Emoji Prediction from Sentence

Final Project Report

TEAM-25

Harsha Meka **Gayathri Sruthi Gannamaneni** **Karthik Pavan Chowdary Bobba**

hmeka@buffalo.edu **ggannama@buffalo.edu** **kbobba2@buffalo.edu**

50546534 50544968 50533620

Abstract:

In contemporary digital communication, emojis have evolved from simple pictorial supplements to essential elements that enhance and clarify textual interactions. Our "Emoji Prediction from Sentence" project endeavors to develop a predictive model that employs deep learning methodologies to accurately suggest emojis based on textual content. Utilizing advanced natural language processing (NLP) techniques and neural network architectures, including Long Short-Term Memory (LSTM) and Transformer models, which investigates the intricate dynamics between text and emojis. We used the Multi Label Annotations dataset 'Tweemoji', which comprises nearly 13 million tweets annotated with emojis, to train and validate the predictive models. This project employs a hybrid approach by integrating Multinomial Naïve Bayes classifiers with LSTM networks to address various challenges in text representation and sequence processing. The objective is to refine the accuracy of emoji predictions, thereby facilitating more nuanced and emotionally resonant interactions on social media platforms. Ultimately, this research contributes to the broader field of sentiment analysis and aims to enhance the user experience by providing contextually appropriate emoji suggestions automatically.

Problem Statement:

In this new age of communication, emojis play a pivotal role in expressing emotions, intentions, and reactions, enriching text-based interactions across various platforms. However, the process of selecting the appropriate emoji can be time-consuming and may not accurately reflect the user's intended sentiment or context. This issue is compounded by the vast array of emojis available, which can overwhelm users and lead to misinterpretations in communication.

The primary challenge addressed by our project is to develop a deep learning model capable of automatically predicting the most appropriate emoji from a given sentence. This entails understanding the nuanced emotional and contextual cues within the text, which vary widely across different languages and cultural contexts. The project aims to leverage advancements in natural language processing (NLP) and machine learning to interpret these cues and suggest emojis that enhance the expressiveness and clarity of digital communications.

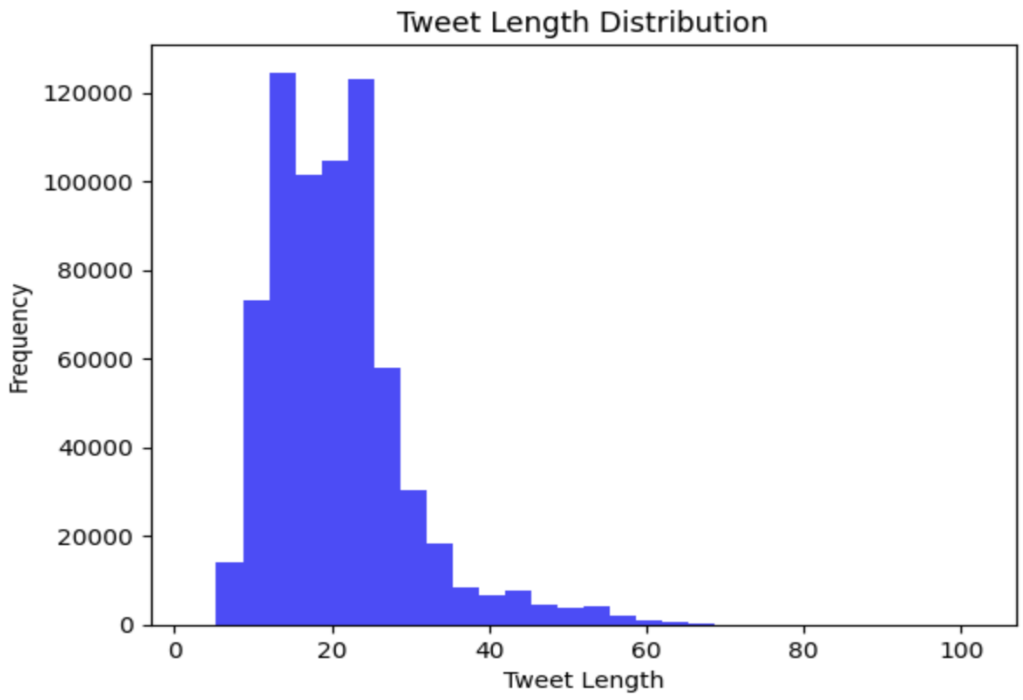
Therefore, the objective of our project is to design and implement a sophisticated model that utilizes state-of-the-art deep learning techniques, such as Long Short-Term Memory (LSTM) networks or Transformer-based models like BERT, to analyze and predict emojis that are contextually and emotionally congruent with the input sentence. By doing so, our project seeks to improve user experience in digital communications, making interactions quicker, more intuitive, and emotionally resonant.

