) and \( P(\text{JJ}|\text{DT}) \) to be high
    - But \( P(\text{DT}|\text{JJ}) \) to be low
  - Compute \( P(\text{NN}|\text{DT}) \) by counting in a labeled corpus:
    \[
P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}
    \]
    \[
P(\text{NN}|\text{DT}) = \frac{C(\text{DT}, \text{NN})}{C(\text{DT})} = \frac{56,509}{116,454} = .49
    \]
Two Kinds of Probabilities

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

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

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$
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/DT reason/NN for/IN the/DT race/NN for/IN outer/JJ space/NN
- How do we pick the right tag?
Disambiguating “race”

(a) Secretariat is expected to race tomorrow

(b) Secretariat is expected to race tomorrow
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,
Statistical POS Tagging using Trigrams

- In general:  \[ \arg \max \ P(\text{Tag Sequence} | \text{Word Sequence}) \]

- Rewrite this:

\[
\arg \max \frac{P(\text{Tag Sequence} \mid \text{Word Sequence})}{P(\text{Word Sequence})} = \arg \max \frac{P(T)}{P(W \mid T)}
\]

\[
\hat{T} = \arg \max \ P(T)P(W \mid T)
\]

\[
P(T)P(W \mid T) = \prod_{i=1}^{n} P(w_i \mid w_{i-1}t_{i-1}t_i)P(t_i \mid w_{i-1}t_{i-1})
\]

- \( P(\text{Tag Sequence} ) = P(t_1)P(t_2 \mid t_1) \prod_{i=3}^{n} P(t_i \mid t_{i-2}t_{i-1}) \)

- This is easy to get from simple word counting and smoothing:

\[
P(t_i \mid t_{i-2}t_{i-1}) = \frac{C(t_{i-2}t_{i-1}t_i)}{C(t_{i-2}t_{i-1})}
\]
Statistical POS Tagging

- Make the assumption that the word depends only on its tag.
  
  $P(\text{Word Sequence} | \text{Tag Sequence})$

  
  \[
P(w_i | w_1 t_1 \cdots w_{i-1} t_{i-1}) = P(w_i | t_i)
  \]

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

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

- Combine the two factors.

  \[
P(t_1)P(t_2 | t_1)\prod_{i=3}^{n} P(t_i | t_{i-2}, t_{i-1}) \left[ \prod_{i=1}^{n} P(w_i | t_i) \right]
  \]

- The state transition probabilities come from the language model.
- The emission probabilities come from $P(\text{word} | \text{tag})$.
- Find the best tag sequence - overall.
Probabilistic Models of Spelling
Some problems with Finite State Machines:
- In the case of a reject, they offer no advice as to why a string was rejected.
- In the case of global ambiguity (two or more accept conditions or paths) FSAs simply provide all or one.

So...we're going to make a brief digression away from FSAs to look at alternative. Then we'll find a way back to FSAs by augmenting them slightly.
The Noisy Channel Model

Many problems in language processing can be viewed as noisy channel problems:
- Optical character recognition,
- Spelling correction,
- Speech recognition,
- Machine translation, etc.
Basic Probability Background

- Basic Probability Background
  - Prior Probability (or unconditional probability): $P(A)$

- Think of $A$ as a proposition as in some simple propositional logic.

- Also useful...

  \[ P(A \land B), P(A \lor B), P(\neg A \land B), etc \]
Relating Conditionals and Priors

\[
P ( A \mid B ) = \frac{P ( A \wedge B )}{P ( B )}
\]

rearranging yields...

\[
P ( A \wedge B ) = P ( A \mid B ) P ( B )
\]

you could also say...

\[
P ( A \wedge B ) = P ( B \mid A ) P ( A )
\]
Bayesian Reasoning

We know ... \( P(A \land B) = P(A \mid B)P(B) \)
and that \( P(A \land B) = P(B \mid A)P(A) \)

so \( P(A \mid B)P(B) = P(B \mid A)P(A) \)

or \( P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \)

or \( P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A)} \)

This is commonly known as Bayes law. The key point is that we can move from \( P(A \mid B) \) to \( P(B \mid A) \) and back given appropriate information.
In applying probability theory to a noisy channel what we're looking for is the most probable source given the observed signal. We can denote this as: \( \arg \max_{\text{Source}} P(\text{Source} \mid \text{Signal}) \)

Unfortunately, we don’t usually know how to compute this, so...

Rewrite ... 
\[
P(\text{Source} \mid \text{Signal}) = \frac{P(\text{Signal} \mid \text{Source}) P(\text{Source})}{P(\text{Signal})}
\]

Giving  
\[
\arg \max_{\text{Source}} = \frac{P(\text{Signal} \mid \text{Source}) P(\text{Source})}{P(\text{Signal})}
\]
Applying Bayes to a Noisy Channel 2/2

How does this help if all we have is the signal and the source is exactly what we don’t know?

We know the space of possible sources. We can plug each into the equation one by one and compute their probabilities using this equation. The source hypothesis with the highest probability wins.
Applying Bayes Noisy Channel to Spelling

- We have some word that has been misspelled and we want to know the real word. The real word is the source, and the misspelled word is the signal.

