DEVELOPMENT OF EMOTION DETECTION SYSTEM USING BIDIRECTIONAL LONG SHORT -TERM MEMORY NETWORKS

BY

AMADI, CHRISTIAN ONYEKACHI
(B.Sc., CSC FUPRE; PGD, CSC FUTO)
20194175748

A THESIS SUBMITTED TO THE POSTGRADUATE SCHOOL FEDERAL UNIVERSITY OF TECHNOLOGY, OWERRI.

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTERS OF SCIENCE (M.SC.) DEGREE IN COMPUTER SCIENCE

NOVEMBER, 2024

CERTIFICATION

This is to certify that this work "Development of Emotion Detection System Using Bidirectional Long Short-Term Network" was carried out by Amadi, Christian Onyekachi, Reg. number (20194175748) in partial fulfillment for the award of the degree of Master of Science (M.Sc.) in Computer Science in the department of Computer Science of the Federal University of Technology, Owerri.

Dr. (Mrs) J. N. Odii
Supervisor — Date: 17/12/2024

Dr. (Mrs) C. L. Okpalla
Co-Supervisor — Date: 16/12/2024

Dr. (Mrs) J. N. Odii
H.O.D. Computer Science — Date: 17/12/2024

Prof. U. F. EZE
Dean, SICT — Date: 18/12/24

Prof. (Mrs) J. N. Nwosu
Dean of PGS — Date: _____

Prof. Moses Okechukwu Onyesolu
External Examiner — Date: _____

DEDICATION

This thesis is dedicated to my dearest wife, Mrs. Amadi Obianuju Joy, and my older brother, Dr. Amadi Faith.

ACKNOWLEDGEMENTS

My immeasurable thanks go to the Head of Computer Science Department who is also my principal supervisor, Dr. (Mrs) J. N. Odii for her ever-readiness to attend to me throughout the period of this work and for expediting the completion of this program. I highly appreciate the efforts of my co-supervisor Dr. (Mrs) C. L. Okpalla for giving her succinct but appropriate corrections when needed. I will never forget the reviewer of this work Dr. I. I. Ayogu for his meticulous efforts to read in between the lines and enforced corrections that brought this work to an acceptable standard. My appreciation goes to the Deans of Schools of SICT and Postgraduate Studies, Prof. (Mrs.) U. F. Eze and Prof. (Mrs) J.N. Nwosu for their lenient responses whenever they are approached. I remain grateful to my lecturers especially Doctors C. N. Njoku, C. G. Onukwugha, C. L. Okpalla, C. Anyiam, I. I. Ayogu, O. A. Njoku, M. E. Benson-Emenike, U. Onyemauche, T. U. Onwuama and all other lecturers in the department of Computer Science and SICT at large whose scrutiny and contributions metamorphosed this thesis into a masterpiece.

I specially appreciate my dear parents, Mr. Israel Amadi and Mrs. Clara Amadi, my brother, Dr. Faith Amadi, my wife Miel and my other concerned siblings for their encouragement and numerous supports.

I will not fail to thank my colleagues: Christopher Ofoegbu, Jenifer Offor, and Ugonna for their spirit of teamwork. Finally, I appreciate the efforts of my friends, so numerous to be mentioned, for the diverse aids rendered to me during the course of this work and the entire programme.

I express my inestimable gratitude to God almighty for the unmerited grace he bestowed on me especially during the course of this program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Emotion detection in the context of chatbots holds immense promise for creating more empathetic and responsive conversational agents. This study presents a novel approach to enhancing chatbot capabilities by integrating emotion detection using Bidirectional Long Short-Term Memory (BiLSTM) neural networks. The primary objective of this research is to equip chatbots with the ability to discern and adapt to the emotional states of users during interactions. Leveraging the advantages of BiLSTM, we develop a model that can capture the temporal dependencies and contextual nuances in user messages, enabling it to accurately identify emotions such as happiness, sadness, anger, and more. The chatbot's architecture is augmented with the emotion detection module, allowing it to continuously analyze user input and provide emotionally tailored responses. Through a comprehensive dataset of conversational exchanges enriched with emotional labels, our model is trained to understand the intricacies of emotional expressions within text. The results of our experiments demonstrate the efficacy of the BiLSTM-based emotion detection approach within the chatbot framework. Users experience more personalized and empathetic interactions as the chatbot adapts its responses to match the detected emotional states. Comparative evaluations against traditional rule-based and non-emotion-aware chatbots underscore the significant improvements in user engagement and satisfaction. In conclusion, this research represents a significant advancement in the field of conversational Artificial Intelligence (AI). The integration of BiLSTM-based emotion detection empowers chatbots to better understand and respond to users' emotions, enhancing user experiences across a range of applications, from customer support to mental health companions. This work paves the way for more emotionally intelligent and empathetic AI-driven conversations, ultimately improving the quality and effectiveness of human-computer interactions.

Keywords: Emotion, Bidirectional, LSTM, RNN, Dataset, Chatbots

CHAPTER ONE

INTRODUCTION

1.1 Background Information

Emotions are central to human communication, influencing decision-making, behavior, and interpersonal interactions. As the world increasingly relies on artificial intelligence (AI) and natural language processing (NLP) for various applications, the integration of emotional intelligence into AI systems has gained significant attention. One of the most promising applications in this area is emotion detection, which enables machines to identify and interpret human emotions through text, speech, or visual cues. Emotion detection serves as a foundation for creating emotionally intelligent systems, such as chatbots, that can mimic human-like interactions and enhance user experience (Poria et al., 2017).

Emotion detection involves analyzing textual data to classify emotions such as joy, anger, sadness, and fear. Unlike speech or video-based emotion recognition, text-based emotion detection relies solely on linguistic cues, which presents unique challenges due to the lack of tonal, facial, and gestural information. Additionally, informal text formats, including abbreviations, emojis, and slang, further complicate the process of extracting emotional meaning. Nevertheless, text-based emotion detection is critical for applications like customer service, mental health support, and educational platforms, where real-time analysis of user sentiment is essential (Binali et al., 2010).

In the field of NLP, Bidirectional Long Short-Term Memory (BiLSTM) networks have emerged as a powerful tool for emotion detection. BiLSTM is a variant of the Long Short-Term Memory

(LSTM) architecture that processes input sequences in both forward and backward directions, enabling the model to capture contextual dependencies from both past and future words in a sentence. This bidirectional approach makes BiLSTM particularly effective for understanding complex linguistic structures and recognizing subtle emotional cues in text (Schuster & Paliwal, 2019). Its ability to handle long-term dependencies and sequential patterns has made it a widely adopted model for tasks requiring high-level semantic understanding, including sentiment and emotion analysis.

The integration of emotion detection into chatbots represents a significant advancement in human-computer interaction. Chatbots equipped with emotion detection capabilities can deliver personalized and empathetic responses, enhancing user satisfaction across diverse applications. For instance, in customer service, emotion-aware chatbots can identify frustrated users and escalate issues promptly. In mental health applications, such chatbots can provide emotional support by detecting signs of distress and responding appropriately. By understanding user emotions, chatbots can build trust and foster meaningful interactions (Fadhil & Gabrielli, 2017).

Despite these advancements, several challenges persist in developing emotion-aware chatbots. These include the complexity of multi-label emotion classification, where a single text may convey multiple emotions simultaneously, and the need to adapt to cultural and linguistic differences in emotional expression. Furthermore, ensuring computational efficiency while maintaining high accuracy is critical for real-time chatbot applications.

This study focuses on developing a BiLSTM-based emotion detection system and its integration into a chatbot framework. By leveraging the strengths of BiLSTM in contextual understanding, the research aims to create a chatbot capable of recognizing and responding to user emotions

effectively. The outcomes of this study are expected to contribute to the design of emotionally intelligent conversational systems, advancing the state of human-computer interaction.

1.2 Problem Statement

Emotion identification is a relatively novel task in text analysis that this new notion touches on. Recurrent neural networks are principally used in this to categorize emotions into a specific or distinct collection of emotion labels. Models based on neural networks are effective in identifying and comprehending contexts. It still has trouble capturing more background information or broader context, which might cause emotions to be misinterpreted. Therefore, this research uses the BiLSTM neural network model to try to alleviate the inefficiencies in text pre-processing in order to get better predictions.

1.3 Aim and Objectives for the study

This study's aim is to develop emotion detection system using bi-directional long short-term memory network. The objectives are to:

a. Use a fundamental BiLSTM model technique to construct a Bi-LSTM model for emotion detection.

b. Use the relevant datasets, assess how well the model created in (1) performs.

c. Incorporate into a chatbot the model created in (1) into a chatbot application and assess how it affects the chatbot's functionality.

1.4 Research Questions

a. Which data augmentation methods work best for class imbalance problems in datasets used for emotion detection?

b. How well do BiLSTM-based models for detecting emotions function in practical applications like chatbots?

c. How can user experience and participation in interactive systems become affected by the accuracy of emotion detection?

