Midterm for 6.864

Name:

30	30	30	30

*G*ood luck!

Part #1 30 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
$S \rightarrow S\ A$	0.1
$S \rightarrow A\ B$	0.4
$S \rightarrow B\ A$	0.5
$A \rightarrow a$	0.2
$A \rightarrow b$	0.8
$B \rightarrow a$	0.3
$B \rightarrow b$	0.7

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

Solution:

- **S**
  - **S**
    - **S**
      - **A**
        - **a**
      - **B** **Probability:** $(0.1)^2(0.4)(0.2)^2(0.3)(0.7)$
        - **b**
    - **A**
      - **a**
  - **A**
    - **b**

- **S**
  - **S**
    - **S**
      - **B**
        - **a**
      - **A** **Probability:** $(0.1)^2(0.5)(0.8)^2(0.3)(0.2)$
        - **b**
    - **A**
      - **a**
  - **A**
    - **b**

Question 2 (10 points)

Consider an application of global linear models to parsing. In this scenario each input x is a sentence. We have a fixed context-free grammar; GEN(x) returns the set of all parses allowed for x under the context-free grammar. The feature vector $\Phi(x, y)$ for any sentence x paired with a parse tree y is defined as

$$\Phi(x, y) = \sum_{\alpha \to \beta \in (x,y)} \bar{\phi}(\alpha \to \beta)$$

where $\bar{\phi}$ is a function that maps a context-free rule $\alpha \to \beta$ to a feature vector, and the notation $\sum_{\alpha \to \beta \in (x,y)}$ refers to a sum over all context-free rules in the parse tree defined by (x, y).

We'd like $\Phi(x, y)$ to be a 3-dimensional feature vector, with the following values for its three components:

$$\begin{aligned}
\Phi_1(x, y) &= \text{number of times } \texttt{S -> NP VP} \text{ is seen in } (x, y) \\
\Phi_2(x, y) &= \text{number of times } \texttt{N -> dog} \text{ is seen in } (x, y) \\
\Phi_3(x, y) &= \text{number of times } \texttt{NP -> NP NP} \text{ is seen in } (x, y)
\end{aligned}$$

Give a definition for the function $\bar{\phi}$ which leads to this definition of $\Phi(x, y)$.

Solution:

$$\bar{\phi}(\alpha \to \beta) = \begin{cases}
\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} & \textbf{if } \alpha = \textbf{S and } \beta = \textbf{NP VP} \\[2ex]
\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} & \textbf{if } \alpha = \textbf{S and } \beta = \textbf{dog} \\[2ex]
\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & \textbf{if } \alpha = \textbf{S and } \beta = \textbf{NP NP} \\[2ex]
\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} & \textbf{otherwise}
\end{cases}$$

Question 3 (10 points)

In the lecture on text segmentation, we considered unsupervised methods for this task. An alternative approach is to model text segmentation as a supervised learning task. During training, we are provided with texts where sentence breaks that correspond to segment boundaries are annotated as positive examples. The rest of the sentence breaks are annotated as negative examples. Assume that a boundary location depends on the word on the left, the word on the right, and on the tag of the previous sentence break (i.e., whether it is a segment boundary or not).

We would like to apply a log-linear tagger to this problem. Define the features of a log-linear tagger which would implement the specification above.

Solution:
We can use features of the following three types, which are defined for all words w and for $t, t_{-1} \in \{+1, -1\}$ which indicate if the is a boundary or no boundary.

$$\phi^1_{w,t} = \begin{cases} 1 & \textbf{if word on the left is } w \textbf{ and current tag is } t \\ 0 & \textbf{otherwise} \end{cases}$$

$$\phi^1_{w,t} = \begin{cases} 1 & \textbf{if word on the right is } w \textbf{ and current tag is } t \\ 0 & \textbf{otherwise} \end{cases}$$

$$\phi^1_{t_{-1},t} = \begin{cases} 1 & \textbf{if the previous tag is } t_{-1} \textbf{ and current tag is } t \\ 0 & \textbf{otherwise} \end{cases}$$