Objectives:

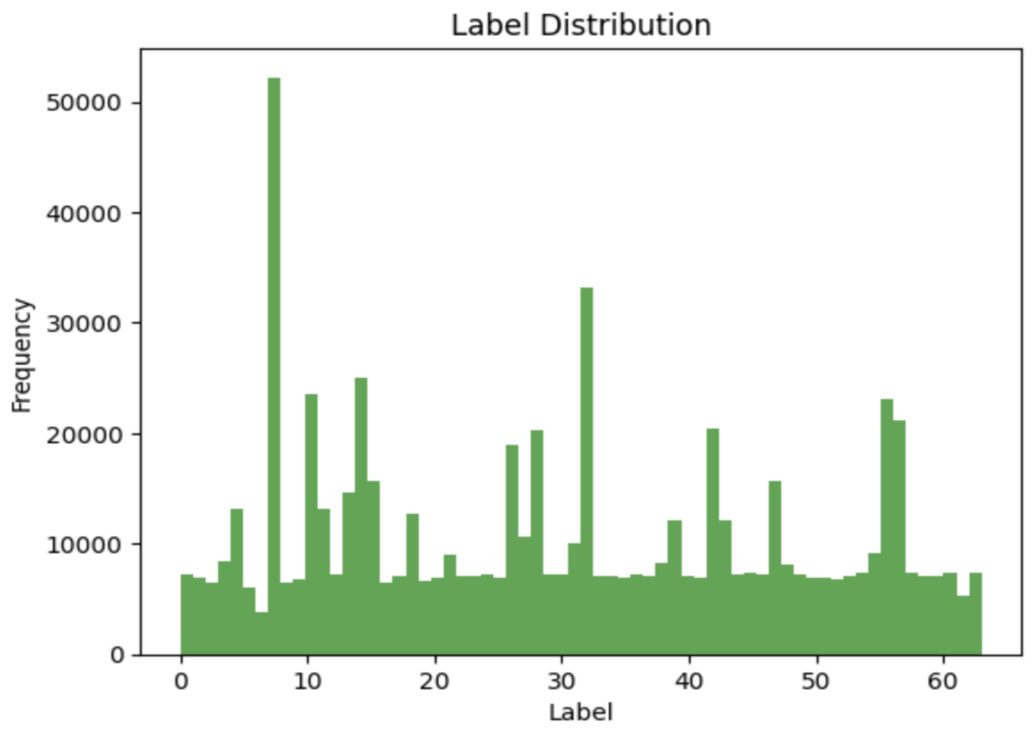
- Develop models to predict emojis from tweets.
- Analyze the relationship between text and emoji usage.
- Enhance model accuracy using various Deep Learning techniques.

Dataset:

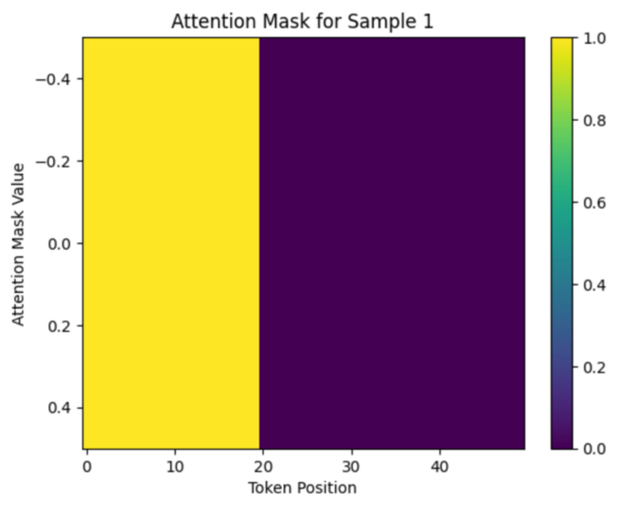
The "Twemoji Dataset" is a comprehensive collection of approximately 13 million tweets designed specifically for training and validating models that predict emojis based on text or images. This dataset is meticulously organized into distinct sets: training, validation, and testing, with each set consisting of tweets annotated with emojis to reflect the contextual or emotional undertones expressed in the text. For research involving image data, subsets of the dataset include tweets that contain images, and for these, image URLs are also provided. This ensures a rich, varied dataset that aids in enhancing machine learning models’ ability to interpret and predict emoji usage in digital communication effectively.