1.5 Justification of the study

Bi-directional Long-Short Term Memory network model plays a major role in precisely identifying emotions. It has demonstrated high performance in a variety of natural processing tasks, such as sentiment analysis and emotion identification. Academic studies have proven how successful BiLSTM models are at detecting emotions in a variety of contexts. These studies have also highlighted the models' capacity to manage variability, generalize across languages and domains, and capture contextual information.

BiLSTM have significant advantages over conventional approaches to natural language processing when it comes to capturing contextual dependencies. By applying the BiLSTM model, a kind of recurrent neural network (RNN), to comprehend how neural networks process and interpret text data to infer emotions, this study endeavor seeks to bridge a significant gap in the field of emotion detection research. The application of BiLSTM in emotion identification is

still mainly unexplored, despite the fact that prior research has examined emotion detection using a variety of methodologies.

The research has significant potential to enhance existing emotion detection techniques by providing breakthroughs in precision, interpretability of the model, and adaptability of input handling. The study uses the upsampling technique to balance data in order to address class imbalance. By using BiLSTM models in this application, systems for perceiving and reacting to human emotion may become more resilient and dependable.

1.6 Scope of the study

This work is limited to evaluating model quality attributes using a neural network-based process, realized using a Bi-directional long-short term memory network, namely;

Reliability: Models ability to be trusted to provide stable outputs in different conditions or data sets.

Usability: describes how easy and intuitive it is for users (data scientists, developers, or end-users) to interact with and use the model effectively.

Efficiency: Efficiency refers to the model's ability to deliver accurate results in a timely manner, using minimal computational resources.

Functionality: Model’s ability to meet its intended purpose and deliver the required features or outputs.

Maintainability: Indicates how easy it is to update, modify, or fix the model over time.

Portability: Portability refers to the model's ability to be transferred or adapted to different environments, platforms, or systems.

1.7 Definition of Terms

a. Bidirectional Long Short-Term Memory, or BiLSTM, is a kind of recurrent neural network (RNN) architecture that may be used to model sequences. It has the ability to gather context from parts in the sequence that are both past and future. Its capacity to comprehend the context and dependencies inside text data for emotion identification is why it is used in this study.

b. Dataset: An organized set of information used to test and train machine learning algorithms. When used in this context, it often refers to a dataset that is used to train and test the BiLSTM model and contains textual data with corresponding emotion labels.

c. Emotion: Emotion encompasses a vast array of sensations, ideas, and physical reactions. It is a complicated psychological and physiological condition. Subjective sensations, behavioral manifestations, and physiological responses to both internal and external stimuli are all included. Human cognition, decision-making, social interaction, and general well-being are all significantly influenced by emotions.

d. Emotion Detection: The method of locating and classifying emotions conveyed in data that is textual, auditory, or visual. It refers to the task of identifying emotions in textual data in the context of this study.

e. RNN: A family of artificial neural networks called Recurrent Neural Networks (RNNs) are made to handle sequential data by preserving internal memory. RNNs are well suited for tasks like time series prediction, natural language processing (NLP), speech recognition, and handwriting recognition because they can capture temporal

dependencies and context in sequential data, unlike traditional feedforward neural networks that process each input independently.

CHAPTER TWO

LITERATURE REVIEW

2.1 Conceptual Framework

The demand in the medical field has led to a recent rise in the popularity of emotion detection (Byoung, 2020). Technology that senses emotions can help people and machines communicate. Additionally, it may facilitate better decision-making. Numerous machine learning models have been put out to identify textual emotions. We are concentrating on the Bidirectional LSTM Model, though. Regular LSTMs are used in conjunction with bidirectional LSTMs, or Bi-LSTMs, to improve the model's performance on sequence classification tasks. Bi-LSTMs train on sequential input by employing two LSTMs. The input sequence is used just as it is in the first LSTM. The input sequence is represented in reverse for usage with the second LSTM. It accelerates our model and adds more context to it (Bahdanau et al., 2021).

2.2 Data Processing Method: Deep Learning

The most prominent study topic in the field of machine learning is deep learning, which is also a research path. The study of how the human brain acquires knowledge is where the idea of deep learning originates (Hasan et al., 2020). Neural networks are designed to implement the automatic feature extraction of input data by mimicking the multi-layer neurons of the human brain to abstract concepts and analyze the process of data (Ren et al., 2022). Deep learning has the advantage of extracting data features through hidden layer nodes and abstracting lower-level characteristics into more expressive high-level features through layer-by-layer extraction. Deep learning is therefore frequently utilized in data analysis across a range of disciplines. According

to Felbo et al. (2020), it has demonstrated remarkable performance in various disciplines such as speech recognition, computer vision, image classification, video analysis, and natural language processing. For this reason, deep learning was the method of choice for our experiment's emotion identification. The use of autonomous learning based on vast data is clearly what sets deep learning apart from conventional pattern recognition. According to Zhang et al. (2020), the unique features enable the outcome of the pattern recognition approach to be enhanced. The features of manual design have dominated a range of pattern recognition applications during the last few decades (Chen et al., 2021).

Big data does not take advantage of itself since manual design mostly depends on the designer's past knowledge and the features of big data. Because of the need on manual parameter adjustment, very few parameters are permitted to exist in feature designs. The majority of deep learning models has the ability to autonomously acquire feature representations from large datasets, and deep neural networks can include thousands of parameters. This time, we went with the BI-LSTM model. There seems to be a difference between machine learning and human learning: it can take five to ten years to manually build a desirable feature. Nevertheless, inside the neural network framework, BILSTM may rapidly acquire new feature representations from data in a matter of hours. For the primary purpose of identifying and categorizing emotions in this investigation, we selected BI-LSTM (Zuang, 2020).

2.3 Algorithm

Diverse models have been built to fit and evaluate the dataset more precisely while dealing with diverse subjects. Generally speaking, natural language processing involves computer translation, speech recognition, language modeling, etc. The chronological order in which the samples

emerge is crucial for this processing (Bengio et al., 2021). A neural network structure known as the recurrent neural network (RNN) has evolved in response to this need. The output of a neuron in an RNN can act directly onto itself at the subsequent date. RNN can resolve issues with computer vision, natural language processing, and other related fields by continuously providing training data to the model. Since RNN's memory is truly finite, it can only retain the data from the earlier steps, and as the sequence lengthens, according to Deng (2020), it is vulnerable to the issues of gradient explosion and gradient disappearance. Long-Short-Term memory (LSTM), a specific recursive neural network, was developed by the researchers. Though they can learn about long-term dependency through a method known as a "gate," LSTM and RNN are fundamentally similar. A time-cycle neural network called Long Short-Term Memory, or LSTM, was created to address the RNN's long-term dependency. The model can transfer pertinent information in the sequence processing, just like the problem we are working with in this paper, "Emotion States Classification," which involves both time series problems and time period problems. By retaining the prior cell unit information and updating the node information with the current cell state value, LSTM can learn the long-distance information in the data set. It can also choose the information we want to output by erasing the irrelevant information. For emotion recognition, the LSTM network is a wise option. The "gate" structure, which comes in three varieties forgetting gates, input gates, and output gates is used to add and remove information. The learning of Long-Short-Term dependencies is a goal that the LSTM algorithm can accomplish. It goes without saying that LSTM has a wide variety of specialized models to meet various needs. Consequently, the LSTM model emerged as the preferred option for this study (Kratzwald et al., 2020).

2.4 Emotion Models

Emotion models serve as the cornerstone of emotion detection systems, providing a definition for emotions and a means of "distinguishing between various emotion states" (Storman et al., 2022). Either emotion categories or emotion dimensions have been employed as supporting models in previous research.

Different emotion classes or labels serve as the foundation for emotional categories. The six basic, autonomous emotions that make up the Eckman's basic emotion model (Zheng et al., 2021) are anger, disgust, fear, happiness, sorrow, and surprise. Other composite emotions, such as pride, humiliation, lust, guilt, and so forth, can be created by combining these six fundamental emotions. The eight primary emotions that exist in pairs of opposite states are "joy vs. sadness, trust vs. disgust, anger vs. fear, and surprise vs. anticipation," according to the Robert Plutchik Model (Byoung, 2020). Each of these emotions can be experienced to varying degrees of intensity to determine the resulting emotional state (Storman et al., 2022).

Unlike discrete emotion models, dimensional emotional models view emotions as connected rather than independent. According to Lin et al. (2022), they "represent affects in dimensional form," meaning that each emotion has a place in a dimensional (1D, 2D, or 3D) space. Emotions are "distributed in a two-dimensional circular space" between the valence (pleasant vs. unpleasant) and arousal (activation vs. deactivation states) dimensions, according to Rusell's Circumplex Model of Affect (Bao et al., 2021; Zheng et al., 2020).

Plutchik's wheel of emotions is a two-dimensional wheel with valence and arousal on the vertical and horizontal axes, respectively. It is another dimensional emotional model. Centric circles are used to symbolize emotions, with the eight basic emotions being the sources of the emotions in

the innermost circles. On the outermost portions of the wheel, the combinations of the primary emotions and the eight fundamental emotions come after them, respectively. Based on where the emotions fall on the wheel, it shows how related they are to one another. Dominance is the third dimension of Russel and Mehrabian's three-dimensional PAD (Pleasure, Arousal Dominance) emotional state model, which measures experiencers' degree of control over the emotion being expressed (Zheng et al., 2020).

