Midterm for 6.864
Name:

<table>
<thead>
<tr>
<th>40</th>
<th>30</th>
<th>30</th>
<th>30</th>
</tr>
</thead>
</table>

Good luck!
Part #1  

We define a PCFG where the non-terminals are \{S, NP, VP, Vt, NN, PP, IN\}, the terminal symbols are \{Mary, ran, home, with, John\}, and the start non-terminal is S. The PCFG has the following rules:

- S -> NP VP 1
- VP -> Vt NP 0.5
- VP -> VP PP 0.5
- NP -> NP PP 0.3
- NP -> NN 0.7
- PP -> IN NP 1
- Vt -> ran 1
- NN -> Mary 0.2
- NN -> John 0.3
- NN -> home 0.4
- NN -> umbrella 0.1
- IN -> with 5

**Question 1** (10 points)

For the input string *MaryRanHomeWithUmbrella*, show two possible parse trees under the PCFG, and show how to calculate their probability.
Consider a language model that combines information from a syntactic language model and a bigram language model. These two sources of information are combined using linear interpolation. I.e., the probability of a text consisting of words $w_1, ..., w_n$ is:

$$\lambda P_{\text{syntax}}(w_1, ..., w_n) + (1 - \lambda)P_{\text{bigram}}(w_1, ..., w_n)$$

**Question 2** (15 points)

Prove that this language model forms a valid probability distribution over all possible texts. You may assume that $P_{\text{syntax}}$ and $P_{\text{bigram}}$ are both valid probability distributions themselves.
In class, we have looked at the algorithm which finds the maximum probability tree when given a sequence of words \(w_1, w_2, \ldots, w_n\) and a PCFG. Your goal is to modify this algorithm so that it finds the maximum probability tree that does not exceed a given height. The height of the tree is defined as the length of the longest path from the root to a leaf.

**Question 3** (6.25 points)

Describe the structure of your dynamic programming table \(\pi\) — specify its dimensions and describe the content of cells in this table.

**Question 4** (6.25 points)

Specify the base cases for your computation.

**Question 5** (6.25 points)

Specify the recursive conditions for your computation.

**Question 6** (6.25 points)

What is the time complexity of your algorithm?
Recall that a log-linear model has the form

\[ P(y|x; \Theta) = \frac{e^{f(x,y) \cdot \Theta}}{\sum_{y'} e^{f(x,y') \cdot \Theta}} \]

where \( y \) is a label, \( x \) is an input, \( f(x,y) \) is an \( d \)-dimensional feature vector, and \( \Theta \) is an \( d \)-dimensional vector of real-valued weights.

Now suppose that we are given a corpus of morphologically segmented text, and we wish to train a log-linear model to make a correct segmentation decision \( y \) given a word \( x \). Our goal here will be to define the feature function \( f(x,y) \).

In our first attempt, we list every possible string up to some very large length (longer than any potential word): \( s_1, \ldots, s_d \). We then define our feature function \( f(x,y) \) as:

\[ f(x,y)_k = \begin{cases} 
1 & \text{if } s_k \text{ appears as a distinct morpheme in word } x \text{ segmented according to } y \\
0 & \text{otherwise}
\end{cases} \]

for all \( k \in \{1, \ldots, d\} \).

**Question 7** (5 points)

Consider a string \( s_k \) which never appears in the corpus, even as a substring of a word (say, “arwqabr”). What effect will \( \Theta_k \) have on the conditional likelihood of our corpus, \( \prod_{x,y} P(y|x; \Theta) \)? If we use a Gaussian prior on \( \Theta \), what will be the MAP estimate for \( \Theta_k \)?
Question 8  (5 points)

Suppose we have learned some $\Theta$ from our corpus. Now we wish to segment a word which never appeared there: “edict”. Let’s assume that the suffix -ed appeared many times in our corpus (“jump-ed” “lump-ed” etc), and its corresponding weight in $\Theta$ is quite large. This model will still not necessarily assign higher probability to “ed-ict” than to “edict”. Why not?
Question 9  (7.5 points)

But let’s suppose that, unfortunately, the model does assign higher probability to the incorrect analysis “ed-ict”. Modify the feature function $f(x, y)$ in a linguistically motivated way to avoid this and similar mistakes.
**Question 10** (7.5 points)

Finally, suppose we wanted to add a feature corresponding to the number of morphemes in the word-segmentation pair \((x, y)\). We consider two options: (a) add a single *integer*-valued feature:

\[
f(x, y)_{d+1} = \text{count of morphemes in } (x, y)
\]

and (b), for each integer \(i\) up to a very large number add a *binary* feature which is only active when there are \(i\) morphemes in \((x, y)\):

\[
f(x, y)_{d+i} = \begin{cases} 
1 & \text{if there are } i \text{ morphemes in } (x, y) \\
0 & \text{otherwise}
\end{cases}
\]

Each of these approaches has merit. Name an advantage of each approach.
In class we discussed n-gram language models and the problem of data sparsity. One approach to handling sparsity is to use smoothing methods. Perhaps the simplest smoothing method we discussed was add-$\alpha$ smoothing, where the parameters are set as:

$$\Theta(w_n|w_1, \ldots, w_{n-1}) \leftarrow \frac{\text{Count}(w_1, \ldots, w_n) + \alpha}{\text{Count}(w_1, \ldots, w_{n-1}) + \alpha|V|}$$

We mentioned in class that this is the MAP estimator for the n-gram language model when we assume that the priors over possible values of $\Theta$ are Dirichlet distributions parameterized by $\alpha$ (all you have to know about the Dirichlet distribution is that it is a distribution over all possible positive $\Theta$ values which sum to one).

Now suppose that in addition to our corpus, we are given a dictionary which lists synonym pairs. That is, we are given a list of non-overlapping word pairs $(w, v)$ and a similarity score $0 < \lambda_{w,v} < 1$ for each such pair. This similarity score tells us how similar in meaning and grammatical usage the two words are. Now, one simple way to take this information into account would be to merge all synonym pairs into single words. E.g., all instances of *jump* or *leap* could be treated as *jumpORleap*. This would be especially useful if one of the words never or rarely appears in the training corpus, but the other word appears more frequently.

There are two problems with this approach: (1) it doesn’t take the degree of similarity–$\lambda_{w,v}$–into account, and (2) it doesn’t account for the possibility that the actual data we observe may not match perfectly with what we expect from the dictionary and should thus override it.

**Question 11** (25 points)

Your job is to describe a new prior distribution on the $\Theta$’s which uses the information from the dictionary. Your solution should use the $\lambda_{w,v}$ values and ideally also Dirichlet distributions (so that the benefits of add-$\alpha$ smoothing remain). You should explain why using a MAP estimate under this prior solves problem (2) above. **You do NOT need to derive a MAP estimator for your prior.**