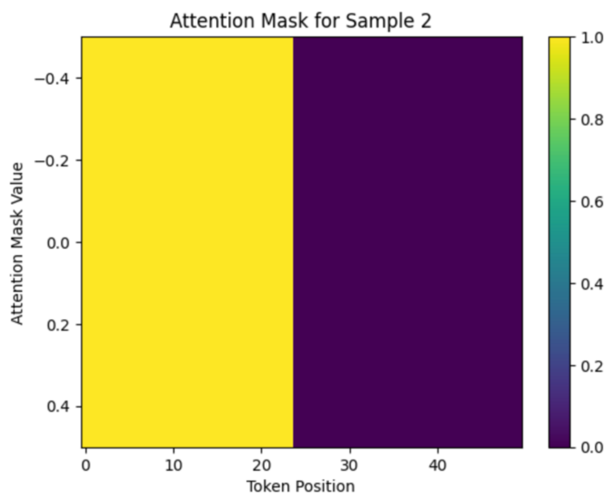


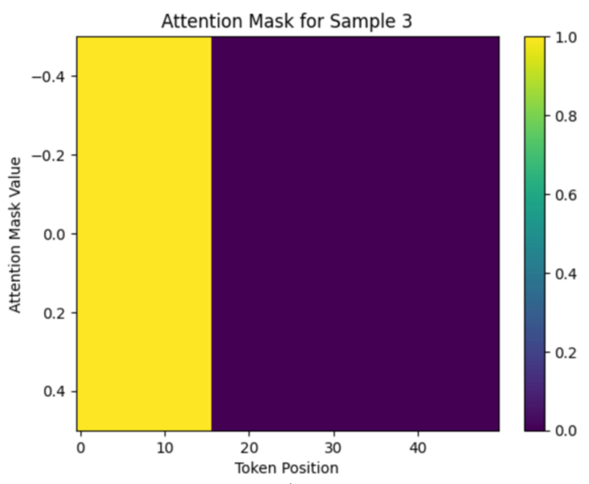
This histogram illustrates the tweet length distribution from Tweemoji dataset, where the x-axis represents tweet lengths from 0 to 100 characters and the y-axis shows their frequencies. The data reveals a bimodal distribution with peaks at approximately 20 and 40 characters, indicating a predominance of shorter tweets. As tweet length increases, the frequency sharply declines, highlighting that fewer tweets approach the 100-character limit. This distribution is key for understanding user engagement and optimizing data processing and model training in tasks like natural language processing, by guiding decisions on text segmentation and padding strategies.

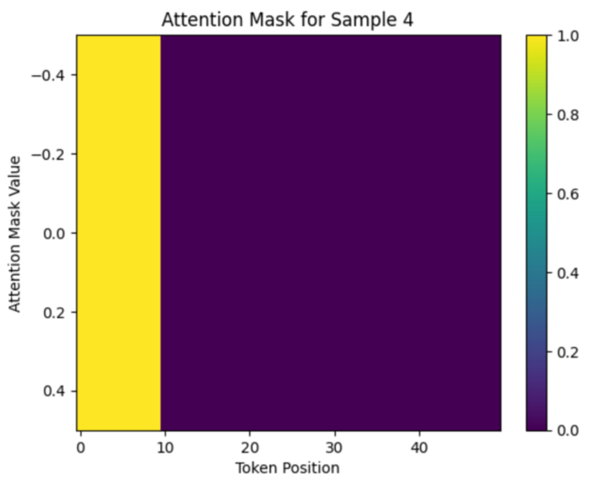


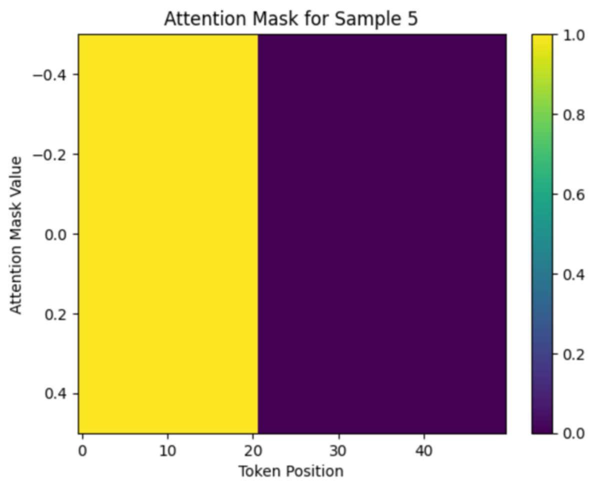
This histogram illustrates the distribution of labels in a dataset, depicted with the label numbers on the x-axis ranging from 0 to over 60 and the frequency of each label on the y-axis. The chart shows several peaks where certain labels have higher frequencies, notably one label around 5 with the highest occurrence exceeding 45,000 instances. Other significant peaks appear at labels around 15, 25, and just over 50. In contrast, many labels have much lower frequencies, suggesting a non-uniform distribution across the dataset. This pattern indicates that certain labels are much more common than others, which could influence the training and performance of machine learning models applied to this data, potentially necessitating techniques such as resampling or reweighting to address label imbalance.



