Natural Language Processing
(CS60075)

Name: Ishaan Sinha
Roll number: 21CS30064

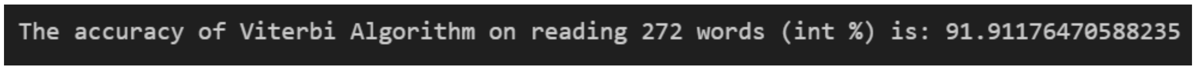
Task - 1 : POS Tagger Implementation

The implementation presented in this report is focused on the Viterbi algorithm for Part-of-Speech (POS) tagging of text data. POS tagging is a fundamental task in natural language processing, where each word in a sentence is assigned its corresponding grammatical category or part of speech.

1. **Implementation Overview**
   Data Preprocessing : - The Treebank corpus from the Natural Language Toolkit (NLTK) was used for training and testing. - The corpus was tokenized into sentences and words, and the frequency of each POS tag was computed.
   Transmission Matrix : - A transition matrix was developed to represent the probabilities of transitioning from one POS tag to another, capturing the syntactical structure of the language. To compute the transition probabilities, occurrences of tag pairs in the training corpus were counted. Additionally, add-1 smoothing (Laplace smoothing) was applied to ensure that even unseen tag transitions received a non-zero probability, making the model more robust in handling rare or previously unseen sequences. The probabilities were then normalized to produce a valid probability distribution across all possible tag transitions.
   Emission Matrix : - An emission matrix was created to model the probabilities of generating a word from a particular POS tag, aiding in the handling of word-level information. The emission probabilities were calculated by counting the frequency of words associated with each tag in the training corpus. To account for unseen word-tag pairs, add-1 smoothing (Laplace smoothing) was applied, ensuring that even words not observed during training were assigned a non-zero probability. Finally, the counts were normalized to produce a valid probability distribution for each tag's possible word emissions.
   Viterbi Algorithm : - The Viterbi algorithm, a dynamic programming technique, was implemented to find the most probable sequence of POS tags for a given sentence. - Initialization, recursion, and termination steps were employed to fill the Viterbi matrix and backpointer matrix. - The algorithm iteratively determined the best sequence of tags for each word in the input sentence, considering both transition and emission probabilities.

2. **Results**
   The implementation was evaluated on sample data from the Treebank corpus, and accuracy

was measured by comparing the predicted POS tags with the ground truth. The accuracy was
found to be around 91.9%, indicating the decent effectiveness of the implemented Viterbi algorithm.



```
The accuracy of Viterbi Algorithm on reading 272 words (int %) is: 91.91176470588235
```

(Note: Due to large size of .ipynb file after generating this result I removed the code segment of checking the accuracy)

3. **Conclusion**
   This implementation successfully demonstrates the Viterbi algorithm for POS tagging from scratch. By leveraging transition and emission probabilities derived from a training corpus, the algorithm efficiently assigns POS tags to words in a sentence. The handling of unknown words and tags with the help of add-1 smoothing ensures robust performance in real-world scenarios. This implementation can be a foundation for more advanced POS tagging systems and can be optimized for specific applications or languages.

Task - 2 : Vanilla Emotion Recognizer

The objective of this task is to classify textual data into one of six emotion labels: joy, sadness, anger, fear, love, or surprise. For this purpose, we utilized the **twitter_messages** corpus, which is pre-split into training, validation, and test subsets, comprising 16,000 training sentences, 2,000 validation sentences, and 2,000 test sentences. The Hugging Face API was employed to efficiently load and manage the dataset.

To prepare the data for classification, we employed **TfidfVectorizer** to generate sentence embeddings. This process involved converting each sentence into a numerical vector, where each dimension represents the significance of a word in the sentence, based on its frequency in the corpus adjusted by the term frequency-inverse document frequency (TF-IDF) score. This approach captures the importance of words in each sentence while minimizing the impact of common but less informative words.

We then trained a **Support Vector Machine (SVM)** classifier with a linear kernel on the vectorized training data. SVM is particularly well-suited for this task due to its effectiveness in handling high-dimensional sparse data, which is typical in text classification scenarios. The model was trained to distinguish between the six emotion categories by maximizing the margin between them in the feature space.

After training, the model was evaluated on the test set, and its performance was measured using standard classification metrics such as accuracy and a detailed classification report, which provides insights into how well the model distinguishes between different emotions.

