How to Do
Part-Of-Speech (POS) Tagging

Why is Part-Of-Speech Tagging Hard?

- Words may be ambiguous in different ways:
  - A word may have multiple meanings as the same part-of-speech
    - *file* – **noun**, a folder for storing papers
    - *file* – **noun**, instrument for smoothing rough edges
  - A word may function as multiple parts-of-speech
    - a *round* table: **adjective**
    - a *round* of applause: **noun**
    - to *round* out your interests: **verb**
    - to work the year *round*: **adverb**

Why is Part-Of-Speech Tagging Needed?

- May be useful to know what function the word plays, instead of depending on the word itself.
- Internally, next higher levels of NL Processing:
  - Phrase Bracketing
    - Can write regexps like (Det) Adj* N+ over the output for phrases, etc.
  - Parsing
    - As input to or to speed up a full parser
    - If you know the tag, you can back off to it in other tasks
  - Semantics
- Applications that use POS tagging:
  - Speech synthesis - Text-to-speech (how do we pronounce “lead”?)
  - Information retrieval — selection of high-content words
  - Word-sense disambiguation
  - Sentiment detection — selection of high-opinion or emotion words

Overview of Approaches

- Rule-based Approach
  - Simple and doesn’t require a tagged corpus, but not as accurate as other approaches
- Stochastic Approach
  - Refers to any approach which incorporates frequencies or probabilities
  - Requires a tagged corpus to learn frequencies
  - N-gram taggers taggers
  - Hidden Markov Model (HMM) taggers
- Other Issues: unknown words and evaluation

Word Class Ambiguity (in the Brown Corpus)

- Recall that words often have more than one word class: another example is the word *this*
  - *This* is a nice day = PRP
  - *This* day is nice = DT
  - You can go *this* far = RB

- Degree of ambiguity in English
  - 40% of word tokens are ambiguous.
  - 11.5% of word types are ambiguous.
    - Unambiguous (1 tag): 35,340
    - Ambiguous (2-7 tags): 4,100

- *the word "still" has 7 tags*

2 tags	3,760
3 tags	264
4 tags	61
5 tags	12
6 tags	2
7 tags	1

(Derose, 1988)

N-gram Approach

- N-gram approach to probabilistic POS tagging:
  - calculates the probability of a given sequence of tags occurring for a sequence of words
  - the best tag for a given word is determined by the (already calculated) probability that it occurs with the n previous tags
  - may be bi-gram, tri-gram, etc

word_{n-1}	…	word_{-2}	word_{-1}	word
tag_{n-1}	…	tag_{-2}	tag_{-1}	??

- Presented here as an introduction to HMM tagging
  - And given in more detail in the NLTK
  - In practice, bigram and trigram probabilities have the problem that the combinations of words are sparse in the corpus
  - Combine the taggers with a backoff approach

N-gram Tagging

- Initialize a tagger by learning probabilities from a tagged corpus

  $word_{n-1}$ … $word_{-2}$ $word_{-1}$ word

  tag_{n-1} … tag_{-2} tag_{-1} XX

  – Probability that the sequence … tag_{-2} tag_{-1} word gives tag XX
  – Note that initial sequences will include a start marker as part of the sequence
- Use the tagger to tag word sequences (usually of length 2-3) with unknown tags
  – Sequence through the words:
    - To determine the POS tag for the next word, use the previous n-1 tags and the word to look up probabilities and use the highest probability tag

Need Longer Sequence Classification

- A more comprehensive approach to tagging considers the entire sequence of words
  - *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.

Thanks to Jim Martin's online class slides for the examples and equation typesetting in this section on HMM's.

Road to 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.
  - i.e. the probability of the tag sequence $t_1 \ldots t_n$ given the word sequence $w_1 \ldots w_n$

*

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

- Hat ^ means "our estimate of the best one"
- Argmax_x f(x) means "the x such that f(x) is maximized"
  - i.e. find the tag sequence that maximizes the probability

Road to HMMs

- This equation is guaranteed to give us the best tag sequence

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n | w_1^n)$$

- 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

Thomas Bayes 1701 - 1761

Using Bayes Rule

- Bayes rule:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

- Apply Bayes Rule:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$$

- Note that this is using the conditional probability, given a tag sequence, what is the most likely word sequence with those tags.
  - Eliminate denominator as it is the same for every sequence

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n|t_1^n)P(t_1^n)$$

Likelihood and Prior

- Further simplify

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \overbrace{P(w_1^n|t_1^n)}^{\text{likelihood}} \overbrace{P(t_1^n)}^{\text{prior}}$$

- Likelihood: assume that the probability of the word depends only on its tag

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

- Prior: use the bigram assumption that the tag only depends on the previous tag

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

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

Two Sets of Probabilities (1)

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

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

Count of DT NN sequence

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

Two Sets of Probabilities (2)

- 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:

