CSCI544: Homework Assignment 3

1. Task 1: Vocabulary Creation
    - To prepare a count of the frequency of words, I've used a Counter variable.
    - I've used a threshold of <2, so any word which has a frequency of just 1 is replaced with the <unk> tag. The unk tag is not just limited to one but there are 13 <unk> tags each in the form "<unk_(POS Tag of unknown word)>". So totally there are around 17401 words tagged as <unk>.
    - The size of the vocabulary is 21170 unique words.
    - The words and the tags have been converted to lowercase for all purposes from the beginning.
2. Task 2: Model Learning
    - I've added a full stop at the start of the first word of the first sentence. So the initial states probabilities is now in the form transition_proabilites["."][<POS_TAG>].
    - Made a dictionary of the state_transition counts, state_counts and the emission transition counts and used a ChainMap to create dictionaries for Transition Probabilities and Emission Probabilities.
    - There are 1378 transition probabilities and 29743 emission probabilities in the dictionary
3. Task 3: Greedy Algorithm
    - In the Greedy Algorithm, the POS tag which is picked is the argument which has the maximum value of transition_probabiltiy*emission_probability
    - In the event that probability is zero because of a case where the transition or the emission does not exist, I've taken the max value of the previous transition_probaility and used that argument as the POS tag for the current word instead.
    - I was able to achieve an accuracy of 93.02% using this approach and the predicted POS tags of the test data is in the greedy.out file
4. Task 4: Viterbi Algorithm
    - In the Viterbi Algorithm, we have to create the trellis diagram using a Dynamic Programming table which is built from the bottom up and then we track the most probable POS tags from the top down after generating the entire table.
    - At each state, we make sure we pick the POS tag which gives us the maximum probability from all incoming tags and we make sure we remember which state gives us this tag through an entry in the dictionary.
    - In the event all the states for a given word have zero probability(because the product of emission_proability*transition_probability is zero for all states and no max value could be found) then I just took the max of the previous states probabilities and assigned it to all the current states.

- After creating the Trellis Table, all I had to do was start from the last observed state and backtrack my way back to the first word to get all the probable POS Tags for a given sentence.
- I was able to achieve an accuracy of 94.4% on the dev data using this approach and the predicted POS tags for the test data is in the viterbi.out file.