**NLP – Semester Project Report**

**Himanish Kopalle**

**11114178**

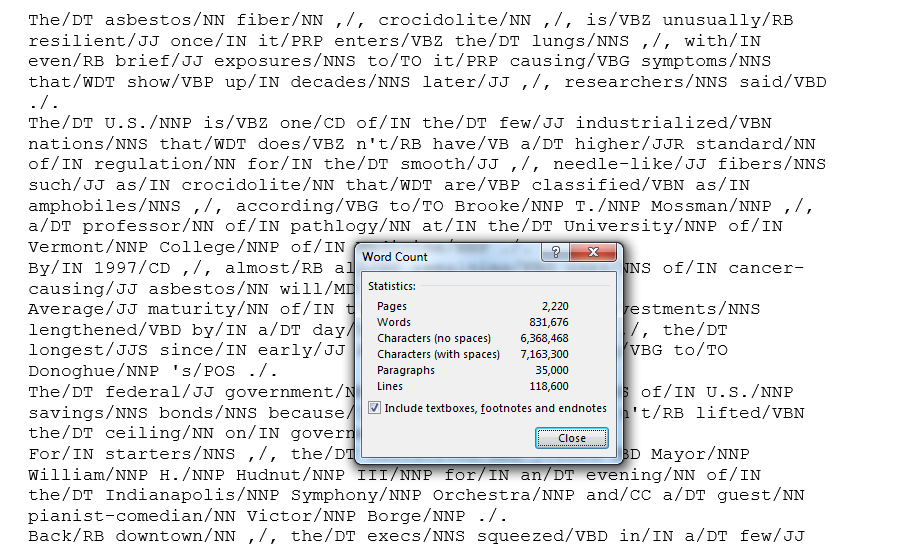
**Proposed model**:

Though we have many POS taggers which implement Viterbi algorithm. But the challenge is to get a better accuracy and assigning accurate tag to a word. I propose a model which will implement Viterbi algorithm over trigrams with the help of a smoothing technique.

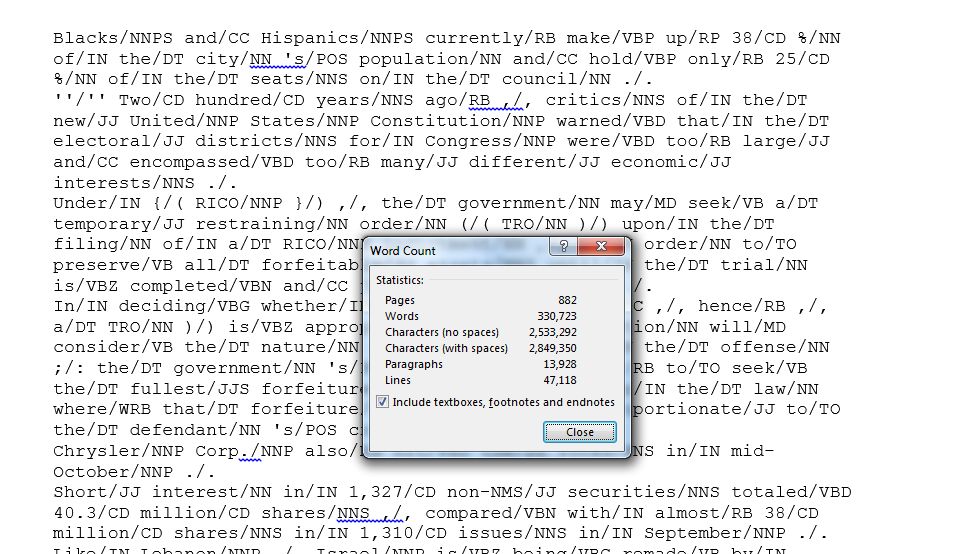
**Dataset used**:

The dataset used in the experiment consists of a corpora of 35000 sentences (paragraphs) for training data file and 13928 sentences for test data file. The corpora has a clear distribution of each word in the sentences tagged with a POS tagger.

Sample set of training data file:



Sample set of test data file:



**Language used**: The project is built in python and no other external libraries are used for the experiment.

**Experiment process**:

The project is to build a language model on trigrams which gives a better accuracy when compared with that of Bigrams.