Part #2 — 30 points

Consider the following HMM model for tagging. We will assume that the vocabulary consists of three words, *the, dog, sleeps.* There are two possible part-of-speech tags, *D, N.*

One set of parameters in the HMM are probabilities of the form $P(word|tag)$, for example $P(the|D)$, $P(the|N)$, etc. In our HMM these probabilities take the following values:

Parameter	Value
$P(the\|D)$	0.7
$P(dog\|D)$	0.2
$P(sleeps\|D)$	0.1
$P(the\|N)$	0.2
$P(dog\|N)$	0.5
$P(sleeps\|N)$	0.3

Another set of parameters in the HMM are of the form P(tag|previous-tag), for example $P(N|D)$, $P(D|N)$, etc. Note that we have a *bigram* HMM tagger, in that each tag depends only on the previous tag (in lecture we saw a *trigram* tagger, where each tag depends on the previous two tags). We take `START` to be a special tag which always appears at the start of a sentence, and `STOP` to be a tag that appears at the end of the sentence. In our model these parameters take the following values:

Parameter	Value
$P(D\|START)$	0.1
$P(N\|START)$	0.2
$P(STOP\|START)$	0.7
$P(D\|D)$	0.6
$P(N\|D)$	0.2
$P(STOP\|D)$	0.2
$P(D\|N)$	0.4
$P(N\|N)$	0.5
$P(STOP\|N)$	0.1

Question 4 (30 points) We would like you to define a PCFG that is "equivalent" to the above HMM. By "equivalent" we mean the following:

- For any symbol sequence x and state sequence y which has probability $P_{HMM}(x, y) > 0$ under the HMM: 1) there should be a parse tree y' such that the pair x, y' gets probability $P_{PCFG}(x, y')$ under the PCFG; 2) this probability should satisfy $P_{HMM}(x, y) = P_{PCFG}(x, y')$

- There should be a one-to-one function $f(y) = y'$ that maps a state sequence in the HMM to a parse tree generated by the PCFG, and that satisfies $P_{HMM}(x, y) = P_{PCFG}(x, f(y))$

In your solution you should: (a) write down your PCFG which is equivalent to the above HMM; (b) define the function $f(y)$ between state sequences and parse trees.

Note: you may find it useful to make use of productions in your PCFG where ϵ (the empty string) is on the right-hand-side of your rule. For example,

$$X \rightarrow \epsilon \quad \text{with } P(X \rightarrow \epsilon | X) = 0.1$$

states that the non-terminal X can rewrite to the empty string with probability 0.1. To ellaborate further, the context-free grammar

$$S \rightarrow a\ X$$
$$X \rightarrow \epsilon$$

Generates one string, i.e., the string `a`.

Solution:

(a)

S → START D'	**0.1**
S → START N'	**0.2**
S → START STOP	**0.7**
D' → D D'	**0.6**
D' → D N'	**0.2**
D' → D STOP	**0.2**
N' → N D'	**0.4**
N' → N N'	**0.5**
N' → N STOP	**0.1**
D → the	**0.7**
D → dog	**0.2**
D → sleeps	**0.1**
N → the	**0.2**
N → dog	**0.5**
N → sleeps	**0.3**
START → *	**1.0**
STOP → *	**1.0**

(b) Given a state sequence $s_1, s_2, ..., s_n$ where $s_i \in \{D, N\}$ and a corresponding word string $w_1, w_2, ..., w_n$ return the following tree:

```
S
    START
        *
    s'_1
        s_1
            w_1
        s'_2
            s_2
                w_2
...
                    s'_n
                        s_n
                            w_n
                        STOP
                            *
```

Part #3 — 30 points

This question considers log-linear models. We'd like to build a model that estimates a distribution $P(tag|word)$ using a log-linear model. The variable tag can take any one of three values, *D, N, V*. The variable $word$ could potentially be any member of a set $\mathcal{V}$ of possible words. The set $\mathcal{V}$ contains the words *the, dog, sleeps*, as well as additional words (i.e., $|\mathcal{V}| > 3$). The distribution should give the following probabilities:

$$\begin{aligned}
P(D|the) &= 0.9 \\
P(N|dog) &= 0.9 \\
P(V|sleeps) &= 0.9 \\
P(D|word) &= 0.6 \text{ for any word other than } the, dog \text{ or } sleeps \\
P(N|word) &= 0.3 \text{ for any word other than } the, dog \text{ or } sleeps \\
P(V|word) &= 0.1 \text{ for any word other than } the, dog \text{ or } sleeps
\end{aligned}$$

