PARTS OF SPEECH IN THE UNIVERSAL DEPENDENCIES TREEBANKS

This part consists of one python file that runs the Viterbi algorithm that assigns part of speech tags to the test set of sentences. It is a dynamic programming algorithm based on hidden Markov models that finds the path of tags with the highest probability.

First, the program loads particular corpora (determined by variable lang) and splits it into the test and training set according to a number of sentences. Then there are multiple functions that prepare our data for training and also run the Viterbi algorithm. The function get_words_and_tags() creates a tuple of words and tags. To create bigrams of tags from the training set bigrams_tags_begend() is used.

Then there are functions to calculate emission probability. It contains a function that calculates it through the counts and also other function that is using smoothing instead. The same is done for the transition probability. In the algorithms below, smoothing is used to avoid zero probabilities for events that were not witnessed in the training corpus.

Viterbi algorithm is implemented by Viterbi() function. In the beginning, it prepares emission and transition probabilities trained on the training set. Then it applies the Viterbi algorithm to each sentence of the test set. It initialises the Viterbi matrix where rows are tags and columns are particular words. Each cell of the matrix then contains two elements where the first one is the maximum probability of that cell and the second one is an index of the tag of the previous word (back pointer). The back pointer is used to get the path of tags with the highest probability.

The first column of the matrix is calculated only by emission probability*transition probability. Where transition probability is given between starting tag <s> and each tag in the row, it then continues and calculates the probability of each word (column) for each tag (row). For each cell, it picks only the maximal probability that is counted by multiplying transition_probability * emission_probability * preceding_probability. The preceding probability is given by the particular path it took before reaching the cell. After it goes through all of the words, the algorithm makes the last terminating step where it calculates the probability of reaching the end of the sentence </s> and picks the highest probability. Then it looks at its back pointer (index of the tag of the last word) and completes the backtracking. Lastly, the list of backtracked tags is reversed and added to the predicted tags. The whole process is repeated until there are no test sentences. In the end, it returns the tuple of words and their predicted tags.

Additionally, the program also contains a greedy algorithm. It starts at the beginning of the sentence with starting tag <s>. Each step always picks the tag that has the highest probability (calculated by emission probability*transition probability) without considering previous tags and words.

The Viterbi algorithm trained and tested on the English corpora achieves the accuracy of 89.56% and greedy algorithm 87.33%.

Comparison of different languages

Language	Unique tags	Number of words	Unique words (forms)	Unique lemmas	Forms/Lemmas ratio	Viterbi accuracy	Greedy accuracy
English	17	193356	19247	14527	1.33	90.93 %	88.71%
Spanish	17	321280	41627	30041	1.38	93.11%	91.6%
Dutch	16	182339	26186	20928	1.25	89.12%	87.24%
Czech	16	220865	40043	18277	2.19	90.83%	88.32%
German	16	226348	44178	36292	1.22	91.08%	87.55%
Persian	16	208121	21704	13967	1.55	94.21%	93.59%