- Assume that \( V \) is the space of all the words we know.

\[
\hat{w} = \arg \max_{w \in V} \frac{P(s \mid w)P(w)}{P(s)}
\]

\( \hat{w} \) denotes the correct word, and \( s \) denotes the misspelling.
Getting the Numbers

- It seems like we need...
  - $P(s/w)$
  - $P(w)$
  - $P(s)$

- Let's first consider the $P(s)$ in the equation...

\[
\hat{w} = \arg \max_{w \in V} \frac{P(s \mid w)P(w)}{P(s)}
\]

What about $P(w)$...

\[
\hat{w} = \arg \max_{w \in V} \frac{P(s \mid w)P(w)}{P(s)}
\]

What about $P(s/w)$...

\[
\hat{w} = \arg \max_{w \in V} \frac{P(s \mid w)P(w)}{P(s)}
\]

Let’s consider the example ... collect statistics on the probability of misspelling “actress” and “acress”.
Spelling Error Patterns

- It is fruitless to try to collect statistics about the misspellings of individual words given a large dictionary. You'll likely never get enough data.

- We need a way to compute $P(s|w)$ without using direct information.

- This is where spelling error patterns come in...
  - Insertion: ther for the
  - Deletion: ther for there
  - Substitution: noq for now
  - Transportation: teh for the
Spelling Error Statistics 1/2

- Collect statistics for each error type from a large corpus.

- For example... asking for $P(\text{acress}|\text{actress})$ is assumed to be the same as asking for the probability that a deletion of "t" happened here.

- So... just collect a large corpus of text (containing errors) and see how often "t" gets deleted.
For the general case we can use:

- del[xy]: number of times in the training set when characters $xy$ were typed as $x$.
- ins[x,y]: number of times when $x$ in the correct word was typed as $xy$.
- sub[x,y]: number of times when $x$ was typed as $y$.
- trans[x,y]: number of times when $xy$ was typed as $yx$. 
Kernighan’s Statistics

\[
P(t \mid c) = \begin{cases} 
  \text{deletion} & \frac{\text{del}[c_{p-1}, c_p]}{\text{count}[c_{p-1}, c_p]}, \\
  \text{insertion} & \frac{\text{ins}[c_{p-1}, t_p]}{\text{count}[c_{p-1}]}, \\
  \text{substitution} & \frac{\text{sub}[t_p, c_p]}{\text{count}[c_p]}, \\
  \text{transposition} & \frac{\text{trans}[c_p, c_{p+1}]}{\text{count}[c_p, c_{p+1}]}.
\end{cases}
\]

- So... for transposing h and e as in “teh” for “the”. Divide the numbers of occurrences of “eh” that are errors by the total number of “eh” occurrences in the corpus.
Kernighan Method...

- Apply all possible single spelling changes to the misspelled word.

- Collect all the resulting strings that are actually words.

- Compute the probability of each of those candidate words.

- Display them ranked to the user.
Results for misspelled word “acress”

\[ \text{freq}(c) = C(c) \]
\[ P(c) = \frac{C(c) + 0.5}{N + 0.5V} \]
\[ N = 44 \text{ Million} \]

<table>
<thead>
<tr>
<th>c</th>
<th>freq(c)</th>
<th>p(c)</th>
<th>p(tlc)</th>
<th>p(tlc)p(c)</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>actress</td>
<td>1343</td>
<td>.0000315</td>
<td>.000117</td>
<td>3.69 \times 10^{-9}</td>
<td>37%</td>
</tr>
<tr>
<td>cress</td>
<td>0</td>
<td>.000000014</td>
<td>.00000144</td>
<td>2.02 \times 10^{-14}</td>
<td>0%</td>
</tr>
<tr>
<td>caress</td>
<td>4</td>
<td>.0000001</td>
<td>.00000164</td>
<td>1.64 \times 10^{-13}</td>
<td>0%</td>
</tr>
<tr>
<td>access</td>
<td>2280</td>
<td>.000058</td>
<td>.000000209</td>
<td>1.21 \times 10^{-11}</td>
<td>0%</td>
</tr>
<tr>
<td>across</td>
<td>8436</td>
<td>.00019</td>
<td>.0000093</td>
<td>1.77 \times 10^{-9}</td>
<td>18%</td>
</tr>
<tr>
<td>acres</td>
<td>2879</td>
<td>.000065</td>
<td>.0000321</td>
<td>2.09 \times 10^{-9}</td>
<td>21%</td>
</tr>
<tr>
<td>acres</td>
<td>2879</td>
<td>.000065</td>
<td>.0000342</td>
<td>2.22 \times 10^{-9}</td>
<td>23%</td>
</tr>
</tbody>
</table>

Conclusion: most likely the intended word was “acres”
Problems

- Doesn't incorporate contextual information:
  a stellar and versatile across whose...

- Chapter 6 will address this.

- There are usually a million ways to instantiate a Bayesian model. It depends on whether you are willing/able to collect the data needed for the model you'd like to choose.

- Big issues in this approach
  - The corpus,
  - Zero counts,
  - What to condition on.