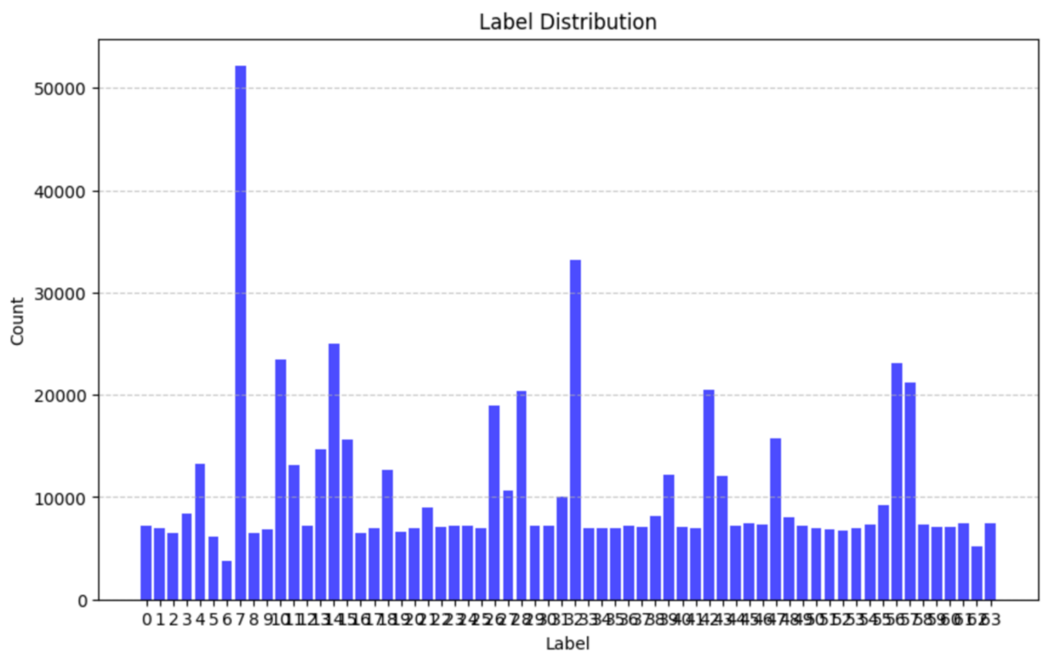




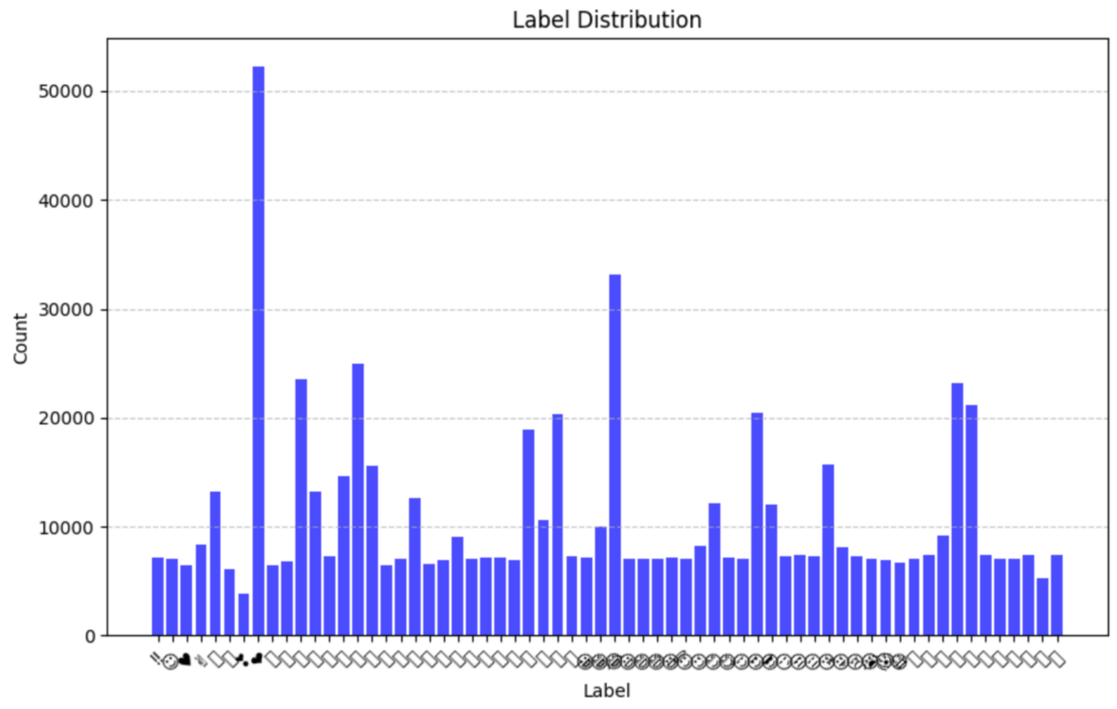




These images depict attention mask visualizations for different samples, each showing how attention is allocated across token positions. In each chart, token positions range from 0 to 40, plotted against attention values from -0.4 to 0.4. These visualizations consistently show two distinct color zones: a yellow segment from positions 0 to 10 and a purple segment from 10 to 40. This sharp transition in color indicates a significant shift in attention values, suggesting that the model focuses less on the initial tokens and more on the latter tokens in each sequence. This pattern across all samples illustrates a consistent attention strategy by the model, highlighting its prioritization of certain parts of the input data for processing and decision-making. Such insights are crucial for understanding the internal workings of the model and its approach to handling input data.



This histogram depicts the label distribution across a dataset, showing the frequency of each label on the y-axis and the label indices on the x-axis, which range from 0 to over 60. The chart highlights significant variability in label frequency, with some labels, particularly those near the indices of 5, 20, and just above 50, exhibiting much higher frequencies—reaching up to approximately 50,000 counts. Conversely, many other labels are much less frequent, with their counts distributed unevenly across the range. This uneven distribution indicates that certain labels are much more common within the dataset, which could pose challenges for machine learning models, such as biases towards more frequent labels. Such insights are crucial for guiding the development of strategies to handle class imbalance, potentially through techniques like resampling or adjusting class weights during model training. This analysis is useful for preparing the dataset for tasks where balanced representation might impact performance and fairness of predictive models.