Although categorical techniques are more widely employed than the other, their restricted number of categories may prevent them from offering sufficient coverage of all states. "Not correlated to a certain emotional state, dimensional models are able to capture subtle emotion concepts that differ only slightly," according to the statement. Additionally, they offer a way to gauge how similar two affective states are to one another (Mirsamadi et al., 2020).

Pattern-based representations are used in the CARER model (Wu, 2022) and are further enhanced with word embeddings before being put to the test on a number of emotion perception tasks. Joy, love, anger, sadness, surprise, and fear are the six emotions that are used. It suggests using a graph-based, semi-supervised approach to provide rich descriptors, which serve as the fundamentals for creating text-based contextualized emotion representations. It uses a graph to represent an emotion corpus. The approach effectively captures the reciprocal worldwide usage of language variants present in textual data. The paper also describes how to gather information from the Twitter API, categorize tweets with hashtags, and preprocess them using a standard pipeline (Wu, 2020).

The suggested system blends a matrix representation of the text's enriched patterns with a multi-layer CNN architecture. After going through a ReLU activation function to create a feature map

matrix, the process' output is fed into a pooling layer, which in turn feeds the results into two hidden layers (Cheng, 2021).

The softmax activation layer is responsible for producing the output. For each emotion, the model's outputs approximate up to 70 f1 scores. The fact that the f1 score only rises to 70 despite the high frequency of emotions like happiness and sorrow is in some ways a disadvantage of the model. When compared to standard TF-IDF vectors, it works well. Additionally, the article discusses evaluating the model using multilingual activities like as Chinese and doing remarkably well on it, as evidenced by their comparable F1 score (Cheng, 2020).

Word embedding vectors, which are contextualized features from pertinent Tweets, are incorporated into the model by Ren et al. (2022) to create a context-based neural network model for Twitter sentiment analysis. They demonstrated that by employing neural pooling functions to represent the context of a specific target tweet, the most beneficial elements could be extracted from tweets, leading to improvements. Teng et al. (2021) have presented an additional context-sensitive technique for sentiment classification. They use bidirectional LSTM to learn the sentiment strength, intensification, and negation of lexicon sentiments in order to compose the sentiment value of a sentence. Their method is based on a basic weighted-sum model.

In order to capture both local information within sentences and long-distance dependencies across phrases, Wang *et al*. (2020) integrated CNN and LSTM. To predict the valence and arousal ratings of text, they suggested a regional CNNLSTM model, which is composed of two parts: regional CNN and LSTM.

Many supervised classification algorithms for emotions have been developed on data collected from microblogs like Twitter, employing hashtags or emoticons as the emotion label for the data, because there is a dearth of emotion-labelled datasets.

Millions of emoji occurrences in social networks were utilized as noisy labels by Felbo et al. (2020) to pre-train neural models to learn representations of emotional contexts. They employed two bidirectional LSTM layers with 1024 hidden units (512 in each direction) to record the context of each word. An attention layer that uses skip-connections to receive input from all LSTM layers is also used. Their findings demonstrated that the models learn more accurate representations of the emotional content in text and perform better when it comes to sentiment, emotions, and sarcasm detection when they are remotely supervised over a wider range of noisy labels.

Yu et al. (2020) suggested a dual attention network and LSTM-based transfer learning technique. To capture the general sentiment words and emotion-specific phrases, they separated the sentence representation into two feature areas. An analogous method applies a customized long short-term memory bank architecture to social media conversation emotion labeling. They used a heuristic technique to label their dataset, and then they used their classifier to reconstitute this heuristic (Gupta et al., 2021).

Koper et al. (2020) used a deep learning technique with enlarged lexicons of emotional norms to predict the intensity of emotions in tweets. They demonstrated the higher performance of domain-specific embeddings (trained on Twitter data) over other embeddings.

According to Polignano et al. (2020), an architecture made up of self-attention and BiLSTM showed promising results across a variety of datasets. The authors employed pre-trained word

embeddings from various domains to implement transfer learning. As such, they might cover a larger range of terms across domains while lowering the computational cost. Their model performs at roughly 62% on the ISEAR dataset.

Kratzwald et al. (2020) have suggested Sent2affect, a type of transfer learning for affective computing. The output layer of their Bi-LSTM network is then tailored to the job of emotion detection after it has been pre-trained for a different purpose (sentiment analysis). After evaluating the performance using six benchmark datasets, they discovered that Bi-LSTM with pre-trained word embeddings was the most effective technique in each experiment.

They also find that in every experiment, the Bi-LSTM seems to perform better than the unidirectional LSTM. In order to collect semantic and emotional information, Batbaatar et al. (2019) presented a semantic-emotion neural network (SENN), which integrates two sub-networks. A BiLSTM is used in the first network to collect semantic data. CNN is the second network used to gather sentiment data. For the ISEAR dataset, the SENN model performs at a rate of roughly 74%. Imran et al. (2020) examined how people from various cultural backgrounds felt about the coronavirus. Using the sentiment140 dataset, LSTM models that estimate sentiment polarity and emotions from extracted tweets have been trained to reach cutting-edge accuracy.

Distributed word vectors are typically used in deep learning-based techniques. The embeddings Word2Vec, GloVe, and FastText are frequently utilized. Words are treated as the smallest units by Word2Vec and GloVe. FastText takes a different tack by treating every word as though it were composed of ngram characters. Because of this, it can handle uncommon terms that aren't in dictionaries (Bojanowski et al., 2020).

In an effort to add context information into traditional word embeddings, contextualized word embeddings, dubbed USE and BERT (Cer et al. 2022), have been proposed more recently. Nevertheless, these word embeddings are limited in their ability to convey emotion information and are not task-specific, thus it has been demonstrated that using a neural network to learn task-specific emotion embeddings is beneficial (Devlin et al., 2021).

In textual discussions, some researchers make emotional predictions. Joy, sadness, anger, and neutral are the four emotion classes that Luo and Wang (2020) select from when fine-tuning the BERT model to predict emotion in dialogues. They make use of datasets that include the anonymous Facebook chat logs known as EmotionPush and scripts from the television series Friends. Chatterjee et al. (2020) selected from four emotion classes—happy, sad, furious, and other—when applying the Bi-LSTM model to infer the underlying emotion from textual dialogues. Another tool created by Al-Omari et al. (2020) for categorizing emotion in dialogues is called EmoDet2. The EmoContext dataset is categorized as happy, sad, furious, and other. They employ features taken from AffectiveTweets and GloVe embeddings. Additionally, they take word contextual embeddings out of the BERT model. To generate predictions, these vectors feed feed-forward and Bi-LSTM models. Their findings demonstrate that extracting BERT embeddings and then feeding them into a Bi-LSTM network improves system performance.

2.5 Sequential Transfer Learning

The goal of transfer learning is to use information from one source task to enhance a model's performance in a related but distinct target task. According to Ruder et al. (2019), sequential transfer learning improves the model's performance on target tasks by applying knowledge from source tasks to target tasks. Source tasks and target tasks are learned sequentially.

The two steps of sequential transfer learning are usually pretraining and adaptation. The model is trained on source tasks in pretraining. The pretrained model is further trained on target tasks during adaption. Fine-tuning and feature extraction are two common adaption techniques. While feature extraction uses the pretrained model as a feature extractor and maintains the parameters constant during the adaptation stage, fine-tuning modifies the pretrained model's parameters (Chen et al., 2020).

Recent years have seen a significant increase in the usage of sequential transfer learning, and deep pretrained language models have shown excellent results on a range of NLP applications (Peters et al., 2023). Although the deep pretrained models adapt very well, they are prone to forgetting, which occurs when the model learns target tasks and forgets what it has previously learnt from source tasks (Howard and Ruder, 2021).

Natural language processing has made use of deep pretrained language models like USE and BERT. Sequential Transfer Learning, which pretrains a language model on large-scale unlabeled data and then adapts it to downstream tasks, has been used to promote pretrained language models for a wide range of NLP applications. Pretrained weights can either be frozen or fine-tuned as part of the adaptation stage. Because of its flexibility, fine-tuning is actually more extensively used (Lan et al., 2020).

The sequential transfer learning method used by DeepEmotex involves learning the source and target tasks sequentially. Accordingly, each task is learnt independently rather than through joint model optimization as in multi-task learning (Ruder et al. 2020).

Sequential transfer learning happens in two stages: an adaptation or fine-tuning phase when the gained knowledge is transferred to a target task or domain, and a pretraining phase where general-purpose representations are learnt on a source task or domain (Ruder et al., 2021).

Classifying emotions using deep neural network architectures is not simple. The tiny emotion datasets are unsuitable for the pretrained models, which is the issue. When they are adjusted with a classifier, they usually get overfitted and experience forgetfulness. This is particularly problematic when there are few fine-tuning data points. Current studies focus mostly on fine-tuning strategies to forget less in order to address the forgetting issue in transferring deep pretrained models. They used a sizable pool of readily available emotion-labeled data from Twitter to fine-tune with less forgetting.

