EMOTION CLASSIFICATION OF TWEETS USING NATURAL LANGUAGE PROCESSING

A PROJECT REPORT

Submitted by

AMANI KAMBHAM – 00799187

LAKSHMI SAI KISHORE SAVARAPU – 00827326

Under the guidance of

Professor Mr. Khaled Sayed

For the Course DSCI-6004-02

NATURAL LANGUAGE PROCESSING

UNIVERSITY OF NEW HAVEN

WEST HAVEN, CONNECTICUT

SPRING 2024

TABLE OF CONTENTS

ABSTRACT

The classification of emotions within tweets stands as a cornerstone in the evolving landscape of Natural Language Processing (NLP), offering profound insights into public sentiment, individual emotional states, and collective reactions to global events. This project underscores the critical importance of sentiment classification in tweets, aiming to harness and interpret the vast and nuanced spectrum of human emotions expressed through social media. Recognizing the complexity of this task, we employ a multifaceted approach that integrates advanced NLP models and innovative methodologies to enhance the accuracy and depth of emotion classification.

Our foundational strategy employs a series of neural network architectures, each selected for their unique strengths in processing sequential data and capturing linguistic patterns: Vanilla Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Bidirectional RNN. These models serve as the primary tools in our endeavor to classify tweets into discrete emotional categories, addressing the challenge of understanding the subtle nuances embedded in text.

In pursuit of optimizing our models' performance, we introduce an innovative approach by training these networks on input sentences in reverse order. This novel technique is designed to mitigate the vanishing gradient problem—a common obstacle in training deep neural networks—by presenting potentially more relevant contextual information at the beginning of the sequence. This methodological adjustment aims to enhance the models' ability to learn and retain important emotional indicators from the text, thereby improving classification accuracy.

Building on the foundation laid by our initial models and novel preprocessing technique, we further explore the potential of stacked network architectures. Stacked Vanilla RNN, Stacked LSTM, Stacked GRU, and Stacked Bidirectional RNN models are developed, employing both the traditional and reverse-order approaches to sentence input. This advanced strategy layers multiple

instances of neural networks, intensifying the model's capacity to learn from complex, long-distance dependencies within the text. The use of stacked architectures, in conjunction with the reverse-order input method, represents a dual approach aimed at pushing the boundaries of model sophistication and performance in emotion classification.

This project not only highlights the importance of accurately classifying emotions in tweets for a deeper understanding of public sentiment but also showcases the efficacy of combining cutting-edge NLP models with innovative methodologies. By employing a comprehensive suite of neural network architectures and exploring both traditional and novel approaches to model training, we aim to set a new benchmark in the field of sentiment analysis, opening new avenues for research and application in the digital age.

INTRODUCTION

In an era where social media platforms like Twitter have become ubiquitous channels for personal expression and public discourse, the ability to accurately classify and understand the emotions embedded in tweets has emerged as a pivotal challenge and opportunity in the field of Natural Language Processing (NLP). Sentiment classification, particularly the nuanced task of emotion classification, provides invaluable insights into public sentiment, enabling a deeper understanding of collective emotions, trends, and reactions to events on a global scale. This project seeks to advance the state of emotion classification in tweets through the innovative application of NLP techniques, employing a comprehensive approach that leverages both established and novel methodologies.

The foundational aspect of our approach involves the deployment of various neural network architectures known for their efficacy in processing sequential and textual data: Vanilla Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and

Bidirectional RNN. Each of these models offers unique advantages in capturing the temporal dynamics and contextual nuances of language, making them well-suited for the task of emotion classification in tweets.

Recognizing the limitations inherent in traditional text processing techniques, particularly the challenge of vanishing gradients which can hinder a model's learning from long text sequences, we introduce a novel preprocessing strategy: the reverse ordering of input sentences. This technique aims to present the model with the most contextually relevant information at the outset of the sequence, thereby enhancing its ability to learn and retain significant emotional indicators from the text.

