**Last Class:**
1. Intro to part-of-speech tagging
2. TBL approach to p-o-s tagging

**Today:**
1. Hidden Markov Model Tagger

---

**HMM Tagger**

Given $W = w_1, \ldots, w_n$, find $T = t_1, \ldots, t_n$ that maximizes

$$P(t_1, \ldots, t_n | w_1, \ldots, w_n)$$

Restate using Bayes’ rule:

$$\frac{(P(t_1, \ldots, t_n) \ast P(w_1, \ldots, w_n | t_1, \ldots, t_n))}{P(w_1, \ldots, w_n)}$$

Ignore denominator...
Make independence assumptions...

---

**Independence Assumptions (factor 1)**

$P(t_1, \ldots, t_n)$: approximate using **n-gram model**

- **bigram** $\prod_{i=1}^{n} P(t_i | t_{i-1})$
- **trigram** $\prod_{i=1}^{n} P(t_i | t_{i-2}t_{i-1})$

---

**Independence Assumptions (factor 2)**

$P(w_1, \ldots, w_n | t_1, \ldots, t_n)$: approximate by assuming that a word appears in a category independent of its neighbors

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

Assuming bigram model:

$$P(t_1, \ldots, t_n) \ast P(w_1, \ldots, w_n | t_1, \ldots, t_n) \approx \prod_{i=1}^{n} P(t_i | t_{i-1}) \ast P(w_i | t_i)$$
Hidden Markov Models

Equation can be modeled by an HMM.

- **states**: represent a possible lexical category
- **transition probabilities**: bigram probabilities
- **observation probabilities, lexical generation probabilities**: indicate, for each word, how likely that word is to be selected if we randomly select the category associated with the node.

Viterbi Algorithm

c: number of lexical categories

\[ P(w_t|t_i): \text{lexical generation probabilities} \]

\[ P(t_i|t_j): \text{bigram probabilities} \]

Find most likely sequence of lexical categories \( T_1, \ldots, T_n \) for word sequence.

**Initialization**

For \( i = 1 \) to \( c \)

\[ \text{SCORE}(i,1) = P(t_i|\phi) \ast P(w_1|t_i) \]

\[ \text{BPTR}(i,1) = 0 \]

**Iteration**

For \( t = 2 \) to \( n \)

For \( i = 1 \) to \( c \)

\[ \text{SCORE}(i,t) = \max_{j=1..c} \left( \text{SCORE}(j, t-1) \ast P(t_i|t_j) \right) \ast P(w_t|t_i) \]

\[ \text{BPTR}(i,t) = \text{index of } j \text{ that gave max} \]

**Identify Sequence**

\( T(n) = i \) that maximizes \( \text{SCORE}(i,n) \)

For \( i = n-1 \) to \( 1 \)

\( T(i) = \text{BPTR}( T(i+1), i+1 ) \)

Results

- Effective if probability estimates are computed from a large corpus
- Effective if corpus is of the same style as the input to be classified
- Consistently achieve accuracies of 96% or better using trigram model
- Cuts error rate in half vs. naive algorithm (90% accuracy rate)
- Can be smoothed using backoff or deleted interpolation...
Extensions

- Can train HMM tagger on unlabeled data using the EM algorithm, starting with a dictionary that lists which tags can be assigned to which words.
- EM then learns the word likelihood function for each tag, and the tag transition probabilities.
- Merialdo (1994) showed, however, that a tagger trained on even a small amount hand-tagged data works better than one trained via EM.