CS4442/9542b
Artificial Intelligence II
prof. Olga Veksler

Lecture 11
Natural Language Processing
Part of Speech Tagging

Many slides from: Joshua Goodman, L. Kosseim, D. Klein, D. Jurafsky, M. Hearst, K. McCoy, Y. Halevi, C. Manning, M. Poesio
Outline

• What is POS and POS tagging
  • POS = part of speech
• Why we need POS tagging
• Different Approaches to POS
  1. rule-based tagging
  2. statistical tagging
What is a Part of Speech?

• Words that behave alike
  • appear in similar contexts
  • perform similar functions in sentences
  • undergo similar transformations

• Terminology
  • **POS** (part-of-speech tag)
  • also called
    • grammatical tag
    • grammatical category
    • syntactic word class
Substitution Test

• Two words belong to the same part of speech if replacing one with another does not change the grammaticality of a sentence

\[
\text{The \{sad, big, green, \ldots\} dog is barking.}
\]
• Perhaps started with Aristotle (384–322 BCE)
• From Dionysius Thrax of Alexandria (c. 100 BCE) the idea that is still with us
  • 8 main parts of speech
• Those 8 are not exactly the ones taught today
  • Thrax: noun, verb, article, adverb, preposition, conjunction, participle, pronoun
  • School grammar: noun, verb, adjective, adverb, preposition, conjunction, pronoun, interjection
How Many POS are there?

- A basic set:
  - N(oun), V(erb), Adj(ective), Adv(erb), Prep(osition), Det(erminer), Aux(iliaries), Part(icle), Conj(unction)

- A simple division: open/content vs. closed/function
  - Open: N, V, Adj, Adv
    - new members are added frequently
  - Closed: Prep, Det, Aux, Part, Conj, Num
    - new members are added rarely

- Many subclasses, e.g.
  - eats/V ⇒ eat/VB, eat/VBP, eats/VBZ, ate/VBD, eaten/VBN, eating/VBG, ...
• Goal: assign POS tag (noun, verb, ...) to text

*The/AT girl/NN put/VBD chairs/NNS on/IN the/AT table/NN.*

• What set of parts of speech do we use?
  • various standard tagsets to choose from, some have a lot more tags than others
  • choice of tagset is based on application
  • accurate tagging possible with even large tagsets
Why do POS Tagging?

- Word sense disambiguation (semantics)
  - limits the range of meanings: *deal* as noun vs. *deal* as verb
- Speech recognition and synthesis
  - how to recognize/pronounce a word:
  - *content/noun* vs. *content/adj*
- Stemming: which morphological affixes word can take
  - adverb - *ly* = noun: *friendly* - *ly* = friend
  - cannot apply to adjectives, example: *sly*
- Partial parsing/chunking
  - to find noun phrases/verb phrases
- Information extraction
  - helps identify useful terms and relationships between them
Common Tagged Datasets

• 45 tags in Penn Treebank
• 62 tags in CLAWS with BNC corpus
• 79 tags in Church (1991)
• 87 tags in Brown corpus
• 147 tags in C7 tagset
• 258 tags in Tzoukermann and Radev (1995)
Penn Treebank

• First syntactically annotated corpus
• 1 million words from Wall Street Journal
• Part of speech tags and syntax trees
**Important Penn Treebank tags**

- **IN**  preposition or subordinating conjunct.
- **JJ**  adjective or numeral, ordinal
- **JJR** adjective, comparative
- **NN**  noun, common, singular or mass
- **NNP** noun, proper, singular
- **NNS** noun, common, plural
- **TO**  "to" as preposition or infinitive marker
- **VB**  verb, base form
- **VBD** verb, past tense
- **VBG** verb, present participle or gerund
- **VBN** verb, past participle
- **VBP** verb, present tense, not 3rd p. singular
- **VBZ** verb, present tense, 3rd p. singular