Building upon the solid foundation established by our initial modeling efforts and our innovative approach to text preprocessing, we further explore the potential of stacked network architectures. By layering multiple instances of neural networks on top of each other—Stacked Vanilla RNN, Stacked LSTM, Stacked GRU, and Stacked Bidirectional RNN—we aim to amplify the models' capacity to learn from complex patterns and dependencies in the data. This approach, applied in conjunction with both traditional and reverse-order sentence inputs, represents a concerted effort to maximize the sophistication and performance of our emotion classification models.

The integration of these advanced neural network architectures with both conventional and novel methodologies underscore our project's commitment to pushing the boundaries of what is possible in sentiment analysis. Through this comprehensive and innovative approach, we aim not only to enhance the accuracy of emotion classification in tweets but also to contribute to the broader field of NLP, opening new avenues for research and practical applications in understanding human emotions in the digital age.

DATASET

For this project, a meticulously curated dataset from Kaggle forms the cornerstone of our research. This dataset is specifically designed for emotion detection from text, making it an invaluable resource for training and evaluating our suite of advanced NLP models.

The dataset consists of approximately 40,000 tweets, each meticulously labeled with one of 13 distinct emotions. This wide range of emotions ensures a comprehensive understanding of the varied emotional expressions people share on Twitter, from joy and surprise to sadness and anger.

Each tweet in the dataset is accompanied by a unique identifier (tweet_id), the emotion label, and the tweet content itself. This structured format facilitates straightforward processing and analysis, enabling the efficient training of our models.

The tweets encapsulate a broad spectrum of topics, reflecting the diverse and dynamic nature of discussions on Twitter. This diversity is crucial for training robust models capable of recognizing and classifying a wide array of emotional expressions in varying contexts.

To explore the potential of reversing the input sentence order—a novel approach aimed at combating the vanishing gradient problem—the dataset's structure allows for easy manipulation of tweet content. By evaluating models trained on both standard and reversed sentence inputs, we can assess the effectiveness of this innovative strategy in enhancing model performance.

This dataset not only serves as the foundation for our exploration of neural network architectures and novel methodologies in emotion classification but also represents a significant step forward in the application of NLP techniques to understand and interpret human emotions expressed through social media. By leveraging this rich dataset, we aim to uncover new insights into the emotional landscape of Twitter, contributing to the advancement of sentiment analysis and NLP.

TEXT CLEANING AND PRE-PROCESSING

In our project, we employed a series of text cleaning and preprocessing techniques pivotal for the preparation of our dataset for model training and evaluation. Below, we outline the key steps taken, leveraging Python and TensorFlow's Keras preprocessing tools, to process the textual data from tweets effectively. Some of the techniques implemented are:

Removal of Punctuations and Special Characters:

To ensure the textual data is devoid of noise that could hinder the model's performance, we removed punctuation and special characters from the tweets. This was accomplished by leveraging Python's string manipulation capabilities, specifically utilizing the `str.translate` and `str.maketrans` methods. Notably, the tilde (`~`) character was preserved to maintain the integrity of certain textual content. This step is crucial for focusing the model's attention on meaningful word content.

Word Tokenization:

The processed tweets were then tokenized into individual words using TensorFlow's Keras `Tokenizer` class. This step converts the clean text into sequences of tokens or words, a form that is more amenable to numerical processing by NLP models. The tokenizer was configured to convert text to lowercase and split on whitespace, ensuring consistency across the dataset.

Label Encoding of Target Variable:

Given the categorical nature of our target variable—tweet emotions—it was essential to convert these categories into a numerical format. We employed label encoding, transforming the emotion labels into unique integers. This was achieved through the Keras `Tokenizer`, treating each unique emotion label as a token and subsequently mapping these tokens to unique integers.

Padding Input Sequences:

To accommodate the input requirements of neural network models, which necessitate uniform input dimensions, we padded the tokenized sequences. The Keras `pad_sequences` method was used to ensure all input sequences (tweets) have the same length by adding zeros ('post' padding) to shorter sequences. This step is vital for preparing the dataset for training and inference with our models.