This histogram visualizes the distribution of label counts across various categories of a dataset. On the x-axis, each category or label is represented, spanning a sequence that appears to cover over 60 distinct labels. The y-axis quantifies the count of occurrences for each label within the dataset. This histogram shows notable variability in label frequency, with several peaks indicating higher occurrences at specific labels. For example, there is a particularly high peak at the eighth label, which significantly surpasses the frequency of others, reaching more than 50,000 occurrences. Other labels exhibit moderate peaks, while many maintain lower frequency counts.

Preprocessing Steps:

For preparing the "Tweemoji Dataset" several crucial preprocessing steps were implemented on it:

1**. Data Cleaning**: The initial stage involved removing extraneous elements such as URLs, usernames, and special characters from the tweets. This step ensures that the models focus on meaningful textual content without being distracted by irrelevant data.

2. **Tokenization**: Subsequently, each tweet was tokenized, breaking the text into individual words or symbols. This process is essential for models to analyze and learn from the structure and composition of the language used in the tweets.

3. **Embedding**: After tokenization, word embeddings were applied to transform these tokens into numerical vectors. These embeddings capture the semantic meanings of words, allowing sophisticated models like LSTMs and Transformers to process and learn from the patterns in textual data effectively.

These preprocessing steps are critical for enhancing the dataset’s utility in training models, ensuring that the data fed into the models is clean, structured, and semantically rich, thereby facilitating more accurate predictions and analyses.

LSTM:

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) capable of learning order dependence in sequence prediction problems. This is crucial in tasks like emoji prediction, where the context and sequence of words significantly influence the output.

The LSTM Model is a neural network designed for sequential data processing, integrating several key components to handle tasks like language modeling effectively. It initializes with an embedding layer that maps vocabulary indices into dense vector representations based on the specified `embedding_dim` and `vocab_size`, effectively converting discrete language tokens into a format suitable for neural processing. The core of the model is composed of LSTM layers, which are configured to manage long-term dependencies within data sequences through multiple stacked layers (`n_layers`), enhanced by dropout (`drop_prob`) to prevent overfitting by randomly omitting units during training. This sequence of data processing concludes with a fully connected linear layer that maps the LSTM outputs to the desired output dimension (`output_dim`), finalizing the model's output for various predictive or classification tasks. The model's `forward` method orchestrates the flow of data through these components, ensuring efficient sequential processing and robust learning capabilities suitable for a wide range of applications that require nuanced sequence understanding.

Bi-LSTM:

Bidirectional Long Short-Term Memory (Bi-LSTM) networks are an advanced type of RNN that improve upon the traditional LSTM architecture by processing data sequences in both forward and backward directions. This dual-path processing enables the network to capture contextual relationships that a standard unidirectional LSTM might miss.

The Bi-LSTM Model is designed for sequential data tasks requiring awareness of both past and future contexts. It starts with an embedding layer that maps input words into vectors, using specified dimensions (`embedding_dim`) and vocabulary size (`vocab_size`). The core of the model is a bidirectional LSTM (`lstm`) layer, allowing it to process sequences in both directions, effectively doubling the contextual information it can leverage. This layer includes multiple stacked units (`n_layers`), enhanced by dropout (`drop_prob`) to reduce overfitting. Outputs from the LSTM are then regularized through a dropout layer before being passed to a fully connected (`fc`) layer, which transforms them into the final output dimension (`output_dim`). The bidirectional nature requires the final linear layer to handle an output size of `hidden_dim * 2` to account for both directional outputs. This model structure is particularly useful for complex tasks where understanding the complete context is crucial, such as text translation or sentiment analysis.

TextCNN:

Text Convolutional Neural Network (TextCNN) is a powerful model for processing text data. It utilizes convolutional neural layers to extract higher-level features from text, applying a series of filters for n-gram detection at different levels of abstraction.