...
Verb inflection tags

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>VBP</td>
<td>base present</td>
<td><em>take</em></td>
</tr>
<tr>
<td>VB</td>
<td>infinitive</td>
<td><em>take</em></td>
</tr>
<tr>
<td>VBD</td>
<td>past</td>
<td><em>took</em></td>
</tr>
<tr>
<td>VBG</td>
<td>present participle</td>
<td><em>taking</em></td>
</tr>
<tr>
<td>VBN</td>
<td>past participle</td>
<td><em>taken</em></td>
</tr>
<tr>
<td>VBZ</td>
<td>present 3sg</td>
<td><em>takes</em></td>
</tr>
<tr>
<td>MD</td>
<td>modal</td>
<td><em>can, would</em></td>
</tr>
</tbody>
</table>
The entire Penn Treebank tagset

<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>WPS</td>
<td>Possessive wh-</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>(‘ or “)</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>
The cat decided to jump on the couch to play with another cat

• Word type
  • Distinct words in the text (vocabulary)
  • text above has 10 word types
    • the, cat, decided, to, jump, on, couch, play, with, another

• Word token
  • any word occurring in the text
  • text above has 13 word tokens
Distribution of Tags

• POS follow typical frequency-based behavior
  • most word types have only one part of speech
  • of the rest, most have two
  • only a small number of word types have lots of parts of speech
    • but these occur with high frequency
Most Word Types not Ambiguous but

<table>
<thead>
<tr>
<th>num. word types</th>
<th>Unambiguous (1 tag) 35 340</th>
<th>Ambiguous (&gt;1 tag) 4 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 tags</td>
<td>3760</td>
<td></td>
</tr>
<tr>
<td>3 tags</td>
<td>264</td>
<td></td>
</tr>
<tr>
<td>4 tags</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>5 tags</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>6 tags</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>7 tags</td>
<td>1 “still”</td>
<td></td>
</tr>
</tbody>
</table>

- but most word types are rare
- Brown corpus (Francis & Kucera, 1982):
  - 11.5% **word types** are ambiguous (>1 tag)
  - 40% **word tokens** are ambiguous (>1 tag)
1. Book/VB that/DT flight/NN
   • book can also be NN
   • Can I read a book on this flight?

2. Does/VBZ that/DT flight/NN serve/VB dinner/NN?
   • that can also be a complementizer
   • My travel agent said that there would be a meal on this flight.
Potential Sources of Disambiguation

1. Lexical information:
   • look up all possible POS for a word in a dictionary
   • “table”: {noun, verb} but not a {adj, prep,...}
   • “rose”: {noun, adj, verb} but not {prep, ...}

2. Syntagmatic information:
   • some tag sequences are more probable than others:
     • DET + N occur frequently but DET+V never occurs
     • ART+ADJ+N is more probable than ART+ADJ+VB
   • Can find the syntagmatic information
     • by talking to the experts
     • or, better, from training corpora
For a is a sequence of tags $t_1, t_2, \ldots, t_k$ compute

$$P(t_1, t_2, \ldots, t_k)$$

tells us how likely this tag sequence is

similar to computing probability of a sequence of words $P(w)$

make the same approximation as before

$$P(t_n \mid t_1, t_2, \ldots, t_{n-1}) = P(t_n \mid t_{n-k} \ldots t_{n-1})$$

for computational efficiency, our assumption is

$$P(t_n \mid t_1, t_2, \ldots, t_{n-1}) = P(t_n \mid t_{n-1})$$
1. rule-based tagging
   • uses hand-written rules

2. statistical tagging
   • uses probabilities computed from training corpus
     • Charniak
     • Markov Model based
Rule-based POS Tagging

- Step 1: assign each word with all possible tags
  - use dictionary

- Step 2: use if-then rules to identify the correct tag in context (disambiguation rules)
Rule-based POS Tagging: Sample rules

• **ART-V rule:**
  tag ART (article) cannot be followed by a tag V (verb)
  
  *...the book...*

  • the: {ART}
  • book: {N, V} --> {N}

• **N-IP rule:**
  tag N (noun) cannot be followed by tag IP (interrogative pronoun)
  
  *... man who ...*

  • man: {N}
  • who: {RP, IP} --> {RP} relative pronoun
Rule-based Tagger

• using only syntagmatic patterns
  • Green & Rubin (1971)
  • accuracy of 77%

• In addition
  • very time consuming to come up with the rules
  • need an expert in English to come up with the rules
Statistical POS Tagger: Charniak 1993

