Python script for designing the POS tagging Model used NumPy module and Counter from collections module

1. For the training data provided, since there are multiple occurrences of every sentence, I shuffled the data and split into 80/20 ratio for training/dev set.
2. From the training data set, extracted the words which had occurred more than once and added them to the list of word vocabulary and extracted the tags and added them to the tag vocabulary list (word_vocabulary, tag_Vocabulary).
3. I considered the indexing of words as per the word_vocabulary and indexing of tags as per the tag_Vocabulary lists when designing the transition and emission matrices.
4. Designed most-frequent tag system that will assign to each input word the tag that it was most frequently assigned to in the training data and for the unknown words the most frequent tag from the whole training data set (Accuracy was 90.5%)

Viterbi algorithm implementation:

5. I implemented the same algorithm from Draft 9 (Ref 4)
6. Unknown words handling:
    a. Programmatically, for the unknown words , assigned emission value as the maximum entry value in the emission matrix
7. Applied Laplace smoothing on the Transition matrix. Firstly the transition matrix is initialized to 1 and then added the bigram count of tags to each entry and subsequently, divided each row element by the sum of that particular row for the probability.
8. Word and tag vocabulary are the lists and Transition, Emission and VIterbi matrices are numpy arrays of multidimensional.
9. Designed the model in such a way that each tag has equal probability of occurring at the first observation (Equal start probability for each tag)
10. Each entry in a Viterbi matrix stores the maximum probability of particular tag being assigned to that observation word considering all the previous observations and the tag probabilities.
11. Back_pointer matrix has the same size as viterbi matrix and keeps track of the index of the previous tag resulting in the highest probability for each tag at that observation stage.
12. With indices I extracted the tags and stored the output in a new file including the position, word, tag in (position"\t"word"\t"tag) format
13. I think that my model will predict the most accurate tags because, on training the model with the 80% of whole training data set the accuracy (92.6%) is high when compared to the baseline system's accuracy (90.5%).

As I am new to python language, references 1, 3 helped me in solving the assignment

References:

1. **https://docs.scipy.org/doc/numpy-dev/user/quickstart.html**
2. **https://web.stanford.edu/~jurafsky/slp3/10.pdf**
3. **https://docs.python.org/3/library/collections.html**
4. **https://web.stanford.edu/~jurafsky/slp3/9.pdf**