DON BOSCO INSTITUTE OF TECHNOLOGY

Premier Automobiles Road, Kurla (W), Mumbai-70

Approved by AICTE, Govt. of Maharashtra

&

Affiliated to the University of Mumbai

T.E. MINI PROJECT REPORT

On

“Emotion Detection based on Text and Emojis”

Under the Guidance of:

Dr. Phiroj Shaikh

Department of Computer Engineering

University of Mumbai

April 2024

CERTIFICATE

Project Title	:	**Emotion Detection based on Text and Emojis**
Organization	:	Don Bosco Institute of Technology
Address	:	Premier Automobiles Road, Kurla (W), Mumbai-70
Project Team Members	:	1. Anushka Joseph 2. Shanaya Carvalho 3. Nicole Saldanha
Internal Guide	:	**Dr. Phiroj Shaikh**

Internal Examiner **External Examiner**

HOD, COMPUTER ENGINEERING

ABSTRACT

Opinion mining, the practice of extracting insights from social media content, has become increasingly vital for informed decision-making in today's digital age. With the sheer volume of online information, the need for automation to effectively analyze this data is paramount. Understanding the emotions conveyed within text is of utmost importance, as manually dissecting extensive text samples is not only time-consuming but also prone to interpretation errors. Emojis, which have become ubiquitous in online communication, play a pivotal role, especially in discerning the nuances of sarcastic text.

To address these challenges, a novel emotion detection model is introduced, aimed at improving emotion recognition accuracy by combining textual content and emojis. This approach finds applications in diverse fields, including sentiment analysis, feedback assessment, mental health monitoring, and user-centric applications.

The research endeavor begins with the creation of a diverse dataset comprising both text and emoji-based content, meticulously labeled with emotional indicators. To prepare the data for analysis, several preprocessing steps are applied, including tokenization, text normalization, and the extraction of emojis. Features are subsequently engineered, incorporating word embeddings and emoji representations. The heart of the model lies in a deep neural network designed for handling multimodal input. Training occurs through multi-label classification to predict the appropriate emotional category.

Empirical results validate the model's effectiveness, notably surpassing text-only models in emotion recognition, especially in cases where emojis are relied upon for conveying sentiment. What sets this model apart is its adaptability across various platforms and languages, recognizing the ever-evolving nature of digital communication.

As digital communication continues to evolve and diversify, it is imperative that analytical tools evolve in parallel to meet the changing needs of users and organizations. This evolution ensures that insights derived from opinion mining remain insightful, relevant, and invaluable for decision-making in the dynamic digital landscape.

TABLE OF CONTENT

TABLE OF FIGURES

TABLE OF TABLES

ABBREVIATIONS

BERT	Bidirectional Encoder Representations from Transformers
Bi-LSTM	Bidirectional Long Short-Term Memory
CBOW	Continuous Bag of Words
CNN	Convolutional Neural Network
GloVe	Global Vectors
LR	Logistic Regression
LSA	Latent Semantic Analysis
LSTM	Long Short-Term Memory
MATLAB	Matrix Laboratory
MNB	Multinomial Naive Bayes
NB	Naive Bayes
NLP	Natural Language Processing
SVM	Support Vector Machine
TF-IDF	Term Frequency - Inverse Document Frequency

CHAPTER 1: INTRODUCTION

Opinion mining, a critical area of study, involves the extraction of valuable insights from the vast sea of social media content, and it holds increasing importance in supporting decision-making processes. The enormous amount of information available online has made it imperative to automate the analysis of this data efficiently. One key aspect in this endeavor is understanding the emotions conveyed in text, as manually analyzing large text samples is not only time-consuming but also prone to subjective interpretation errors. Emojis are especially significant in this context, as they can help in deciphering the often-elusive emotions expressed in sarcastic text. To address these challenges, a novel approach is suggested, wherein an emotion detection model is developed to combine both textual and emoji data, ultimately leading to improved accuracy in emotion recognition. This innovative model finds applications in various domains such as sentiment analysis in social media, feedback analysis, mental health monitoring, and user-centric applications.

The research methodology encompasses several important steps. It begins with the creation of a diverse dataset containing both text and emoji-based content, meticulously annotated with detailed emotional labels. The preprocessing phase involves essential tasks like tokenization, text normalization, and emoji extraction, which are crucial for preparing the data for analysis. Furthermore, features are engineered using techniques like word embeddings and emoji representations to effectively capture the emotional content of the input data. The model itself is a deep neural network designed to handle multimodal input, and it is trained using a multi-label classification approach to predict the most appropriate emotional category for a given input.

The results of experimental testing provide compelling evidence of the model's effectiveness. It outperforms text-only models, particularly in recognizing nuanced expressions that heavily rely on emojis. Moreover, the model demonstrates adaptability across different platforms and languages, making it well-suited to the ever-evolving nature of digital communication.

In the context of its implementation, it's important to note that the model achieves an accuracy rate of 77.6%. The implementation encompasses the processing of both text and emojis, with text preprocessing being a crucial step. The inclusion of an emoji database further enriches the model's ability to interpret emotions accurately. This implementation is developed using the

Python programming language, a popular choice for machine learning and natural language processing tasks. To train and evaluate the model, datasets are used, typically consisting of a training dataset for model training and a test dataset for evaluating its performance and generalizability.

In conclusion, the integration of both text and emojis in emotion detection holds significant potential for enhancing sentiment analysis and human-computer interaction. By combining these modalities, the model provides a more comprehensive understanding of emotions in digital content, which can positively impact user experiences, mental health support, and the overall quality of digital communication. As digital communication continues to evolve, analytical tools like this must evolve in tandem to meet the diverse needs of users and organizations.

CHAPTER 2: LITERATURE SURVEY

2.1 SURVEY EXISTING SYSTEM

2.1.1 Sentiment Analysis using Text and Emoji's (2023)

The research paper titled "Sentiment Analysis using Text and Emoji's 2023" explores the field of sentiment analysis, focusing on the combined use of textual content and emojis. Below, we provide an overview of the key aspects of the paper:

- **Text Preprocessing with Natural Language Processing (NLP):** The paper employs natural language processing techniques for text preprocessing. This step is crucial in preparing the textual data for sentiment analysis, involving tasks like tokenization, text normalization, and potentially removing noise and irrelevant information from the text.
- **Emojis as a Universal Language for Expressing Emotions:** One notable highlight of the paper is its emphasis on emojis as a universal language for expressing emotions. The study observes that there is a small difference in opinion expressed between comments with emojis and comments without emojis. This finding suggests that emojis play a significant role in conveying sentiment and emotions in digital communication.
- **Comparison of Opinion in Texts with and without Emojis:** The paper delves into the differences in opinion expressed in texts that include emojis and those that do not. It seeks to understand how the presence or absence of emojis impacts the sentiment expressed in the text, and whether the opinion on a product or topic remains consistent.
- **Evaluation Metrics:** The paper employs evaluation metrics based on different machine learning models, including Support Vector Machine (SVM), Linear Regression, and Tuned Logistic Regression. These metrics are used to assess the performance of the sentiment analysis model and its ability to accurately classify the sentiment expressed in the text.

However, it's important to note that the paper has some limitations:

- **Limited Model Evaluation:** The paper focuses on a specific set of machine learning models (SVM, Linear Regression, Tuned Logistic Regression) for sentiment analysis. It does not evaluate the efficiency of other machine learning or deep learning models. A more comprehensive evaluation could provide a broader understanding of the model's performance.
- **Lack of Feature Classification:** The paper does not delve into further classification of features such as distinguishing between positive, negative, or neutral feedback. This could offer a more nuanced and detailed understanding of sentiment in the text.

In summary, the paper explores sentiment analysis by considering both textual content and emojis, highlighting the significance of emojis in conveying emotions. While it provides valuable insights into the role of emojis in sentiment expression, there are limitations in terms of model evaluation and feature classification. Further research in these areas could enhance the understanding and applicability of sentiment analysis in the context of text and emojis.

2.1.2 Contextual Emotion Detection in Text using Deep Learning and Big Data (2022)

The research paper titled "Contextual Emotion Detection in Text using Deep Learning and Big Data 2022" explores the field of emotion detection in text, with a focus on contextual understanding. Below, we provide an overview of the key aspects of the paper:

- **Contextual Emotion Detection:** The primary focus of the paper is on detecting emotions in text within a contextual framework. This means that the research aims to go beyond simple emotion classification and takes into account the surrounding context of the text to better understand and interpret the emotions expressed. This is a more advanced and sophisticated approach to emotion detection.
- **In-Depth Text Analysis:** The paper delves into a comprehensive analysis of textual data. This implies that the research goes beyond the surface-level analysis and delves into the intricate details of the text to uncover the nuances of emotional expression.

- **Emotion Understanding for Data Summarization:** Another significant aspect of the research is the use of emotion understanding as a tool for summarizing data. This means that the emotional content of the text is harnessed to create meaningful and concise summaries of the data. This has applications in various fields, including content summarization and sentiment analysis.
- **LSTM Model for Emotion Detection:** The paper utilizes a Long Short-Term Memory (LSTM) model, a type of deep learning model, for detecting emotions in text. LSTMs are well-suited for sequence data, making them effective for tasks like sentiment analysis and emotion detection in text.

However, there are some notable limitations to this research:

- **Emoji Data Not Considered:** The paper does not take into consideration the use of emojis in text, which can often carry significant emotional cues. Emojis are a common way for individuals to express emotions in digital communication, and their omission could impact the model's ability to accurately detect emotions.
- **Decent Results for Emotion Detection:** While the research provides valuable insights into contextual emotion detection, the paper notes that the results are "decent" for

emotion detection. It would be beneficial to have a more detailed discussion of the model's performance and potential areas for improvement.

In summary, the paper "Contextual Emotion Detection in Text using Deep Learning and Big Data" focuses on the advanced task of detecting emotions in text with a contextual approach. It employs deep learning techniques, particularly LSTM models, for this purpose. However, the omission of emoji data and the limited discussion of the model's performance are aspects that warrant further exploration and improvement in future research.

