# Midterm, COMS 4705

Name:

| 15 | 10 | 10 | 15 | 15 |
|---|---|---|---|---|
| | | | | |

*G*ood luck!

## Part #1 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 assume that our vocabulary $\mathcal{V} = \{\text{the}\}$, that is, the vocabulary has a single word *the*. We would like to define the parameters of a bigram language model such that

$$p(\text{STOP}) = 0$$

$$p(\text{the STOP}) = 0.4$$

$$p(\text{the the STOP}) = 0.4 \times 0.6$$

$$p(\text{the the the STOP}) = 0.4 \times 0.6^2$$

$$\ldots$$

(In general the probability of a sentence which has the word *the* $n$ times, for $n \geq 1$, is $0.4 \times 0.6^{n-1}$.)

**Question 1** (7 points) Write down the parameters of the language model such that it gives the above distribution over sentences (i.e., $p(x) = 0.4 \times 0.6^{n-1}$ if $x$ is a sentence of $n$ consecutive *the*'s, followed by the STOP symbol).

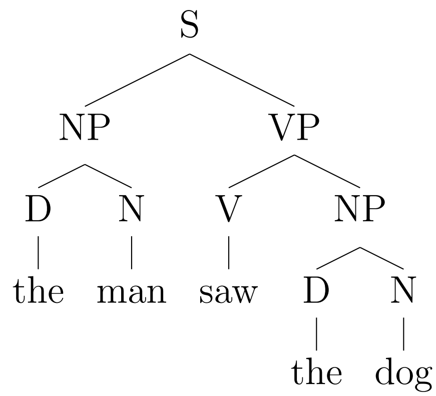
**Question 2** (8 points) Write down a PCFG such that:

1. Any sentence consisting of the word *the* $n$ times in a row, where $n \geq 1$, has probability
$$0.4 \times 0.6^{n-1}$$
2. Any other sentence has probability 0.

(I.e., this is the same distribution as in the last question)

## Part #2 (10 points)

Consider the following parse tree:




And in addition consider the following rules that can be used to lexicalize the parse tree (note that these rules do not necessarily make sense from a linguistic perspective):

- For the rule `S -> NP VP`, we define `NP` to be the head
- For the rule `NP -> D N`, we define `D` to be the head
- For the rule `VP -> V NP`, we define `V` to be the head

Recall that for a lexicalized PCFG in Chomsky Normal form, each rule takes one of the following forms:

- $X(h) \rightarrow_1 Y_1(h)Y_2(m)$ where $X, Y_1, Y_2$ are non-terminals, and $h, m$ are words
- $X(h) \rightarrow_2 Y_1(m)Y_2(h)$ where $X, Y_1, Y_2$ are non-terminals, and $h, m$ are words
- $X(h) \rightarrow h$ where $X$ is a non-terminal, and $h$ is a word

**Question 3** (10 points) If we lexicalize the above parse tree, then build a lexicalized PCFG with all rules seen in the tree, what is the complete set of rules in the grammar? (You do not need to include probabilities for the rules, just list the rules in the grammar.)

## Part #3 (10 points)

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:

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

  with all other parameter values equal to 0.

**Question 4** (10 points) 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 $u \in \mathcal{K}_{k-1}$, $v \in \mathcal{K}_k$,

$$\pi(k, u, v) = \max_{w \in \mathcal{K}_{k-2}} \left(\pi(k-1, w, u) \times q(v|w, u) \times e(x_k|v)\right)$$

- **Return** $\max_{u \in \mathcal{K}_{n-1}, v \in \mathcal{K}_n} \left(\pi(n, u, v) \times q(\text{STOP}|u, v)\right)$

Figure 1: The basic Viterbi Algorithm.

## Part #4 — 15 points

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

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

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 $y_{n+1} =$ STOP. The Viterbi algorithm is shown in figure 1.

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

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

We have assumed in this definition that $y_0 = y_{-1} = y_{-2} = *$, 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 5 (15 points)** In the box below, give a version of the Viterbi algorithm that takes as input a sentence $x_1 \dots x_n$, and finds

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

for a skip tagger, as defined in Eq. 2. (Note: it is fine if the runtime of your algorithm is $O(n|\mathcal{K}|^3)$.)

**Input:** a sentence $x_1 \dots x_n$, parameters $q(w|v)$ and $e(x|s)$.
**Definitions:** Define $\mathcal{K}$ to be the set of possible tags. Define $\mathcal{K}_{-1} = \mathcal{K}_0 = \{*\}$, and $\mathcal{K}_k = \mathcal{K}$ for $k = 1 \dots n$.
**Initialization:**

**Algorithm:**

**Return:**

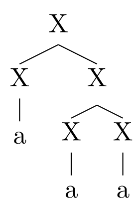
## Part #5 (15 points)

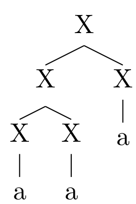
In this question our goal is to design an algorithm that takes a sentence $s$ and a context-free grammar in Chomsky normal form as input, and as its output returns *the number of parse trees for the sentence $s$* as its output.

For example, if $s$ is the sentence `a a a`, and the context-free grammar is

X $\rightarrow$ X X
X $\rightarrow$ a

with start symbol `X`, the algorithm should return the value 2, because there are two parses for the sentence under this grammar:



**Question 6** (15 points) Complete the following algorithm so that it returns the number of possible parse trees for the input sentence $s$.

**Input:** a sentence $s = x_1 \ldots x_n$, a context-free grammar $G = (N, \Sigma, S, R)$.
**Initialization:**
For all $i \in \{1 \ldots n\}$, for all $X \in N$,

$$\pi(i, i, X) \quad = \quad \begin{cases} 1 & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

**Algorithm:**

- For $l = 1 \ldots (n-1)$
  - For $i = 1 \ldots (n-l)$
    - Set $j = i + l$
    - For all $X \in N$, calculate

$$\pi(i, j, X) = \sum_{\substack{X \rightarrow Y\,Z \in R, \\ s \in \{i \ldots (j-1)\}}} \underbrace{\hspace{6cm}}_{\text{COMPLETE THE DEFINITION HERE}}$$

**Output:** Return $\pi(1, n, S)$