• Simplest statistical tagger
• From corpus, calculate most probable tag for each word that is the one maximizing
  \[
  \frac{\text{count(word has tag } t\text{)}}{\text{count(word)}}
  \]
• Equivalent to maximizing
  \[
  \text{count(word has tag } t\text{)}
  \]
• Charniak tagger assigns most probable POS tag to a word
• Given a word to tag,
  1. for each possible tag \( t \) for this word, compute
     \[
     \text{count(word has tag } t\text{)}
     \]
  2. choose tag \( t \) that maximizes the above
Statistical POS Tagger: Charniak 1993

- Accuracy of 90%
  - contrast with 77% accuracy of the rule-based tagger!
  - evidence of power of statistical over rule-based methods
  - MUCH better than rule based, but not very good...
    - 1 mistake every 10 words
  - funny fact: every word will have only one POS assigned to it
    - book will always be assigned the noun tag
- This tagger is used mostly as baseline for evaluation
- How do we improve it?
  - take the context of the surrounding words into account
  - some sequence of tags are much more likely than others
Statistical Tagger: Markov Model Based

- Tag sentence of words \( w_{1,n} = w_1 \ w_2 \ \ldots \ \ w_n \)
- Denote tag sequence as \( t_{1,n} = t_1 \ t_2 \ \ldots \ \ t_n \)
  - \( t_i \) is a tag for word \( w_i \)
- Find the best tagging \( t_{1,n} \) out of all possible taggings
- Two sources of information
  1. \( P(t_{i+1} | t_i) \): how likely is tag \( t_{i+1} \) after tag \( t_i \)
  2. \( P(w_i | t_i) \): if tag is \( t_i \) how likely word is \( w_i \)
    - Example: \( P(\text{book} | \text{verb}) > P(\text{book} | \text{noun}) \)
      - there are many more nouns than verbs
      - say 1,000 verbs and 10,000 nouns
Markov Model Tagger

- The best tagging is the one that maximizes
  \[ P(t_{1,n} \mid w_{1,n}) \]
- Hard to estimate directly
- Using Bayes law
  \[ P(t_{1,n} \mid w_{1,n}) = \frac{P(w_{1,n} \mid t_{1,n})P(t_{1,n})}{P(w_{1,n})} \]
- Bottom does not effect maximization,
  - constant over all possible taggings \( t_{1,n} \)
- Find tagging that maximizes
  \[ P(w_{1,n} \mid t_{1,n})P(t_{1,n}) \]
We will make two simplifying assumptions

First simplifying assumption:

1. given its tag, probability of word is independent of tags of other words in a sentence:

   \[
   P\left(w_{1,n} \mid t_{1,n}\right) \prod_{i=1}^{n} P\left(w_i \mid t_i\right)
   \]

   - \( P( \text{book} \mid \text{verb} ) \) is independent of what are the tags of other words in the sentence
   - Reasonable assumption. For example, if the next tag is \textit{adverb}, does not change much about \( P( \text{book} \mid \text{verb} ) \)
Markov Model Tagger: First Assumption

\[ P(w_{1,n} \mid t_{1,n}) = \prod_{i=1}^{n} P(w_i \mid t_i) = P(w_1 \mid t_1) P(w_2 \mid t_2) ... P(w_n \mid t_n) \]

- \( P(w_i \mid t_k) \) estimated from tagged corpus:

\[
\frac{C(w_i \text{ has tag } t_k)}{C(t_k)}
\]

- i.e. \( P(\text{book} \mid \text{verb}) \) is count of how many times \text{book} has tag \text{verb} divided by how many times tag \text{verb} occurs in corpus
Markov Model Tagger: Second Assumption

2. Each tag depends only on one previous tag:

\[
P(t_{1,n}) = \prod_{i=1}^{n} P(t_i \mid t_{i-1}) = P(t_1 \mid t_0) P(t_2 \mid t_1) \ldots P(t_n \mid t_{n-1})
\]

- this is **Markov** assumption we saw in language modeling
- estimate as in language modeling:

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

- \(P(t_1 \mid t_0)\) stands for \(P(t_1)\), estimated by

\[
P(t_1) = \frac{C(t_1)}{N}
\]
Markov Model Tagger

• Using these 2 assumptions, find tagging that maximizes
\[
\prod_{i=1}^{n} P(w_i | t_i)P(t_i | t_{i-1})
\]  (1)