In summary, this emotion recognizer implements a classic machine learning pipeline, consisting of text vectorization using TF-IDF and classification using an SVM. This approach establishes a solid baseline for emotion recognition, with potential improvements achievable through advanced text processing techniques and model tuning.

Task - 3 : Improved Emotion Recognizer Implementation

Objective: The primary objective of this task was to enhance the performance of the emotion recognition model by incorporating Part-of-Speech (POS) tags into the feature set. The aim was to investigate whether adding syntactic information could improve the classifier's ability to correctly identify emotions in text.

Methodology:

1. **POS Tagging:**
   - The dataset was first processed using a POS Tagger (from Task-1) to assign POS tags to each word in the text. This added an additional layer of syntactic information to the dataset, potentially aiding the model in better understanding the context and structure of the sentences.
2. **Feature Integration:**
   - Instead of using the words alone as features, I concatenated each word with its corresponding POS tag. This created a new feature set where each element of the text was a combination of a word and its POS tag (e.g., "happy_JJ"). The idea behind this was to preserve the semantic meaning while also providing the model with information about the grammatical role of each word.
3. **Training the Classifier:**
   - The same classifier used in Task-2 (SVM) was employed for this task. The new feature set, which included the concatenated word-tag pairs, was used to train the classifier. The goal was to assess whether this syntactic information would improve the model's performance on the emotion recognition task.

Results:

- The POS-tag-enhanced model achieved an accuracy of 0.74 on the test set, which was lower than the 0.88 accuracy achieved by the baseline model in Task-2.

Comparison and Analysis:

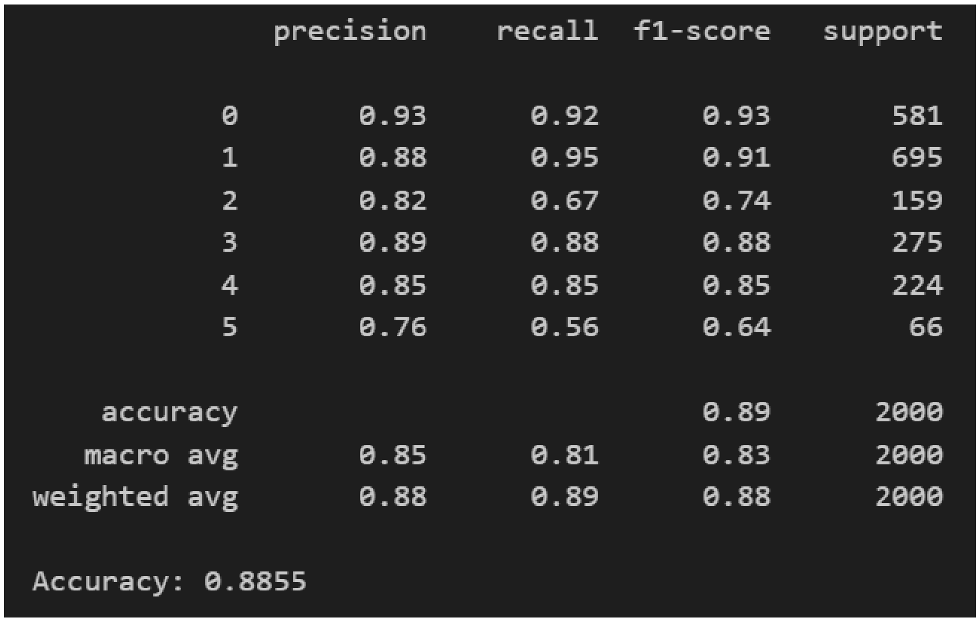
- **New Words:** One significant challenge was the presence of new words in the Twitter messages that were not present in the Treebank corpus used for POS tagging. The model might have struggled to correctly classify these words, as the POS tags assigned might not have been as accurate due to the limited vocabulary in the training corpus.

- **Viterbi Algorithm Limitations:** While the Viterbi algorithm provided decent accuracy for POS tagging on random sentences from the Treebank corpus, its performance degraded when applied to the larger and more diverse dataset of Twitter messages. This likely contributed to inaccuracies in the POS tags, which in turn affected the overall model performance.
- **Increased Complexity:** By concatenating the POS tags with words, the dataset's complexity increased. This additional complexity might have overwhelmed the classifier, making it harder to identify patterns in the data and thus leading to a decrease in accuracy.