The majority of earlier research (Kiros et al., 2023) concentrated on various pretraining objectives to acquire general-purpose word or phrase representations. A few studies have looked at how to modify the pretrained model for a certain target task and how to fine-tune it (Ragheb et al., 2020). Two popular methods are used for fine-tuning: The first method involves using the pre-training network as a feature extractor (Donahue et al. 2020), with the exception of the last layer, where all model layers are frozen while fine-tuning on the target task. In this method, a downstream model makes use of the pre-trained representations. As an alternative, the parameters of the previously trained model are unfrozen and adjusted for a new task. This method makes it possible to modify a general-purpose representation for a variety of applications (Dai et al. 2021).

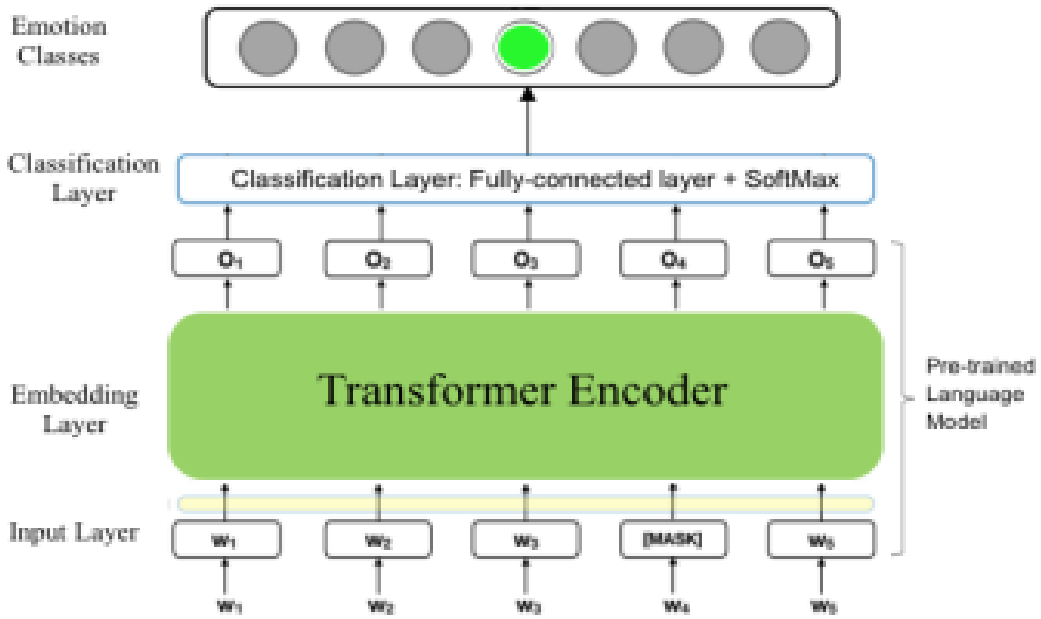


Figure 2.1: DeepEmotex model (Dai et al., 2020).

The input text's semantic textual information is contained in an embedding that is learned by the embedding layer. In order to anticipate emotions, the learnt representations are input into the classification layer (Dai et al., 2020).

The input words are represented by their embeddings in this model. The above model has a transformer encoder after the embedding layer and a SoftMax classification layer after that. Making the most of pre-trained representations requires a deeper comprehension of the adaptation step. Thus, two cutting-edge pre-trained models—BERT (Devlin et al., 2021) and Universal Sentence Encoder—are used by DeepEmotex. They applied the knowledge gleaned from a big corpus to this emotion categorization model by means of these models. Next, according to Cer et al. (2020), they adjust the DeepEmotex model to fit their target emotion datasets.

2.6 Transfer Learning Model

Using a deep neural network, the Universal Sentence Encoder (USE) (Cer et al., 2020) generates universal sentence embeddings, or pretrained embeddings, which are obtained by training deep learning models on a large corpus. These pretrained (generic) embeddings can be used in a wide range of natural language processing (NLP) tasks, such as text classification, semantic similarity, and clustering.

Sentences, phrases, and brief paragraphs are examples of text longer than a word for which the Universal Sentence Encoder is trained and optimized. A text with varying length is the input. The incoming text is encoded into 512 dimensional embeddings via the Universal Sentence Encoder. In order to dynamically accommodate a wide range of natural language comprehension tasks that require modeling the meaning of word sequences rather than just individual words, the USE embeddings are trained on various data sources and tasks.

In essence, the USE models come in two forms. In the initial iteration, phrase embeddings are generated by feeding input embeddings for words and bi-grams via a feed-forward deep neural network (DNN) after the input embeddings are first averaged together (Cer et al. 2022). The second version is more complicated than the Deep Averaging Network (DAN). The DAN encoder's compute time is linear with the length of the input sequence, which is its main benefit.

The transformer-network based sentence encoding model is used in the second edition. The N = 6 identical layers that make up the transformer encoder stack are called layers. Two sublayers make up each layer. A multi-head self-attention mechanism is the first, and a straightforward, position-wise entirely layer normalization comes after a connected feed-forward network (Cer et al., 2020).

Their findings show that the transformer-based encoder performs the best overall on the transfer task. Nevertheless, computing time and memory consumption increase significantly with sentence length as a result of this. We choose the transformer-based encoder for our emotion classification challenge since it outperforms the DAN encoder in terms of overall performance (Cer et al., 2022).

We use the LSTM method to tackle our emotion classification job by utilizing previous knowledge from pre-trained embeddings. The Universal Sentence Encoder is utilized by our sub-network for sentence embedding. We use our gathered emotion-labeled dataset to refine the sentence embeddings.

We construct a feed-forward neural network using the rectified linear activation function (ReLU) and two hidden dense layers. The vanishing gradient issue is solved with ReLU. Deep neural networks that experience the vanishing and explosion gradient problem will benefit from this (Zhang et al., 2020).

Learning features are provided by a dense layer, which uses all possible combinations of the features from the preceding layer. 512 feature vectors produced by the Universal Sentence Encoder technology are the input used in our model. After that, the vector is passed into layers that are fully connected, ending with a layer called softmax. Next, we use the tagged tweets obtained in Section V-A to refine our model. We set the trainable parameter to true in order to fine-tune the embedding weights. Here, we make use of transfer learning using pre-trained embeddings (Zhang et al., 2020).

Figure 2.2 displays the DeepEmotex-USE overall design. A tweet is used as the model's input. First, a sentence is mapped into its embedding vector by the embedding layer using the pre-

trained USE model. Our approach divides the sentence into tokens, embeds each token, and then concatenates them into 512-dimension embeddings that take context into account. Subsequently, a feedforward neural network with ReLU activation is fed the embeddings. In order to generate a classification probability, it projects the input into 256-dimension embeddings and feeds them to the classification layer. Our model produces a label for the classification of emotions. Predicting each tweet's emotion accurately is the primary goal, according to Zhang et al. (2020).

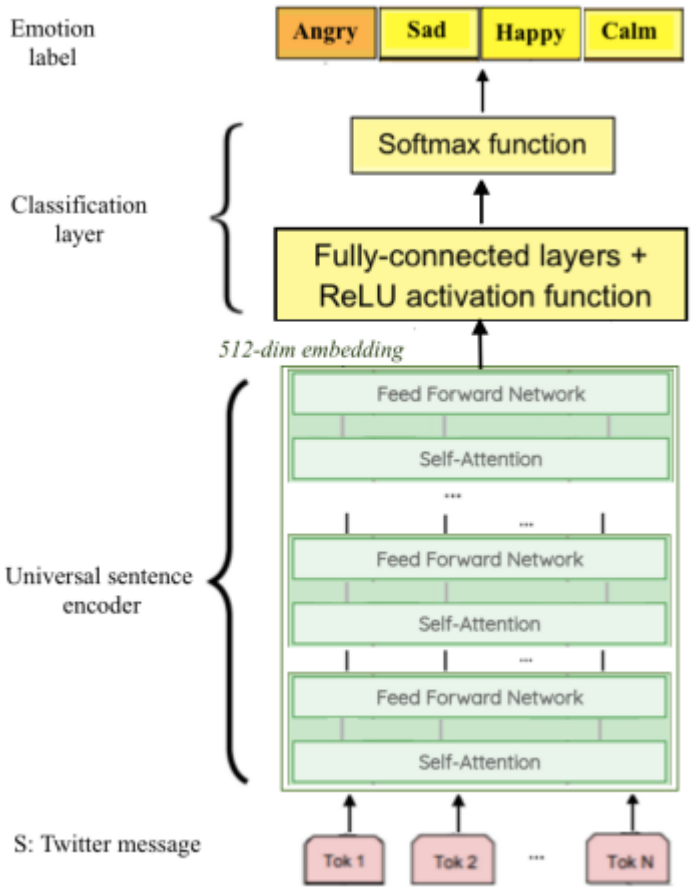


Figure 2.2 Model of DeepEmotex-USE to classify emotion in text messages using Universal Sentence Encoder (Zhang et al. 2021).

