CS5012 Language and Computation

Practical 1 – Report POS tagging with HMM

Introduction

The aim of practical is to POS tag words of the sentences using HMM algorithms, Viterbi, Beam Search, Forward and backward probability.

To use these algorithms, it was required to train the model by getting the smoothed emission and transition probabilities using tagged Brown corpus and NLTKs Witten-Bell smoothing function.

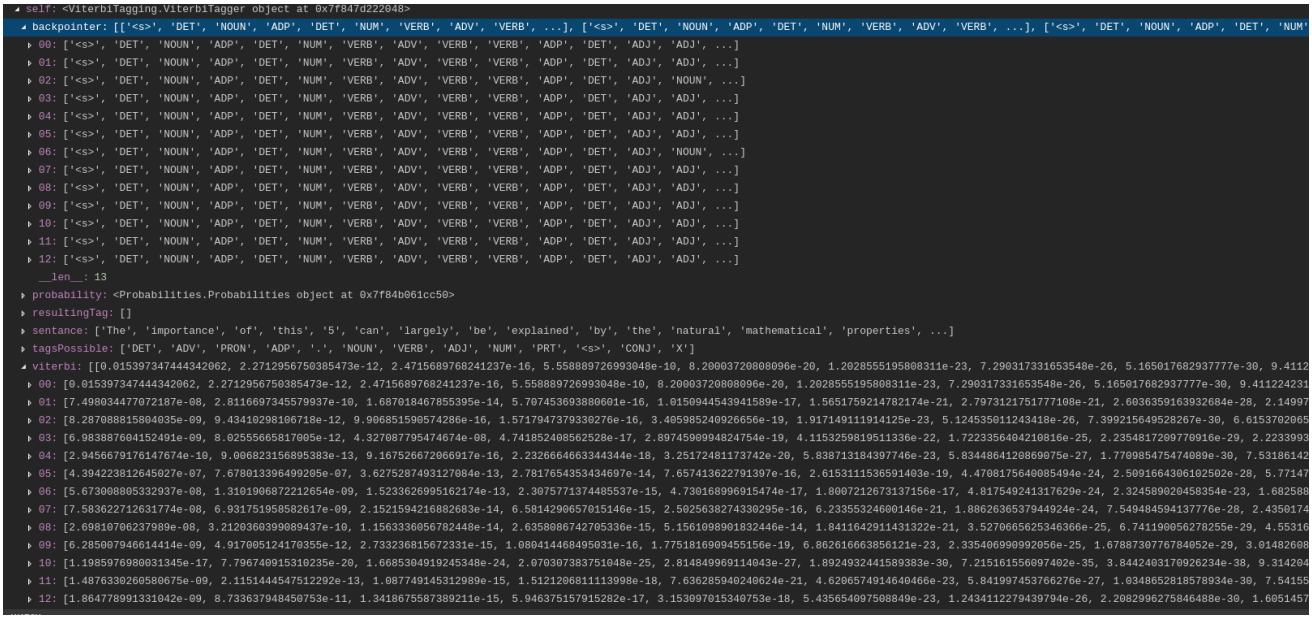
For this practical OOP approach is used using Python 3.7 (Worked in lab machines). There are 3 class python files and 1 main file which runs them.

Training

This part is done in the Probabilities.py file, which contains class Probabilities. Constructor saves sentences emitted from the corpus. It also contains uniqueTags list which contains all tags which may appear in corpus. Emission and transition probabilities are created and stored in dictionary variable, so they can be accessed via getEmissionProbabilities and getTransitionProbabilities. Bigrams for the transition probabilities are created manually, without nltks ngrams method.

Algorithm 1

Viterbi algorithm is implemented using 2d list which stored all probabilities. Another list backpointer is used to store the tags where these probabilities are coming from.



Using recursion, the tags are taken from the backpointer list. By training first 10000 words, and next 500 words for testing viterbi gave 0.9500389644125032 accuracy.

Algorithm 2

Used the same approach. However, Beam search has additional functions called keepMaximum. It takes K maximum from each column and makes other values –1. While calculating next stage probabilities –1 probabilities are ignored. The reason why 0 is not used, is underflow. It would not be wise to make artificial 0s, which could lead to the fake underflows.

The accuracy of beam search is:

beam K=(1) Accuracy: 0.9332409732444368

beam K=(2) Accuracy: 0.9489133258290761

For K = 13 it gives the same probability as Viterbi. As it keeps all values from the probability table, which makes it operate like viterbi algorithm

Algorithm 3

Algorithm 3, forward and backward is implemented. However, its accuracy is 0.9279591306606633. Forward and backward probability tables are calculated and the in the third stage these values were used and multiplied by emission and transition probabilities of each position in table. The difference of forward probability table here, was summation instead of the maximisation.

Run

To run the application, the main.py file should be run. Using

python3 Main.py