HMM p-o-s 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:

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

Ignore denominator...

Make independence assumptions...
Independence Assumptions (factor 1)

\(P(t_1, \ldots, t_n):\) approximate using \textbf{n-gram model}

\textbf{bigram} \( \prod_{i=1,n} P(t_i | t_{i-1}) \)

\textbf{trigram} \( \prod_{i=1,n} P(t_i | t_{i-2}t_{i-1}) \)
Independence Assumptions (factor 2)

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

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

Assuming bigram model:

\[
P(t_1, \ldots, t_n) \ast P(w_1, \ldots, w_n \mid t_1, \ldots, t_n) \approx
\prod_{i=1,n} P(t_i \mid t_{i-1}) \ast P(w_i \mid t_i)
\]
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

$P(w_t|t_i)$: lexical generation probabilities

$P(t_i|t_j)$: bigram probabilities

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

**Initialization**

For $i = 1$ to $c$ do

$\text{SCORE}(i,1) = P(t_i|\phi) \times P(w_1|t_i)$

$\text{BPTR}(i,1) = 0$
Iteration

For $t = 2$ to $n$

For $i = 1$ to $c$

$SCORE(i,t) =$

$$MAX_{j=1..c}(SCORE(j, t-1) \times P(t_i | t_j)) \times P(w_t | t_i)$$

$BPTR(i,t) =$ index of $j$ that gave max

Identify Sequence

$T(n) =$ $i$ that maximizes $SCORE(i,n)$

For $i = n-1$ to $1$ do

$T(i) =$ $BPTR( T(i+1), i+1 )$
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 97% or better using trigram model
- Cuts error rate in half vs. naive algorithm (90% accuracy rate)
- Can be smoothed using backoff or interpolation or discounting...
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.