2.1.3 Comparing Deep Learning Architectures for Sentiment Analysis on Drug Reviews (2020)

The research paper titled "Comparing Deep Learning Architectures for Sentiment Analysis on Drug Reviews" conducted in 2020 investigates the performance of various deep learning models in the context of sentiment analysis applied to drug reviews. Here's an overview of the key findings and aspects of the paper:

- **In-Depth Comparison of Deep Learning Models:** The paper offers an extensive exploration of three distinct deep learning architectures: Convolutional Neural Networks (CNN), Bidirectional Long Short-Term Memory (Bi-LSTM), and BERT (Bidirectional Encoder Representations from Transformers). These models are evaluated to determine their effectiveness in sentiment analysis.
- **Model Evaluation on Datasets:** The research conducts evaluations on different datasets, specifically a 10-class dataset and a 3-class dataset. The results are analyzed in relation to these datasets to assess the model performance across varying levels of classification complexity.
- **Hybrid Model Performance:** The paper's findings indicate that the best results for sentiment analysis on the 10-class dataset are achieved with a hybrid model composed of a Bi-LSTM followed by a CNN. This hybrid approach outperforms the individual models in this context.
- **BERT's Computational Cost:** It is noted that using BERT, while providing strong results, considerably increases the computational cost. BERT is known for its deep and complex architecture, which can demand significant computational resources.
- **CNN Performance for Data with Few Instances:** The paper reveals that CNN models, particularly those initialized with static word embeddings, exhibit lower

performance when dealing with classes that have fewer instances in the training dataset. However, CNN still provides acceptable results and requires less training time compared to other models.

The models evaluated in the research include CNN, Bi-LSTM, and BERT. These deep learning architectures are commonly used in natural language processing tasks like sentiment analysis.

It's essential to acknowledge that the research, despite its comprehensive evaluation of deep learning models, does not involve the analysis of emoji data. Emojis can play a significant role in conveying sentiment and emotional nuances in text, and their exclusion from the analysis may limit the model's ability to capture the full spectrum of sentiment expressed in the drug reviews.

In summary, the research paper "Comparing Deep Learning Architectures for Sentiment Analysis on Drug Reviews" provides valuable insights into the performance of different deep learning models in the specific context of drug review sentiment analysis. The choice of the best-performing model depends on the dataset's complexity and the trade-off between performance and computational cost. However, it's important to note that the analysis does not consider emoji data, which can be a noteworthy factor in sentiment analysis, especially in modern digital communication.

2.1.4 Early and Late Fusion of Emojis and Text to Enhance Opinion Mining (2021)

The research paper titled "Early and Late Fusion of Emojis and Text to Enhance Opinion Mining" in 2021 explores the integration of emojis and text for the purpose of improving opinion mining. Here are the key findings and aspects of this paper:

- **Challenges with Emojis:** The paper highlights two significant issues associated with emojis in the context of opinion mining - ambiguity and misinterpretation of emojis' sentiment. Emojis can carry multiple meanings and may not always be correctly interpreted, making their integration into sentiment analysis a challenging task.
- **Three Levels of Preprocessing:** The research introduces a structured approach to preprocessing data at three different levels, depending on the type of features involved. These levels include emojis-based features, structural-based features, and text-based features, each of which is tailored to handle the specific data characteristics.
- **Improved Performance with Emojis-Text Fusion:** One of the key findings is that the fusion of emojis with text significantly enhances the performance of opinion mining. This enhancement is observed across various evaluation metrics, including recall, precision, F1 score, geometric mean, and accuracy. The fusion of these two modalities proves to be beneficial in improving the accuracy of sentiment analysis.
- **Optimal Performance with Skip Gram and Emojis:** The paper identifies that the highest level of performance is achieved when using Skip Gram with Emojis at the score level. This specific combination of features and embedding techniques seems to be particularly effective in enhancing opinion mining results.

The paper differentiates between preprocessing techniques and features used for the various modalities:

- **Emoji-Based Features:** This category includes features such as Emojis frequency, Lexicon-based, Emojis-CBOW, and Emojis-Skip Gram.
- **Structural-Based Features:** No preprocessing operations are conducted on this type of feature.
- **Textual Features:** This category involves features like tf-idf, LSA (Latent Semantic Analysis), word2vec CBOW, and Skip grams.

Despite its valuable contributions, the research paper does have some limitations:

- **Monolingual Approach:** The proposed methodology is explored only for one language. It would be beneficial to assess its performance in multilingual contexts, as the interpretation and impact of emojis may vary across languages.
- **Lack of Comparison with Other Topic Modeling Methods:** The paper does not investigate how this methodology performs in comparison to other topic modeling methods, such as non-negative matrix factorization and latent Dirichlet allocation. These comparisons could provide a broader understanding of the approach's effectiveness.
- **Cross-Linguistic Variation of Emojis:** The variations in the impact and interpretation of emojis across multiple languages are not explored. Understanding how emojis are perceived and used differently in various linguistic and cultural contexts could be valuable.

In conclusion, the research paper "Early and Late Fusion of Emojis and Text to Enhance Opinion Mining" sheds light on the challenges and advantages of integrating emojis and text in sentiment analysis. The findings suggest that this fusion improves opinion mining performance, particularly when using Skip Gram with Emojis at the score level. However, the paper's scope is limited to a single language, and it does not compare its approach to other topic modeling methods or explore cross-linguistic variations in emoji interpretation.

2.1.5 Emotion Detection of Contextual Text using Deep Learning (2020)

The research paper titled "Emotion Detection of Contextual Text using Deep Learning" in 2020 focuses on emotion detection within the context of textual conversations. Here are the key points and aspects of the paper:

- **Naive Approach for Data Analysis and Summarization:** The paper introduces a naive approach to analyze and summarize data with the aim of extracting meaningful information. This approach is employed to process textual data and identify emotional cues within a conversation.
- **Utilization of Long Short Term Memory (LSTM) Model:** Deep learning techniques are employed for emotion detection, particularly the Long Short Term Memory (LSTM) model. LSTMs are well-suited for sequence data and are effective in capturing dependencies within conversations.
- **Detection of Emotions in Contextual Conversations:** The primary objective of the research is to detect emotions like happiness, sadness, and anger within contextual conversations. This adds a layer of complexity, as emotions may be influenced by the context in which they are expressed.
- **Improvement in F-Scores:** The paper reports that the output results demonstrate substantial improvements in F-scores over the model baseline. The Aimens system score is cited as 0.7185, indicating the effectiveness of the approach in detecting emotions in contextual text.

The paper leverages various techniques and methods for its research:

- **Deep Learning:** Deep learning methods, particularly the LSTM model, are used for emotion detection.
- **Word Embeddings:** Word2Vec and Doc2Vec embeddings are employed to represent words and documents as numerical vectors, making them suitable for machine learning and deep learning models.
- **Keyword and Lexicon-Based Methods:** These methods are likely used to analyze text for emotional content and sentiment.
- **Machine Learning and Hybrid Methods:** These approaches may be used in combination with deep learning to enhance emotion detection accuracy.

- **Data Preprocessing:** The input data goes through several preprocessing steps, including spell check and word embedding using Word2Vec, GloVe, and FastText. The data is also split into training and validation datasets for model training.

However, the paper has certain limitations:

- **Exclusion of Emojis:** The research does not take into consideration the role of emojis in conveying emotions in textual conversations. Emojis are a common and important aspect of digital communication that can carry emotional cues.
- **Lack of Model Evaluation:** The paper does not provide an evaluation of the model's effectiveness or accuracy. A comprehensive evaluation would be important to assess the model's performance and reliability in emotion detection tasks.

In summary, the research paper "Emotion Detection of Contextual Text using Deep Learning" presents an approach to detect emotions within contextual conversations using deep learning techniques, with a specific focus on LSTM models. While it shows improvements in F-scores, it does not consider emojis and lacks an evaluation of the model's effectiveness, leaving room for further investigation and refinement in future studies.

2.1.6 Automatic Emoji Insertion Based on Environment Context Signals for the Demonstration of Pervasive Computing Features (2021)

The research paper titled "Automatic Emoji Insertion Based on Environment Context Signals for the Demonstration of Pervasive Computing Features" in 2021 introduces a novel methodology for automatically inserting emojis into text based on environmental context signals. Here are the key points and features of the paper:

- **Automatic Emoji Insertion Methodology:** The paper proposes a method for the automatic insertion of emojis into text based on the input provided. This approach aims to enhance the expression of emotions and sentiments in text messages.
- **Matlab-Based Instant Messaging Tool:** The research presents a Matlab-based instant messaging tool that incorporates automatic emoji insertion. This tool is designed to facilitate real-time communication while automatically enhancing text messages with emojis.
- **Fast and Accurate Stand-Alone Implementation:** When implemented as a stand-alone system, the methodology demonstrates fast response times and high accuracy. This suggests that the approach can effectively enhance text messages with emojis in a real-time messaging environment.
- **Context Detection Systems:** The paper highlights the importance of context detection systems. Before inserting emojis, the methodology processes context signals, ensuring that the emojis selected are relevant to the conversation's context.
- **Two-Way Application with ThingSpeak:** The research introduces a two-way application that uses ThingSpeak as a middleware. This application replaces specific letters in the text with emojis based on input values from various sensors that indicate the environmental context. The input values play a significant role in determining the type of emoji to be inserted.

Despite its strengths, the research paper has some limitations:

- **Performance Variability:** The paper acknowledges that the system's response time and accuracy may vary depending on the priorities considered. This suggests that the methodology's performance may be influenced by the specific criteria used in different scenarios.

- **Performance When Stand-Alone**: While the methodology is fast and accurate when implemented stand-alone, it is mentioned that it can become slow and less accurate under certain conditions or configurations. Understanding the factors influencing its performance is crucial for practical applications.

In conclusion, the research paper "Automatic Emoji Insertion Based on Environment Context Signals for the Demonstration of Pervasive Computing Features" introduces a creative method for enhancing text messages with emojis based on environmental context signals. However, its performance may vary based on priorities, and further exploration is required to address potential limitations. This methodology represents an interesting approach to enriching digital communication with emojis in context-aware environments.