• Naïve algorithm: given sentence \(w_1,n\) go over all possible tag assignments \(t_1,n\) and compute (1)

• Choose final tagging \(t_1,n\) which maximizes (1)
  • efficiency: for each word try only tags given by the dictionary
  • example: for fly, possible tags are noun, verb and also adjective (meaning keen or artful, mainly in England)
Markov Model Tagger

- Naïve algorithm: given sentence $w_{1,n}$ go over all possible tag assignments $t_{1,n}$
- 40% words have more than 1 tag
- too many tag assignments to try
- if 2 tags per word, then $2^n$ possible assignments
- exhaustive search is exponential
Markov Model Tagger

• Side note: Markov tagger becomes Charniak’s tagger if tags are assumed independent, i.e.

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

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

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

\[
= \prod_{i=1}^{n} P(w_i, t_i)
\]
Markov Model Tagger

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

word 1 \quad word 2 \quad word 3 \quad \ldots \quad word n

ADJ \quad ADJ \quad PREP

NOUN \quad NOUN \quad NOUN

VERB

PREP

NOUN

VERB

DETER
Markov Model Tagger: DP

- Use DP (dynamic programming) to significantly speed up
  - also called Viterbi algorithm
- If \( k \) tags per word and \( n \) words, can find best tagging in \( O(k^2n) \)
- To avoid floating point underflows, take logarithms

\[
\log \left[ \prod_{i=1}^{n} P(w_i | t_i)P(t_i | t_{i-1}) \right] = \sum_{i=1}^{n} \left( \log P(w_i | t_i) + \log P(t_i | t_{i-1}) \right)
\]

- how likely word \( w_i \) is for tag \( t_i \)
- how likely tag \( t_i \) to follow tag \( t_{i-1} \)
Markov Model Tagger: DP

• Turn maximizing:

\[
\sum_{i=1}^{n} \log P(w_i \mid t_i) + \sum_{i=1}^{n} \log P(t_i \mid t_{i-1})
\]

• Into equivalent minimizing

\[
-\sum_{i=1}^{n} \log P(w_i \mid t_i) - \sum_{i=1}^{n} \log P(t_i \mid t_{i-1})
\]
Markov Model Tagger: DP

• Find a sequence of tags $t_1, t_2, \ldots, t_n$ to minimize

$$ \sum_{i=1}^{n} - \log P(w_i | t_i) + \sum_{i=1}^{n} - \log P(t_i | t_{i-1}) $$

• In the new notation, find tags $t_1, t_2, \ldots, t_n$ to minimize:

$$ \sum_{i=1}^{n} L(w_i | t_i) + \sum_{i=1}^{n} L(t_i | t_{i-1}) $$
Markov Model Tagger: DP

\[ \sum_{i=1}^{n} L(w_i | t_i) + \sum_{i=1}^{n} L(t_i | t_{i-1}) \]
• Change notation just for the first word:

\[
L(w_1 | t_1) = -\log[P(w_1 | t_1)] - \log[P(t_1 | t_0)]
\]

\[
\sum_{i=1}^{n} L(w_i | t_i) + \sum_{i=1}^{n} L(t_i | t_{i-1}) \Rightarrow \sum_{i=1}^{n} L(w_i | t_i) + \sum_{i=2}^{n} L(t_i | t_{i-1})
\]
• Each node has cost $L(w_i | t_i)$
• Each edge has cost $L(t_i | t_{i-1})$

Cost of a path: $\sum_{i=1}^{n} L(w_i | t_i) + \sum_{i=2}^{n} L(t_i | t_{i-1})$
• Find minimum cost path that starts at some node corresponding to word 1 and ends at some node corresponding to word n
Markov Model Tagger: Main Step of DP

- Main Step: for every node at word $w_i$, find smallest cost path that leads into it, starting at any node at word $w_1$
• First compute the best path that ends at any node for \( w_1 \)
• Then compute the best path that ends at any node for \( w_2 \)
• …..
• Finally compute the best path that ends at any node for \( w_n \)
• The best path overall is smallest cost path that end at \( w_n \)

Compute the best path that ends here and here. Cheapest of these two is the final answer
• For word \( w_i \) tag \( t \) node is \((w_i, t)\)

• \( C(w_i, t) \) cost of best path that starts at any \((w_1, t)\) and ends at \((w_i, t)\)