Handling Unique Words and Vocabulary:

The tokenizer's `word_index` property was utilized to identify all unique words in the dataset, forming the basis of our model's vocabulary. This approach not only helps in reducing computational complexity but also ensures that the models are trained on relevant and context-specific vocabulary.

Preparing the Labels for Model Training:

The emotion labels were processed using the Keras `Tokenizer` to fit the numerical format required for model training. This involved creating a mapping of emotion labels to integers and adjusting these mappings to align with the expected format of our neural network models. Additionally, an inverse mapping was created to interpret the model's predictions back into human-readable emotion labels.

This comprehensive preprocessing pipeline was essential for transforming raw tweet data into a structured format suitable for training sophisticated NLP models. By meticulously cleaning and preparing the data, we laid a strong foundation for our models to learn and predict the emotional content of tweets accurately, demonstrating the critical role of effective data preprocessing in the success of NLP projects.

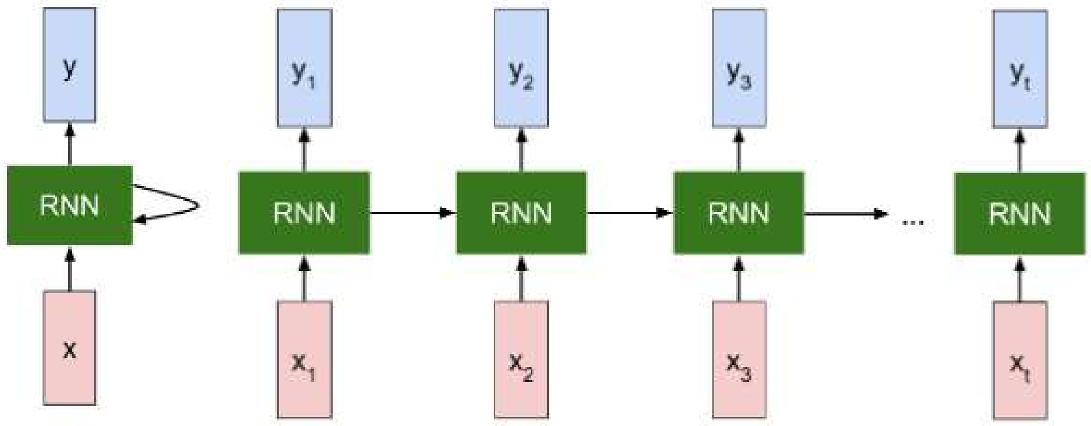
MODELS

In this project, we employ a suite of sophisticated Natural Language Processing (NLP) models, each chosen for their unique strengths in handling the complexities of language and emotional expression in textual data. Our approach encompasses both conventional models and innovative adaptations, including the exploration of stacked architectures and the novel preprocessing technique of reversing input sentences. Here's an overview of the models utilized:

Vanilla Recurrent Neural Network (RNN):

Description: The simplest form of recurrent neural networks, designed to process sequences of data by maintaining a 'memory' of previous inputs through its hidden layers. Ideal for modeling temporal dependencies in text.