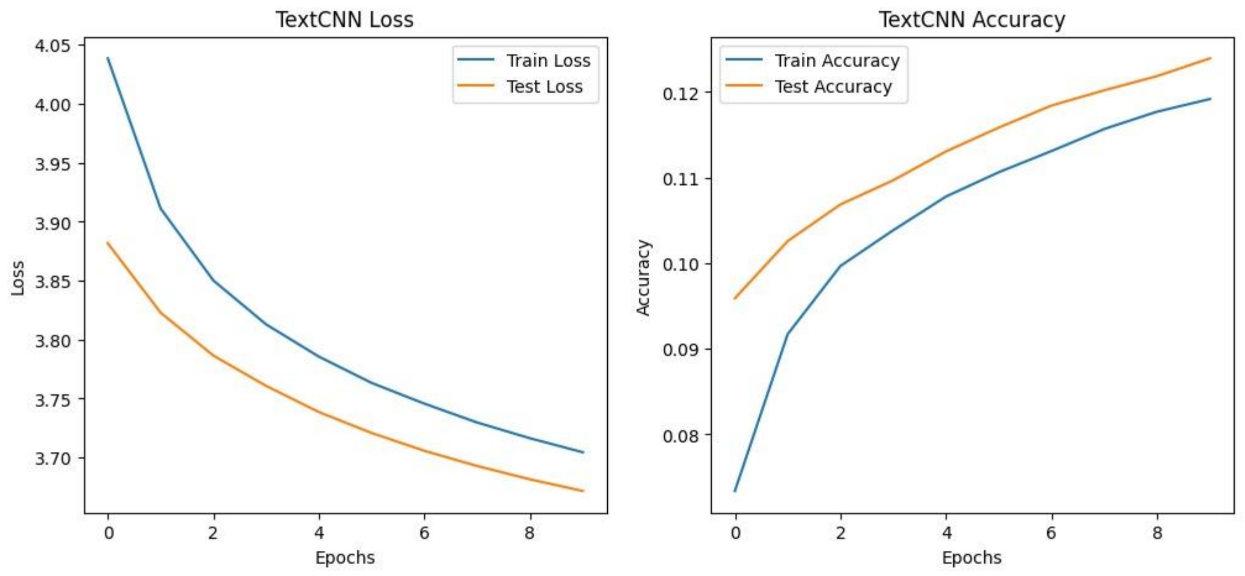
The `TextCNN` model is structured for text classification tasks, combining embeddings and convolutional layers to process and classify textual data efficiently. It begins by converting input word indices into dense vector representations using an embedding layer, with the vectors corresponding to a predefined vocabulary size and embedding dimension. The core of the model consists of multiple convolutional layers organized in a `ModuleList`, each employing a different filter size to capture textual features at various granularities. These layers use 2D convolution operations, tailored for text data by treating the sequence length and embedding dimension as spatial dimensions. After convolution, a ReLU activation is applied followed by max pooling to extract the most significant features. The output from all convolutional filters is concatenated and passed through a dropout layer for regularization, and finally, a fully connected layer maps the condensed features to the specified number of output classes. This configuration enables the model to leverage local contextual features effectively, making it suitable for complex text-related tasks like sentiment analysis or topic classification.

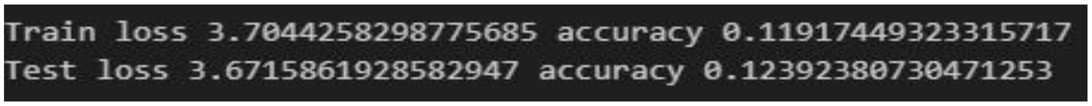
BERT:

BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art NLP model developed by Google. It uses the Transformer architecture to process words in relation to all other words in a sentence, rather than one-by-one in order. This approach enables the model to capture the context of a word based on all of its surroundings (left and right of the word).

Results and Analysis:

Performance Metrics:



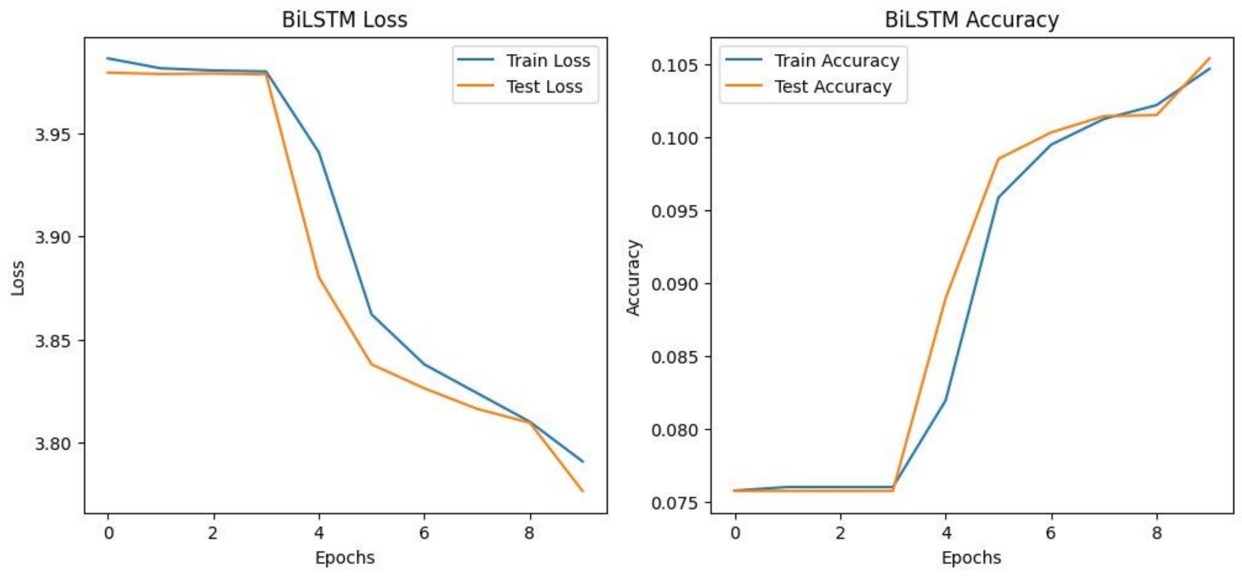


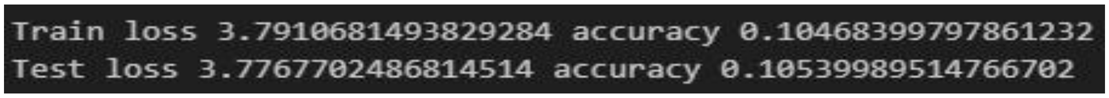
```
Train loss 3.7044258298775685 accuracy 0.11917449323315717
Test loss 3.6715861928582947 accuracy 0.12392380730471253
```

The image shows two graphs, each representing the performance metrics of a TextCNN model across 10 training epochs. TextCNN Loss graph shows the model's loss values during training, with the x-axis representing the number of epochs (from 0 to 10) and the y-axis displaying the loss magnitude. The lines depict a clear downward trend in both training (blue line) and test (orange line) loss, indicating that the model's prediction error decreases as it learns from the data over successive epochs.

