**Hate Speech Classification Task Using Word2Vec + NN, RNN & Transformer (BERT)**

**Name: Varala Nandu Swapnik**

**Roll no: 21MT10061**

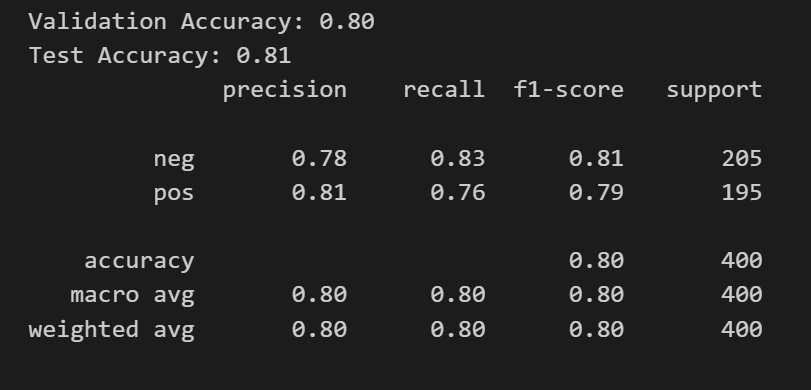
**Code:** [**https://github.com/swapnik-iitkgp/NLP-Course-Project**](https://github.com/swapnik-iitkgp/NLP-Course-Project)

**Task 1: Using Neural Network Classifier:**

* In Task 1, a Part-of-Speech (POS) tagger was implemented from scratch using the NLTK library and the Treebank corpus.
* The Treebank corpus was used for training data, which consists of sentences with words and their corresponding POS tags.
* Transition and emission probabilities were calculated based on the training data, which are essential for the Viterbi Algorithm used in POS tagging.
* A custom POS tagger was created to tag words in sentences, assigning each word a POS tag based on the probabilities learned from the training data.

**Task 2: Using Recurrent Neural Network Classifier:**

* In Task 2, a baseline sentiment analyser was built using the movie\_reviews corpus, which contains movie reviews labelled as 'positive' or 'negative.'
* TF-IDF vectorization was used to convert the reviews into numerical feature vectors.
* A Multinomial Naive Bayes classifier was trained on these TF-IDF features to classify the sentiment of movie reviews into 'positive' or 'negative.'
* The performance of this baseline model was evaluated using metrics like accuracy and a classification report.



* **Results of Vanilla Sentiment Analyser on movie\_reviews corpus**

**Task 3: Using BERT Classifier:**

Step 1: POS Tagging with Task 1's POS Tagger:

* Used the POS Tagger I have implemented in Task 1 to tag the words in the movie\_reviews dataset. Each word in a review will be associated with its POS tag.

Step 2: Sentence Embeddings

* For sentence embeddings, we can use techniques like TF-IDF, Word2Vec, or Doc2Vec to represent the text as numerical vectors. Since I've already used TF-IDF in Task 2, I continued with that for consistency.

Step 3: Integration of POS Tags and Sentence Embeddings

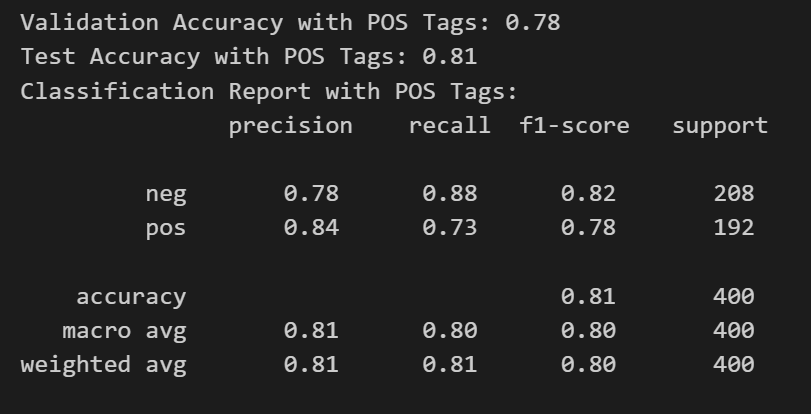
Integrated POS tag features with sentence embeddings, I used the following strategy:

* Concatenated the POS tag embeddings with the sentence embeddings. This resulted in a longer feature vector that includes both the word-level POS tag information and the sentence-level information.
* For example, if the sentence embedding has 100 dimensions and we have a POS tag vocabulary of 50 tags, we will have a final feature vector of 150 dimensions.

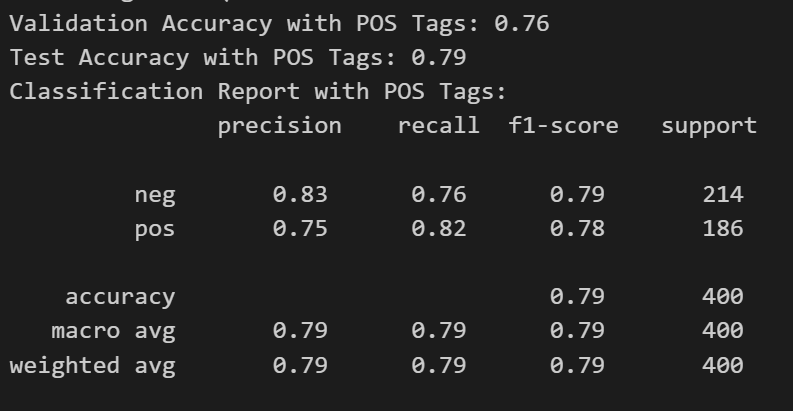
This approach reduces the dimensionality of the POS tag information to match the dimensionality of the sentence embeddings.

Step 4: Classifier Training

* Trained the same classifier that I used in Task 2 (e.g., Multinomial Naive Bayes or another classifier of your choice) on the integrated feature vectors. The features will now be a combination of sentence embeddings and POS tag embeddings.



**Results of Improved Sentiment Analyser using nltk POS tagger on movie\_reviews corpus with an execution time of approx. 90 seconds**



**Results of Improved Sentiment Analyser using Viterbi POS tagger on movie\_reviews corpus with an execution time of approx. 40 minutes**