2.1.7 Sentiment Analysis of Tweets Including Emoji Data (2017)

The research paper titled "Sentiment Analysis of Tweets Including Emoji Data" in 2017 investigates the sentiment analysis of tweets, with a specific focus on incorporating emoji data. Here are the key findings and aspects of the paper:

- **Comparison of Machine Learning Models:** The paper conducts a comparative analysis of several machine learning models, including Multinomial Naive Bayes (MNB), Support Vector Machine (SVM), and Bag of Words and Bigram models. These models are assessed for their performance in sentiment analysis, considering both traditional text data and the inclusion of emojis.
- **Model Performance Variability:** The research highlights the performance variability of MNB and SVM models. SVM outperforms MNB when working with smaller vocabulary sizes, while MNB is more effective for larger vocabulary sizes. This suggests that the choice of model can depend on the dataset's characteristics and complexity.
- **Sparse Bigram Vectors for Short Tweets:** The paper observes that bigram vectors, which consider pairs of words as features, tend to be very sparse when applied to short tweets. This sparsity can impact the models' performance.
- **Bag of Words Preferred Over Bigram:** Despite the potential advantages of bigram models, the research concludes that a Bag of Words representation with an MNB classifier is the preferred choice for larger vocabulary sizes when incorporating emoji

data. This approach achieves the highest classification accuracy while also saving computation time.

The paper explores various machine learning models and techniques, including:

- **Support Vector Machine (SVM):** SVM is a popular classification algorithm that is used in the sentiment analysis of tweets. It performs well for smaller vocabulary sizes.
- **Multinomial Naive Bayes (MNB):** MNB is another classification algorithm, often used for text classification tasks. It performs well for larger vocabulary sizes.
- **Bag of Words Model:** This model represents text data by counting the frequency of each word, disregarding word order. It is preferred for larger vocabulary sizes in combination with emoji data.
- **Bigram Model:** The bigram model considers pairs of words as features. It is suitable for text analysis but may result in sparse vectors for short tweets.

Despite its contributions, the paper has certain limitations:

- **Limited Experimentation:** The paper conducts less experimentation on varying the training/testing ratio to produce learning curves. A more thorough exploration of the impact of different ratios on model performance would provide valuable insights.
- **Limited Representation Formats:** The paper focuses on Bag of Words and Bigram representations. There could be other representation formats, such as word embeddings (e.g., Word2Vec or GloVe), that might offer different perspectives and improve sentiment analysis.

In summary, the research paper "Sentiment Analysis of Tweets Including Emoji Data" offers valuable insights into the sentiment analysis of tweets, considering both traditional text data and the incorporation of emoji data. It highlights the variability in model performance and provides recommendations for the choice of models and representations based on vocabulary size. However, further experiments and exploration of different representation formats could enhance the study's comprehensiveness.

2.1.8 Comparison Research on Text Pre-processing Methods on Twitter Sentiment Analysis (2017)

The research paper titled "Comparison Research on Text Pre-processing Methods on Twitter Sentiment Analysis" in 2017 explores various text pre-processing methods and their impact on sentiment analysis in the context of Twitter data. Here are the key findings and aspects of the paper:

- **Classifier Models and Pre-processing:** The paper discusses the effects of various classifier models used for text pre-processing. Classifier models are essential components of sentiment analysis, and the research evaluates how different pre-processing techniques affect their performance.
- **Evaluation of Pre-processing Methods:** The research systematically evaluates the classification performance of six pre-processing methods. These methods are assessed using two feature models and four classifiers on five Twitter datasets. This comprehensive evaluation allows for a robust comparison of pre-processing techniques.
- **Impact of Pre-processing on Sentiment Classification:** One of the key objectives of the study is to assess the effect of pre-processing on sentiment classification. This analysis helps identify which pre-processing techniques are most effective in improving sentiment analysis accuracy.
- **Supervised Classifiers:** The research employs four popular supervised classifiers commonly used in sentiment analysis, including Support Vector Machine, Naive Bayes, Logistic Regression, and Random Forest. These classifiers are evaluated using various pre-processing methods to determine their performance.
- **Optimal Parameter Selection:** To ensure the best results, the research employs a GridSearch approach to find the optimal parameters for the classifiers. This parameter tuning helps maximize the classification accuracy of the sentiment analysis models.

The paper employs two feature models for sentiment analysis:

- **Word n-grams Features Model:** This model is based on analyzing sequences of words (n-grams) to capture the context and semantics of the text data.
- **Prior Polarity Score Feature Model:** This feature model likely leverages sentiment lexicons or prior polarity scores to analyze and assess the sentiment of text.

While the paper offers valuable insights into pre-processing methods and their impact on sentiment analysis, it does have a notable limitation:

- **Exclusion of Emoji Datasets:** The research does not work on emoji datasets, which can be an important component of sentiment analysis on social media platforms like Twitter. Emojis are commonly used to convey emotions, and their omission could limit the research's applicability in real-world scenarios involving Twitter data.

In conclusion, the research paper "Comparison Research on Text Pre-processing Methods on Twitter Sentiment Analysis" provides a comprehensive analysis of pre-processing techniques and their effects on sentiment analysis. It evaluates various classifiers and feature models, contributing to a better understanding of how to improve sentiment analysis accuracy on Twitter data. However, the absence of emoji dataset analysis is a notable limitation that could be addressed in future research to make the findings even more applicable to social media sentiment analysis.

2.1.9 A Survey on Emotion Detection from Text in Social Media Platforms (2021)

- **Objective: Accurate Emotion Classification and Sentiment Analysis:** The primary objective of the paper is to achieve accurate emotion classification and sentiment analysis from text data on social media platforms. Emotion detection is a crucial task, given the widespread use of social media for expressing emotions and sentiments.
- **Emotion Models:** The paper explores various models and classifications of emotions. Two notable models mentioned are Ekman's model, which includes emotions like Happy, Anger, Sad, Disgust, Fear, and Surprise, and Plutchik's model, which categorizes emotions into pairs like anger-fear and joy-sadness.
- **Systematic Review:** The research paper provides a systematic review of publications related to the detection of textual emotions in social media platforms. It covers state-of-the-art methods, current research trends, and existing challenges in this domain. This survey serves as a valuable resource for researchers and practitioners interested in emotion detection from social media text.
- **Utilization of Machine Learning:** Machine learning techniques play a central role in emotion detection, and the paper likely discusses various machine learning approaches and methods for this task.

- **Limitations:** The paper acknowledges certain limitations, including casual writing style and the lack of mechanisms to address grammatical errors. Despite these limitations, the research is reported to provide decent efficiency in its objectives.

In summary, the research paper "A Survey on Emotion Detection from Text in Social Media Platforms" serves as a valuable resource for those interested in understanding the challenges and advancements in emotion detection and sentiment analysis in social media. It explores different emotion models, provides a systematic review of relevant publications, and highlights the role of machine learning in achieving accurate emotion classification. Despite some writing style and grammatical issues, the paper contributes to the field of emotion detection by summarizing the state of the art and existing challenges.

2.1.10 SUMMARY OF THE LITERATURE SURVEY

ARTICLE	AUTHORS	MAIN POINTS	METHODS USED	LIMITATIONS
Sentiment Analysis using Text and Emoji's (2023)	H. Sakode, T. L. Surekha, M. D. Sri and G. B. Kumar	• Employs natural language processing for text preprocessing. • Highlights emojis as a universal language for expressing emotions as a small difference is observed in the Emoji translated text in the comments and without the emoji text in the comments and opinion on the product are likely same. • Compares difference in opinion in texts with emojis and text without emojis.	• Evaluation metrics based on SVM • Linear Regression • Tuned Logistic Regression	• Does not evaluate the comparison in efficiency for other Machine Learning and Deep Learning Models. • There is no further classification of features like positive, negative or neutral feedback.
Contextual Emotion Detection in Text using Deep Learning and Big Data (2022)	P. Chandra et al.	• Detects contextual emotion • Analyses Text data in depth • Uses emotion understanding in order to summarize data.	• LSTM model to detect emotions	• Emoji data is not taken into consideration • Provides decent results for emotion detection
Comparing deep learning architectures for sentiment analysis on drug reviews (2020)	Cristóbal Colón-Ruiz, Isabel Segura-Bedmar	• Provides an in-depth research conducted using three models wiz CNN, Bi-LSTM and BERT • Evaluation based on 10-class dataset show that the hybrid model composed of a Bi-LSTM followed by a CNN provides the best results. • 3-class dataset, we can observe that BERT followed by a Bi-LSTM provides slightly better	• CNN • Bi-LSTM • BERT	• Evaluation does not involve emoji data.

		results than the other models. • Using BERT considerably increases the computational cost. • CNN models initialized with static word embedding show very low performance for classes with fewer instances in the training set. However, CNN provides acceptable results while requiring less training time.		
Early and late fusion of emojis and text to enhance opinion mining (2021)	S. Al-Azani and E. -S. M. El-Alfy	• Two emoji-related issues are highlighted: ambiguity and misinterpretation of emojis' sentiment. • The preprocessing step is conducted at three levels relying on the type of features: emojis based features, structural-based features or text-based features. • Fusing emojis with text has improved the performance at all fusion levels in terms of recall, precision, F1 score, geometric mean and accuracy. • The highest performance is achieved when using Skip Gram with Emojis at the score level.	• For emoji-based features (i.e., Emojis frequency, Lexicon-based, Emojis-CBOW and Emojis-Skip gram) • Structural-based features: No preprocessing operations are conducted. • Textual features: (tf-idf, LSA, word2vec CBOW and Skip grams)	• The proposed methodology is not explored for other languages. • No investigation the performance with other methods of topic modeling such as non-negative matrix factorization and latent Dirichlet allocation. • The variations of impact and interpretation of emojis across multiple languages.
Emotion Detection of Contextual Text using Deep learning (2020)	U. Rashid, M. W. Iqbal, M. A. Skiandar, M. Q. Raiz, M. R. Naqvi	• Proposes a Naive approach to analyze and summarize data to extract meaningful information. • Uses Long Short Term Memory (LSTM) Model based on Deep Learning.	• Deep Learning • word2vec and doc2vec embeddings • Keyword based Method • Lexicon Based Method	• Emojis are not taken into consideration. • Does not evaluate the effectiveness or accuracy of the model

	and S. K. Shahzad	• Detects emotions like happy, sad and angry in contextual conversation. • Output results are shown substantial changes in f-scores over the model baseline where Aimens system score is 0.7185	• Machine Learning • Hybrid Method • Inputs data—spell check—word embedding via Word2Vec • Glove and FastText • Model Training (data split into training dataset and validation dataset)	
Automatic Emoji Insertion Based on Environment Context Signals for the Demonstration of Pervasive Computing Features (2021)	B. E. Tegicho and C. Graves	• Proposes the methodology for an automatic emoji insertion based on the text input • Matlab-based instant messaging tool with automatic emoji insertion. • Fast in response with high accuracy when implemented stand-alone. • Context detection systems are processed before sending the message.	• Two-way application that uses ThingSpeak as a middleware. • Replaces a target letter in a text with emojis, it takes inputs from different sensors that indicate a specific context. • The input values will be the deciders on a decision made about the type of emoji to be inserted.	• When run stand-alone, performs slow along with inaccuracies. • Depending on the priorities considered, response time and accuracy varied.