• \( P(w_i, t) \) is parent of node \((w_i, t)\) on this path

• After all \( C(w_i, t) \) computed, min of \( C(w_n, t) \) over all \( t \) gives best path
• First compute the best path that ends at any node for \( w_1 \)
  • trivial, since the path has just one node
• For all tags of the first word \( t \):

\[
C(w_1, t) = L(w_1 | t) \\
P(w_1, t) = \text{null}
\]
Markov Model Tagger: DP Iteration

- Computed $C(w_i, t)$ and $P(w_i, t)$ for all tags $t$ and $i < k$

```
word 1  · · ·  word k-1  word k  · · ·  word n
ADJ     · · ·  ADJ  NOUN  · · ·  NOUN
NOUN    · · ·  NOUN PREP  · · ·  VERB
VERB    · · ·  ADV
```

all the best paths are computed
Markov Model Tagger: DP Iteration

- Now compute $C(w_k, t)$ and $P(w_k, t)$ for $k$
- Consider node $(w_k, \text{ADJ})$

The best path from $w_1$ to $(w_k, \text{ADJ})$ goes through either
1. $(w_{k-1}, \text{ADJ})$: then it follows best path from $w_1$ to $(w_{k-1}, \text{ADJ})$
2. $(w_{k-1}, \text{NOUN})$: then it follows best path from $w_1$ to $(w_{k-1}, \text{NOUN})$

- because a sub-path of the best path is a best path itself
• \( C(w_k, \text{ADJ}) \) is the smaller of two quantities:

1. \( C(w_{k-1}, \text{ADJ}) + L(\text{ADJ} | \text{ADJ}) + L(w_k | \text{ADJ}) \)
   
   • then \( P(w_k, \text{ADJ}) = (w_{k-1}, \text{ADJ}) \)

2. \( C(w_{k-1}, \text{NOUN}) + L(\text{ADJ} | \text{NOUN}) + L(w_k | \text{ADJ}) \)
   
   • then \( P(w_k, \text{ADJ}) = (w_{k-1}, \text{NOUN}) \)
Markov Model Tagger: DP Iteration

- In general, $C(w_k, t)$ is computed as follows:

\[
C(w_k, t) = \min_{t' \in T(w_{k-1})} \left\{ C(w_{k-1}, t') + L(t | t') \right\} + L(w_k | t)
\]

- $P(w_k, t) = (w_{k-1}, t^*)$ where $t^*$ is the tag for word $w_{k-1}$ minimizing the expression above.
Markov Model Tagger: DP Termination

• After computed all $C(w_i, t)$ best cost path is found as the minimum of $C(w_n, t)$ over all tags $t$
• Parents on the path traced back using $P(w_i, t)$

<table>
<thead>
<tr>
<th>word 1</th>
<th>word 2</th>
<th>...</th>
<th>word n-1</th>
<th>word n</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>ADJ</td>
<td></td>
<td>ADJ</td>
<td>PREP</td>
</tr>
<tr>
<td>NOUN</td>
<td>NOUN</td>
<td></td>
<td>NOUN</td>
<td>VERB</td>
</tr>
<tr>
<td>VERB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$C(w_2, \text{NOUN})$ is smallest, $P(w_2, \text{NOUN}) = (w_1, \text{VERB})$

$C(w_n, \text{VERB})$ is smallest, $P(w_n, \text{VERB}) = (w_{n-1}, \text{ADJ})$

• Final tagging is: VERB   NOUN   ...   ADJ   VERB
**MMT Example**

\[
\begin{align*}
L(\text{book}|\text{ADJ}) &= 10 \\
L(\text{book}|\text{VERB}) &= 1 \\
L(\text{book}|\text{NOUN}) &= 2 \\
L(\text{that}|\text{PRON}) &= 2 \\
L(\text{that}|\text{CONJ}) &= 4 \\
L(\text{flight}|\text{NOUN}) &= 2 \\
L(\text{flight}|\text{VERB}) &= 1
\end{align*}
\]

