Per the CSC384 A4 handout, during the training phase, I need to learn three probability tables:

- The initial probabilities over the POS tags (how likely each POS tag appears at the beginning of a sentence)
- The transition probabilities from one POS tag to another.
- The emission probabilities from each POS tag to each observed word.

Upon preprocessing the training data, I create a dictionary of the total number of unique parts of speech tags that are seen. For the initial probability table, I create a numpy array of the length of unique parts of speech tags. I go through the numpy array and assign

The transition probability is the likelihood of a specific sequence occurring in order, such as the possibility that a noun will be followed by a verb, or a noun will be followed by an adjective, etc. For the transition matrix, I initialize a square numpy array of the length of unique parts of speech tags by itself. I go through the distinct parts of speech tags minus one and assign a probability based on the occurrence of a specific coming after the next. This can be interpreted as the probability of the following tag given the first tag. This is based on counting the occurrences of the tags in the training data. Finally, I normalize this matrix.

The emission probability is the observation likelihood. It is the conditional probability of observing a word from a given tag. For the emission matrix, I initialize a numpy array of the length of the unique parts of speech by the unique words that appear in the training file. I go through the total word list from the training file and find the probability of the word given the tag. Finally, I normalize this matrix.

Finally, after I have these three probabilities, I can apply Viterbi algorithm to it. The goal is to find the most likely path through the graph. The likelihood of a path is the product of the transition probabilities along the path, and the probabilities of the observations at each state. We first find the probability for time step 0 and then recursively find the probabilities for time step 1 to the length of sequence of observations minus 1.

Some cases to consider are if a word shows up on the test file that doesn't show up in the training file. The way I go about resolving this is by using a very small epsilon value. Currently, for the transition probability in this case, it would give me 0 since it's unseen data and thus not tag the word correctly. However, if I manually assign that case to a small number instead of 0, say for example 0.0001, then the possibility of an incorrect tag or error is mitigated.