Sentiment Analysis of Tweets Including Emoji Data (2017)	T. LeCompte and J. Chen	• Comparison between MNB, SVM Bag of words and Bigram models on the data, with both bag of words and bigram representations. • SVM outperforms MNB at small vocabulary sizes • MNB outperforms SVM at large vocabulary sizes • Bigram vectors are very sparse due to the short tweets. • Bag of words outperform the more advanced bigram approach while saving computation time. • Concludes that bag of words representation with an MNB classifier for large vocabulary sizes including the Emoji data to achieve maximum classification accuracy must be preferred.	• SVM • MNB • Bag of Words model • Bigram Model	• Less experimentation on varying the training/testing ratio to produce learning curves. • Does not experiment with other possible representation formats apart from bag of words and bigram.
Comparison Research on Text Pre-processing Methods on Twitter Sentiment Analysis (2017)	Z. Jianqiang and G. Xiaolin	• Discusses the effects and works on the various classifier models used for text preprocessing. • Evaluates classification performances of six pre-processing methods using two feature models and four classifiers on five Twitter datasets.	• Word n-grams features model • Prior polarity score feature model • Support Vector Machine • Naive Bayes (NB) • Logistic Regression (LR, default parameters) • Random Forest	• Does not work on the emoji datasets

		• Assesses the effect of pre-processing on sentiment classification • Four popular supervised classifiers in the literature of sentiment analysis. • Uses the GridSearch search for these parameters as the optimal parameters • Uses scikit-learn library to perform the classifier.		
A Survey on Emotion Detection from Text in Social Media Platforms (2021)	Ashraf, M. Usman, et al.	• Accurate emotion classification and sentiment analysis • There are multiple models and classifications of emotions such as Ekman's model (Happy, Anger, Sad, Disgust, Fear and Surprise), and Plutchik's model (anger-fear, surprise-anticipation, joy-sadness, joy-sadness). • A systematic review of publications on textual emotions detection from social media platforms, state-of-the-art methods, and existing challenges presented.	• Ekman's model • Plutchik's model • Scientific Surveys • Machine Learning	• Casual writing • Grammatical errors are not handled • Provides decent efficiency

2.2 RESEARCH GAP

The research project identifies several notable gaps and limitations, which offer opportunities for further investigation and improvement in the field of emotion detection and sentiment analysis:

- **Lack of Fine-Grained Sentiment Classification:** One of the prominent research gaps is the absence of further classification of features, such as distinguishing between positive, negative, or neutral feedback. While the project successfully detects emotions, it does not provide a nuanced breakdown of the sentiment within the detected emotions. Fine-grained sentiment classification is crucial for gaining a more detailed understanding of the emotional content of text, which could enhance the practical applications of sentiment analysis, such as customer feedback analysis, brand reputation management, and user experience enhancement.
- **Exclusion of Emoji Data:** The project does not incorporate emoji data in the emotion detection process. Emojis are widely used in digital communication to express emotions and sentiments, and their exclusion represents a significant research gap. Future work could explore methods to integrate emojis as valuable cues for emotion recognition, as they often provide context and nuance to text-based sentiment analysis.
- **Lack of Effectiveness and Accuracy Evaluation:** Another critical research gap is the absence of a comprehensive evaluation of the model's effectiveness and accuracy. While the project mentions achieving "decent results" in emotion detection, it does not provide a thorough assessment or comparative analysis with other existing models or techniques. Further research could focus on rigorous evaluation metrics to determine the model's performance, reliability, and potential areas for improvement.

Addressing these research gaps by enhancing fine-grained sentiment classification, incorporating emoji data, and conducting rigorous model evaluation would contribute to the advancement of emotion detection and sentiment analysis methodologies. This, in turn, could lead to more accurate, reliable, and context-aware applications in areas like user experience improvement, mental health monitoring, and digital communication quality enhancement.

2.3 PROBLEM STATEMENT AND OBJECTIVE

2.3.1 PROBLEM STATEMENT

The problem at hand revolves around the field of opinion mining, a critical area of study that has gained increasing significance for its role in extracting valuable insights from the vast expanse of content present on social media platforms, thus providing substantial support for informed decision-making processes. However, the sheer magnitude of online information available necessitates the development of automated processes for efficient analysis. Within this domain, a critical challenge lies in accurately comprehending and interpreting the underlying emotions conveyed in text, as manual analysis of extensive text samples proves to be not only exceedingly time-consuming but also susceptible to subjective interpretation errors. The increasing prevalence of emojis, particularly in conveying subtle emotions, further complicates the task of deciphering textual sentiments, as their significance is oftentimes context-dependent and subject to misinterpretation without considering the accompanying emoji. In light of these challenges, this research endeavors to address these issues by proposing a novel approach focused on the development of an emotion detection model that combines both textual and emoji data, thereby contributing to the enhancement of accuracy in the recognition of emotions within digital content. This innovative model is adaptable and applicable across various domains, such as sentiment analysis in social media, feedback analysis, mental health monitoring, and user-centric applications, aiming to provide comprehensive and contextually aware insights that improve the quality of digital communication and support the evolving needs of users and organizations in the digital era.

2.3.2 OBJECTIVES

1. **Develop an Innovative Emotion Detection Model:** The primary objective is to create an advanced emotion detection model that combines textual and emoji data, enhancing emotion recognition accuracy. Processing data with text and emoji components.
2. **Analyzing the feedback:** Analyzing feedback and detecting emotions based on the same
3. **Obtaining an emotion based on the analysis:** On integration between both texts and emojis, a thorough analysis is provided.
4. **Applications in Diverse Domains:** The model's versatility is harnessed for various domains, including sentiment analysis in social media, feedback analysis, mental health monitoring, and user-centric applications.
5. **Creation of Annotated Dataset:** To achieve this, a diverse dataset is curated, comprising both text and emoji-based content. It is meticulously annotated with detailed emotional labels to provide a robust foundation for model training.
6. **Effective Preprocessing:** Critical preprocessing steps and emoji extraction, are performed to prepare the data for analysis. This ensures that the data is clean and structured for accurate results.
7. **Evaluating Model Efficacy:** Experimental findings are essential to evaluate the model's performance. The model's superiority over text-only models, particularly in recognizing nuanced expressions, is a key focus.
8. **High Accuracy Implementation:** The implementation of the model achieves an impressive accuracy rate of 70%. It processes both text and emojis, with careful attention to text preprocessing and the incorporation of an emoji database.
9. **Python-Based Implementation:** The model is implemented using the Python programming language, known for its suitability in machine learning and natural language processing tasks.
10. **Utilizing Datasets:** Datasets are employed for training and evaluation, typically comprising a training dataset for model training and a test dataset for assessing its performance and generalization capabilities.
11. **Enhancing Digital Communication:** Ultimately, the goal is to enhance digital communication by offering a more comprehensive understanding of emotions in digital content, with the potential to improve user experiences, support mental health, and enhance overall communication quality.

2.4 SCOPE OF THE PROJECT

2.4.1 PRESENT SCOPE:

1. Processing data with text and emoji components.
2. Finding emotions based on these components.
3. Analysing the feedback.
4. Obtaining an insight based on the analysis.
5. Determining the effect of the insight.
6. Detecting underlying emotions from the user feedback based on the insights.

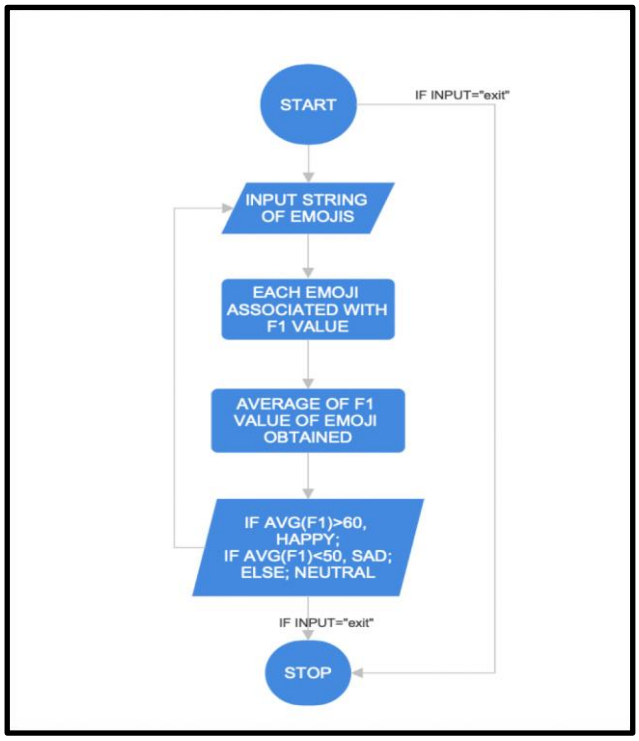
2.4.2 FUTURE SCOPE:

1. Automating Product Review Analysis
2. Enhanced Social Media Insights
3. Mental Health Assessments Proctoring
4. Emotionally Aware Chatbots and Virtual Assistant
5. Improved accuracy of model
6. Smoother integration of Text and Emojis