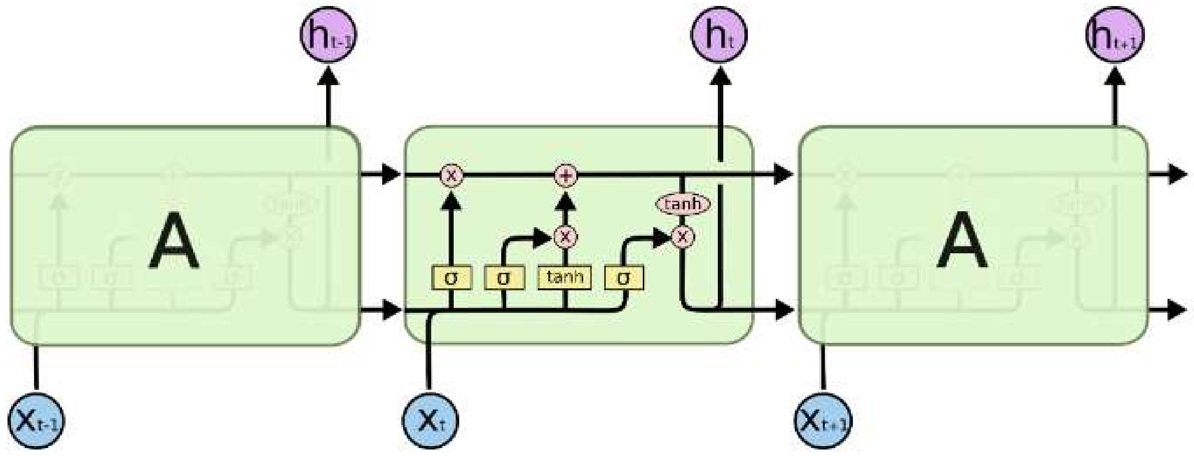
Application: Used as a baseline for emotion classification, understanding the fundamental capabilities and limitations in handling sequential data.



Long Short-Term Memory (LSTM):

Description: An advanced RNN variant equipped with special gates (forget, input, and output) to control the flow of information, effectively addressing the vanishing gradient problem and improving the model's ability to learn from long-distance dependencies.

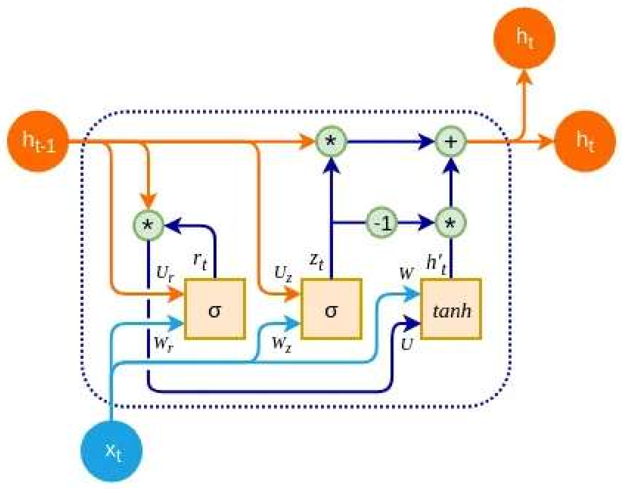
Application: Employed to capture complex emotional nuances in tweets, leveraging its proficiency in retaining relevant context over long sequences of text.



Gated Recurrent Unit (GRU):

Description: Similar to LSTM but with a simplified architecture, combining the input and forget gates into a single update gate. This results in fewer parameters, making GRU models faster to train without significantly compromising performance.

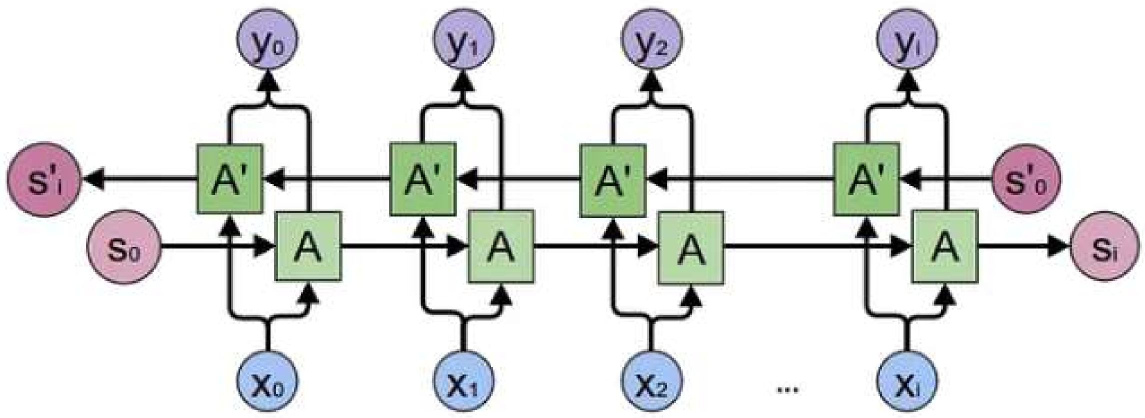
Application: Utilized for efficient emotion classification, particularly in scenarios where computational resources or training time are constrained.