In the process, the language model uses five dictionaries for unigrams, bigrams, trigrams, to count the word-tag and likelihood probability i.e., word-tag probability (uniTagCount, biTagCount, triTagCount, wordTagCount, wordTagProb). The language model results the trigrams and their count, likelihood probabilities, tagset [] - list the tags of the training data, wordTypes [] - set of all unique words.

The prior probability is calculated by smoothing the counts using Laplace smoothing. Two matrices, tag x tag and word x tag are formed for the HMM model.

The predict\_tags () functions takes the sentences from the corpus and the model as parameters. Four lists are used for prior probabilities, likelihood probabilities, and set of tags and list of all unique words.

The function holds two Viterbi models, one for calculating Viterbi values from the prior and likelihood matrices, and second for holding the maximum probability value for the Viterbi values of word given tag.

For predicting the tags for unknown words, we are giving a uniform distributed value using Naïve Bayes technique.

**Viterbi algorithm functionality**:

In the project, Viterbi has been implemented by using two matrices named Viterbi\_value and Viterbi\_loc. A predefined python function utils.py is used in the experiment for gathering the words and their tags from the dataset.

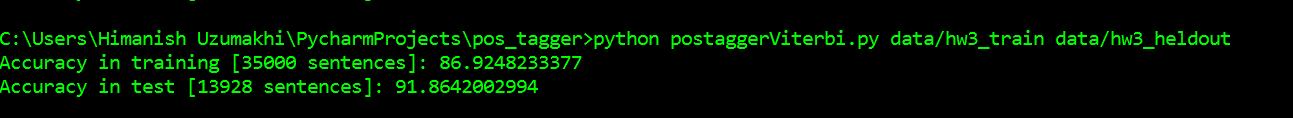
* First matrix have the words of a sentence as rows and tags in tagset as columns. This makes the size of the first matrix to be the length (sentence) X length (tagset). First matrix stores the Viterbi values calculated according to the HMM Viterbi algorithm.
* Second matrix has the same size of the first matrix where each element consists a list of size two; where first element is Row number, second element is column number. So this list is used to find location of the element in the first matrix with which current Viterbi value got maximized.
* The second matrix is used for backtracking. First, we will find the maximum value in last row and assign that tag to last word of a sentence. Later based on second matrix the tags for all the words is assigned from last word of a sentence to first word of a sentence.

**Comparisons**:

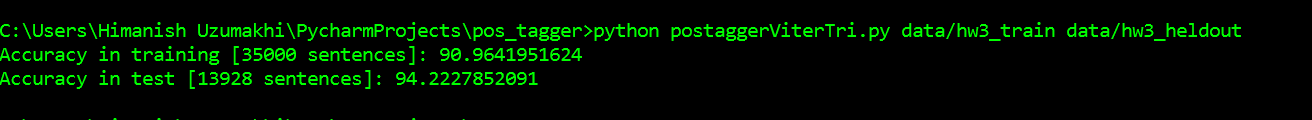
The proposed trigrams HMM model gives a better accuracy over the previous bigrams model with 4.4% improvement on training data and 3.1% improvement on test data.

**Results**:

Accuracies for Bigram model:



Accuracies for Trigram model:



**Conclusion**:

With the given corpora and the results of the proposed model, it can be concluded that the accuracy for a language model would increase with the increase in the n-grams. Though the project uses uniform distribution for identifying the unseen words, the experiments with other techniques such as applying morphological rules is also giving a close accuracy results.

**Future enhancement**:

The future work would clearly be a step towards experimenting the language modal and Viterbi algorithm using other smoothing techniques such as Good Turing, Kartz Backoff technique and Interpolation technique. By comparing the results of these smoothing techniques on both bi grams and trigrams we can define which one is a better solution.

**References**:

* <http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/hmms.pdf>
* <https://web.stanford.edu/~jurafsky/slp3/9.pdf>
* <http://files.asimihsan.com/courses/nlp-coursera-2013/notes/nlp.html#the-viterbi-algorithm-for-hmms>