Note that we have intentionally left the values for the following probabilities undefined: $P(N|the)$, $P(V|the)$, $P(D|dog)$, $P(V|dog)$, $P(D|sleeps)$, $P(N|sleeps)$. It is assumed that these probabilities could take any values such that $\sum_{tag} P(tag|word) = 1$ is satisfied for $word =$ *the, dog, sleeps*, or indeed any other word in $\mathcal{V}$.

Question 5 (10 points)

Define the features for a log-linear model that can model this distribution $P(tag|word)$ perfectly. Each feature should be an indicator function: i.e., each feature $\phi(x, y)$ can take only the values 0 or 1 depending on the values of x and y. Your model should make use of as few features as possible. We will give you 10 points for using 6 features, and will penalise you for using more than 6 features. (It may be possible to use only 5 features, but a model with 6 features may be more straightforward to analyse.)

Solution:

$$\phi_1(t, w) = \begin{cases} 1 & \textbf{if } t = D \textbf{ and } w = the \\ 0 & \textbf{otherwise} \end{cases}$$

$$\phi_2(t, w) = \begin{cases} 1 & \textbf{if } t = N \textbf{ and } w = dog \\ 0 & \textbf{otherwise} \end{cases}$$

$$\phi_3(t, w) = \begin{cases} 1 & \textbf{if } t = V \textbf{ and } w = sleeps \\ 0 & \textbf{otherwise} \end{cases}$$

$$\phi_4(t,w) = \begin{cases} 1 & \textbf{if } t = D \textbf{ and } w \notin \{the, dog, sleeps\} \\ 0 & \textbf{otherwise} \end{cases}$$

$$\phi_5(t,w) = \begin{cases} 1 & \textbf{if } t = N \textbf{ and } w \notin \{the, dog, sleeps\} \\ 0 & \textbf{otherwise} \end{cases}$$

$$\phi_6(t,w) = \begin{cases} 1 & \textbf{if } t = V \textbf{ and } w \notin \{the, dog, sleeps\} \\ 0 & \textbf{otherwise} \end{cases}$$

Question 6 (10 points)

Write an expression for each of the probabilities

$$P(\text{D}|\text{cat})$$
$$P(\text{N}|\text{laughs})$$
$$P(\text{D}|\text{dog})$$
$$P(\text{V}|\text{sleeps})$$

as a function of the parameters in your model. (Assume that the words *laughs* and *cat* are both members of the set $\mathcal{V}$.)

Solution:

$$P(D|cat) = \frac{e^{\theta_4}}{e^{\theta_4} + e^{\theta_5} + e^{\theta_6}}$$

$$P(N|laughs) = \frac{e^{\theta_5}}{e^{\theta_4} + e^{\theta_5} + e^{\theta_6}}$$

$$P(D|dog) = \frac{1}{e^{\theta_2} + 1}$$

$$P(D|cat) = \frac{e^{\theta_3}}{e^{\theta_3} + 1}$$

Question 7 (10 points)

What value do the parameters in your model take to give the distribution described above?

Solution:

$$
\begin{aligned}
P(D|cat) &= \frac{e^{\theta_3}}{e^{\theta_3}+1} \\
0.9 &= \frac{e^{\theta_3}}{e^{\theta_3}+1} \\
0.9 + 0.9e^{\theta_3} &= e^{\theta_3} \\
0.9 &= 0.1e^{\theta_3} \\
\theta_3 &= \log 9
\end{aligned}
$$