Bidirectional RNN:

Description: Enhances the traditional RNN by processing data in both forward and reverse directions, allowing it to capture context from both the past and the future within a sequence. This bidirectional processing provides a more comprehensive understanding of the text.

Application: Applied to ensure that the emotional content of tweets is fully contextualized, improving the accuracy of classification.



Innovative Adaptations:

Reverse Order Processing: All models are also trained on datasets with input sentences in reverse order, aiming to bring contextually relevant information to the forefront and address the challenges associated with learning from long text sequences.

Stacked Architectures: Each of the above models is further explored in a stacked configuration—layering multiple instances of the same model type to deepen the network and enhance its learning capacity. This approach is applied to both traditional and reverse-ordered sentence inputs, investigating its impact on model performance in emotion classification.

These models, along with the innovative strategies employed, represent a comprehensive effort to advance the state of emotion classification in tweets. By exploring the capabilities of these neural network architectures and experimenting with novel approaches to model training, the project aims to set new benchmarks in the accuracy and depth of sentiment analysis within the realm of NLP.

Key Components in the model Architecture:

Activation Function: Utilized the softmax activation function for the output layer, suitable for multiclass classification problems.

Optimizer: Employed the Adam optimizer for its efficient computation and adaptability to different data and parameters.

Loss Function: Chose categorical cross-entropy as the loss function, ideal for multiclass classification tasks.

EVALUATION METRICS

In this project, evaluating the performance of our NLP models is crucial for understanding their effectiveness in classifying emotions from tweets. We utilize a comprehensive set of evaluation metrics and visualization techniques to assess and interpret the models' performance. Below is an overview of the evaluation metrics and methods applied:

Accuracy:

Accuracy measures the proportion of correctly predicted observations to the total observations. It gives a quick indication of the overall correctness of the model but may not reflect the performance on imbalanced datasets effectively.

Recall (Sensitivity):

Recall measures the ability of the model to capture relevant data points. It's the ratio of correctly predicted positive observations to all actual positives. High recall indicates the model is good at minimizing false negatives.

Precision:

Precision assesses the accuracy of positive predictions made by the model, calculated as the ratio of correctly predicted positive observations to the total predicted positives. High precision indicates the model is good at minimizing false positives.

F1 Score:

The F1 score is the harmonic mean of precision and recall, providing a balance between the two metrics. It's especially useful when the class distribution is uneven. An F1 score closer to 1 indicates better model performance.

METHODOLOGY

This project was implemented on the Google Colab platform, leveraging its powerful computational resources and ease of use for complex NLP tasks. Below is a detailed report of our methodology.

Process 1: Standard Text Processing

Library and Module Imports: Initiated by importing necessary Python libraries and modules, including TensorFlow, Keras for modeling, and Pandas and Numpy for data manipulation.

Dataset Importation: The dataset was imported by uploading to the notebook.

Dataframe Conversion: The imported data was meticulously converted into a structured dataframe for further processing.

Exploratory Data Analysis (EDA): Conducted EDA to understand the dataset's characteristics, including distribution of classes, presence of outliers, and data completeness.

Text Cleaning and Pre-processing: Implemented several text cleaning and preprocessing techniques such as word tokenization, removal of punctuations, padding of input sequences, and label encoding the target variable. The total number of unique tokens in the document was also identified.

Train-test Split: The preprocessed data was then divided into training and testing sets to facilitate model training and evaluation.

Feature Extraction: Extracted features from the preprocessed text to prepare it for model input.

Model Building: Built neural network models tailored for multiclass classification, incorporating a softmax activation function in the output layer to handle multiple classes effectively.

