BBM497: Introduction to Natural Language Processing Lab. **(Due: 20/04/2020)**

Submission Assignment #2

Instructor: Necva BOLUCU *Name:* Mehmet Emin TUNCER*, Netid:* 21591022

1 Reading Data

I read the train file with the dataset () function, and I returned a list with each paragraph forming a sentence. I kept each sentence as a tuple to be a word and tag (word, tag). I did the same to extract the sentences in the test.txt file using the dataset () function.

2 Creating HMM model

I have used a class structure for HMM model. In the HMM () function, I have created count dictionaries suitable for tokens for 3 different possibilities using the read sentences, such as initial prob, transition prob, emission prob. I created probability dictionaries of the HMM model by using dictionaries in which counts of tokens are kept.

3 Viterbi Algorithm

For the Viterbi Algorithm, I created and used viterbi() funciton.

3.1 Creating Matrices

In this section, I used the matrix for each word tag prediction. I created matrices for each sentence with creatematrix () function separately. I created the number of matrices according to the number of words in the sentence and the number of rows according to all tag numbers.

3.2 Predicting the tags

I have used three probability dictionarties of HMM model for predicting the tags. I used the initial and emission possibilities for the tag prediction of the first word. I converted results to log probabilities. I added them to the first columns of the matrices. Then, for each other row and column, I calculated the probability of emission and transition for each tag, summed it with the previous probability and added the highest value to the matrix.

3.3 Handling the Unknown Words and Situations

I did not use smoothing when calculating probabilities. Because the goal was to choose the path with higher probabilities, instead of smoothing, I put ”-20” value for non-probability values. Because when the smoothing was already done, the probability that came was very small compared to the others, so I thought it wasn't necessary.

3.4 Finding the Tags

While finding the tags of the words, I put it in a list with the tag corresponding to the highest value in each column in the matrix.

4 Accuracy and Conclusion

I compared the predicted tags and the actual tags of the words, with using ACC() function, and the correct guess percentage came to about 84.6 percent. I think that is enough result. But if I did smoothing, maybe I could get higher results.