2.7 BERT

A Transfer Learning Model Employing Transformer Bidirectional Encoder Representations displays the DeepEmotex-BERT model architecture employing Bidirectional Encoder Representations from Transformers (BERT). A tweet serves as the model's input, and an emotion label serves as its output. For the purpose of creating text representations, we employ the pretrained BERT model (Devlin et al. 2021). Using a bidirectional Transformer encoder that has been previously trained on the language modeling problem, BERT acquires text representations. Sequence-to-sequence model architecture is used in transformers. There are independent encoder and decoder components in every transformer. Their use of attention, which is referred to as self-attention, differs. A stack of encoders that are fully coupled to a stack of decoders makes up the core architecture. A feed forward component and a self-attention component make up each encoder. A feed forward component, an encoder-decoder attention component, and a self-attention component make up each decoder (Vaswani et al. 2022).

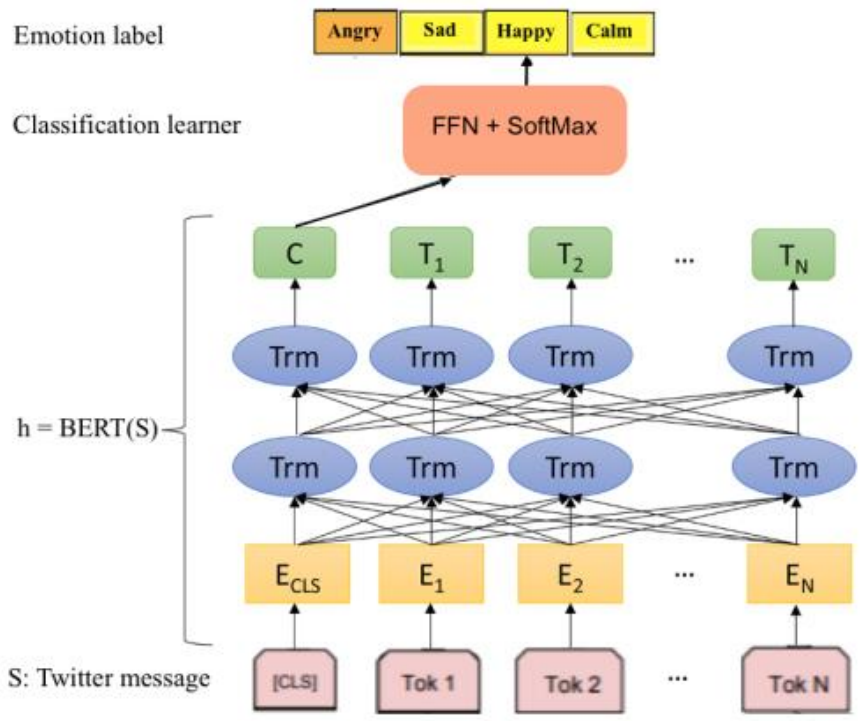


Figure 2.3: Using pretrained BERT, the DeepEmotex model classifies emotion in text messages (Vaswani et al., 2022).

2.8 Overview of an Emotional Chatbot

A particular kind of AI-powered chatbot called an emotional chatbot is made to recognize and react to human emotions throughout a discussion. In order to deliver sympathetic and encouraging conversations, emotional chatbots are able to recognize and react to a variety of user emotions, including joy, sadness, frustration, and anger (Bickmore & Cassell, 2020).

In a variety of settings, such as mental health, customer service, education, and entertainment, emotional chatbots can be employed. For instance, D'Mello et al. (2023) suggest that emotional chatbots can be employed to offer emotional support to individuals undergoing stress, anxiety, or despair.

Machine learning algorithms, sentiment analysis, natural language processing (NLP), and other methods are used by emotional chatbots to identify emotions. Additionally, they can deduce emotions from user behavior, tone of voice, and facial expressions (Gaffney et al., 2020).

Emotional chatbots provide several advantages, such as enhanced customer pleasure, less stress, and higher user engagement. Emotional chatbots do, however, have certain drawbacks, including the requirement for precise emotion detection and the possibility of inadvertent biases (Liao et al., 2021).

There are several sorts of emotional chatbots, including the following:

1. Empathetic chatbots: These chatbots are made to be sensitive to the feelings of the user and react accordingly. They recognize emotional indicators in the user's language and react appropriately by utilizing machine learning techniques and natural language processing (NLP) (Tian et al., 2021).
2. Therapeutic chatbots: These chatbots are intended to offer users therapy and emotional assistance. To assist clients in controlling their emotions and enhancing their mental health, they employ methods including mindfulness and cognitive behavioral therapy (CBT) (Gaffney et al., 2019).
3. Companion chatbots: These chatbots are made to be users' virtual friends. According to D'Mello et al. (2020), they are designed to converse and interact with users in order to offer companionship and emotional support.
4. Coaching chatbots: By offering direction and emotional support, these chatbots are intended to assist users in reaching their objectives. To assist users in achieving their

intended results, they employ strategies including goal-setting and motivational interviewing (Bickmore et al., 2021).

5. Entertainment chatbots: These chatbots are made to make people laugh and have fun. In order to uplift users' spirits, they have lighthearted interactions with them and offer jokes, memes, and other entertainment (Liao et al., 2021).

2.9 Bidirectional Recurrent Neural Networks Methods

A specific class of neural networks called recurrent neural networks, or RNNs, are used to analyze sequential input. A collection of data points is referred to as sequential data. For example, text is sequential because it is created from a combination of letters, music is sequential because it is a combination of a succession of sound elements, and video is sequential because it is made up of a sequence of video frames. Preserving the data that was discovered from the earlier examples is necessary for modeling sequential data. For instance, you have to take into account the prior argument made by the participants in the discussion if you are to forecast the subsequent argument during that debate. You structure your arguments so that they follow the natural flow of the debate. Similarly, in order to make a decision, an RNN must learn and retain the data, which is reliant on prior learning (Graves et al., 2023).

An RNN does not cap the input or output as a set of fixed-sized vectors, in contrast to a standard neural network. Furthermore, it doesn't reduce the quantity of computational steps needed to train a model. Alternatively, it enables us to use sequential data—a series of vectors—to train the model.

It's interesting to note that an RNN keeps model parameters constant across the network. To account for different sequence data lengths, it uses parameter sharing. It would not be

computationally viable nor practicable to generalize the data values throughout the series if we were to take into account different parameters for different data pieces. Generalization refers to the recurrence of values inside a sequence. In a song, a note might appear somewhere else; In order for an RNN to determine the dependency that is still present in the data, this must be recorded. An RNN therefore transfers learnt information to the subsequent layers rather than beginning from scratch at each learning point (Bahdanau et al., 2021).

An RNN uses loops to allow for parameter sharing and information persistence.

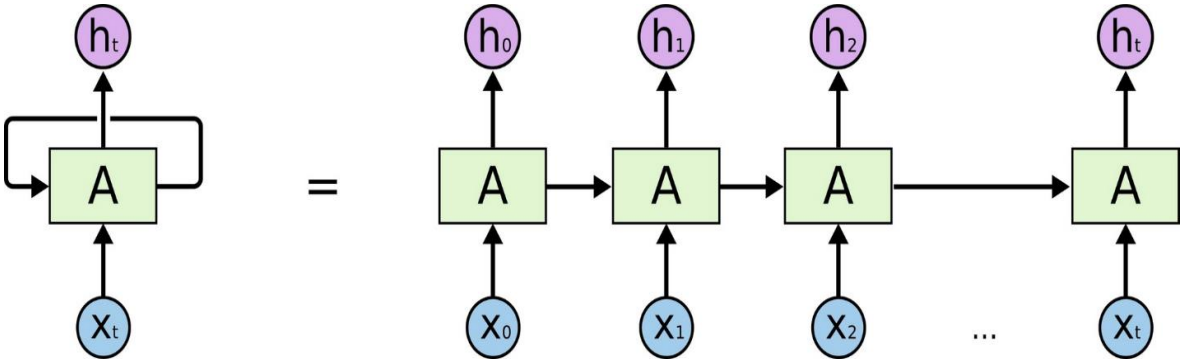


Figure 2.4: Unfolding an RNN (Bahdanau et al., 2022).

Recently, recurrent neural networks (RNN) have been effectively trained for time series modeling, and they have been applied to accomplish state-of-the-art outcomes in supervised tasks such as speech and handwriting recognition (Mohamed et al., 2022). RNNs have also been effectively applied to time series unsupervised learning. The unsupervised situation is further highlighted by the recent application of RNNs to create sequential data in a machine translation scenario. Perhaps because bidirectional RNNs have not been extensively researched as generative models, Bahdanau et al. (2020) employed a bidirectional RNN to encode a phrase into a vector but ultimately opted for a unidirectional RNN to decode it into a translated phrase. More recently, a deep bidirectional RNN was employed in speech recognition by Maas et al. (2021),

producing text as output. Reconstructing missing values from scratch is intriguing in at least three ways: first, it can handle real-world data that contains missing values; second, the ability to reconstruct values from scratch can be used to gauge performance in unsupervised learning (Raiko et al., 2021). Lastly, reconstruction of artificially missing values can be utilized as a training criterion (Goodfellow et al., 2020).

