Questions for Flipped Classroom Session of COMS 4705
Week 3, Fall 2014. (Michael Collins)

Question 1  Consider a trigram HMM tagger with:

- The set $\mathcal{K}$ of possible tags equal to \{D, N, V\}
- The set $\mathcal{V}$ of possible words equal to \{the, dog, barks\}
- The following parameters:

\[
\begin{align*}
q(D|*, *) &= 1 \\
q(N|*, D) &= 1 \\
q(V|D, N) &= 1 \\
q(\text{STOP}|N, V) &= 1 \\
e(\text{the}|D) &= 1 \\
e(\text{dog}|N) &= 0.4 \\
e(\text{barks}|N) &= 0.6 \\
e(\text{dog}|V) &= 0.1 \\
e(\text{barks}|V) &= 0.9
\end{align*}
\]

with all other parameter values equal to 0.

**Question:** Write down the set of all pairs of sequences $x_1 \ldots x_n, y_1 \ldots y_{n+1}$ such that the following properties hold:

- $p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) > 0$
- $x_i \in \mathcal{V}$ for all $i \in 1 \ldots n$
- $y_i \in \mathcal{K}$ for all $i \in 1 \ldots n$, and $y_{n+1} = \text{STOP}$
**Input:** a sentence $x_1 \ldots x_n$, parameters $q(s|u, v)$ and $e(x|s)$.

**Definitions:** Define $\mathcal{K}$ to be the set of possible tags. Define $\mathcal{K}_{-1} = \mathcal{K}_0 = \{^*\}$, and $\mathcal{K}_k = \mathcal{K}$ for $k = 1 \ldots n$.

**Initialization:** Set $\pi(0, ^*, ^*) = 1$.

**Algorithm:**

- For $k = 1 \ldots n$,
  - For $u \in \mathcal{K}_{k-1}$, $v \in \mathcal{K}_k$,
    $$\pi(k, u, v) = \max_{w \in \mathcal{K}_{k-2}} (\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v))$$
- Return $\max_{u \in \mathcal{K}_{n-1}, v \in \mathcal{K}_n} (\pi(n, u, v) \times q(\text{STOP}|u, v))$

Figure 1: The basic Viterbi Algorithm.

**Question 2** Consider a trigram HMM, as introduced in class. We saw that the Viterbi algorithm could be used to find

$$\max_{y_1 \ldots y_{n+1}} p(x_1 \ldots x_n, y_1 \ldots y_{n+1})$$

where the max is taken over all sequences $y_1 \ldots y_{n+1}$ such that $y_i \in \mathcal{K}$ for $i = 1 \ldots n$, and $y_{n+1} = \text{STOP}$. (Recall that $\mathcal{K}$ is the set of possible tags in the HMM.)

In a trigram tagger we assume that $p$ takes the form

$$p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i|y_{i-2}, y_{i-1}) \prod_{i=1}^n e(x_i|y_i) \tag{1}$$

Recall that we have assumed in this definition that $y_0 = y_{-1} = ^*$, and $y_{n+1} = \text{STOP}$. The Viterbi algorithm is shown in figure 1.

Now consider a four-gram tagger, where $p$ takes the form

$$p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i|y_{i-3}, y_{i-2}, y_{i-1}) \prod_{i=1}^n e(x_i|y_i) \tag{2}$$

We have assumed in this definition that $y_0 = y_{-1} = y_{-2} = ^*$, and $y_{n+1} = \text{STOP}$.
Question: In the box below, give a version of the Viterbi algorithm that takes as input a sentence $x_1 \ldots x_n$, and finds

$$\max_{y_1 \ldots y_{n+1}} p(x_1 \ldots x_n, y_1 \ldots y_{n+1})$$

for a four-gram tagger, as defined in Eq. 4.