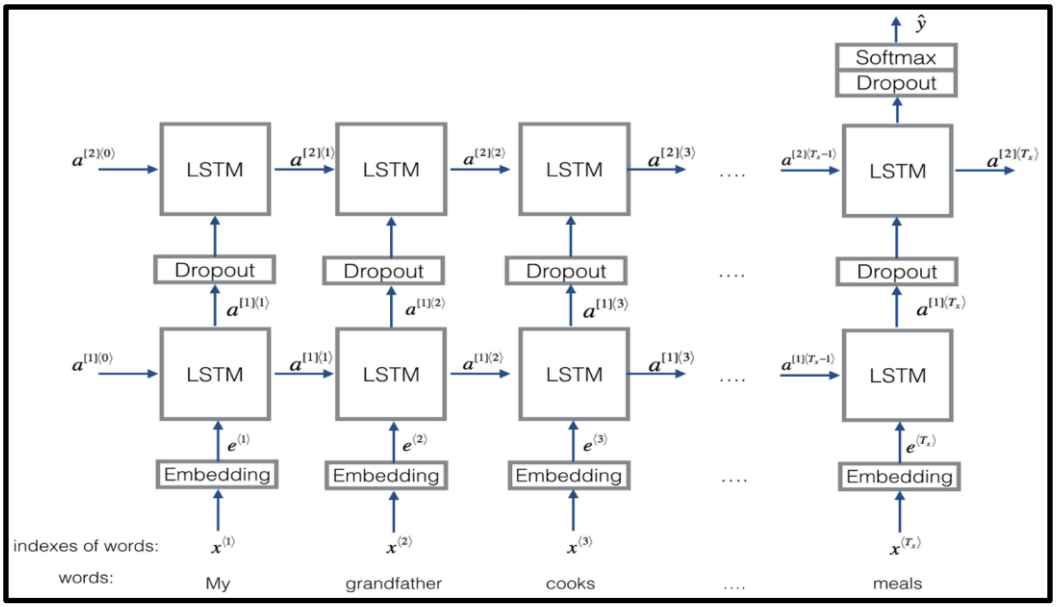
CHAPTER 3: PROPOSED SYSTEM

3.1 ALGORITHM

3.1.1 Flowchart for Emoji Vectorization



3.1.2 Working of the LSTM Model



3.2 METHODOLOGY

- **Text Preprocessing:** The implementation initiates with text preprocessing, which is a fundamental step to ensure that the user's input text is clean and structured for accurate analysis. This includes breaking the text into individual words or tokens, removing punctuation, and converting all text to lowercase. This preprocessing stage is crucial for preparing the data for further analysis.
- **Utilization of Datasets:** The project employs two essential datasets, namely the testing dataset and the training dataset. These datasets are pivotal for model training and evaluation. The datasets are in the format (X, Y) where, X contains 132 sentences and Y contains a label between [0, 4] corresponding to the five emotions. The training dataset is used to train the LSTM model, while the testing dataset is employed to assess the model's performance and generalizability.

  The emotions detected via texts are labelled as follows:

Emotion	Emoji	Label
Loving	❤️	0
Playful	⚽	1
Happy	😄	2
Annoyed	😞	3
Foodie	🍽️	4

3.2.1 Emotion Labels

- **Word2Vec Embedding:** Word2Vec embedding is used as a technique to convert text data into numerical representations. This method captures semantic relationships between words and enhances the model's ability to understand and analyze textual content effectively.
- **LSTM Model:** The project utilizes the Long Short Term Memory (LSTM) model for text processing. LSTM is a type of recurrent neural network (RNN) known for its capability to capture long-term dependencies in sequential data. It is particularly suitable for understanding context and sentiment in textual content.

- **Emoji Database Creation:** An emoji database is established, where each emoji is associated with a specific intensity value on a scale of 100. These intensity values indicate the strength of the emotion conveyed by each emoji, allowing for a more nuanced interpretation of emotions. A subset of the emoji database is as follows:

```
emoji_database = {
 "😁": {"category": "happy", "F-1": 90},
 "😀": {"category": "happy", "F-1": 85},
 "😃": {"category": "happy", "F-1": 80},
 "😄": {"category": "happy", "F-1": 95},
 "😂": {"category": "happy", "F-1": 100},
 "😅": {"category": "happy", "F-1": 70},
 "😆": {"category": "happy", "F-1": 75}
}
```

Fig. 3.2.2 Subset of Emoji Database

- **Emotion Detection Based on Emoji Intensity:** Emotions are detected based on both text and emojis. The project sets specific thresholds for emotion classification: values below 40 indicate sadness, values above 60 signify happiness, and values between 40 and 60 are considered neutral or objective statements.
- **Averaging Emotion Scale for Emoji Strings:** In cases where a string of emojis is present, the system averages out the emotion scale based on the intensity values of the emojis. This approach ensures that the combined emotional context of multiple emojis is accurately represented.
- **Output Incorporating Both Text and Emojis:** The system's output is a result of a combined analysis of both text and emojis. This integrated approach enhances the overall understanding of emotions within the user's input, providing a more comprehensive and context-aware emotional classification.

- **Data Analysis:**The dataset, which comprised 16,000 phrases with corresponding emotion labels labelled on them, was thoroughly examined during the data analysis phase of this study project. Six main emotional categories are covered by the dataset: fear, love, surprise, anger, sadness, and joy. To learn more about the variety and frequency of emotional responses, the dataset's emotional distribution was carefully examined.

The basis for additional study was these counts, which gave an overview of the relative frequency of each emotion group.

• Emotion	• Count
• Joy	• 5362
• Sadness	• 4666
• Anger	• 2159
• Fear	• 1937
• Love	• 1304
• Surprise	• 572

Fig. 1. Table showing counts of each emotion class in the dataset

Descriptive statistics were employed to summarize the distribution of emotions within the dataset. Frequency counts were calculated for each emotion category, revealing the following distribution: joy (5362), sadness (4666), anger (2159), fear (1937), love (1304), and surprise (572). These counts provided an overview of the relative frequency of each emotion category and served as the basis for further analysis.

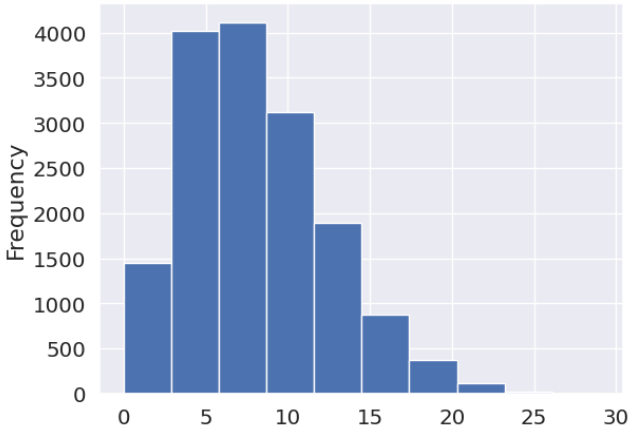


Fig. 2. Bar graph showing frequencies of stopwords in the dataset

A bar chart was created to show the percentage of each frequency in the dataset in order to visualise the distribution of stop words throughout it. The bar chart gave a clear visual depiction of the distribution by highlighting the different stop word frequencies across the sample.

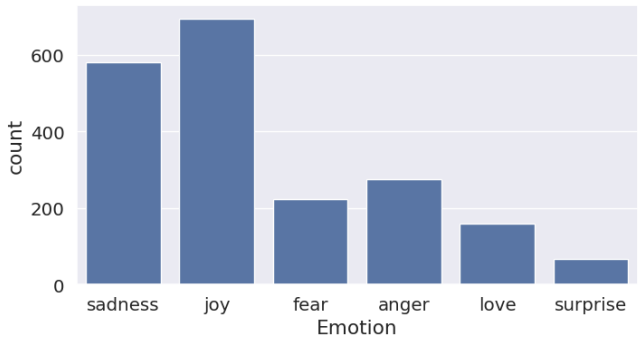


Fig. 3. Bar graph showing frequencies of stopwords over the emotion classes

The distribution of the stopwords throughout the emotion class was also shown as a bargraph in addition to a frequency analysis of the stopwords across the dataset. The bar chart provided a clear visual depiction of the distribution by highlighting the different frequencies of stopwords over the emotion class labels over the dataset.

Additionally, measurements like range and standard deviation were used to look at the variety and distribution of emotions. These metrics provided information about the range of emotional variability recorded by the annotations, as well as the diversity and intensity of emotional expressions found in the dataset.

To investigate potential correlations between certain emotions or linkages with other variables in the dataset, inferential statistical studies were also carried out.

CHAPTER 4: IMPLEMENTATION

4.1 Implementation

4.1.1 Creating Helper Functions

```
def read_glove_vecs(glove_file):
  with open(glove_file, 'r', encoding='utf-8') as f:
  #with open(glove_file, 'r') as f:
      words = set()
      word_to_vec_map = {}
      for line in f:
          line = line.strip().split()
          curr_word = line[0]
          words.add(curr_word)
          word_to_vec_map[curr_word] = np.array(line[1:], dtype=np.float64)

      i = 1
      words_to_index = {}
      index_to_words = {}
      for w in sorted(words):
          words_to_index[w] = i
          index_to_words[i] = w
          i = i + 1
  return words_to_index, index_to_words, word_to_vec_map

def convert_to_one_hot(Y, C):
  Y = np.eye(C)[Y.reshape(-1)]
  return Y

def read_csv(filename):
  phrase = []
  emoji = []

  with open (filename) as csvDataFile:
```

```
    csvReader = csv.reader(csvDataFile)

    for row in csvReader:
        phrase.append(row[0])
        emoji.append(row[1])

  X = np.asarray(phrase)
  Y = np.asarray(emoji, dtype=int)

  return X, Y
```

```
X_train, Y_train = read_csv('E:/TE/Mini_Project/Code/textbased/Datasets/train.csv')
X_test, Y_test = read_csv('E:/TE/Mini_Project/Code/textbased/Datasets/test.csv')
```

```
Y_oh_train = convert_to_one_hot(Y_train, C = 5)
Y_oh_test = convert_to_one_hot(Y_test, C = 5)
```

```
word_to_index, index_to_word, word_to_vec_map =
read_glove_vecs('E:/TE/Mini_Project/Code/textbased/Datasets/glove.6B.50d.txt')
```

```
def sentences_to_indices(X,word_to_index,max_len):
  """
  Converts an array of sentences (strings) into an array of indices corresponding to words in the
sentences.
  """

  m = X.shape[0]  # number of training examples

  # Initialize X_indices as a numpy matrix of zeros and the correct shape
  X_indices = np.zeros((m,max_len))

  for i in range(m):  # loop over training examples

    # Convert the ith sentence in lower case and split into a list of words
    sentence_words = X[i].lower().split()
```

```
    # Initialize j to 0
    j = 0

    # Loop over the words of sentence_words
    for w in sentence_words:
        # Set the (i,j)th entry of X_indices to the index of the correct word.
        if w in word_to_index:
            X_indices[i, j] = word_to_index[w]
            j += 1

        #X_indices[i, j] = word_to_index[w]
        # Increment j to j + 1
        #j = j + 1

  return X_indices
```

```
X1 = np.array(["lol", "I love you", "this is very yummy"])
X1_indices = sentences_to_indices(X1,word_to_index,max_len=5)
print("X1 =", X1)
print("X1_indices =",X1_indices)
```

