ASSIGNMENT -3

MT16019

Algorithm Implemented:

Training:

- Read training corpus line by line.
- Keep track of count of each POS tag in dictionary named context, count of each unique 'tag' followed by 'word' in dictionary emit and count of each tag followed by another tag in dictionary transition.
- Keep track of all unique words of corpus.
- Find the count of transition from last tag followed by end of sentence (</s>).

Testing: Forward algorithm

- Create two dictionaries "best_score" to keep track of best scores to go from one node to other and "best_edge" to keep track of best path from one node to other.
- Initialize best_score['0 <s>'] =0 and best_edge['0 <s>']=null.
- Compute best scores and best edge to go to current state from all possible previous states(all possible POS).
- Best path to a node is calculated using **lowest negative log probability.**
- Smoothing is applied for unknown words in emission probability.

Testing: Backward Algorithm

- Back track the best_edge dictionary to find best paths starting from position of last word followed by </s>.
- Extract tags from edges and store it in a list.
- Reverse the tags.

Choices Made and Assessment of Corpus:

- Read the file completely and converted it into list of all sentences.
- Stored every count in dictionaries.
- Checked the format of training and testing file and processed it accordingly (that is removed \n and \t).
- Created a vocabulary.

I think it works well as:

- All possible POS tags are available in training corpora, all transition probabilities are known beforehand.
- For unknown words smoothing is done using the formula:

$$P_E(x_i|y_i) = \lambda\ P_{ML}(x_i|y_i) + (1-\lambda)\ 1/N$$

- Lowest negative log likelihood gives the best possible path from one node to other, so it works well to predict best next possible tag while testing.