Course Policy: Read all the instructions below carefully before you start working on the assignment, and before you make a submission. The course assignment policy is available at http://nlp.cs.princeton.edu/cos484.

- This assignment contains 5 problems. 3 theory problems and 2 programming problems.
- We highly recommend that you typeset your submissions in \LaTeX. Use the template provided on the website for your answers. If you’ve never used \LaTeX, you can refer to the short guide here: http://bit.ly/WorkingWithLaTeX. Include your name and NetIDs with your submission. If you wish to submit hand written answers, you can scan and upload the pdf.
- Assignments must be uploaded to Gradescope before class (9:30am Eastern) on the due date mentioned above.
- As per the late day policy outlined on the course website, you have 96 allowed late hours (about 4 days) overall to use at your discretion throughout the semester. Once you run out of late hours, late submissions will incur a penalty of 10% for each day, up to a maximum of 3 days beyond which submissions will not be accepted.
- For the programming questions, you only need to complete the coding part specified in the .ipynb notebook. You need to compress the two .ipynb files for programming problem 1 and 2 into a single .zip file and upload this one file. Make sure the cells are properly executed and have the outputs printed.
**Problem 1: Hidden Markov Models for Tagging**  

We are interested in labeling each word in the following corpus of text with a set of sentiment tags:

- the bread smells delicious
- the coffee was bitter and awful

The set of tags available to us is \{+, −, O\}, representing positive sentiment, negative sentiment and neutral sentiment, respectively. We assume Markov assumption and output independence, i.e. each tag only depends on the previous tag and each word only depends on its corresponding tag. This results in a hidden Markov model (HMM) over the tags and words. We are also given the following parameters of the HMM:

<table>
<thead>
<tr>
<th>and</th>
<th>awful</th>
<th>bitter</th>
<th>bread</th>
<th>coffee</th>
<th>delicious</th>
<th>smells</th>
<th>the</th>
<th>was</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>0.0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.1</td>
<td>0.6</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>-</td>
<td>0.0</td>
<td>0.5</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>O</td>
<td>0.15</td>
<td>0.05</td>
<td>0.05</td>
<td>0.1</td>
<td>0.1</td>
<td>0.05</td>
<td>0.1</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Table 1: Emission probabilities, \( \phi \). Rows denote hidden tags \( y_j \) and columns are observations \( x_j \).

<table>
<thead>
<tr>
<th></th>
<th>+</th>
<th>−</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>\∅</td>
<td>0.1</td>
<td>0.2</td>
<td>0.7</td>
</tr>
<tr>
<td>+</td>
<td>0.3</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>−</td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>O</td>
<td>0.2</td>
<td>0.2</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Table 2: Transition probabilities, \( \theta \). Rows denote tags \( y_j \) and columns are over \( y_{j+1} \). Transition from null state represents the initial probability of a state, i.e. \( \theta_{∅,+} = π(+) \).

(a) (4 points) Write down the formula to compute the joint probability of the tag sequence \( y = \langle O, O, O, + \rangle \) and the sentence \( x = \langle \text{the bread smells delicious} \rangle \) given the above parameters.

(b) (9 points) Given a word sequence \( x = \langle \text{coffee smells bitter} \rangle \), what is the most likely tag sequence for this text using Viterbi decoding?

**Problem 2: Probabilistic Context-Free Grammars (PCFGs)**  

(a) Consider the following PCFG \((N, Σ, S, R, q)\) with nonterminals \( N = \{NP, PP, NNS, CC, IN\} \), terminals \( Σ = \{\text{cats, and, in}\} \), start symbol = NP, and the following rules and their probabilities:

<table>
<thead>
<tr>
<th></th>
<th>( q(r) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP (→) NP PP</td>
<td>0.4</td>
</tr>
<tr>
<td>NP (→) NP CC NP</td>
<td>0.4</td>
</tr>
<tr>
<td>NP (→) NNS</td>
<td>0.2</td>
</tr>
<tr>
<td>PP (→) IN NP</td>
<td>1.0</td>
</tr>
<tr>
<td>NNS (→) cats</td>
<td>1.0</td>
</tr>
<tr>
<td>CC (→) and</td>
<td>1.0</td>
</tr>
<tr>
<td>IN (→) in</td>
<td>1.0</td>
</tr>
</tbody>
</table>

(i) (3 points) For the sentence \( \text{cats and cats} \), how many valid parses (with probability > 0 under this grammar) are there? Show all valid parses.