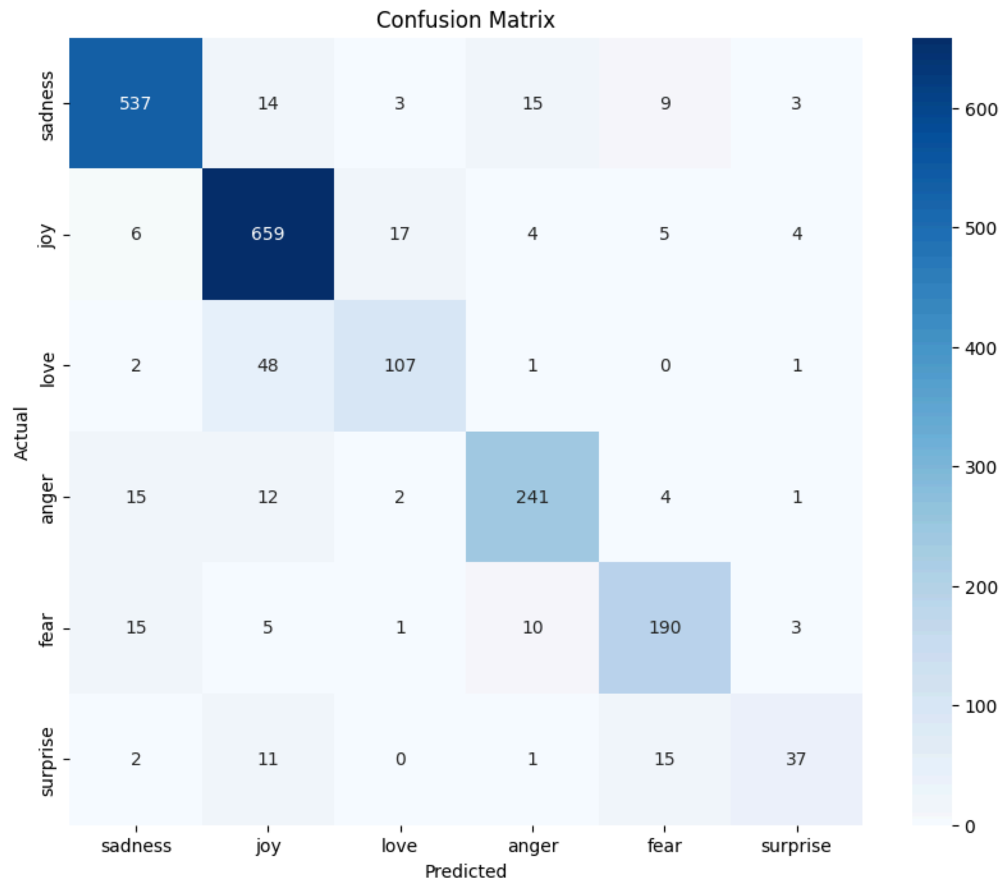
Conclusion: Incorporating POS tags into the emotion recognition pipeline did not lead to an improvement in accuracy; instead, it introduced challenges that outweighed the potential benefits. The increased complexity of the feature set and the limitations of the POS tagging process likely contributed to the reduced performance. Nonetheless, this experiment provides valuable insights into the limitations and trade-offs associated with adding syntactic features to text classification tasks.

Classification Reports and Confusion Matrix

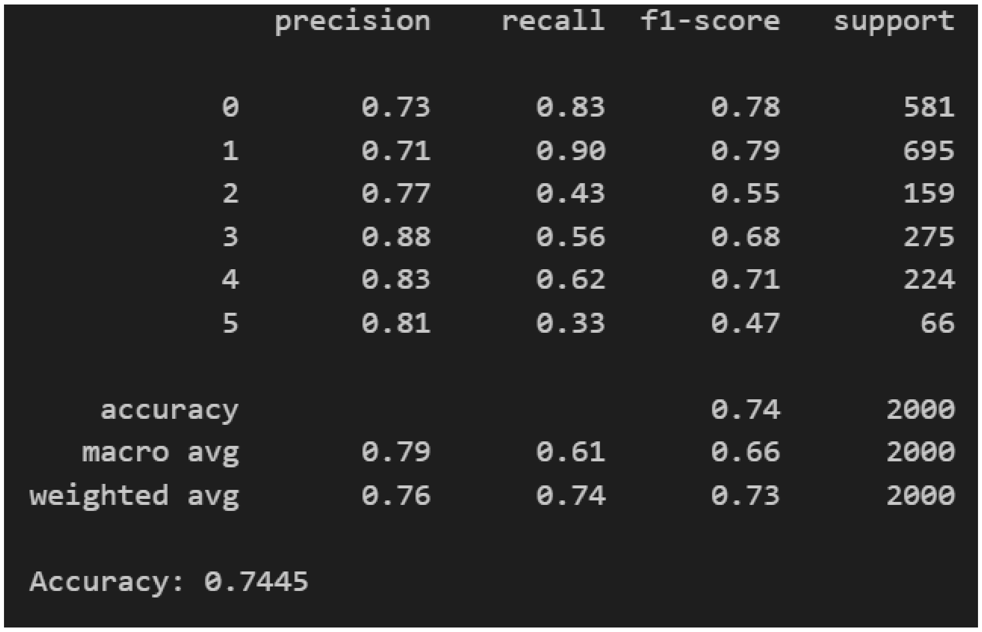
Task - 2:



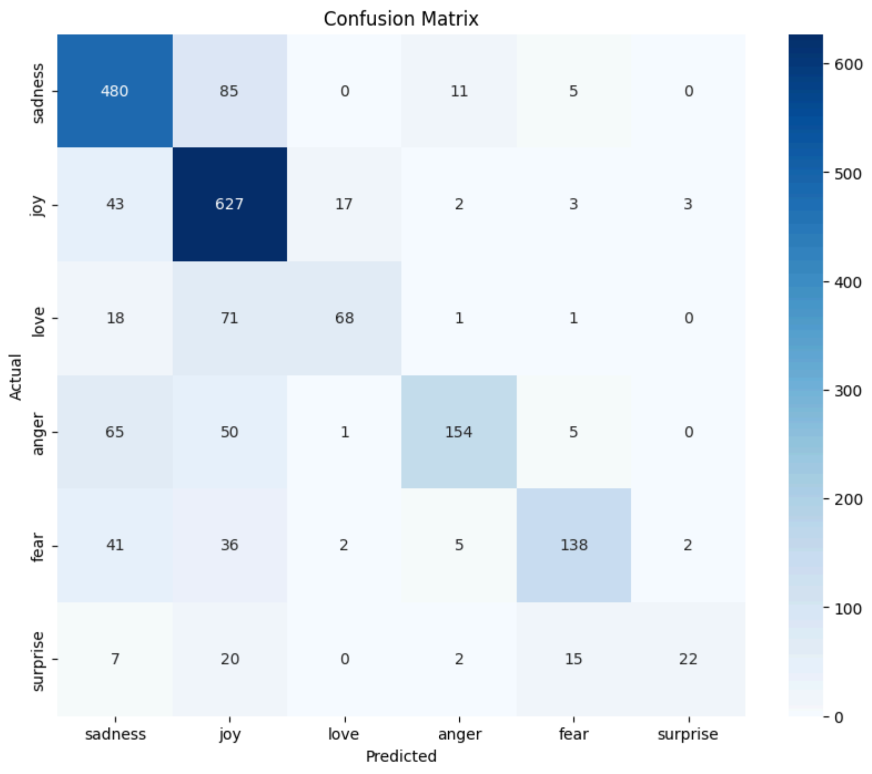
```
              precision    recall  f1-score   support

           0       0.93      0.92      0.93       581
           1       0.88      0.95      0.91       695
           2       0.82      0.67      0.74       159
           3       0.89      0.88      0.88       275
           4       0.85      0.85      0.85       224
           5       0.76      0.56      0.64        66

    accuracy                           0.89      2000
   macro avg       0.85      0.81      0.83      2000
weighted avg       0.88      0.89      0.88      2000

Accuracy: 0.8855
```



Task - 3:



```
              precision    recall  f1-score   support

           0       0.73      0.83      0.78       581
           1       0.71      0.90      0.79       695
           2       0.77      0.43      0.55       159
           3       0.88      0.56      0.68       275
           4       0.83      0.62      0.71       224
           5       0.81      0.33      0.47        66

    accuracy                           0.74      2000
   macro avg       0.79      0.61      0.66      2000
weighted avg       0.76      0.74      0.73      2000

Accuracy: 0.7445
```


Confusion Matrix
Actual
Predicted
sadness
joy
love
anger
fear
surprise
480 85 0 11 5 0
43 627 17 2 3 3
18 71 68 1 1 0
65 50 1 154 5 0
41 36 2 5 138 2
7 20 0 2 15 22
0
100
200
300
400
500
600