- **book**
  - ADJ: \(L(\text{PRON}|\text{VERB}) = 3\)
  - VERB: \(L(\text{CONJ}|\text{VERB}) = 4\)
  - NOUN: \(L(\text{PRON}|\text{NOUN}) = 2\), \(L(\text{CONJ}|\text{NOUN}) = 1\)
  - PRON: \(L(\text{PRON}|\text{ADJ}) = 1\)
  - CONJ: \(L(\text{CONJ}|\text{ADJ}) = 2\)

- **that**
  - PRON: \(L(\text{NOUN}|\text{PRON}) = 1\)
  - VERB: \(L(\text{VERB}|\text{PRON}) = 10\)
  - CONJ: \(L(\text{VERB}|\text{CONJ}) = 2\)
  - NOUN: \(L(\text{NOUN}|\text{CONJ}) = 4\)

- **flight**
  - NOUN: \(L(\text{VERB}|\text{NOUN}) = 2\)
  - VERB: \(L(\text{VERB}|\text{VERB}) = 1\)
L( book|ADJ ) = 10
L( book|VERB ) = 1
L( book|NOUN ) = 2

• **Iteration 1:**
  - C(book,ADJ) = 10, P(book,ADJ) = null
  - C(book,VERB) = 1, P(book,VERB) = null
  - C(book,NOUN) = 2, P(book,NOUN) = null
MMT Example

L(PRON|ADJ) = 1
L(PRON|VERB) = 3
L(PRON|NOUN) = 2
L( that|PRON ) = 2
L( that|CONJ ) = 4

• Iteration 2:
  • C( that, PRON ) = 6, P( that, PRON ) = (book, VERB)
• Iteration 2:
  • $C(\text{that}, \text{CONJ}) = 8$, $P(\text{that}, \text{CONJ}) = (\text{book}, \text{NOUN})$
• Iteration 3:
  • \( C(\text{flight}, \text{NOUN}) = 9, \ P(\text{flight}, \text{NOUN}) = (\text{that}, \text{PRON}) \)
• Iteration 3:
  • $C(fl\text{ight}, \text{VERB}) = 11$, $P(fl\text{ight}, \text{VERB}) = (\text{that}, \text{CONJ})$

L(VERB|PRON) = 10
L(VERB|CONJ) = 2
L( flight|NOUN ) = 2
L( flight|VERB ) = 1

C( book, \text{ADJ} ) = 10, \quad P( book, \text{ADJ} ) = \text{null}
C( book, \text{VERB} ) = 1, \quad P( book, \text{VERB} ) = \text{null}
C( book, \text{NOUN} ) = 2, \quad P( book, \text{NOUN} ) = \text{null}
C( that, \text{PRON} ) = 6, \quad P( that, \text{PRON} ) = ( book, \text{VERB} )
C( that, \text{CONJ} ) = 8, \quad P( that, \text{CONJ} ) = ( book, \text{NOUN} )
Final Tagging: Book<verb> that <pron> flight<noun>
for each $t \in \text{Tags}(w_1)$ do

$C(w_1, t) = L(w_1 | t), \ P(w_1, t) = \text{null}$

for $i \leftarrow 2$ to $n$ do

for each $t \in \text{Tag}(w_i)$ do

$C(w_i, t) = -\infty$

for each $t' \in \text{Tag}(w_{i-1})$ do

nextCost = $C(w_{i-1}, t') + L(t | t') + L(w_i | t)$

if nextCost < $\text{cost}(w_i, t)$ do

$C(w_i, t) = \text{nextCost}$

$P(w_i, t) = t'$
Unknown Words

• Simplest method: assume an unknown word could belong to any tag; unknown words are assigned the distribution over POS over the whole lexicon
  
  - \( P(\text{“karumbula”} | \text{verb}) = P(\text{“karumbula”} | \text{noun}) = \)
  
  - \( P(\text{“karumbula”} | \text{adjective}) = \ldots \) etc

• Some tags are more common than others
  
  - for example a new word can be most likely a verb, a noun etc. but not a preposition or an article

• Use morphological and other cues
  
  - for example words ending in \(-ed\) are likely to be past tense forms or past participles
Tagging Accuracy

- Ranges from 96%-97%
- Depends on:
  - Amount of training data available
  - The tag set
  - Difference between training corpus and dictionary and the corpus of application
  - Unknown words in the corpus of application
- A change in any of these factors can have a dramatic effect on tagging accuracy – often much more stronger than the choice of tagging method