Team 15 : Categorising Sentiments of Pandemic Tweets using NLP

Nusaiba Alam - **21301274**
Abrar Ahmed - **21301309**
Md Tawhidur Rahman - **21301308**
Tahiya Mysara Yusha - **21101209**

1 Introduction

The world came to a complete halt due to the coronavirus epidemic, which drastically altered our way of life. The epidemic altered industry, education, and even our idea of health, compelling society to innovate and adapt to unexpected difficulties. Social separation kept us physically separate, but human resilience and technology allowed individuals to connect in new and creative ways. A lot of individuals used social media services like Twitter to express their personal views and feelings. Our research aims to develop an effective model that can recognize and categorize tweet sentiment by gathering some of the coronavirus pandemic-related tweets and applying natural language processing (NLP) techniques.

The initial stage in this project was to remove any extraneous data from the dataset, including links, unnecessary whitespace, and other pointless columns. After tokenizing the tweets, we eliminated stopwords that have little to no significance in sentiment analysis and divided the text into smaller parts. Subsequently, word embeddings and other approaches were used to prepare the data for model training by capturing word context.

We used TensorFlow and Keras-built deep learning models for sentiment analysis. Using layers such as embedding layers for word vector processing, Long Short-Term Memory and Gated Recurrent Unit layers for handling sequential input, and dense layers for classification, a Sequential Neural Network architecture was employed. These algorithms have been programmed to anticipate whether a tweet will be good, neutral, negative or other classifications.

This project's ultimate objective is to use sentiment analysis to categorize tweets on pandemics into five groups: positive, neutral, negative, highly positive, and extremely negative. In order to help policymakers, researchers, and public health professionals better comprehend society reactions to the crisis, the initiative plans to precisely identify these views and give insights into public opinion and emotional responses during the COVID-19 epidemic.

2 Dataset

Source: Kaggle Pandemic Tweet Challenge.
Data Size: 41,157

For this project, we worked with a dataset comprising 41,157 rows and two columns: "Original Tweet" and "Sentiment". To ensure effective sentiment analysis, the tweets were preprocessed by some steps.

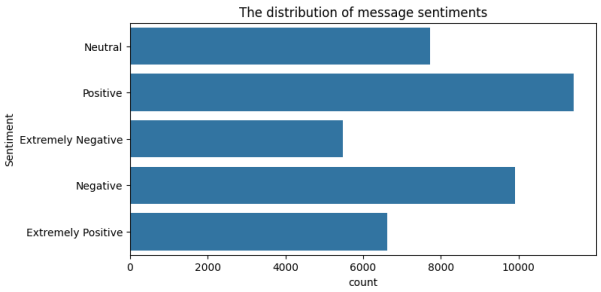

Figure 1: Distribution of Sentiment Classes

2.1 Imported Necessary Libraries:

- To perform data cleaning, processing, and model preparation, essential Python libraries like pandas, numpy, nltk, re, and tensorflow were imported.
- These libraries help with handling data structures, tokenization, text cleaning, and building machine learning models.

2.2 Data Cleaning:

- The "Original Tweet" column was cleaned to remove unnecessary elements that could hinder the sentiment analysis process.

- **Whitespace Removal:** Extra spaces were removed to ensure consistent formatting.
- **URL Removal:** URLs, which do not contribute to sentiment analysis, were eliminated.
- **Stop Words Removal:** Common stop words (e.g., "the", "and") were excluded, as they do not provide meaningful sentiment information.
- **Retaining Specific Punctuation & Alphanumeric Characters:** Kept punctuation like .,!?;:'"()/- and alphanumeric characters since they can convey sentiment (e.g., an exclamation mark to show strong emotion).
- **Reduced Repetitive Punctuation:** Converted multiple punctuation marks (like "!!!" or "??") into single occurrences for standardization.

2.3 Kept Relevant Columns:

- Only the cleaned tweet and sentiment information were retained.
- A new column, "Cleaned Tweet", was created after preprocessing.
- The original "Sentiment" column was retained but eventually converted during one-hot encoding, which allows classification into multiple categories.

2.4 One-Hot Encoding of Sentiment:

- The sentiment labels (Negative, Extremely Negative, Positive, Extremely Positive, Neutral) were converted into numerical vectors using one-hot encoding.
- For example, a tweet classified as "Positive" becomes [0, 0, 1, 0, 0].
- This encoding facilitates multinomial classification, where the model predicts one of these five sentiment classes.

2.5 Train-Test-Validation Split (80%-10%-10%):

The dataset was divided into three parts:

- Training Data (80%): Used to train the machine learning model.
- Validation Data (10%): Used during model training to fine-tune and validate the model's performance.
- Test Data (10%): Held back to evaluate how well the model generalizes to unseen data. This split ensures that the model is robust and not overfitted to the training set.

2.6 Tokenization:

The text data was tokenized, meaning each word in the tweets was converted into numerical tokens based on its frequency in the training data. This is crucial for feeding the data into a machine learning model. The tokenizer was fitted on the training data to ensure the model learns from the actual patterns it will encounter. In deployment, the model will generalize based on these learned patterns.

2.7 Padding:

After tokenization, the sequences of tokens were padded to a uniform length, ensuring that all input data has the same shape. Padding is necessary because models like RNNs or LSTMs expect inputs of consistent dimensions.

2.8 Undersampling:

Training features Undersampling (To make all the classification even and equal, means will not be biased towards any feature while training)

These preprocessing steps ensure that the data is clean, correctly formatted, and ready for input into a sentiment analysis model.

textbfEmbeddings: GloVe word embeddings were applied to convert each token into a dense vector, capturing the semantic meaning of words. GloVe embeddings are pre-trained on a large corpus and help the model understand word relationships better, which is useful for sentiment analysis.

