Viterbi Algorithm

During the process of assignment, I found few challenges and found solutions to tackle the challenge.

Structure of the dictionary:
Initial task of reading the words and tags and filling them in a python data structure was a challenge. I had lot of options to choose from. The possible design option was to utilize a list of bag of words and tags and have another list containing the list of their count. I did not choose this option, because whenever I need a word-tag count, it would lead to unnecessary computation and lead to extra step during each step of computation.
Solution: I chose to use a dictionary of words and each word contains another dictionary that has set of tags that has its counts of occurrence. So whenever observational probability must be found out, I would just need to hash into dictionary to find the count.

Unknown Words:
To find the unknown words, I had to pass through the dictionary to find the words that had a low occurrence. For this I introduced a threshold value, below which the word is defined UNK. So for this I made sure, if a word has more than one tag and in one of the tag it has occurrence more than threshold value, that tag will persist, while the other tag which is below a certain threshold becomes UNK and is added to the UNK entry.

Calculating Probability in the beginning:
I saw an opportunity to either calculate the probabilities at the beginning or whenever it was required. Calling the probability function when it was required made the code more modular and it made it easy to find it during the calculations of observational and transitional probabilities.
Solution: Modularity of code made it easier.

Start tags and end tags:
For the purpose of some clarity I used both start and stop tags. It helped a lot during the Viterbi algorithm to exactly identify the start and end states. The burden was just to remove it before submitting the results.

Sentences of Viterbi testing:
So during the initial stages of coding I used small test cases to verify the correctness of the code. As I started to develop into stage of dev set testing, I was posed with a question of running Viterbi over thousands of columns of words. So I segregated them into sentences and output of each sentence was appended until end of the document.

K-Fold testing:
For testing purposes I used K-fold as my testing algorithm. Using 10-fold I was able to achieve accuracy of 95% and all test cases did over 90%. Intuitively, the model performed well and I believe this would validate that it performed well in test set also.

Smoothing:

To smooth the transitional probability, I used Laplacian smoothing. It was just add one to all the tags pairs that were not present in the tag-pair dictionary. Then I had to divide it with the tag added with its count of tags.

Optimal Threshold for UNK:
To find the optimal threshold, using K-Fold I ran over various test cases and found both threshold value equal to 2,3 performed marginally better than having a threshold 1. So this will make the system resistant to face the unknown words.