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

Count of is tagged with VBZ

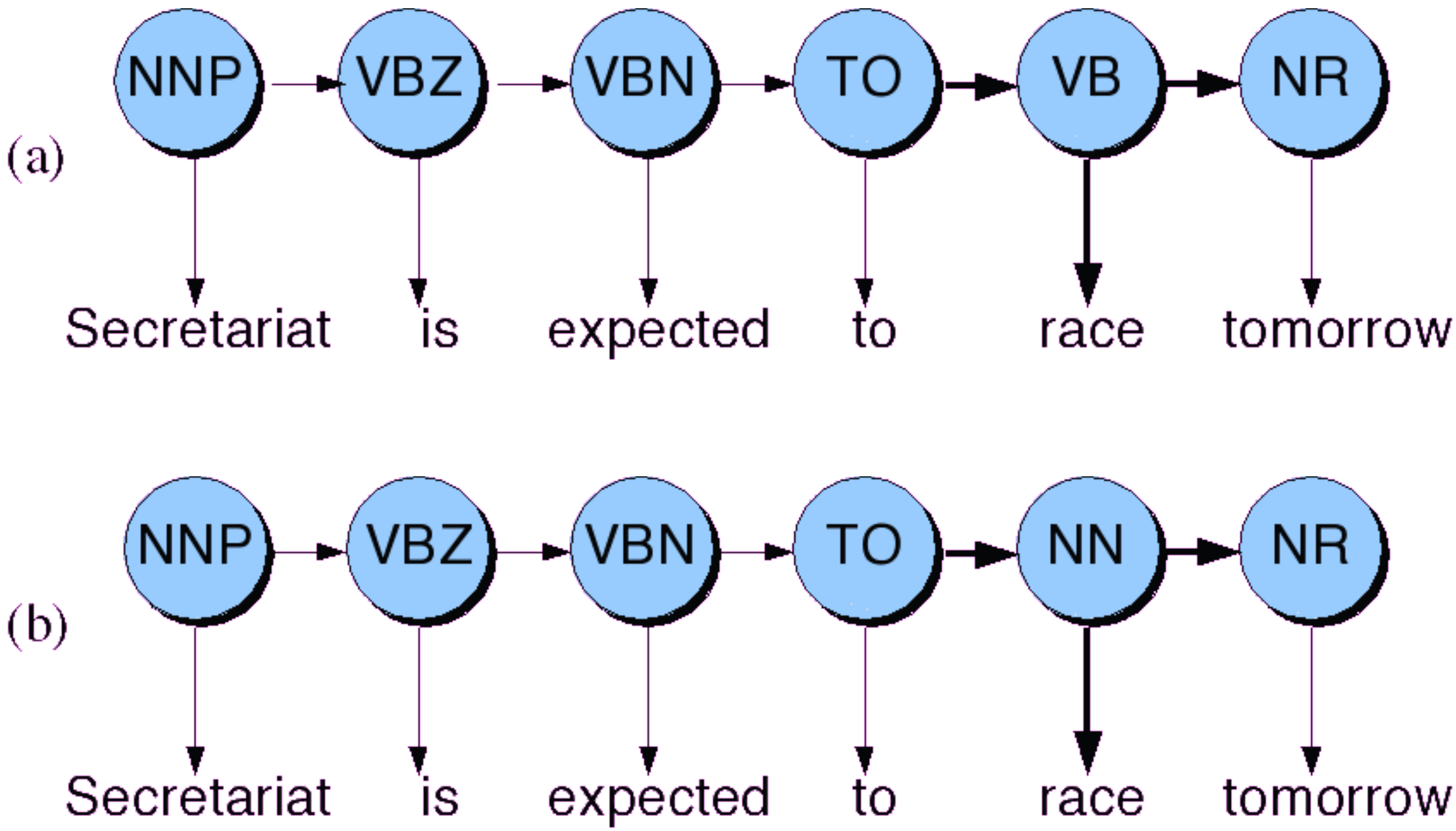
$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

An Example: the word “race”

- The word “race” can occur as a verb or as a noun:
  - 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"

Which tag sequence is most likely?



Example

- *The equations only differ in "to race tomorrow"*
- $P(NN|TO) = .00047$
- $P(VB|TO) = .83$
- $P(race|NN) = .00057$
- $P(race|VB) = .00012$
- $P(NR|VB) = .0027$
- $P(NR|NN) = .0012$

The tag transition probabilities P(NN|TO) and P(VB|TO)

Lexical likelihoods from the Brown corpus for 'race' given a POS tag NN or VB.

Tag sequence probability for the likelihood of an adverb occurring given the previous tag verb or noun

- $P(VB|TO)P(NR|VB)P(race|VB) = .00000027$
- $P(NN|TO)P(NR|NN)P(race|NN)=.00000000032$
- *So we (correctly) choose the verb tag.*