Model Training: The models were trained using the prepared dataset, optimizing performance through iterations. Each model was trained over 25 epochs, allowing us sufficient iterations to observe the learning trends and performance stabilization over time.

Model Evaluation: Evaluated the models on several metrics, including accuracy, precision, and recall. Additionally, generated classification reports and confusion matrices to assess model performance in detail.

Performance Comparison: Compared the performance of all models to identify the most effective architecture and preprocessing techniques.

Process 2: Training with Reversed Text Sentences

Text Sentence Reversal: Modified the dataframe by reversing the text sentences, aiming to investigate the impact of this approach on model performance.

Repeat Steps (2-11 from Process 1): Repeated the steps from Process 1 for the dataset with reversed sentences, maintaining consistency in methodology while exploring the efficacy of reversed text input.

RESULTS AND INFERENCE:

Performance of NLP models (Input sequence in usual order):

Metrics	Vanilla RNN Model	LSTM Model	GRU Model	Bidirectional Vanilla RNN Model
Accuracy	0.18	0.24	0.22	0.11
Precision	0.10	0.24	0.22	0.11
Recall	0.10	0.24	0.22	0.11
F1 Score	0.10	0.24	0.22	0.11

Performance of NLP models (Input sequence in reverse order):

Metrics	Vanilla RNN Model	LSTM Model	GRU Model	Bidirectional Vanilla RNN Model
Accuracy	0.12	0.25	0.22	0.11
Precision	0.12	0.25	0.22	0.11
Recall	0.12	0.25	0.22	0.11
F1 Score	0.12	0.25	0.22	0.11

Inference on Performance of Basic Models (Usual Order vs. Reverse Order):

Vanilla RNN and Bidirectional Models: These models show lower performance compared to LSTM and GRU models, with particularly poor results from the Bidirectional Vanilla RNN model, regardless of input sequence order. This suggests that simple recurrent architectures may struggle with the complexity or the imbalanced nature of the dataset.

LSTM and GRU Models: Both models demonstrate relatively better performance, with LSTM showing a slight improvement in reverse order (accuracy of 0.25). This suggests that more complex architectures like LSTM and GRU are better at capturing dependencies in the data, potentially benefiting from the different perspectives provided by reversing the input sequence.

Performance of Stacked NLP models (Input sequence in usual order):

Metrics	Stacked Vanilla RNN Model	Stacked LSTM Model	Stacked GRU Model	Stacked Bidirectional Vanilla RNN Model
Accuracy	0.11	0.19	0.20	0.19
Precision	0.11	0.19	0.20	0.19
Recall	0.11	0.19	0.20	0.19
F1 Score	0.11	0.19	0.20	0.19

Performance of Stacked NLP models (Input sequence in reverse order):

Metrics	Stacked Vanilla RNN Model	Stacked LSTM Model	Stacked GRU Model	Stacked Bidirectional Vanilla RNN Model
Accuracy	0.22	0.19	0.20	0.15
Precision	0.22	0.19	0.20	0.15
Recall	0.22	0.19	0.20	0.15
F1 Score	0.22	0.19	0.20	0.15

Performance of Stacked Models (Usual Order vs. Reverse Order):

General Observation: Stacked models did not show significant improvement over their single-layer counterparts in the usual order. This might indicate that simply increasing model complexity through stacking does not necessarily translate to better performance without specific tuning or enhancements like bidirectional processing or attention mechanisms.

Stacked Vanilla RNN: Interestingly, the performance of the Stacked Vanilla RNN model significantly improves in reverse order (accuracy of 0.22), suggesting that for simpler models, reversing the input sequence might help mitigate issues related to short-term memory by presenting more relevant information earlier in the sequence.

Stacked GRU and LSTM Models: These models maintain consistent performance across different configurations but do not show substantial improvements, indicating that the stacking of LSTM and GRU layers might require further optimization or might be constrained by other factors like dataset size or feature representation.