TextCNN Accuracy graph shows the model's accuracy, again plotted against 10 epochs. It illustrates an upward trajectory for both training accuracy (blue line) and test accuracy (orange line), signaling improvements in the model's performance in correctly predicting outcomes as training progresses. The training accuracy is consistently higher than the test accuracy, which is common as models tend to perform better on data they have seen (training data) compared to new data (test data).

Both loss and accuracy graphs together indicate that the model is learning effectively from the training data over time, as evidenced by decreasing loss and increasing accuracy. The smoothness of both curves suggests that the training process is stable.



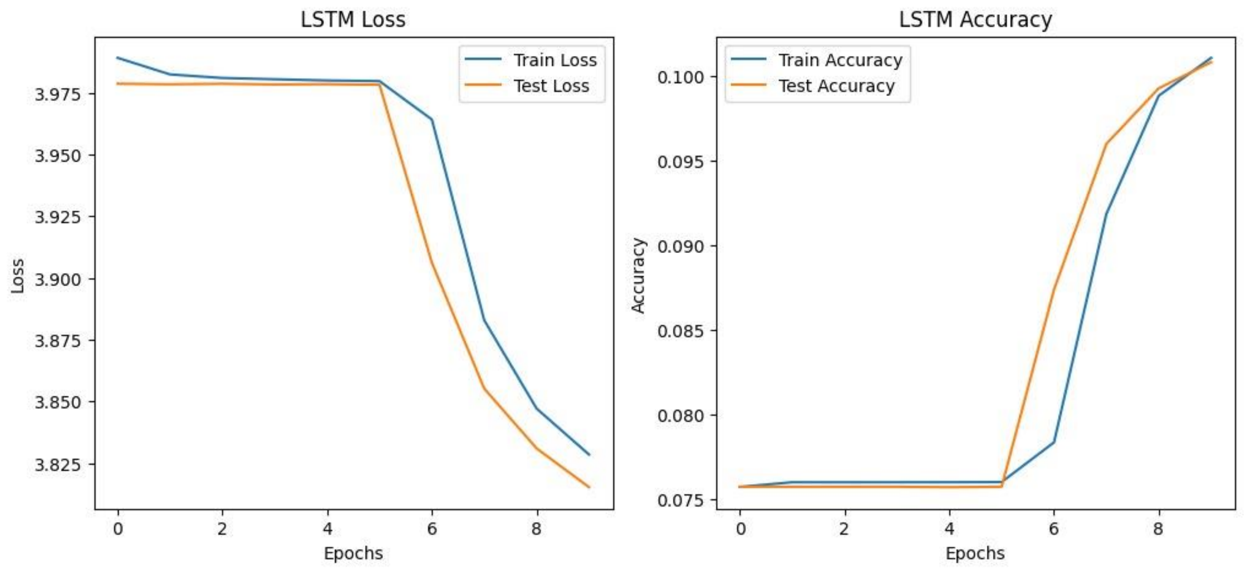


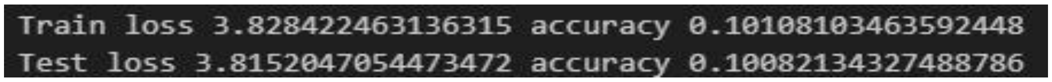
```
Train loss 3.7910681493829284 accuracy 0.10468399797861232
Test loss 3.7767702486814514 accuracy 0.10539989514766702
```

The image shows two graphs depicting the training dynamics of a Bidirectional Long Short-Term Memory (Bi-LSTM) model over a series of epochs, ranging from 0 to 10. The Bi-LSTM Loss graph tracks the model's loss for both training (blue line) and testing (orange line). Initially, both lines start at approximately the same level, but they quickly diverge, with the training loss decreasing more sharply before converging closely with the test loss towards the end of the epochs. This trend indicates that the model is effectively learning from the training data and generalizing well to unseen test data, with both losses converging to a low value, suggesting minimal overfitting.

Bi-LSTM Accuracy graph illustrates the accuracy of the model on both the training (blue line) and testing (orange line) sets. Both accuracy metrics start at similar levels, with a sharp increase after the initial epochs and continuing to rise steadily. The lines remain close throughout the training process, reflecting consistent improvements and effective learning without significant disparity between training and test performance. This close alignment between the train and test accuracy suggests that the Bi-LSTM model is robust and generalizes well, making it effective for the tasks it is trained to perform.