Training to reconstruct longer gaps can encourage the model to focus on longer-term predictions, even when typical RNN training criteria are associated with one-step prediction. Keep in mind that the one-step

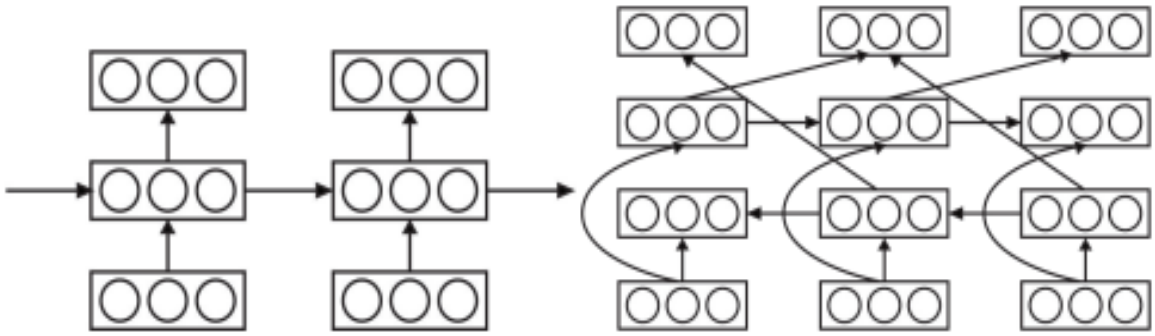


Figure 2.5: Structure of the simple RNN (left) and the bidirectional RNN (right) (Bahdanau et al., 2020)

The prediction criterion is commonly employed in methods that primarily focus on modeling long-term interdependence. Sequentially drawing samples from the model is simple when utilizing unidirectional RNNs as generative models. In smoothing jobs, on the other hand, when we wish to assess probability for missing values in the middle of a time series, inference is not simple. Inference with gap sizes of one is possible for discrete data; however, the cost of inference increases exponentially with increasing gap sizes. Depending on the gap size, even sampling can become extremely costly. Explicitly training the model using missing data is one method for

building models that are used to fill in the gaps. To our knowledge, nevertheless, no such criterion has yet been applied.

Not every case calls for learning from the facts that came before it in order. Imagine you are attempting to guess a sentence from another sentence that was introduced in a book or article some time ago. This necessitates keeping in mind not just the data that came right before, but also the data that came before. Because of the parameter sharing technique, an RNN always employs the same weights. Because of this, the gradient either bursts or disappears during backpropagation, meaning that the network learns very little from data that is far from the present point (Sutskever et al., 2021).

We employ Long Short-Term Memory Networks, or LSTMs, to tackle this issue. One can learn long-term dependencies with an LSTM.

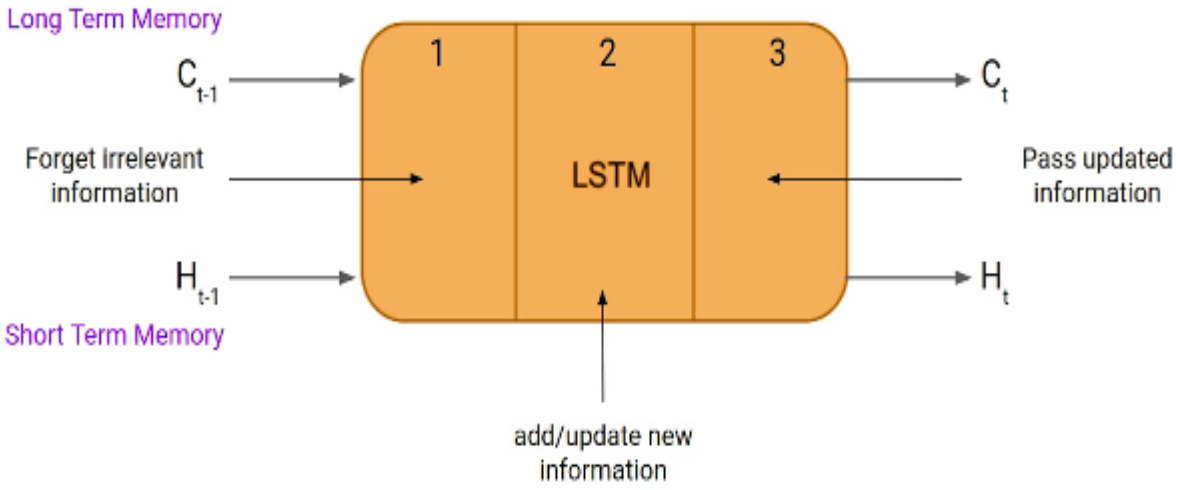


Figure 2.6: Long Short-Term Memory Networks (Sutskever et al., 2021).

An LSTM block performs a few more functions than an RNN, which has a single layer in a network block. It retains the important information and discards the irrelevant information it learns from the network by using input, output, and forget gates (Goodfellow et al., 2020).

2.9.1 Bidirectional LSTM

Recurrent neural networks like Bidirectional LSTM (BiLSTM) are mainly employed in natural language processing. In contrast to a regular LSTM, this one may use data from both sides and accept input in both directions. Additionally, Sutskever et al. (2021) state that it is an effective tool for simulating the sequential interdependence between words and phrases in both directions of the sequence.

In conclusion, BiLSTM flips the direction of information flow by adding an additional LSTM layer. In short, it indicates that in the extra LSTM layer, the input sequence runs backward. The outputs from the two LSTM layers are then combined using a variety of methods, including concatenation, multiplication, average, and sum (Goodfellow et al., 2023).

The unrolled BiLSTM is displayed in the following graphic to provide context:

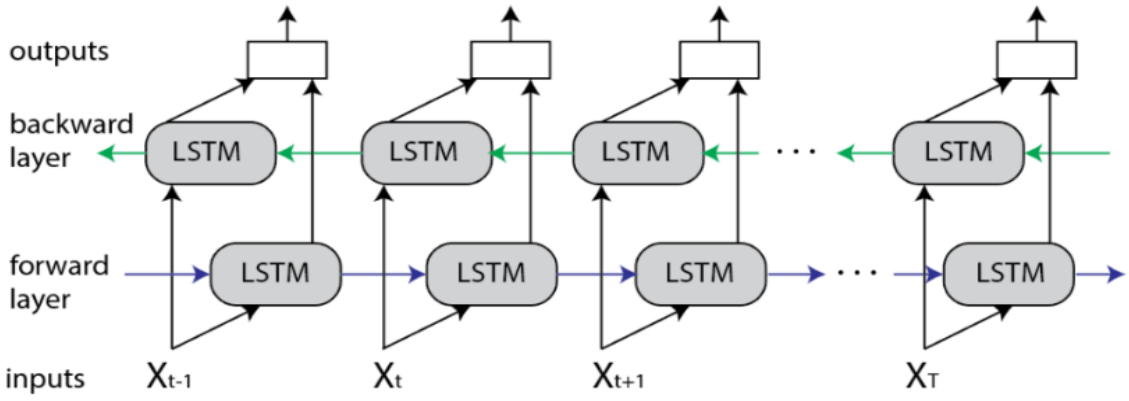


Figure 2.7: Bi-LSTM illustration (Goodfellow et al., 2023).

2.10 Challenges of Emotion Detection Using RNN

Recurrent Neural Networks (RNNs) for mood detection provide a number of issues that must be resolved to raise the models' level of accuracy and resilience. Here are a few instances of difficulties:

a. Limited Access to Data: The scarcity of labeled data makes mood detection with RNNs a significant difficulty. The creation and assessment of RNN models is hampered by the challenging and time-consuming nature of gathering huge volumes of labeled data for mood detection. In order to overcome this difficulty, more labeled data have been made accessible for RNN model training through the use of data augmentation techniques as voice synthesis and data resampling (Khorrami et al., 2022).

b. Inter-Subject Variability: Developing RNN models that generalize to various individuals is challenging since mood is a subjective experience that can change greatly amongst people. Personalized RNN models that can adjust to each user's unique mood patterns have been developed as a solution to this problem (Yang et al., 2020).

c. Variability in Input Modalities: There are several ways to convey one's mood, including through language, text, facial expressions, and bodily functions. However, the efficacy of various modalities in detecting moods can differ, and they can call for distinct processing methods. In order to increase mood detection performance, multi-modal RNN models that can integrate data from several modalities have been developed (Zhou et al., 2019).

d. Noisy and Incomplete Data: RNNs might be difficult to use for mood detection because of noisy and incomplete input data, which can produce inaccurate or untrustworthy

predictions. In order to overcome this difficulty, RNN models that can deal with missing or corrupted data have been created. For example, denoising autoencoders have been used to reconstruct corrupted data (Zhang et al., 2021).

e. Interpretability and Transparency: Since it can be challenging to comprehend how the model is generating predictions, interpretability and transparency issues can also arise when employing RNNs for mood detection. In order to solve this difficulty, explainable RNN models have been created. These models can offer insights into the model's decision-making process by highlighting significant aspects in the input data through the use of attention mechanisms (Wu et al., 2021).