Comparison Between Usual and Reverse Order Inputs:

Reversed Input Advantage: Models trained with reversed text sequences occasionally show improved or comparable performance to those trained on usual order sequences. This suggests that input manipulation could be a useful strategy for certain types of models, especially simpler ones or where temporal dynamics are crucial.

Overall Model Evaluation:

The best-performing model in standard configurations is the LSTM model trained on reversed sequences, showing the highest accuracy and F1 score among the basic models. This indicates that LSTM benefits from its ability to retain longer-term dependencies, which might be better captured when the input sequence is reversed.

Among stacked models, the Stacked Vanilla RNN model trained on reversed sequences shows a significant jump in performance, highlighting how modifying input structures can beneficially impact even simpler model architectures in certain contexts.

CONCLUSION:

The project has explored various neural network architectures and preprocessing techniques to classify emotions from tweets. The investigation highlighted the capabilities and limitations of models like Vanilla RNN, LSTM, GRU, and Bidirectional RNN under standard and reversed text inputs. Additionally, the use of stacked models provided insights into the effects of increased model complexity on performance.

Future Scope:

To further improve accuracy and fulfil the objectives of emotion classification in tweets, the following techniques and areas of exploration can be performed:

Hyperparameter Optimization: Systematic tuning of model parameters, such as learning rate, number of layers, dropout rate, and batch size, could potentially enhance model performance. Techniques like grid search or random search may be employed to find the optimal set of parameters.

Advanced Model Architectures: Incorporating more sophisticated neural network architectures like Transformer models or those utilizing attention mechanisms could improve the ability to focus on relevant parts of the input data, thereby enhancing classification accuracy.

Data Augmentation: To address issues of overfitting and improve the model's generalization capability, data augmentation techniques like synonym replacement, random insertion, or more sophisticated NLP-based approaches could be used to expand the training dataset artificially.

Ensemble Methods: Combining the predictions from multiple models through techniques such as stacking, boosting, or bagging could lead to improvements in accuracy and robustness, as ensemble methods can capitalize on the strengths of individual models and mitigate their weaknesses.

Deeper Semantic Analysis: Employing techniques that can capture more complex semantic representations of text, such as sentiment embeddings or context-aware embeddings from models like BERT or GPT, may enhance the model's understanding of emotional nuance.

Extended Training: Experimenting with longer training durations or more epochs for models showing steady improvements might uncover additional gains in performance that were not fully realized in the initial training phase.

Post-training Refinement: Techniques like model pruning or quantization could be explored to enhance deployment efficiency without significant losses in performance, making the models more practical for real-world applications.

In conclusion, while the project has made significant strides in emotion classification from tweets, the findings underscore the necessity for ongoing research and refinement of techniques to fully realize the potential of NLP in understanding and interpreting human emotions in social media. Future work in this area promises to refine these approaches further, contributing valuable insights to both the field of NLP and the broader domain of AI-driven emotional analysis.

<u>REFERENCES:</u>

1. W. Y. Chong, B. Selvaretnam and L. -K. Soon, "Natural Language Processing for Sentiment Analysis: An Exploratory Analysis on Tweets," 2014 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology, Kota Kinabalu, Malaysia, 2014, pp. 212-217, doi: 10.1109/ICAIET.2014.43. keywords: {Sentiment analysis;Twitter;Semantics;Feature extraction;Machine learning algorithms;Tagging;Sentiment Analysis; Natural Language Processing; Tweets},

2. S. M. et. al., "Detecting Emotion from Natural Language Text Using Hybrid and NLP Pre-trained Models", TURCOMAT, vol. 12, no. 10, pp. 4095–4103, Apr. 2021.

3. L. Mathew and V. R. Bindu, "A Review of Natural Language Processing Techniques for Sentiment Analysis using Pre-trained Models," 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2020, pp. 340-345, doi: 10.1109/ICCMC48092.2020.ICCMC-00064.

4. Kishori K. Pawar, Pukhraj P Shrishrimal, R. R. Deshmukh," Twitter Sentiment Analysis: A Review" International Journal of Scientific & Engineering Research, Volume 6, Issue 4, April-2015