(ii) (3 points) For the sentence \( \text{cats and cats in cats} \), how many valid parses (with probability > 0 under this grammar) are there? Show all valid parses.
(iii) (4 points) The rules involving tag-word pairs in the grammar are called the lexicon (the second half of the table). Suppose we want to use add-α smoothing to the lexicon so that all the pairs between words (cats, and, in) and part-of-speech tags (NNS, CC, IN) have non-zero probabilities. For example, in this case NNS → and and NNS → in would be added to the grammar, as would similar extra rules for CC and IN. Now, for the sentence cats and cats, how many valid parses (with probability > 0 under this grammar) are there? Show all valid parses.

(b) Consider another PCFG \( (N, \Sigma, S, R, q) \) with nonterminals \( N = \{S, VP, PP, NP, NNS, CC, Vi, IN\} \), terminals \( \Sigma = \{cats, and, slept, in\} \), start symbol \( S \), and the following rules and their probabilities:

\[
\begin{align*}
S &\rightarrow NP \ VP & r &\rightarrow 1.0 \\
VP &\rightarrow Vi \ PP & q(r) &\rightarrow 1.0 \\
PP &\rightarrow IN \ NP & &\rightarrow 1.0 \\
NP &\rightarrow NP \ CC \ NP & a &\rightarrow \\
NP &\rightarrow NNS \ PP & b &\rightarrow \\
NP &\rightarrow NNS & 1 - a - b &\rightarrow \\
NNS &\rightarrow cats & 1.0 &\rightarrow \\
CC &\rightarrow and & 1.0 &\rightarrow \\
Vi &\rightarrow slept & 1.0 &\rightarrow \\
IN &\rightarrow in & 1.0 &\rightarrow \\
\end{align*}
\]

Two variables in this PCFG \( a \) and \( b \) are not defined yet. To have a legal PCFG, we must have \( 0 \leq a, b \leq 1, a + b \leq 1 \).

(i) (2 points) If \( a = 0 \) and \( b = 0 \), what is the distribution over lengths of sentences produced by this grammar? What is its expected value?

(ii) (2 points) If \( a = 0 \) and \( b = 1 \), what is the distribution over lengths of sentences produced by this grammar? What is its expected value?

(iii) (3 points) If \( a = 0 \) and \( b = 0.5 \), what is the expected value of the sentence length produced by this grammar?

---

**Problem 3: Dependency Parsing** (15 points)

Consider the following two sentences:

1. I ate pasta with meatballs
2. I ate pasta with chopsticks

(a) (6 points) Draw the typed dependency trees for the above two sentences. Note that the dependency tree depends on the chosen dependency representations. In this problem, we are going to use the Universal Dependency relations described in (de Marneffe et al., 2014). You can read J & M 18.1 for more information. However, what you need to know for this problem are only the following dependency labels:

- `nsubj`: nominal subject.
- `dobj`: direct object.
- `case`: prepositions, postpositions and other case markers.
- `nmod`: nominal modifier.

\footnote{NNS: plural noun, CC: coordinating conjunction, IN: preposition or subordinating conjunction. See more in https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html}
Note that you probably also need the label root if you add a fake node ROOT in the transition-based system.

(b) (6 points) Describe the transition sequences that lead to the above parse trees separately, using the Arc-Standard algorithm that we learned in the class. Hint: each transition step is either SHIFT or LEFT-ARC(r), or RIGHT-ARC(r), where r is a dependency label. If there are multiple transition sequences, please use the “shortest stack” strategy (prefer LEFT-ARC over SHIFT).

(c) (3 points) As you have seen so far, the two sentences only differ in one word (meatballs vs chopsticks) but they have distinct parse trees and transition sequences. Describe what information/features are crucial for the model to make correct predictions for these two sentences.

**Programming 1: Hidden Markov Model for Named Entity Recognition** (30 points)

In the following programming problems, you are going to implement models for the Named Entity Recognition (NER) task. NER is the task to associate the words in a sentence with their proper name tags. For example, “Marie Curie” may correspond to the tag PER (person) and “Princeton University” may correspond to the tag ORG (organization). In this programming assignment, will use a total of 5 tags: PER (person), ORG (organization), LOC (location), MISC (miscellaneous), and O (non-entity). For example, the correct tagging of the sentence “Steve Jobs founded Apple with Steve Wozniak.” is ⟨PER, PER, O, ORG, O, PER, PER⟩. Note that when consecutive words constitute a named entity, such as “Steve Jobs” in the previous example, they should both be tagged as PER.

In programming problem 1, you will implement the hidden Markov model (HMM) for this task.

Link to notebook: Colab notebook.

**Programming 2: Max-Entropy Markov Model for Named Entity Recognition** (25 points)

The task is the same as the programming problem above. In programming problem 2, you will implement the MEMM model for this task.

Link to notebook: Colab notebook.