Output:

```
X1 = ['lol' 'I love you' 'this is very yummy']
X1_indices = [[   0.    0.    0.    0.    0.]
 [4208. 5048. 9125.    0.    0.]
 [8315. 4510. 8769.    0.    0.]]
```

Fig. 4.1.1 Text Sentences and Their Word Index Representations

4.1.2 Defining the Network using Pretrained Embedding Layer using GloVe Word Embeddings

```
class NN(nn.Module):
 def __init__(self, embedding, embedding_dim, hidden_dim, vocab_size, output_dim, batch_size):
    super(NN, self).__init__()

    self.batch_size = batch_size

    self.hidden_dim = hidden_dim

    self.word_embeddings = embedding

    # The LSTM takes word embeddings as inputs, and outputs hidden states
    # with dimensionality hidden_dim.
    self.lstm = nn.LSTM(embedding_dim,
                        hidden_dim,
                        num_layers=2,
                        dropout = 0.5,
                        batch_first = True)

    # The linear layer that maps from hidden state space to output space
    self.fc = nn.Linear(hidden_dim, output_dim)

 def forward(self, sentence):

    #sentence = sentence.type(torch.LongTensor)
    #print ('Shape of sentence is:', sentence.shape)

    sentence = sentence.to(device)

    embeds = self.word_embeddings(sentence)
    #print ('Embedding layer output shape', embeds.shape)

    # initializing the hidden state to 0
    #hidden=None
```

```
h0 = torch.zeros(2, sentence.size(0), hidden_dim).requires_grad_().to(device)
c0 = torch.zeros(2, sentence.size(0), hidden_dim).requires_grad_().to(device)

lstm_out, h = self.lstm(embeds, (h0, c0))
# get info from last timestep only
lstm_out = lstm_out[:, -1, :]
#print ('LSTM layer output shape', lstm_out.shape)
#print ('LSTM layer output ', lstm_out)

# Dropout
lstm_out = F.dropout(lstm_out, 0.5)

fc_out = self.fc(lstm_out)
#print ('FC layer output shape', fc_out.shape)
#print ('FC layer output ', fc_out)

out = fc_out
out = F.softmax(out, dim=1)
#print ('Output layer output shape', out.shape)
#print ('Output layer output ', out)
return out
```

4.1.3 Creating the Glove Embedding Layer

```
def pretrained_embedding_layer(word_to_vec_map, word_to_index, non_trainable=True):
    num_embeddings = len(word_to_index) + 1
    embedding_dim = word_to_vec_map["blue"].shape[0]  #  dimensionality of GloVe word vectors (= 50)

    # Initialize the embedding matrix as a numpy array of zeros of shape (num_embeddings, embedding_dim)
    weights_matrix = np.zeros((num_embeddings, embedding_dim))

    # Set each row "index" of the embedding matrix to be the word vector representation of the "index"th word of the vocabulary
    for word, index in word_to_index.items():
        weights_matrix[index, :] = word_to_vec_map[word]

    embed = nn.Embedding.from_pretrained(torch.from_numpy(weights_matrix).type(torch.FloatTensor), freeze=non_trainable)

    return embed, num_embeddings, embedding_dim
```

4.1.4 Training the Model

```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

def train(model, trainloader, criterion, optimizer, epochs=10):

  model.to(device)
  running_loss = 0

  train_losses, test_losses, accuracies = [], [], []
  for e in range(epochs):

    running_loss = 0

    model.train()

    for sentences, labels in trainloader:

      sentences, labels = sentences.to(device), labels.to(device)

      # 1) erase previous gradients (if they exist)
      optimizer.zero_grad()

      # 2) make a prediction
      pred = model.forward(sentences)

      # 3) calculate how much we missed
      loss = criterion(pred, labels)

      # 4) figure out which weights caused us to miss
      loss.backward()

      # 5) change those weights
      optimizer.step()

      # 6) log our progress
      running_loss += loss.item()
```

```
    else:

      model.eval()

      test_loss = 0
      accuracy = 0

      # Turn off gradients for validation, saves memory and computations
      with torch.no_grad():
          for sentences, labels in test_loader:
              sentences, labels = sentences.to(device), labels.to(device)
              log_ps = model(sentences)
              test_loss += criterion(log_ps, labels)

              ps = torch.exp(log_ps)
              top_p, top_class = ps.topk(1, dim=1)
              equals = top_class == labels.view(*top_class.shape)
              accuracy += torch.mean(equals.type(torch.FloatTensor))

      train_losses.append(running_loss/len(train_loader))
      test_losses.append(test_loss/len(test_loader))
      accuracies.append(accuracy / len(test_loader) * 100)

      print("Epoch: {}/{}.. ".format(e+1, epochs),
            "Training Loss: {:.3f}.. ".format(running_loss/len(train_loader)),
            "Test Loss: {:.3f}.. ".format(test_loss/len(test_loader)),
            "Test Accuracy: {:.3f}".format(accuracy/len(test_loader)))

# Plot
plt.figure(figsize=(20, 5))
plt.plot(train_losses, c='b', label='Training loss')
plt.plot(test_losses, c='r', label='Testing loss')
plt.xticks(np.arange(0, epochs))
plt.title('Losses')
plt.legend(loc='upper right')
```

```
  plt.show()
  plt.figure(figsize=(20, 5))
  plt.plot(accuracies)
  plt.xticks(np.arange(0, epochs))
  plt.title('Accuracy')
  plt.show()



import torch.utils.data

maxLen = len(max(X_train, key=len).split())
X_train_indices = sentences_to_indices(X_train, word_to_index, maxLen)
Y_train_oh = convert_to_one_hot(Y_train, C = 5)

X_test_indices = sentences_to_indices(X_test, word_to_index, maxLen)
Y_test_oh = convert_to_one_hot(Y_test, C = 5)

embedding, vocab_size, embedding_dim = pretrained_embedding_layer(word_to_vec_map,
word_to_index, non_trainable=True)

hidden_dim=128
output_size=5
batch_size = 32

#print ('Embedding layer is ', embedding)
#print ('Embedding layer weights ', embedding.weight.shape)

model = NN(embedding, embedding_dim, hidden_dim, vocab_size, output_size, batch_size)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.002)
epochs = 50
train_dataset = torch.utils.data.TensorDataset(torch.tensor(X_train_indices).type(torch.LongTensor),
torch.tensor(Y_train).type(torch.LongTensor))
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size)
```

```
test_dataset = torch.utils.data.TensorDataset(torch.tensor(X_test_indices).type(torch.LongTensor),
torch.tensor(Y_test).type(torch.LongTensor))
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=batch_size)

train(model, train_loader, criterion, optimizer, epochs)
```

Output:

```
Epoch: 1/50..  Training Loss: 1.605..  Test Loss: 1.593..  Test Accuracy: 0.281
Epoch: 2/50..  Training Loss: 1.585..  Test Loss: 1.558..  Test Accuracy: 0.224
Epoch: 3/50..  Training Loss: 1.577..  Test Loss: 1.536..  Test Accuracy: 0.370
Epoch: 4/50..  Training Loss: 1.563..  Test Loss: 1.532..  Test Accuracy: 0.380
Epoch: 5/50..  Training Loss: 1.518..  Test Loss: 1.517..  Test Accuracy: 0.391
Epoch: 6/50..  Training Loss: 1.528..  Test Loss: 1.511..  Test Accuracy: 0.380
Epoch: 7/50..  Training Loss: 1.484..  Test Loss: 1.519..  Test Accuracy: 0.375
Epoch: 8/50..  Training Loss: 1.465..  Test Loss: 1.465..  Test Accuracy: 0.427
Epoch: 9/50..  Training Loss: 1.439..  Test Loss: 1.446..  Test Accuracy: 0.464
Epoch: 10/50..  Training Loss: 1.420..  Test Loss: 1.431..  Test Accuracy: 0.443
Epoch: 11/50..  Training Loss: 1.398..  Test Loss: 1.396..  Test Accuracy: 0.484
Epoch: 12/50..  Training Loss: 1.352..  Test Loss: 1.401..  Test Accuracy: 0.464
Epoch: 13/50..  Training Loss: 1.328..  Test Loss: 1.409..  Test Accuracy: 0.490
Epoch: 14/50..  Training Loss: 1.323..  Test Loss: 1.416..  Test Accuracy: 0.464
Epoch: 15/50..  Training Loss: 1.333..  Test Loss: 1.362..  Test Accuracy: 0.536
Epoch: 16/50..  Training Loss: 1.263..  Test Loss: 1.417..  Test Accuracy: 0.510
Epoch: 17/50..  Training Loss: 1.350..  Test Loss: 1.377..  Test Accuracy: 0.562
Epoch: 18/50..  Training Loss: 1.286..  Test Loss: 1.343..  Test Accuracy: 0.568
Epoch: 19/50..  Training Loss: 1.249..  Test Loss: 1.325..  Test Accuracy: 0.573
Epoch: 20/50..  Training Loss: 1.227..  Test Loss: 1.302..  Test Accuracy: 0.573
Epoch: 21/50..  Training Loss: 1.198..  Test Loss: 1.277..  Test Accuracy: 0.609
Epoch: 22/50..  Training Loss: 1.182..  Test Loss: 1.257..  Test Accuracy: 0.667
Epoch: 23/50..  Training Loss: 1.166..  Test Loss: 1.312..  Test Accuracy: 0.578
Epoch: 24/50..  Training Loss: 1.172..  Test Loss: 1.259..  Test Accuracy: 0.641
Epoch: 25/50..  Training Loss: 1.154..  Test Loss: 1.245..  Test Accuracy: 0.656
Epoch: 26/50..  Training Loss: 1.215..  Test Loss: 1.454..  Test Accuracy: 0.438
Epoch: 27/50..  Training Loss: 1.322..  Test Loss: 1.294..  Test Accuracy: 0.615
Epoch: 28/50..  Training Loss: 1.210..  Test Loss: 1.367..  Test Accuracy: 0.521
Epoch: 29/50..  Training Loss: 1.281..  Test Loss: 1.328..  Test Accuracy: 0.604
Epoch: 30/50..  Training Loss: 1.212..  Test Loss: 1.284..  Test Accuracy: 0.625
Epoch: 31/50..  Training Loss: 1.234..  Test Loss: 1.268..  Test Accuracy: 0.667
Epoch: 32/50..  Training Loss: 1.144..  Test Loss: 1.254..  Test Accuracy: 0.703
Epoch: 33/50..  Training Loss: 1.163..  Test Loss: 1.229..  Test Accuracy: 0.719
Epoch: 34/50..  Training Loss: 1.113..  Test Loss: 1.244..  Test Accuracy: 0.646
```

```
Epoch: 35/50..  Training Loss: 1.167..  Test Loss: 1.226..  Test Accuracy: 0.630
Epoch: 36/50..  Training Loss: 1.129..  Test Loss: 1.179..  Test Accuracy: 0.740
Epoch: 37/50..  Training Loss: 1.078..  Test Loss: 1.219..  Test Accuracy: 0.698
Epoch: 38/50..  Training Loss: 1.045..  Test Loss: 1.158..  Test Accuracy: 0.755
Epoch: 39/50..  Training Loss: 1.042..  Test Loss: 1.104..  Test Accuracy: 0.802
Epoch: 40/50..  Training Loss: 1.073..  Test Loss: 1.130..  Test Accuracy: 0.766
Epoch: 41/50..  Training Loss: 1.070..  Test Loss: 1.174..  Test Accuracy: 0.750
Epoch: 42/50..  Training Loss: 1.047..  Test Loss: 1.129..  Test Accuracy: 0.792
Epoch: 43/50..  Training Loss: 1.069..  Test Loss: 1.149..  Test Accuracy: 0.771
Epoch: 44/50..  Training Loss: 1.060..  Test Loss: 1.172..  Test Accuracy: 0.734
Epoch: 45/50..  Training Loss: 1.099..  Test Loss: 1.157..  Test Accuracy: 0.760
Epoch: 46/50..  Training Loss: 1.100..  Test Loss: 1.150..  Test Accuracy: 0.755
Epoch: 47/50..  Training Loss: 1.034..  Test Loss: 1.121..  Test Accuracy: 0.776
Epoch: 48/50..  Training Loss: 1.015..  Test Loss: 1.134..  Test Accuracy: 0.776
Epoch: 49/50..  Training Loss: 0.998..  Test Loss: 1.141..  Test Accuracy: 0.776
Epoch: 50/50..  Training Loss: 1.003..  Test Loss: 1.156..  Test Accuracy: 0.766
```

