EECS 487 Introduction to NLP
HW2

Assigned date: January 23rd, 2023
Due date: February 6th, 2023

Instructions:

- Please submit all the materials in one .zip file to Canvas before the due date. Name your file hw2_<uniqname>.zip, where <uniqname> should be replaced by your unique name.
- Your .zip file should contain three files: **hw2_<uniqname>.pdf** that includes your answers to written questions and analysis questions of the programming part, **hw2.ipynb** that has the output for the programming part (please make sure that you run all cells before submission), and **hmm.py**.
- Please refer to the syllabus for information regarding the late policy.

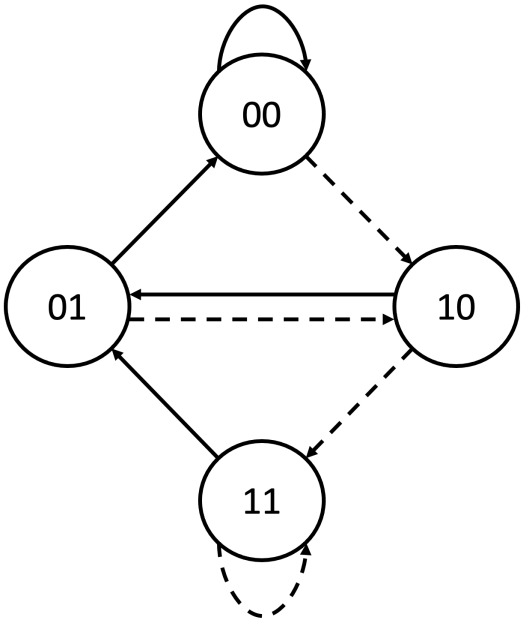
1 Written Questions

1.1 Hidden Markov Model [15 points]

Although we have studied the Viterbi decoding algorithm in the context of natural language processing, it originated as a method for correcting errors during the transmission of digital data. For instance, say there are two people named Alice and Bob who wish to communicate using radios. Alice transmits a message—a string of ones and zeros—to Bob. Due to interference, however, each bit she transmits has the potential to be "flipped" by the environment with probability p. ("Flipping" in this context refers to the phenomenon wherein Alice transmitted a 0 but Bob received a 1, or vice versa).

In 1967, Andrew Viterbi invented a method to correct these transmission errors using Hidden Markov Models and a decoding algorithm that came to be known as the Viterbi algorithm.

Through a process that is outside of the scope of this homework, Alice can take a string of bits and convert them into a sequence of bit pairs (i.e. 00, 01, 10, and 11). The transitions between these bit pairs take the form of the following Hidden Markov model:



Alice's original message can be recovered by looking at the connections between bit pairs. Solid lines represent ones, and dotted lines represent zeroes. So, for instance, the sequence $[00, 10, 11, 01]$ refers to the original message of $[0, 0, 1]$.

Using this scheme, Bob then "observes" the bit pairs transmitted by Alice (although his received bits can be flipped with probability p).

This entire process can be considered as a Hidden Markov Model with the following probability matrices for π (the initial probabilities), A (the transition probabilities), and B (the emission probabilities). Alice's sequence of bit pairs are the "hidden states", while Bob's sequence of bit pairs are the "observations" of these hidden states.

Initial Probability	
Bit Pair	Prob
00	1/4
01	1/4
10	1/4
11	1/4

Transition Probability				
Bit Pair	00	01	10	11
00	1/2	0	1/2	0
01	1/2	0	1/2	0
10	0	1/2	0	1/2
11	0	1/2	0	1/2

Emission Probability				
Bit Pair	00	01	10	11
00	$(1-p)^2$	$p(1-p)$	$p(1-p)$	p^2
01	$p(1-p)$	$(1-p)^2$	p^2	$p(1-p)$
10	$p(1-p)$	p^2	???	???
11	p^2	$p(1-p)$	???	???

1. [1 points] Calculate in terms of p the unknown probabilities in the emission probability matrix (each unknown probability is marked with three question marks).
2. [1 points] Suppose the probability of a single bit flip is $p = 0.05$. Calculate the values of the emission probability matrix B.
3. [5 points] Suppose Alice transmitted the bit string $[01, 00, 10, 11]$ and Bob received the bit string $[01, 00, 10, 10]$. Bob knows his observation must be wrong since the two observation states at the end are both 10, and 10 does not map to itself in the HMM graph. Compute the probability that Alice's original string produced Bob's message.
4. [8 points] Alice sends another message, and Bob receives the string $[00, 10, 11]$. Use the Viterbi algorithm to determine the most likely sequence of hidden states. What is the corresponding sequence of bits (not bit pairs)? Please show your work in terms of p and q (hint: the math becomes easier using p and q rather than decimals), where $q = 1 - p$. It is safe to assume that $p < 0.5$.

