Quiz 1, COMS 4705

Name:

10	30	30	20

*G*ood luck!

Part #1 (10 points)

Question 1 (10 points) We define a PCFG where non-terminal symbols are $\{S, A, B\}$, the terminal symbols are $\{a, b\}$, and the start non-terminal (the non-terminal always at the root of the tree) is S. The PCFG has the following rules:

Rule	Probability
$S \rightarrow S\ S$	0.3
$S \rightarrow A\ S$	0.2
$S \rightarrow B\ B$	0.5
$A \rightarrow a$	0.2
$A \rightarrow b$	0.8
$B \rightarrow a$	0.4
$B \rightarrow b$	0.6

For the input string *abab*, show two possible parse trees under this PCFG, and show how to calculate their probability.

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 $u \in \mathcal{K}$, $v \in \mathcal{K}$,

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

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

Figure 1: The Viterbi algorithm for trigram HMM taggers.

Part #2 — 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

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

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 **bigram** HMM tagger, where we instead have the following definition:

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

where $y_0 = y_{-1} = *$, and $y_{n+1} = \text{STOP}$. The parameters of the bigram model take the form $q(s|v)$ and $e(x|s)$. Note that we have replaced $q(y_i|y_{i-2}, y_{i-1})$ with $q(y_i|y_{i-1})$ in this definition, so intuitively each state only depends on the previous state.

Question 2 (30 points) In the box below, give a version of the Viterbi algorithm that finds

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

for a bigram HMM tagger, as defined in Eq. 2. You will get 30 points on the question if you have a correct algorithm, which runs in $O(n|\mathcal{K}|^2)$ time, where n is the length of the sentence, and $|\mathcal{K}|$ is the number of tags. You will get a maximum of 15 points on the question if you have a correct algorithm, but it runs in slower than $O(n|\mathcal{K}|^2)$ time.

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

Initialization:

Algorithm:

Return:

Part #3

30 points

Consider the CKY algorithm for finding the maximum probability for any tree when given as input a sequence of words $x_1, x_2, \ldots, x_n$. As usual, we use N to denote the set of non-terminals in the grammar, and S to denote the start symbol.

The base case in the recursive definition is as follows: for all $i = 1 \ldots n$, for all $X \in N$,

$$\pi(i,i,X) \quad = \quad \begin{cases} q(X \to x_i) & \text{if } X \to x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

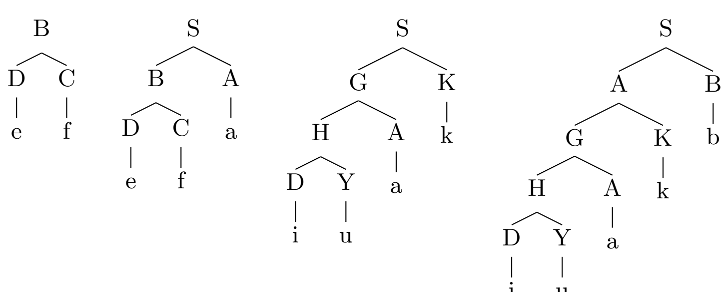
and the recursive definition is as follows: for all (i,j) such that $1 \leq i < j \leq n$, for all $X \in N$,

$$\pi(i,j,X) = \max_{\substack{X \to Y Z \in R, \\ s \in \{i \ldots (j-1)\}}} \left(q(X \to Y Z) \times \pi(i,s,Y) \times \pi(s+1,j,Z) \right)$$

Finally, we return

$$\pi(1,n,S) = \max_{t \in \mathcal{T}_G(s)} p(t)$$

Now assume that we want to find the maximum probability for any *left-branching* tree for a sentence. Here are some example left-branching trees:



(**Question continued on next page**)

It can be seen that in left-branching trees, whenever a rule of the form `X -> Y Z` is seen in the tree, then the non-terminal `Y` must directly dominate a terminal symbol.

Question 3 (30 points) Complete the recursive definition below, so that the algorithm returns the maximum probability for any **left-branching** tree underlying a sentence $x_1, x_2, \ldots, x_n$.

Base case: for all $i = 1 \ldots n$, for all $X \in N$,

$$\pi(i, i, X) \quad = \quad \begin{cases} q(X \to x_i) & \text{if } X \to x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

Recursive case: (Complete below)

Return:

$$\pi(1, n, S) = \max_{t \in \mathcal{T}_G(s)} p(t)x$$

Part #4 — 20 points

In lecture we saw how to build trigram language models using *discounting methods*, and the *Katz back-off* definition. We're now going to build a *four-gram* language model based on these ideas. A four-gram language model gives estimates

$$q(w|t, u, v)$$

where t, u, v, w is any sequence of four words.

Assume we have a corpus, and that $c(t, u, v, w)$ is the number of times the four-gram t, u, v, w is seen in the data. Then take the following definitions:

$$\mathcal{A}(t, u, v) = \{w : c(t, u, v, w) > 0\}$$

and

$$\mathcal{B}(t, u, v) = \{w : c(t, u, v, w) = 0\}$$

Define $c^*(t, u, v, w)$ to be the discounted count for the four-gram (t, u, v, w), as follows:

$$c^*(t, u, v, w) = c(t, u, v, w) - 0.5$$

Assume that for any trigram u, v, w, $q_{BO}(w|u, v)$ is an estimate of the trigram probability, using the backed-off method described in lecture.

Finally, we define the four-gram model as

$$q_{BO}(w|t, u, v) = \begin{cases} \frac{c^*(t,u,v,w)}{c(t,u,v)} & \text{If } w \in \mathcal{A}(t, u, v) \\ \alpha(t, u, v) \times \frac{q_{BO}(w|u,v)}{\sum_{w \in \mathcal{B}(t,u,v)} q_{BO}(w|u,v)} & \text{If } w \in \mathcal{B}(t, u, v) \end{cases}$$

Question 4 (20 points) How would you define

$$\alpha(t, u, v)$$

?