Overall, these graphs indicate a successful training process where the model not only learns effectively but also maintains a strong ability to generalize from its training data to new, unseen data, a key attribute in machine learning models aimed at real-world applications.





```
Train loss 3.828422463136315 accuracy 0.10108103463592448
Test loss 3.8152047054473472 accuracy 0.10082134327488786
```

The image shows two graphs shows the training and testing results for a Long Short-Term Memory (LSTM) model over 10 epochs. The graph on the left shows the model's loss ("LSTM Loss"), and the graph on the right shows the model's accuracy ("LSTM Accuracy").

LSTM Loss graph both the training (blue line) and testing (orange line) loss start at similar values, just under 4.0. As the epochs progress, there is a steep decline in both lines, indicating that the model is effectively learning and minimizing errors. By the 10 epochs, both lines converge closer to a loss value of about 3.975, with the training loss slightly lower than the test loss, suggesting good model generalization without significant overfitting.

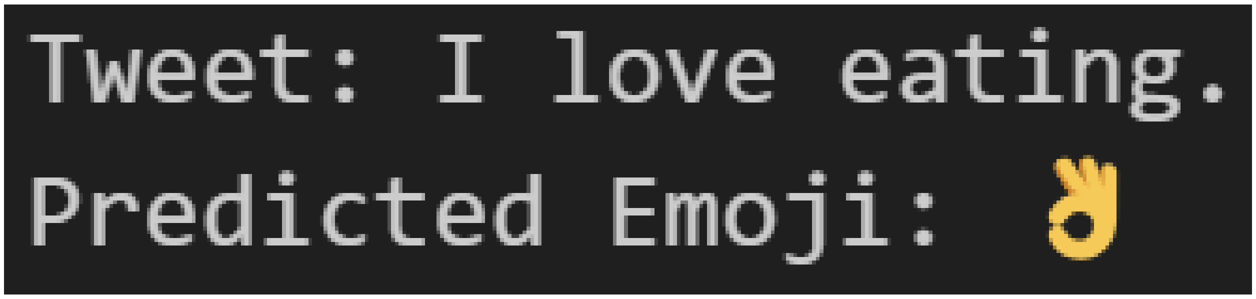
LSTM Accuracy graph shows a dramatic increase in performance over time. Both training and test accuracies start at around 7.5% and experience a sharp rise after the initial epochs. By the end of the nineth epoch, the accuracies plateau near 10%, with the training accuracy marginally outperforming the test accuracy. This progression illustrates that the model not only improves its predictive accuracy as it trains but also maintains a close performance between seen and unseen data, critical for the model's utility in practical applications.

Overall, the graphs collectively indicate a well-tuned training process where the LSTM model effectively reduces error and enhances predictive accuracy simultaneously, characteristics desirable in robust predictive models used in real-world scenarios.

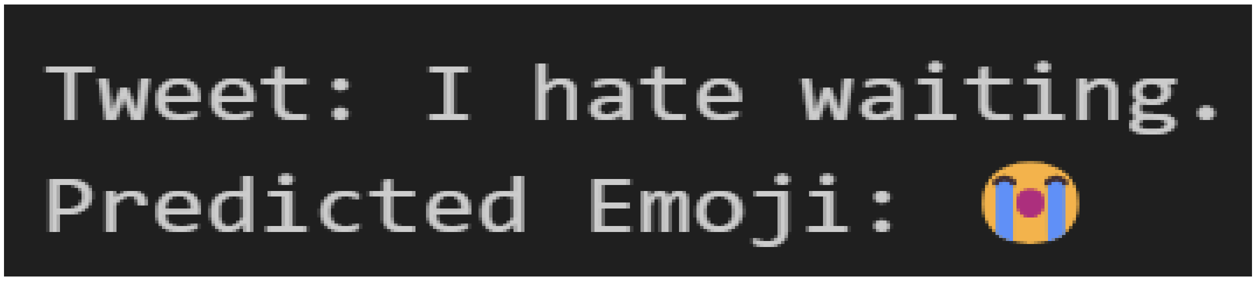
Model Performance Comparison:

- Out of all the implemented models, Text based CNN trained faster compared to Bidirectional LSTM to LSTM to BERT.
- Textbased CNN >> BidirectionalLSTM >> LSTM >> BERT.
- Performance accuracy, BERT>> Bidirectional LSTM>> Textbased CNN LSTM.
- We trained the BERT Model on our dataset to benchmark the performance of our models. The performance difference between the models over 10 epochs was found out to be 9%, while BERT taking the largest amount of time to train.

Predictions:



```
Tweet: I love eating.
Predicted Emoji: 👌
```



```
Tweet: I hate waiting.
Predicted Emoji: 😭
```

Future Work:

Further research will focus on improving the model’s accuracy with more complex neural network architectures and expanding the dataset to include diverse forms of text beyond tweets. Additionally, exploring multilingual emoji prediction to cater to a global audience will be considered.

References:

[1] Emojify: Prediction Emoji from Sentence

[2] Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm.

[3] Context-Aware Emoji Prediction Using Deep Learning:

[4] Emoji Prediction: Extensions and Benchmarking