Quiz 1, COMS 4705

Name:

30	10	15	25

*G*ood luck!

Input: a sentence $x_1 \dots x_n$, parameters $q(s|u,v)$ and $e(x|s)$.
Initialization: Set $\pi(0,*,*) = 1$, and $\pi(0,u,v) = 0$ for all (u,v) such that $u \neq *$ or $v \neq *$.
Algorithm:

- For $k = 1 \dots n$,
  - For $u \in \mathcal{K}$, $v \in \mathcal{K}$,

$$\pi(k,u,v) = \max_{w \in \mathcal{K}} (\pi(k-1,w,u) \times q(v|w,u) \times e(x_k|v))$$

- **Return** $\max_{u \in \mathcal{K}, v \in \mathcal{K}} (\pi(n,u,v) \times q(\text{STOP}|u,v))$

Figure 1: The Viterbi algorithm for trigram HMM taggers.

Part #1 — 30 points

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

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

where the max is taken over all sequences $y_1 \dots y_{n+1}$ such that $y_i \in \mathcal{K}$ for $i = 1 \dots n$, 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

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

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

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

We have assumed in this definition that $y_0 = y_{-1} = y_{-2} = *$, and $y_{n+1} = \text{STOP}$.

Question 1 (15 points) In the box below, give a version of the Viterbi algorithm that takes as input a sentence $x_1 \ldots x_n$, and finds

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

for a four-gram tagger, as defined in Eq. 2.

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

Algorithm:

Return:

Question 2 (15 points) 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. 1. **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.**

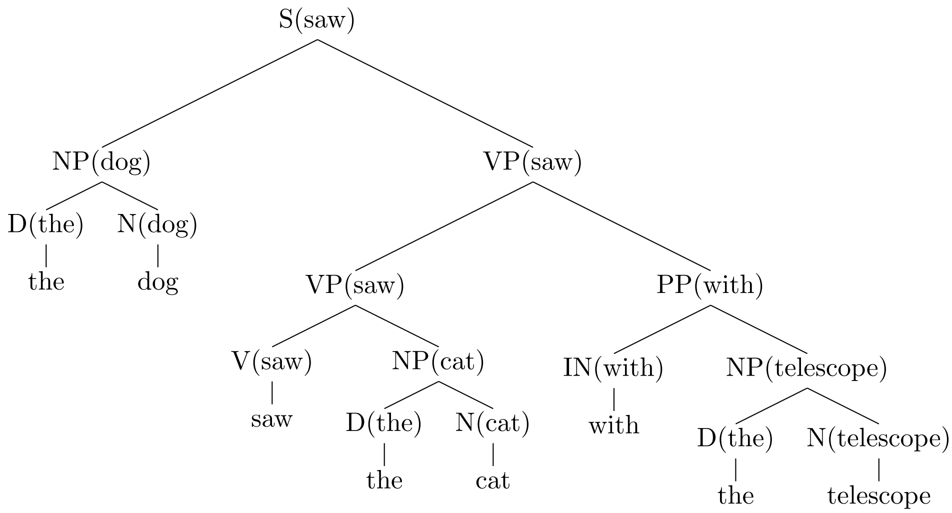
Input: an integer n, parameters $q(w|u, v)$ and $e(x|s)$.
Initialization:

Algorithm:

Return:

Part #2 10 points

Consider the lexicalized tree below:



Question 3 (10 points) Complete the head-finding rules below that would give this lexicalized tree:

For `S` → `NP VP` choose `VP` as the head

For `NP` → `D N` choose ______ as the head

For `VP` → `VP PP` choose ______ as the head

For `VP` → `V NP` choose ______ as the head

For `PP` → `IN NP` choose ______ as the head

Part #3 15 points

Consider the following definition of *bigram* language models (it is very similar to the definition of trigram language models seen in class):

Definition 1 (Bigram Language Model) *A bigram language model consists of a finite set $\mathcal{V}$, and a parameter*

$$q(w|v)$$

for each bigram v, w such that $w \in \mathcal{V} \cup \{STOP\}$, and $v \in \mathcal{V} \cup \{*\}$. The value for $q(w|v)$ can be interpreted as the probability of seeing the word w immediately after the word v. For any sentence $x_1 \ldots x_n$ where $x_i \in \mathcal{V}$ for $i = 1 \ldots (n-1)$, and $x_n = STOP$, the probability of the sentence under the bigram language model is

$$p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i|x_{i-1})$$

where we define $x_0 = *$.

Now consider the following two language models:

Language Model 1

$\mathcal{V} = \{the, dog\}$

$q(the|*) = q(dog|the) = q(STOP|dog) = 1$

All other q parameters are equal to 0.

Language Model 2

$\mathcal{V} = \{the, a, dog\}$

$q(the|*) = q(a|*) = 0.5$

$q(dog|a) = q(dog|the) = q(STOP|dog) = 1$

All other q parameters are equal to 0.

Question 4 (5 points) For language model 1, list all sentences $x_1 \ldots x_n$ such that $p(x_1 \ldots x_n) > 0$. For each sentence, write down its probability under language model 1.

Question 5 (5 points) For language model 2, list all sentences $x_1 \ldots x_n$ such that $p(x_1 \ldots x_n) > 0$. For each sentence, write down its probability under language model 2.

Question 6 (5 points) Now assume that we have a test sentence consisting of a single sentence,

the dog STOP

Which language model (model 1 or 2) gives **lower** perplexity on this test set?

Part #4 — 25 points

Consider a PCFG with the following rules

```
S → V N
S → D N
D → a
D → the
N → dog
V → saw
V → like
```

and the following parameters:

$q(\texttt{S} \rightarrow \texttt{V N}) = 0.6$
$q(\texttt{S} \rightarrow \texttt{D N}) = 0.4$
$q(\texttt{D} \rightarrow \texttt{a}) = 0.2$
$q(\texttt{D} \rightarrow \texttt{the}) = 0.8$
$q(\texttt{N} \rightarrow \texttt{dog}) = 1$
$q(\texttt{V} \rightarrow \texttt{saw}) = 0.6$
$q(\texttt{V} \rightarrow \texttt{like}) = 0.4$

For any sentence x, define $\mathcal{T}(x)$ to be the set of parse trees for x, and define

$$p(x) = \sum_{t \in \mathcal{T}(x)} p(t)$$

where $p(t)$ is the probability of parse tree t under the PCFG shown above.

Question 7 (10 points) List all sentences x such that $p(x) > 0$, where $p(x)$ is defined through the above PCFG. For each sentence, write down its probability.

Question 8 (15 points) A bigram language model has the following definition:

Definition 2 (Bigram Language Model) *A bigram language model consists of a finite set $\mathcal{V}$, and a parameter*

$$q(w|v)$$

for each bigram v, w such that $w \in \mathcal{V} \cup \{STOP\}$, and $v \in \mathcal{V} \cup \{*\}$. The value for $q(w|v)$ can be interpreted as the probability of seeing the word w immediately after the word v. For any sentence $x_1 \ldots x_n$ where $x_i \in \mathcal{V}$ for $i = 1 \ldots (n-1)$, and $x_n = STOP$, the probability of the sentence under the bigram language model is

$$p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i|x_{i-1})$$

where we define $x_0 = *$.

Define a bigram language model that gives the same probability distribution $p(x)$ over sentences as the PCFG shown above. The vocabulary in the language model should be $\mathcal{V} = \{$a, the, dog, saw, like$\}$. You should specify the parameters of the language model.