EECS595: Natural Language Processing
Homework 1 Assignment

Due date: September 27 11:59pm, extended to **September 30, 11:59**

September 28, 2019

Updated on 9/28: (1) the due date is updated; (2) add a readme to the submission (see the submission section)

1 Description

In this programming assignment, you will use Python programming language to implement a first-order Hidden Markov Model (HMM) for part-of-speech tagging using the Viterbi algorithm. You are provided with the following three data files.

- **wsj1-18.train**. This file contains the training data where each word is annotated with a POS tag. You will train the HMM model using this training data.
- **wsj19-21.test**. This file contains the testing data. You will apply the trained model to predict POS tags for each word.
- **wsj19-21.truth**. This file contains ground-truth tags for the testing data. You will compare the results returned by the HMM model with the ground-truth tags to evaluate the model. You need to report the accuracy of your model.

The data files can be downloaded from the assignment directory on CANVAS. Please do not distribute the data as this is part of the Penn Treebank which is licensed by Linguistic Data Consortium. The format of data is straightforward. Each word is followed by a tag. Punctuation marks are considered as words.

Notes:

- The path probabilities during the calculation will be small. So you should use log probabilities and store these in memory. It is important to do this to prevent underflow. Essentially, suppose $P_1 = P_2 \times P_3$, You can use log as following: $\log(P_1) = \log(P_2) + \log(P_3)$. When $P = 0$, how do you represent $\log(P)$? You could use any trick (e.g., set to some convention, or simply -Inf) to make the program work. You can choose to use natural logarithm or logarithm based on 2, etc. Natural logarithm is a bit faster.
- In the testing data: wsj19-21.testing, infrequent words have already been replaced by *UNKA*. You need to find some strategy to handle unknown words. We have discussed it in the class (e.g., words occurring less than 3 times in training data should be mapped to the word token *UNKA*).
- You should experiment with different smoothing techniques when estimating parameters. Submit the model that achieves the best performance on the testing data.

1.1 Implementation

You will need to provide a python program “pos.py” and a pickled pre-trained model ”model.pyc” (https://docs.python.org/3/library/pickle.html).

Your python program must be able to do the following: (1) must import model.pyc (i.e., the trained model) from the current directory, (2) must take two files as the arguments where the first file contains the test data and the second file contains the ground truth tags, and (3) output the accuracy on the test data.

For example, your program should run like the following:

```
$ python3 pos.py wsj19-21.test wsj19-21.truth
```

Will outpout ”The accuracy is 93.7%”.

Your program will be evaluated against additional held-out test data (which is not available to you).

2 Submission

In the CANVAS system, submit the following

- Your python program implementing Viterbi algorith: pos.py
- The trained model file: model.pyc
- A readme file to give a short explanation on what configuration you used to train the model you submitted. E.g., whether you use any type of smoothing for the transition probabilities and/or for the emission probabilities; how you handle the ”unknown” words.