2.11 Early Works and Technological Development of Emotion Detection System

Recurrent Neural Networks (RNNs) have formed the basis for a number of technology advancements in mood detection in recent years. Table 2.1 shows an overview of the early research and technology advancements in the field of emotion detection.

Table 2.1: An overview of the early research and technology advancements in the field of emotion detection.

Year	**Methods**	**Features**
2021	Attention Mechanism	Allows models to focus on important features of the input data, which can improve its accuracy.
2020	Hybrid RNN	This model has shown promising results in recognizing different moods.

2017	Word Embeddings	This technique allows the model to capture the semantic meaning of words and improve its accuracy.
2020	Contextual Features	Provide additional information that can enhance the model's performance.
2020	Transfer Learning	Involves using a pre-trained model for a related task and fine-tuning it for mood detection.
2020	Deep Bidirectional RNN	It captures both forward and backward contextual dependencies in the input data.
2021	Transformer-based RNN	it uses self-attention mechanism of transformers to capture long-term dependencies in the input data.
2021	Self-Supervised Learning	Trains the model on tasks related to the input data, such as predicting missing words or reconstructing corrupted data.
2021	Attention-Based Hybrid RNN	combines the benefits of attention mechanisms and hybrid RNNs.
2020	Multi-Task Learning	Involves training the model on multiple related tasks simultaneously.

2.12 Review of Related Works on Emotion Detection Using Bi-LSTM

Recurrent neural networks (RNNs) have been extensively studied in the field of mood detection in recent years. The studies covered below offer a variety of RNN models that use various methods to increase mood detection accuracy. The summary of other related works on mood detection using a Bi-LSTM are presented in Table 2.2.

Table 2.2: Summary of Other Related works on mood detection using a Bi-LSTM

Author(s) & Year	**Title**	**Work done**	**Limitation**
Wu *et al.,* 2023	Mood Detection of Twitter Users Based on Bi-directional LSTM.	Proposed a BiRNN-based model that captures the contextual information of tweets to predict the mood of users.	The authors do not extensively discuss the impact of hyperparameter tuning or potential challenges faced during training and evaluation.
Chen *et al*., 2022	MoodLens: An Emotion-Based Intelligent Music Recommendation System.	An intelligent music recommendation system that employs BiRNNs for mood detection based on textual descriptions.	The authors do not provide an in-depth analysis of potential challenges or limitations specific to the BiRNN-based approach.
Li *et al*., 2020	Mood Classification of COVID-19-Related	The Bi-GRU model captures temporal dependencies in tweets	The study is limited to mood detection specifically for

	Tweets Using a Bi-GRU Neural Network.	and classifies them into different mood categories.	COVID-19-related tweets, which may not generalize well to other domains.
Zhao *et al*., 2022	Emotion Analysis of Textual Conversations Using Recurrent Neural Networks.	The study demonstrates that BiRNNs effectively capture the sequential nature of conversations for accurate emotion analysis.	The paper does not explicitly outline the limitations specific to BiRNNs but rather provides a more general overview of emotion analysis using RNNs.
Wang *et al*., 2022	Multi-Modal Sentiment Analysis with Word-Level Fusion and Reinforcement Learning.	The work utilizes BiRNNs to capture the sequential dependencies in textual data.	The paper does not solely focus on the limitations of BiRNNs but rather presents a broader framework for multi-modal sentiment analysis.

Perera *et al*., 2020	Contextual Sentiment Analysis Using Bidirectional LSTM Models Trained on Twitter.	The BiRNN model captures contextual information to predict sentiment or mood in different contexts.	The paper primarily focuses on sentiment analysis rather than explicitly discussing the limitations specific to mood detection using BiRNNs.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 Analysis of the Existing system

An existing study model was chosen for this and labelled ‘Deep Emotex’. It is a digital mental health intervention that uses artificial intelligence to provide evidence-based therapy and support for individuals experiencing mental health challenges such as anxiety and depression. One of the ways it does this, is through check-ins. It prompts users to enter their moods/emotions and details explaining their mood and respond by suggesting tools skills, and strategies to help. In addition to messaging, users can view a chart of their mood entries over time, and view psychoeducational media.

While there exists several application and website to detect emotion, majority of them are proprietary and not readily available for use, which also brings about the difficulty in gaining access to their internal framework and conceptual design. These emotion detectors are on pay and get basis and in many instances very expensive to purchase. This formed the basis for choosing the selected existing emotion as a study model, because it is a free and readily available system for any interested online user to engage with. After reviewing the selected existing system, we found need for improvement in the system most especially in its demanding users to input their mood.

Hence, by requesting users to input their mood, it shows that there was not much work done to detect the users’ mood from having a conversation with the chatbot, which does not portray smartness of the algorithm used in developing the system. There is a possibility for such

inefficiency to hamper the system's ability to select appropriate emotion been experienced at that point in time and thus, provide the necessary empathy and validated responses to the user.

3.1.1 Limitations of Existing System

a. Ambiguity in Text: The system may misinterpret sarcasm or mixed emotions due to limited context.

b. Dataset Bias: Emotion-labeled datasets often have class imbalances, affecting prediction accuracy for underrepresented categories.

c. Limited Number of Emotion Category: Emotions can be categorized into different form of emotions, but the existing limited it to five categories.

3.2 Analysis of the Proposed System

an improved system is proposed to achieve better emotion detection and empathy. The proposed system is pre-named 'EmoBot', a stand-alone application which can display empathy and detect emotions in a more efficient manner. The core of emotion detection is the preprocessing phase, which involves word and sentence tokenization as well as sentence scoring and ranking. Hence, the proposed model provides an efficient algorithm that can effectively handle these tasks and deliver a more reliable prediction. A major advantage of the proposed model is in the manner in which it classifies emotion, detects emotion, and empathize with the aim of producing meaningful responses.

3.3 Justification of the Proposed System

The goal of this research is to create a Deep Neural system that can identify emotions from text data and classify these emotions to accurately predict the actual emotional class. The proposed

research uses a Bidirectional Long Short-Term Memory (BiLSTM) model for textual data emotion detection. Two deep learning architectures (DLAs) that are frequently used in emotion classification tasks—the Transformer-based model and BiLSTM—will be employed in the initial phase of experimentation to evaluate and adapt these architectures for use in emotion detection, utilizing their advantages in feature extraction and representation learning.

Furthermore, the GloVe pre-trained word embeddings, which are used to convert word to vectors are utilized to the BiLSTM network model. These embeddings, which may be employed directly in the BiLSTM model, are obtained by unsupervised learning techniques from big text datasets.

3.4 Techniques/Method(s)

The following is the approach for categorizing emotions:

Step 1: Gathering and Preparing Data:

a. A dataset of text samples with emotion labels tagged was gathered during this phase. This dataset was obtained from data.world's pre-existing Twitter mood datasets.
b. Tokenizing the sentences into words, eliminating stopwords, and stemming or lemmatizing the text data to normalize the vocabulary were the steps in the preprocessing procedure.
c. To capture semantic associations, words in the text data were converted into dense vector representations using GloVe, a word embedding technique that has been trained beforehand.

Step 2: Dividing the Information:

a. To appropriately assess the model's performance, the dataset was split into three categories: test sets, validation sets, and training sets.

b. To prevent biases, it was made sure that every set included a proportionate distribution of data from various emotion classes.

Step 3: Architecture of the BiLSTM Model: A BiLSTM neural network architecture for emotion categorization was created as the model architecture. Typically, the architecture consists of:

a. Layer of input: The word embeddings for the input text are delivered to this layer.
b. Layer(s) of the BiLSTM: This layer records context from the input sequence both forward and backward.
c. Optional extra layers to enhance model performance and generalization (such as dropout regularization and attention mechanisms).
d. Layer of output: This layer uses softmax activation to generate probability distributions over the target emotion classes.

Step 4: Model Training:

a. Using backpropagation and gradient descent optimization algorithms—specifically, Adam—the BiLSTM model was trained on the training data in this phase.
b. To maximize model performance, hyperparameters including learning rate, batch size, and number of LSTM units were tweaked at this phase using cross-validation on the validation set.

c. In order to avoid overfitting, this phase will also assess performance indicators (such as accuracy and loss) on the validation set and cease training early.

Step 5: Assessing the Model:

a. This phase will involve testing the trained BiLSTM model on the test set to determine how well it performs in categorizing emotions.
b. To measure how well the model captures various emotion categories, this phase will calculate evaluation metrics like accuracy, precision, recall, and F1-score.

The confusion matrix will also be analyzed in this phase to determine the model's advantages and disadvantages for categorizing different emotions.

Step 6: Visualization and Interpretation of the Model:

a. In this case, the model's predictions and attention weights are visualized to help understand how the model decides what to forecast and what sections of the input text have the greatest influence on the results.

b. Misclassifications and errors will be identified in order to obtain information about possible areas for model biases or improvement.

Step 7: Optimizing and fine-tuning

a. To further improve classification performance, this phase refines the BiLSTM model and experiments with various architectural modifications, hyperparameters, and training procedures.

b. To adapt the model to new datasets or domains with limited labeled data, further strategies like domain adaptation or transfer learning will be investigated.