| Input: | a sentence $x_1 \ldots x_n$, parameters $q(w|t, u, v)$ and $e(x|s)$. |
|---|---|
| Definitions: | Define $\mathcal{K}$ to be the set of possible tags. Define $\mathcal{K}_{-2} = \mathcal{K}_{-1} = \mathcal{K}_0 = \{^*\}$, and $\mathcal{K}_k = \mathcal{K}$ for $k = 1 \ldots n$. |
| Initialization: | |
| Algorithm: | |
| Return: | |
**Question:** In the box below, give a version of the Viterbi algorithm that takes as input an integer $n$, and finds

$$\max_{y_1 \ldots y_{n+1}, x_1 \ldots x_n} p(x_1 \ldots x_n, y_1 \ldots y_{n+1})$$

for a trigram tagger, as defined in Eq. 3. **Hence the input to the algorithm is an integer $n$, and the output from the algorithm is the highest scoring pair of sequences $x_1 \ldots x_n, y_1 \ldots y_{n+1}$ under the model.**

**Input:** an integer $n$, parameters $q(w|u, v)$ and $e(x|s)$.

**Definitions:** Define $\mathcal{K}$ to be the set of possible tags. Define $\mathcal{K}_{-1} = \mathcal{K}_0 = \{\ast\}$, and $\mathcal{K}_k = \mathcal{K}$ for $k = 1 \ldots n$. Define $\mathcal{V}$ to be the set of possible words.

**Initialization:**

**Algorithm:**

**Return:**
**Question 3**  Consider a trigram HMM, as introduced in class. We saw that the Viterbi algorithm could be used to find

$$\max_{y_1 \ldots y_{n+1}} p(x_1 \ldots x_n, y_1 \ldots y_{n+1})$$

where the max is taken over all sequences $y_1 \ldots y_{n+1}$ such that $y_i \in K$ for $i = 1 \ldots n$, and $y_{n+1} = \text{STOP}$. (Recall that $K$ is the set of possible tags in the HMM.)

In a trigram tagger we assume that $p$ takes the form

$$p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^n e(x_i | y_i)$$  \hspace{1cm} (3)$$

Recall that we have assumed in this definition that $y_0 = y_{-1} = *$, and $y_{n+1} = \text{STOP}$. The Viterbi algorithm is shown in figure 1.

Now consider a “skip” tagger, where $p$ takes the form

$$p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i | y_{i-2}) \prod_{i=1}^n e(x_i | y_i)$$  \hspace{1cm} (4)$$

We have assumed in this definition that $y_0 = y_{-1} = *$, and $y_{n+1} = \text{STOP}$. Note that a “skip” tagger replaces the term $q(y_i | y_{i-2}, y_{i-1})$ in a regular trigram tagger with

$$q(y_i | y_{i-2})$$

We call it a skip tagger because $y_{i-1}$ is now omitted from the conditioning information.
Question: In the box below, give a version of the Viterbi algorithm that takes as input a sentence $x_1 \ldots x_n$, and finds

$$\max_{y_1 \ldots y_{n+1}} p(x_1 \ldots x_n, y_1 \ldots y_{n+1})$$

for a skip tagger, as defined in Eq. 4. (Note: it is fine if the runtime of your algorithm is $O(n|\mathcal{K}|^3)$.)

**Input:** a sentence $x_1 \ldots x_n$, parameters $q(w|v)$ and $e(x|s)$.

**Definitions:** Define $\mathcal{K}$ to be the set of possible tags. Define $\mathcal{K}_{-1} = \mathcal{K}_0 = \{\ast\}$, and $\mathcal{K}_k = \mathcal{K}$ for $k = 1 \ldots n$.

**Initialization:**

**Algorithm:**

**Return:**
**Question 4** Say we have a training set consisting of two tagged sentences:

the/DT can/NN is/VB in/IN the/DT shed/NN

the/DT dog/NN can/VB see/VB the/DT cat/NN

We train a bigram tagger of the form

\[ p(x_1 \ldots x_n, y_1 \ldots y_{n+1}) = \prod_{i=1}^{n+1} q(y_i | y_{i-1}) \prod_{i=1}^{n} e(x_i | y_i) \]

using simple maximum-likelihood estimates for the \( q \) and \( e \) parameters.

If we then use the Viterbi algorithm to find the maximum probability tag sequence for each of the training sentences, show that the tagger tags both sentences correctly.

**Question 5** Now come up with a training set such that when we train a bigram tagger using maximum likelihood estimates, the resulting model makes at least one mistake on the training set.