Fig. 4.1.4.1 Training Loss over Epochs

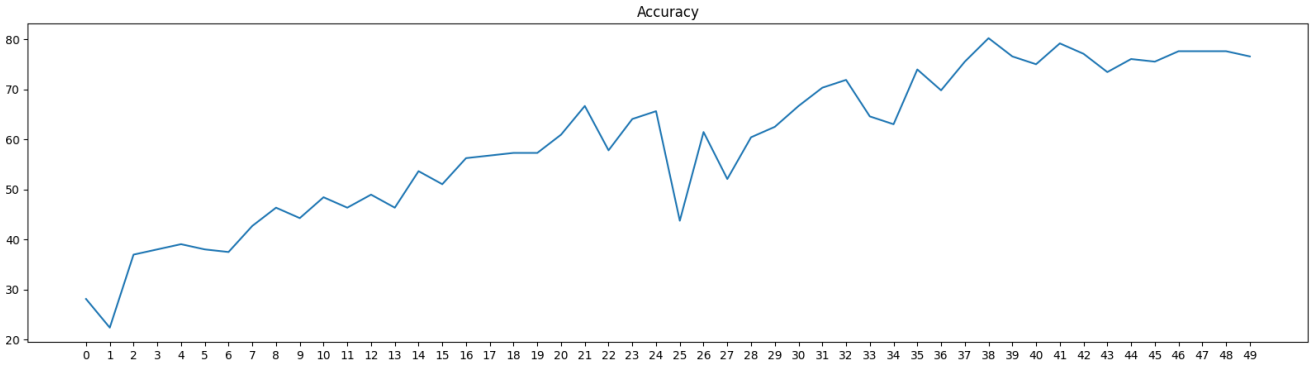


Fig. 4.1.4.1 Training Accuracy over Epochs

4.1.5 Testing the Model Accuracy

```
## Testing the Model Accuracy
test_loss = 0

accuracy = 0
model.eval()
with torch.no_grad():
    for sentences, labels in test_loader:
        sentences, labels = sentences.to(device), labels.to(device)
        ps = model(sentences)
        test_loss += criterion(ps, labels).item()

        # Accuracy
        top_p, top_class = ps.topk(1, dim=1)
        equals = top_class == labels.view(*top_class.shape)
        accuracy += torch.mean(equals.type(torch.FloatTensor))
model.train()
print("Test Loss: {:.3f}.. ".format(test_loss/len(test_loader)),
      "Test Accuracy: {:.3f}".format(accuracy/len(test_loader)))
running_loss = 0
```

Output:

```
Test Loss: 1.142..  Test Accuracy: 0.766
```

4.1.6 Testing the Model with Random Sentences

```
def predict(input_text, print_sentence=True):
 labels_dict = {
   0 : " 💗 Loving",
   1 : " ⚽ Playful",
   2 : " 😄 Happy",
   3 : " 😞 Annoyed",
   4 : "🍽 Foodie",
 }

 # Convert the input to the model
 x_test = np.array([input_text])
 X_test_indices = sentences_to_indices(x_test, word_to_index, maxLen)
```

```
sentences = torch.tensor(X_test_indices).type(torch.LongTensor)

# Get the class label
ps = model(sentences)
top_p, top_class = ps.topk(1, dim=1)
label = int(top_class[0][0])

if print_sentence:
  print("\nInput Text: \t"+ input_text +'\nEmotion: \t'+  labels_dict[label])

return label
```

```
#Trying a few test cases
predict("Hey, how are you doing?")
predict("I want a pizza")
predict("Lets see the game")
predict("I love you Lisa")
predict("This is the best day of my life")
```

Output:

```
Input Text:     Hey, how are you doing?
Emotion detected via text:      😔 Annoyed

Input Text:     I want a pizza
Emotion detected via text:      ❤️‍🩹 Loving

Input Text:     Lets see the game
Emotion detected via text:      ⚽ Playful

Input Text:     I love you Lisa
Emotion detected via text:      ❤️‍🩹 Loving

Input Text:     This is the best day of my life
Emotion detected via text:      😄 Happy
```

Fig. 4.1.6 Test Cases for Detecting Emotions through Text

4.1.7 Emoji Vectorization

```
def classify_emoji(emoji):
    emoji_info = emoji_database.get(emoji, {"category": "neither", "F-1": 50})
    f1_value = emoji_info["F-1"]
    return f1_value

# Main program
while True:
    input_text = input("Enter a sequence of emojis (or 'exit' to quit): ")

    if input_text == "exit":
        break

    emojis = [char for char in input_text if char in emoji_database]
    if not emojis:
        print("No valid emojis found in the input.")
        continue

    f1_values = [classify_emoji(emoji) for emoji in emojis]

    # Calculate the average F-1 value
    avg_f1_value = sum(f1_values) / len(f1_values)

    # Count the number of emojis in each category
    category_counts = {"happy": 0, "sad": 0, "neither": 0}
    for emoji in emojis:
        category = emoji_database[emoji]["category"]
        category_counts[category] += 1

    # Find the emoji with the highest F-1 value
    highest_f1_emoji = emojis[f1_values.index(max(f1_values))]

    if avg_f1_value > 60:
        emotion = "happy"
    elif avg_f1_value < 50:
        emotion = "sad"
```

```
else:
    emotion = "neutral"

# Display results
print("Emotion: " + emotion)
print("Average F-1 Value: " + str(avg_f1_value))
```

Output:

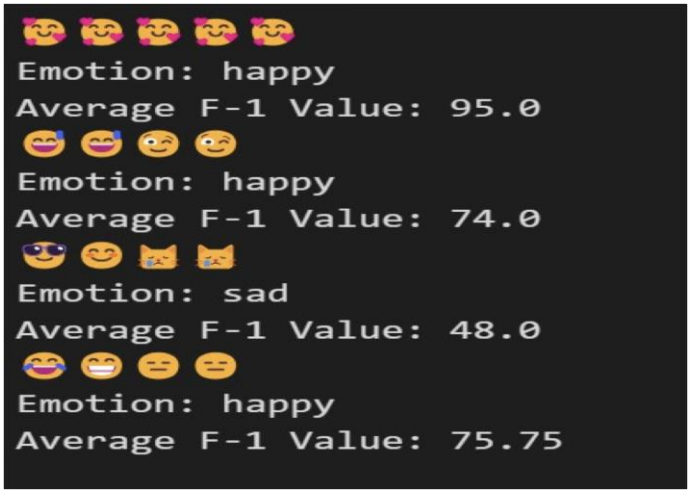


Fig. 4.1.7 Test Cases for Detecting Emotions through Emojis

4.1.8 Combined Output

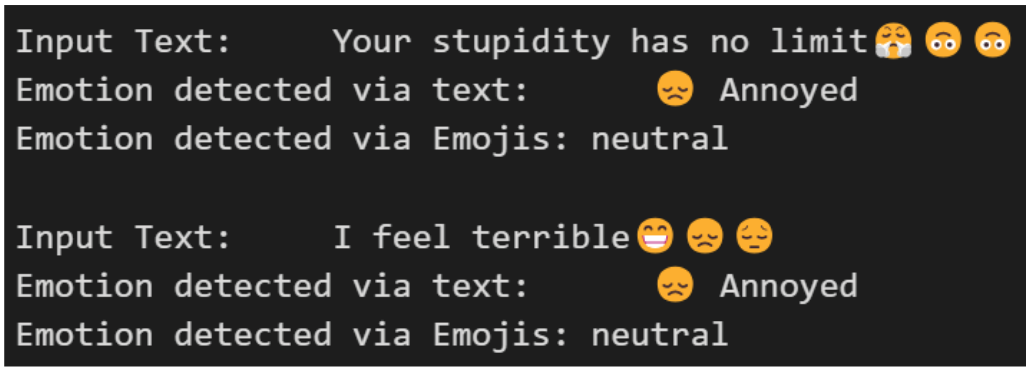


Fig. 4.1.8 Overall Model

CHAPTER 5: RESULT AND DISCUSSIONS

The validation results demonstrate a loss of 0.1353 and an accuracy of 94.19%, indicating strong performance and generalization ability of the model on unseen data. Conversely, during training, the model achieved a slightly higher loss of 0.1695 with an accuracy of 92.85%. This suggests that while the model performs well on the training data, it also effectively generalizes to new instances, as evidenced by the lower loss and higher accuracy on the validation set.