Similarly $\theta_1 = \theta_2 = \log 9$

$$
\begin{aligned}
0.6 &= \frac{e^{\theta_4}}{e^{\theta_4}+e^{\theta_5}+e^{\theta_6}} \\
0.3 &= \frac{e^{\theta_5}}{e^{\theta_4}+e^{\theta_5}+e^{\theta_6}} \\
0.1 &= \frac{e^{\theta_6}}{e^{\theta_4}+e^{\theta_5}+e^{\theta_6}} \\
e^{\theta_4} &= 6e^{\theta_6} \\
e^{\theta_5} &= 3e^{\theta_6}
\end{aligned}
$$

Let $\theta_6 = \log 2$**, then** $\theta_5 = \log 6$ **and** $\theta_4 = \log 12$**.**

Part #4

30 points

Consider the task of aligning two English translations of the same story (extracts from two translations of Andersen's tale are shown below.) In the sentence alignment task, the goal is to identify groups of sentences in one translation that correspond to groups of sentences in the other translation.

1) It was a Princess who was standing outside the door.
2) She was in a sad condition.
3) The water trickled down from her hair.
4) Her clothes clung to her body.
5) She said she was a real Princess.

A) It was a princess standing out there in front of the gate.
B) The water ran down from her hair and clothes, and her dress clung to her body.
C) And yet she said that she was a real princess.

Question 8 (10 points)

We first need to introduce a function that compares similarity between two sentences. Recall the definition for the cosine distance between two vectors x and y, as seen in lecture:

$$cosine(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x^2}\sqrt{\sum_{i=1}^{n} y^2}}$$

How would you define a mapping from sentences s to vectors $f(s)$ such that the cosine distance $cosine(f(s_1), f(s_2))$ is a measure of the similarity between two sentences s_1 and s_2? We'll give full marks for any reasonable mapping.

Solution:

$f(s)$ = **a vector that contains a feature for each word** w **that counts the number of itmes** w **is seen in** s

Question 9 (10 points)

Now, we will define a dynamic programming algorithm that computes sentence alignment of two translations. The alignment score is the sum of the similarity scores of the aligned sentences. Our goal is to find an alignment with the highest score.

We will consider alignments of the following form:

- a sentence can be aligned to an empty sentence. This happens when one of the translators omits a sentence (for example, sentence (2) in the above example would most likely be aligned to an empty sentence).
- a sentence can be aligned to exactly one sentence (for example, sentences (1) and (A) would most likely be aligned to each other).
- a sentence can be aligned to two sentences. This happens when one of the translators either breaks or joins sentences (for example, sentences (3) and (4) would most likely be aligned to sentence (B)).

Our sentence alignment algorithm recursively computes the alignment matrix F indexed by i and j, one index for each sentence. The value stored in $F(i,j)$ is the score of the best alignment between the first i sentences of x and the first j sentences of y. $s(x_i, y_j)$ is the similarity between sentence x_i and sentence y_j.

- Define $F(0,0)$, $F(i,0)$ and $F(0,j)$.

  Solution:

$$\begin{aligned} F(0,0) &= 0 \\ F(i,0) &= 0 \\ F(0,j) &= 0 \end{aligned}$$

- Define $F(i,j)$ for $i > 0$ and $j > 0$.

  Solution:

$$\begin{aligned} F(i,j) = max(&F(i-1,j-1) + s(i,j), \\ &F(i-1,j), \\ &F(i,j-1), \\ &F(i-2,j-1) + s(i-1,j) + s(i,j), \\ &F(i-1,j-2) + s(i,j-1) + s(i,j)) \end{aligned}$$

Question 10 (10 points)

Next, we'll modify the aligment score to be the same as before, but to include a fixed penalty p each time a sentence is aligned to an empty sentence. p is a parameter, which is ≥ 0, chosen by the user of the algorithm. Describe a modified dynamic programming method which takes this new penalty into account.

Solution:

$$\begin{aligned} F(0,0) &= 0 \\ F(i,0) &= -ip \\ F(0,j) &= -jp \end{aligned}$$

$$\begin{aligned} F(i,j) = max(&F(i-1,j-1) + s(i,j), \\ &F(i-1,j) - p, \\ &F(i,j-1) - p, \\ &F(i-2,j-1) + s(i-1,j) + s(i,j), \\ &F(i-1,j-2) + s(i,j-1) + s(i,j)) \end{aligned}$$