Assignment IV: Implementation of Viterbi Algorithm

Introduction:

Viterbi algorithm is used to predict part-of-speech tags for a given sentence using probabilistic approach. Viterbi algorithm uses training corpus to calculate transition and emission probabilities of POS tags. I have used a training dataset containing around 40000 sentences, tagged using Penn-Treebank tagger available in Python's NLTK library.

Approach:

This implementation of Viterbi algorithm takes name of training file and name of testing file as command line arguments. Python's argparse module checks validity of command line arguments and handles most of the errors.

Training work of Viterbi is done by 'trainModel' function which takes name of the training file as a parameter. It iterates over every line in the file and calculates tag frequency, emission count and transition count of each tag and stores them in respective dictionaries. It also stores these dictionaries in binary files for future use. Python's pickle library provides functions to load and dump objects from/to binary files.

If pickled files already exist on the system, the program skips the training part and directly load dictionaries using pickle. These dictionaries are used to calculate emission and transition probabilities during Viterbi implementation.

Once the dictionaries are loaded, the program attempts to open the test file. If the test file is successfully opened, it reads each line from the file and passes it to viterbi function. 'Viterbi' function takes current line and previously obtained dictionaries as parameters.
It returns list of tags corresponding to each individual word in the given line. It uses helper functions to calculate emission and transition probabilities as well as to obtain list of words in the given line. The list of words contain each word and punctuation mark separated and in order in which it appears in the given line.

The program writes output to 'output.txt' in 'word_tag' format. It also obtains tag sequence for a given line using POS tagger available in NLTK library. It stores tag sequence obtained from Viterbi implementation and POS tagger in NLTK in respective lists. Finally it calculates

precision, recall, f-measure and support using Scikit Learn's 'precision_recall_fscore_support' function. Calculated statistics is stored in the file for the reference.

Discussions:

I have discussed my approach with Vaibhav Gawali and Anurag Kolhe.

References:

1. Official Documentation of argparse module
2. Official Documentation of nltk module
3. Official Documentation of Scikit Learn module