3 Models

For text categorization in our project, we used Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM). By using various techniques, each model was created to accommodate the sequential nature of text input and capture long-term relationships.

3.1 LSTM Model

Long-range dependencies are handled by the LSTM model, which also helps to reduce the vanishing gradient issue that regular RNNs may have. Information flow is controlled by input, output, and forget gates, which manage the memory cells that make up LSTM units. Because of this, the model may choose to keep or discard information over

long sequences, which is very helpful for works involving natural language processing.
Our LSTM model's architecture is as follows:

- **Embedding Layer:** Using an embedding layer occupied with pre-trained word embeddings, the input text is represented as dense word vectors. The pre-trained embeddings' semantic data is used by the embedding layer, which is configured to be non-trainable (trainable=False) to avoid additional changes.modification.

- **LSTM Layer:** To extract the sequential patterns from the input data, we employed a single, 128-unit LSTM layer with a tanh activation function.

- **Dropout Layer:** A dropout layer with a rate of 0.2 is added to randomly change a portion of the input units to zero during each update in order to prevent overfitting.

- **Dense Output Layer:** The dense layer in the text classification work predicts the class probabilities for each of the five output classes using a softmax activation function.
  The categorical cross-entropy loss function and Adam optimizer were used to construct the model. Early stopping was used with patience for three epochs in order to minimize overfitting. If the performance worsened, the validation loss was monitored and the optimal weights were restored. The model was trained using a training-validation split to track performance across a maximum of 25 epochs with a batch size of 50.

3.2 GRU Model

The GRU model is structurally similar to the LSTM, but it uses GRU units in place of LSTM cells. The vanishing gradient problem is also addressed by GRU, a more straightforward version of LSTM that utilizes fewer gates—reset and update gates—while maintaining the model's ability to capture dependencies in sequences.
Similar to the LSTM model, the GRU model has the same architecture:

- **Embedding Layer:** Initialized using pre-trained embeddings, the embedding layer is the same as in the LSTM model.

- **GRU Layer:** A GRU layer consisting of 128 units was used with a relu activation function. The goal of GRU design is to achieve the same performance with less complexity.

- **Dropout Layer:** To reduce overfitting, a dropout rate of 0.2 was employed, same like with the LSTM model.

- **Dense Output Layer:** The output layer stays the same and classifies using a softmax activation.
  To deal with the sequential nature of text input, we used LSTM and GRU models, taking advantage of their capacity of identifying dependencies and minimizing the effects of the vanishing gradient issue. The performance of the two models was evaluated for text categorization after being tuned using early stopping.

4 Results

Model Performance:

4.1 LSTM:

- **Training Accuracy:** 70%

- **Testing Accuracy:** 63%

- After the 10th epoch, validation accuracy started to decline, indicating some overfitting.

4.2 GRU:

- **Training Accuracy:** 67%

- **Testing Accuracy:** 60%

- Compared to the LSTM model, overfitting occurs as early as the 7th epoch.

In comparative metrics, LSTM outperformed GRU in all key metrics:

- **Accuracy:** LSTM (63%) vs GRU (60%)

- **Precision, Recall, F1-Score:** LSTM performed better in all sentiment categories in comparison to GRU.

Overall, the LSTM model proved more effective than GRU, especially in accuracy, precision, recall, and F1-score for sentiment analysis in this dataset.

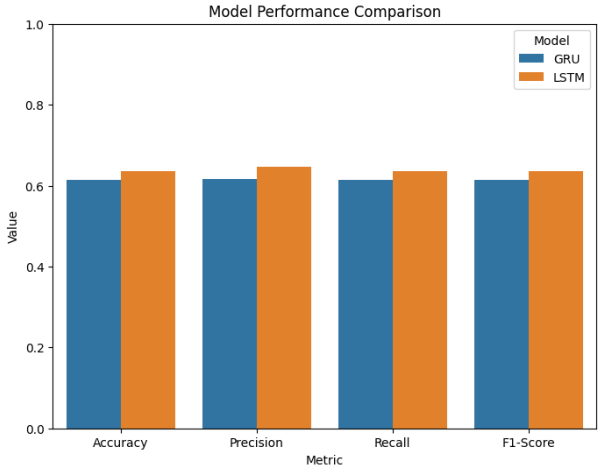

Figure 2: Distribution of Sentiment Classes

5 Conclusion

In the sentiment analysis project focused on pandemic tweets, various challenges were addressed. Effective data preprocessing included thorough text cleaning, tokenization, and padding for varying tweet lengths. Class imbalance was tackled using SMOTE for oversampling minority classes, ensuring fairness. The embedding matrix was constructed with pre-trained GloVe vectors, enhancing word representation. Models using LSTM and GRU architectures were optimized through hyperparameter tuning and early stopping to prevent overfitting. Comprehensive evaluation metrics were employed for accurate performance assessment, and emphasis was placed on data quality through careful curation and consistent labeling. These strategies led to a robust sentiment analysis model.

References

N. Janakiev. Practical text classification with python and keras. Accessed: October 3, 2024.

D. Jurafsky and J. H. Martin. 2008. *Speech and Language Processing, 2nd Edition*. Pearson Prentice Hall.

Nuzulul Khairu Nissa. 2022. Text messages classification using lstm, bi-lstm, and gru. Retrieved October 3, 2024.

(Janakiev) (Jurafsky and Martin, 2008) (Nissa, 2022)