1.2 Logistic Regression [12 points]

We are trying use logistic regression to classify transactions as fraudulent (label 1) or non-fraudulent (label 0). We plan to use the following features:

1. Amount: the amount of the transaction
2. Time: the time of the transaction
3. Location: the location of the transaction(domestic: 0, overseas: 1)
4. Device Type: the type of device used for the transaction (laptop: 0, phone: 1, tablet: 2)
5. Account Age: the length of time the user has had their account

We obtained a dataset below.

Transaction ID	Amount	Time	Location	Device Type	Account Age	Fraudulent
1	500	9	0	0	2	No
2	800	12	0	1	3	No
3	200	15	0	2	1	No
4	50	20	0	0	4	No
5	9000	1	1	1	6	Yes

Based on the features and the dataset, you trained a logistic regression model with weights: $w = [0.001, 0.1, 10, -1, 0.2]$, $b = -2$. Answer the following questions:

1. [3 points] Suppose a transaction has the feature vector $[100, 8, 0, 3, 20]$. What is the probability that it is fraudulent?

2. [3 points] For the transaction above, what is the respective loss value (cross entropy loss) when it is fraudulent and when it is not fraudulent? Use the natural logarithm to find your answer.

3. [2 points] When applying this model to real-world situations, we observe that, whenever the Location is 1, the model always classifies transactions as fraudulent. What is the cause of this problem, and how could you prevent it from occurring in the future? Previously there was a typo in this question: we accidentally wrote 0 above when we meant to write 1.

4. [4 points] To find the optimal w and b such that it minimises this loss function, we need to first derive the general formula of the partial derivative of the loss function with respect to the weight that corresponds to the Time feature. What is the result for the feature vector $[100, 8, 0, 3, 20]$ and the label 1?

1.3 Probabilistic Context-Free Grammars [18 points]

1. i. [4 points] Consider the following grammar, how many parse trees are there for the sentence "cats and cats in cats"? Calculate their probabilities accordingly.

   ii. [4 points] Suppose we want to introduce more rules so that all the pairs between words (cats, and, in) and tags (N, CC, IN) have non-zero probabilities. For example, we will introduce "N $\rightarrow$ and" and "N $\rightarrow$ in" to the grammar, along with similar rules for CC and IN. Now, for the sentence "cats and cats", how many valid parse trees (with probability > 0 under this grammar) are there? Draw all the parse trees.

Rules			P
S	$\rightarrow$	NP	1.0
NP	$\rightarrow$	NP PP	0.3
NP	$\rightarrow$	NP CC NP	0.4
NP	$\rightarrow$	N	0.3
PP	$\rightarrow$	IN NP	1.0
N	$\rightarrow$	cats	1.0
CC	$\rightarrow$	and	1.0
IN	$\rightarrow$	in	1.0

Table 1: Grammar for question 1.3.1

2. i. [2 points] Consider the grammar below. How could we turn it into Chomsky normal form?

   ii. [8 points] Using the grammar below (without any modifications from the last question), use the probabilistic CKY algorithm to parse the sentence *Fed raises interest rates greatly.* You should clearly show the filled table by the CKY algorithm.

Rules			P
S	→	NP VP	1.0
NP	→	NP NP	0.4
NP	→	N	0.6
VP	→	V NP	0.3
VP	→	V	0.3
VP	→	VP Adv	0.4
V	→	raises	0.5
V	→	rates	0.4
V	→	interest	0.1
Adv	→	greatly	1.0
N	→	Fed	0.4
N	→	interest	0.3
N	→	rates	0.3

Table 2: Grammar for question 1.3.2

2 Programming Problems

In this part, you need to solve a programming problem about Hidden Markov Models. hw2.ipynb contains more detailed instructions for coding and serves as a "driver" for the code you will write. hmm.py is where you will fill in your answers to the coding questions as directed by hw2.ipynb.

2.1 Coding [52 points]

See hw2.ipynb for more details.

In the Jupyter notebook, cells 1.5, 1.6, and 1.11 were modified to add an "</s>" token to the end of the sample sentences. This change should not affect your underlying code. Make sure to add these tokens onto your validation and test sets within the functions hmm.search_k(), hmm.search_beam_width(), and hmm.test().

Additionally, for add-k smoothing, (1) calculate the probability distribution without smoothing, (2) add k to each element, and (3) normalizing the distribution so that it sums to 1. This should be done for the initial probabilities, the transition probabilities, and the emission probabilities. For example, to get the transition probability of (A, B), where A and B are tags, you need to add transition_k to the transition count (no matter whether the count of (A, B) is 0 or not) and then divide it by the new sum of all transition counts that start with A.

2.2 Analysis Questions [3 points]

After you have completed 2.1.1, come back and answer these reflection questions.

1. (2 points) Did a larger beam width result in higher decoding accuracy? Comment on potential differences between large beam width and narrow beam width (especially regarding time/space complexity and accuracy).

2. (1 point) In the last homework, you were instructed to make all text lowercase, whereas in this homework, you were instructed to do the opposite. Why might case be a relevant signal for named entity recognition?