Problem Statement

HMMs and Viterbi algorithm for POS tagging

You have learnt to build your own HMM-based POS tagger and implement the Viterbi algorithm using the Penn Treebank training corpus. The vanilla Viterbi algorithm we had written had resulted in ~87% accuracy. The approx. 13% loss of accuracy was majorly due to the fact that when the algorithm encountered an unknown word (i.e. not present in the training set, such as 'Twitter'), it assigned an incorrect tag arbitrarily. This is because, for unknown words, the emission probabilities for all candidate tags are 0, so the algorithm arbitrarily chooses (the first) tag.

In this assignment, you need to modify the Viterbi algorithm to solve the problem of unknown words using **at least two techniques**. Though there could be multiple ways to solve this problem, you may use the following hints:

- Which tag class do you think most unknown words belong to? Can you identify rules (e.g. based on morphological cues) that can be used to tag unknown words? You may define separate python functions to exploit these rules so that they work in tandem with the original Viterbi algorithm.
- Why does the Viterbi algorithm choose a random tag on encountering an unknown word? Can you modify the Viterbi algorithm so that it considers only one of the transition or emission probabilities for unknown words?

Sample test sentences Download

Data

For this assignment, you'll use the Treebank dataset of NLTK with the 'universal' tagset. The Universal tagset of NLTK comprises only 12 coarse tag classes as follows: Verb, Noun, Pronouns, Adjectives, Adverbs, Adpositions, Conjunctions, Determiners, Cardinal Numbers, Particles, Other/ Foreign words, Punctuations.

Note that using only 12 coarse classes (compared to the 46 fine classes such as NNP, VBD etc.) will make the Viterbi algorithm faster as well.

Goals

You can split the Treebank dataset into train and validation sets. Please **use a sample size of 95:5** for training: validation sets, i.e. keep the validation size small, else the algorithm will need a very high amount of runtime.

You need to accomplish the following in this assignment:

implement these techniques, you can either write separate functions and call them from the main Viterbi algorithm, or modify the Viterbi algorithm, or both.

- Compare the tagging accuracy after making these modifications with the vanilla Viterbi algorithm.
- List down at least three cases from the sample test file (i.e. unknown word-tag pairs) which were incorrectly tagged by the original Viterbi POS tagger and got corrected after your modifications.

You can download the Jupyter notebook from below. The dataset has been read and stored in the variable 'nltk_data'. Use this notebook as your starter code.

Viterbi Modifications Download

Submission: You need to submit final Jupyter notebook with the required modifications for this assignment. Also, please show the two modification techniques separately (as two separate functions) to make it easier to evaluate.