VITERBI ALGORITHM

Introduction:

This assignment computes one of the sequence labeling tasks – Part Of Speech (POS) Tagging. For a given sentence, the program predicts the part-of-speech of every word in the sentence, considering the current context. Viterbi algorithm is used to determine best tag sequence for each input sentence. This algorithm uses bigram and lexical probabilities computed from already labelled corpus.

Logic:

Step 1: Generate Labelled Corpus

1) After browsing through internet, I was able to find already labelled corpus. The tags used are from the Penn Treebank Tagset.

Step 2: Compute Bigram and Lexical Probabilities from labelled corpus

1) The program creates two dictionaries transmission_prob and emission_prob to store the bigram and lexical probabilites resp.

2) 'transmission_prob' dictionary contains a tuple (pos_1, pos_2) which stores the probability of occurance of one part-of-speech given another part-of-speech. This computation takes place as follows -
- While traversing the training corpus word by word, if tag1 is followed by tag2, the program checks if the dictionary contains the key (tag1, tag2). If it does, increment the value by 1. Else, add the key (tag1, tag2) in dictionary with value set to 1.
- After the entire training corpus is traversed, replace the counts in the dictionary with the probability using the formula – transmission_prob[(tag1, tag2)] = count(tag1, tag2)/count(tag2)

3) 'emission_prob' dictionary contains a tuple (word, pos) which stores the probability of occurance of word given pos as part-of-speech. This computation takes place similar to computing transmission_prob.
- emission_prob[(word, pos)] = count(word, pos)/count(pos)

4) The program simultaneously counts the occurance of each tag in the entire labelled_corpus and stores in the dictionary 'pos_count'.

Step 3: Find the best tag sequence for each sentence using the bigram and lexical probabilites computed above.

1) The program takes each sentence from test set and computes the best tag sequence for each word using Viterbi Algorithm.

2) It forms a matrix of dimension (N+2, T) where N is the total number of tags in Penn Treebank Tagset and T is the total number of words / observations which are to be tagged.

3) This matrix is filled using the below formula =>

matrix[i][j] = max from k = 1 to N (matrix[k][j – 1] * transmission_prob[(curr_tag, prev_tag)] * emission_prob[(obs, prev_tag)])

where ‘i’ loops through all the tags/states, ‘j’ loops through all observations and ‘k’ loops through all tags.

The argmax from k = 1 to N (matrix[k][j-1] * transmission_prob[(curr_tag, prev_tag)]) gives the tag label (equal to prev_tag) for the previous observation and this tag acronym is then appended to the word using underscore (eg: the_DT) and stored in the list.

4) This list is then written into the output file ‘output.txt’ by joining it using space character.

Program Input and Output:

1) Training_set (Labelled Corpus)
2) Test_set (txt file with one sentence on each line)
3) Output file (txt file)

Example:
Input: They make the arguement in the letters to the agency
Output: They_PRP make_VBP the_DT arguement_NN in_IN the_DT letters_NN to_TO the_DT agency_NN

Resources:

1) Speech and Language Processing (2nd Edition) for Viterbi Algorithm
2) Python Documentation of numpy