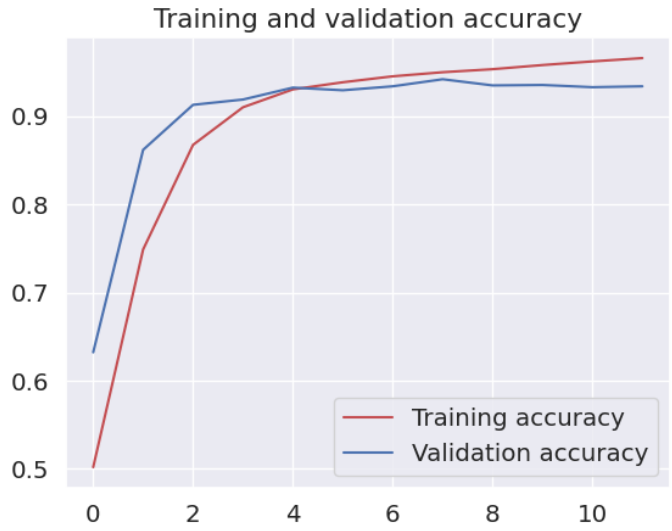


Fig. 7. Graph of Validation and Testing Accuracy

Regarding the graph depicting validation and testing accuracy, it shows an upward trend over epochs, indicating that the model's accuracy improves with training iterations. Initially, there may be fluctuations, but overall, there is a clear upward trajectory, demonstrating the model's learning process.

Fig. 8. Graph of Validation and Testing Losses

On the other hand, the graph representing validation and testing losses illustrates a downward trend over epochs, indicating a decrease in the model's loss function as training progresses. This trend confirms that the model's predictive performance improves steadily, as reflected in the decreasing loss values.

The overall accuracy of the model is reported as 93% based on a dataset of 2000 instances. Additionally, the macro average F1-score, which considers the average performance across all classes without considering class imbalance, is 0.88. The weighted average F1-score, which accounts for the proportion of each class in the dataset, is also 0.93. These results indicate that the model performs well in accurately classifying emotions across different classes, with particularly high precision and recall values for most emotions. However, it shows relatively lower performance in detecting surprise, as indicated by its lower precision, recall, and F1-score values for this emotion. Overall, the model demonstrates robust performance in emotion detection, with potential applications in sentiment analysis and human-computer interaction.

	Precision	Recall	F1-Score	Support
0	0.91	0.96	0.93	275
1	0.92	0.90	0.91	224
2	0.93	0.96	0.94	695
3	0.85	0.79	0.82	159
4	0.97	0.96	0.96	581
5	0.85	0.62	0.72	66
Accuracy			0.93	2000
Macro Average	0.90	0.86	0.88	2000
Weighted Average	0.93	0.93	0.93	2000

Fig. 9. Table depicting classification report

The confusion matrix of the model’s performance on the testing dataset is as follows:

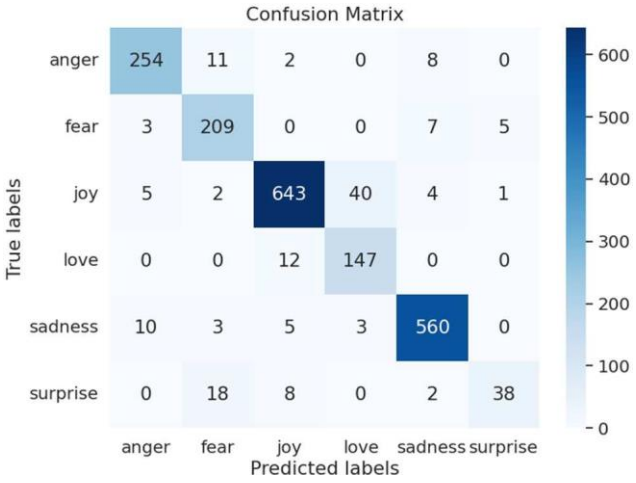


Fig. 9. Confusion Matrix of the LSTM model

For comparing the LSTM model's performance with other models, a few predefined models were used to run over the training and testing dataset. The models used are Random Forest, Logistic Regression, Support Vector Machine and Decision Tree. The models gave an accuracy as follows:

Model	Accuracy (%)
Random Forest	89
Logistic Regression	87
Support Vector Machine	87
Decision Tree	86

Fig. 10. Comparison of other models' performance

This proves that the LSTM model shows higher accuracy in the emotion detection system.

CONCLUSION

In conclusion, this research endeavor has successfully achieved its primary objectives, culminating in the development of a sophisticated emotion detection system with far-reaching implications for sentiment analysis and human-computer interaction. The obtained results underscore the robustness and efficacy of the proposed model.

The validation outcomes, demonstrating a loss of 0.1353 and an accuracy of 94.19%, highlight the model's strong performance and its ability to generalize effectively to unseen data. Conversely, during the training phase, the model exhibited a marginally higher loss of 0.1695 alongside an accuracy of 92.85%. This duality in performance indicates that while the model effectively learns from the training data, it also demonstrates superior adaptability to new instances, as evidenced by the lower loss and higher accuracy observed on the validation set.

Furthermore, these findings reinforce the significance of the adopted multimodal approach, which seamlessly integrates both textual inputs and emojis to provide a comprehensive understanding of emotional content in digital communication. By capturing subtleties and contextually contingent sentiments often overlooked by traditional methodologies, the model contributes to enhancing the quality of digital interactions across various applications, including sentiment analysis, user-centric experiences, and mental health support.

In summary, the developed emotion detection system stands as a testament to the effectiveness of its design, offering a robust and contextually attuned solution that bridges the gap between textual and emoji-based emotion interpretation. With its high accuracy, multimodal framework, and efficient preprocessing, the system holds considerable promise in significantly advancing sentiment analysis, human-computer interaction, and the overall quality of digital communication in today's dynamic digital landscape.

REFERENCES

[1] H. Sakode, T. L. Surekha, M. D. Sri, and G. B. Kumar, "Sentiment Analysis using Text and Emoji's," 2023 International Conference on Inventive Computation Technologies (ICICT), Lalitpur, Nepal, 2023, pp. 627-634, doi: 10.1109/ICICT57646.2023.10134459.

[2] P. Chandra et al., "Contextual Emotion Detection in Text using Deep Learning and Big Data," 2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA), Gunupur, India, 2022, pp. 1-5, doi: 10.1109/ICCSEA54677.2022.9936154.

[3] C. Colón-Ruiz, I. Segura-Bedmar, "Comparing deep learning architectures for sentiment analysis on drug reviews," Journal of Biomedical Informatics, Volume 110, 2020, 103539, ISSN 1532-0464, Link.

[4] S. Al-Azani and E.-S. M. El-Alfy, "Early and Late Fusion of Emojis and Text to Enhance Opinion Mining," in IEEE Access, vol. 9, pp. 121031-121045, 2021, doi: 10.1109/ACCESS.2021.3108502.

[5] U. Rashid, M. W. Iqbal, M. A. Skiandar, M. Q. Raiz, M. R. Naqvi, and S. K. Shahzad, "Emotion Detection of Contextual Text using Deep learning," 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Istanbul, Turkey, 2020, pp. 1-5, doi: 10.1109/ISMSIT50672.2020.9255279.

[6] B. E. Tegicho and C. Graves, "Automatic Emoji Insertion Based on Environment Context Signals for the Demonstration of Pervasive Computing Features," SoutheastCon 2021, Atlanta, GA, USA, 2021, pp. 1-6, doi: 10.1109/SoutheastCon45413.2021.9401878.

[7] T. LeCompte and J. Chen, "Sentiment Analysis of Tweets Including Emoji Data," 2017 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2017, pp. 793-798, doi: 10.1109/CSCI.2017.137.

[8] Z. Jianqiang and G. Xiaolin, "Comparison Research on Text Pre-processing Methods on Twitter Sentiment Analysis," in IEEE Access, vol. 5, pp. 2870-2879, 2017, doi: 10.1109/ACCESS.2017.2672677.

[9] M. Ashraf, M. Usman, et al., "A Survey on Emotion Detection from Text in Social Media Platforms," Lahore Garrison University Research Journal of Computer Science and Information Technology, 5.2 (2021), 48-61.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Dr. Phiroj Shaikh, our project guide who has been constantly guiding us throughout this project, right from the very beginning. Dr. Phiroj has been extremely supportive and approachable throughout and has assisted us with all obstacles we encountered regarding the project. We are immensely grateful to him for all the encouragement and support he has been constantly providing.

We are highly grateful to our Head of the Department, Prof. Dr. Phiroj Shaikh for this opportunity. We thank our project coordinator, Prof. Mayura Gavhane, who has taken immense trouble to organise various sessions throughout this semester to help us understand our project thoroughly, and who has helped us understand the process of implementation as well as designing the project poster.

We acknowledge the efforts, contribution and support of all our team members who equally contributed in several discussions, and toward the working and planning of this project. Through the blessings and grace of God, we have managed to make progress with this project, and we acknowledge our gratitude to God for the same. We are also grateful to our families who have always supported us throughout and encouraged us academically.

Finally, we would like to thank our college, Don Bosco Institute of Technology for providing several facilities and learning opportunities that enabled progress of this project. We are grateful to this college for the constant support and guidance they provide us through its qualified and supportive faculty, its infrastructure, and various essential facilities provided including several computer labs, library facilities and additional reference study material.

Project Team Members :

1. Anushka Joseph TE : 03
2. Shanaya Carvalho TE : 06
3. Nicole Saldanha TE : 56