Practical 1: POS tagging with HMMs

Student ID: 180029539

Common Implementation for all the algorithms:

First, I loaded the brown corpus with universal tagset. Then split the corpus into two data sets of training and testing sentences. Then, I extracted all the tags and words into two lists. Then I started training the HMM. I made two methods which calculate the transition and emission probabilities. Now, I have a trained HMM. Then, the way of testing is different for each algorithm. Now, let's see the implementation differences of all the different algorithms.

Description of the Implementation of Viterbi Algorithm:

For the Viterbi, I have taken the testing sentences one by one and created a 2-dimensional array with a number of rows equal to the number of tags and number of columns equal to the number of words in the sentence. Then, I filled the array using emission and transition probabilities from the training part. Then, I travelled back through the matrix to identify the best tags for each word in the sentence. Finally, I compare the predicted tags with the actual tags and get accuracy.

Description of the Implementation of Eager Algorithm & Beam Width:

For the implementation of this algorithm, I retained the same structure as Viterbi but made some minor additions. As soon as you finish filling a column, I delete the (number of unique tags - beam width) elements with max value. So for example, if I have 10 tags and my beam length is 5. I keep the 5 largest elements in that column and replace the smaller ones with zero.

Description of the Implementation of the Forward-Backward Algorithm:

For the forward and backward algorithm, I retain the array structure of the Viterbi algorithm and use different formulae to fill in the matrix in a forward direction with the forward probabilities. Then, I create another array to traverse the columns in the backward direction and fill in with the backward probabilities. Then, I multiplied forward and backward probabilities to get the probability of a word occurring with the given tag.

Comparison of the three algorithms:

- The Viterbi algorithm runs through all the tags and all the columns before making a decision. Due to this, the execution will be longer than the general algorithm with a variable beam width.
- The forward and backward algorithm has to calculate the probabilities in both directions. Due to this, it will be much slower and occupies more space than the first two algorithms.
- Now, let's look at the accuracies of each algorithm.

Algorithm	**Beam width (only for General algorithm)**	**Accuracy (approximation)**
Viterbi	(number of tags in tagset)	94%
General Algorithm	K=1	93%
General Algorithm	K=2	95%
General Algorithm	K=7	94%
General Algorithm	K=(number of tags in tagset)-1	94%
General Algorithm(Viterbi)	K=(number of tags in tagset)	94%
Forward-Backward Algorithm	N/A	96%

`

- In, the above table accuracy means the number of correctly classified tags divided by the total tags in all the test sentences.
- As we can see from the table the least accurate is the algorithm using K=1, as we are not considering that the value is only dependent on the previous word, unlike the Viterbi where we wait till the end of processing the last word to assess the probabilities.
- But, forward-backwards supersedes the Viterbi, due to the fact that not only consider the previous words but also the following words to assess the probabilities.

Extensions:

- I have implemented the variable beam length and found that the accuracy even though should increase varies with each K value
- I have also tested with different smoothing techniques and found out that Wittenbell is the most accurate as it allocates uniform probability mass to even the occurrences that never happened.

Smoothing Technique	**Accuracy**
MLEProbDist	44
Laplace	90
ELEProbDist	91