Step 8: Integration and Deployment:

a. The trained BiLSTM model is included into a chatbot during this step.

b. In this instance, we keep an eye on the model and make updates to guarantee its longevity and efficacy while taking user preferences and shifting data distributions into account.

3.4.1 Pre-Processing

Preprocess the dataset by carrying out the subsequent actions:

a. Tokenization: In this case, the text would be divided into distinct words or subwords. b. Lowercasing: To maintain consistency, we will also change all text to lowercase during this phase.
b. Eliminating stopwords: Common terms that don't add to the emotional content should be removed.
c. Managing special characters: Eliminating non-alphanumeric characters and punctuation. e. Padding sequences: By padding or truncating as needed, you may make sure that every input sequence is the same length.
e. Encoding labels: Use one-hot encoding, for example, to translate emotion labels into numerical representations.
f. Handling Numbers: Depending on how pertinent the numbers are to the emotion detection task, determine whether to leave them alone, change them to a generic token (like ""), or maintain them as is.

g. Managing Emoticons and Emoji: Since Twitter will be the source of this data, emojis may be encountered. In this stage, we want to determine how to handle emoticons and emoji and whether to replace them when this occurs.

h. Addressing URLs and Emails: Either eliminate or replace URLs and email addresses with generic tokens (such as "" or "") as they might not be relevant to the purpose of classifying emotions.

i. Padding Sequences: By truncating longer sequences or padding shorter sequences with a specific token (such as ""), this step will help to ensure that all input sequences have the same length. Batch processing in neural networks requires this.

3.4.2 Train-Validation-Test Split

Divided into two subgroups using the preprocessed dataset:

a. Training set: This is where the BiLSTM model is trained. This is where the majority of the data and the data used to train the model will be concentrated. The training data ratio is 80 percent.

b. Test set: Applied to assess the trained model's ultimate performance. The model's ability to generalize to new data is tested using the data that was omitted from the model during training. The test data has a 20% ratio.

3.5 Hyperparameter Tuning

To maximize model performance at this stage, various hyperparameters were experimented with:

The quantity of LSTM units and layers.

a. The rate of dropout.

b. The optimizer and learning rate (like Adam).

c. The number of epochs and the batch size.

3.6 Dataset Visualization

a. The visualization will display the accurate amount of data that is available and eliminate any outliers that do not follow the format requirements.

b. MATPlotlib and seaborn are suggested as appropriate visualization tools for the learnt dataset's visual output.

3.7 Datasets

The emotive Tweeter dataset was employed. This dataset includes emotional annotations on tweets. Fine-grained emotion categories including fear, rage, sadness, and happiness are among them. Table 3.1 shows the data distribution among different classes.

Table 3.1: Data distribution among different classes

Classes	Anger	Relief	Bored	Happiness	hate	Fun	Love	Surprise	worry	Enthusiasm	Sad	empty	Neutral
Size	**1** **2** . . . **8638**	**1** **2** . . . **8459**	**1** **2** . . . **5209**	**1** **2** . . . **5165**	**1** **2** . . . **3842**	**1** **2** . . . **2187**	**1** **2** . . . **1776**	**1** **2** . . . **1526**	**1** **2** . . . **1323**	**1** **2** . . . **827**	**1** **2** . . . **759**	**1** **2** . . . **179**	**1** **2** . . . **110**

3.8 Evaluation Methods

a. Accuracy and F1 Score: These are widely used metrics to assess how well emotion detection models perform overall.
b. Confusion Matrix: Offers information on the model's performance in terms of accuracy, recall, and F1 score for every class in relation to various emotion categories.
c. Cross-validation: By training and assessing the model on several subsets of the data, this technique aids in determining the model's capacity for generalization.
d. Emotion Intensity Prediction: Some research assesses models not just for their ability to classify emotions but also for their capacity to predict the degree or intensity of emotions conveyed in text data.

3.9 Model Equations

1.Input Embedding Layer:

i. The input sequence $X = \{x_1, x_2, ..., x_T\}$, where T is the length of the sequence, is converted into dense vector representations using an embedding matrix.
ii. Let E be the embedding matrix of dimensions $V \times d$, where V is the vocabulary size and d is the dimensionality of the embeddings.
iii. The embedding lookup operation yields the embedded input sequence $X_{\text{embed}} = \{e_1, e_2, ..., e_T\}$.

2. Bidirectional LSTM Layer:

a. Forward LSTM:

i. The forward LSTM processes the input sequence X_{embed} in the forward direction.
ii. At each time step t, the forward LSTM computes:

a. Forget gate f_t^{fwd}

b. Input gate i_t^{fwd}

c. Candidate value $\tilde{c}_t^{fwd}$

d. Update cell state c_t^{fwd}

e. Output gate o_t^{fwd}

f. Hidden state h_t^{fwd}

b. Backward LSTM:

i. The backward LSTM processes the input sequence X_{embed} in the backward direction.

ii. At each time step *t*, the backward LSTM computes similar operations as the forward LSTM but with backward context.

iii. Denote the corresponding quantities with superscript bwd.

c. Merge Layer:

i. Concatenate the forward and backward hidden states at each time step to obtain the merged hidden state: $h_t = [h_t^{fwd} ; h_t^{bwd}]$

d. Output Layer:

i. Apply a dense layer followed by softmax activation to predict the probability distribution over emotion classes.

ii. Let W_{out} be the weight matrix and b_{out} be the bias vector of the output layer.

iii. The output probabilities $\hat{y}$ for each class *c* are computed as: $\hat{y}^t,c$ = softmax $(W_{\text{out}}\, h_t + b_{\text{out}})\, c$

In summary, the equations for emotion detection using BiLSTM can be expressed as:

a. Embedding lookup: $X_{\text{embed}} = E \cdot X$

b. Forward LSTM operations at time step *t*:

I. $f_t^{fwd} = \sigma\,(w_f^{fwd} \cdot [h_{t-1}^{fwd}, x_t] + b_f^{fwd})$

II. $i_t^{fwd} = \sigma\,(w_i^{fwd} \cdot [h_{t-1}^{fwd}, x_t] + b_i^{fwd})$

III. $\tilde{c}_t^{fwd} = \tanh\,(w_c^{fwd} \cdot [h_{t-1}^{fwd}, x_t] + b_c^{fwd})$

IV. $c_t^{fwd} = f_t^{fwd} \cdot c_{t-1}^{fwd}\, i_t^{fwd} \cdot \tilde{c}_t^{fwd}$

V. $o_t^{fwd} = \sigma\,(w_o^{fwd} \cdot [h_{t-1}^{fwd}, x_t] + b_o^{fwd})$

VI. $h_t^{fwd} = o_t^{fwd} \cdot \tanh(c_t^{fwd})$

c. $f_t^{fwd} = \sigma\,(w_f^{fwd} \cdot [h_{t-1}^{fwd}, x_t] + b_f^{fwd})$:

I. This equation computes the forget gate f_t^{fwd} at time step *t*.

II. σ is the sigmoid activation function.

III. w_f^{fwd} is the weight matrix for the forget gate.

IV. h_{t-1}^{fwd} is the previous hidden state.

V. x_t is the current input.

VI. b_f^{fwd} is the bias vector for the forget gate.

d. $i_t^{fwd} = \sigma\,(w_i^{fwd} \cdot [h_{t-1}^{fwd}, x_t] + b_i^{fwd})$:

I. This equation computes the input gate i_t^{fwd} at time step *t*.

II. Similar to the forget gate equation, but computes the input gate instead.

e. $\tilde{c}_t^{fwd} = \tanh(W_C^{fwd} \cdot [h_t^{fwd}, x_t] + b_c^{fwd})$:

I. This equation computes the candidate value $\tilde{c}_t^{fwd}$ at time step *t*.

II. tanh is the hyperbolic tangent activation function.

III. W_C^{fwd} is the weight matrix for the candidate value.

IV. Similar to previous equations, but computes the candidate value instead.

f. $c_t^{fwd}=f_t^{fwd}\cdot c_{t-1}^{fwd}+i_t^{fwd}\cdot \tilde{c}_t^{fwd}$:

I. This equation updates the cell state c_t^{fwd} at time step *t*.

II. Combines the previous cell state c_{t-1}^{fwd} with the forget gate and input gate values.

g. $o_t^{fwd} = \sigma\,(w_o^{fwd}\cdot[h_{t-1}^{fwd},x_t] + b_o^{fwd})$:

I. This equation computes the output gate o_t^{fwd} at time step *t*.

II. Similar to previous equations, but computes the output gate instead.

h. $h_t^{fwd} = o_t^{fwd}\cdot \tanh(C_t^{fwd})$:

I. This equation computes the hidden state h_t^{fwd} at time step *t*.

II. Multiplies the output gate with the hyperbolic tangent of the updated cell state to produce the hidden state.

i. Similar equations for the backward LSTM with superscript bwd.

j. Merge Layer: $h_t = [h_t^{fwd};h_t^{bwd}]$

k. Output Layer: $y\hat{}t=\text{softmax}(w_{0ut}^{ht}+b_{\text{out}})$

These equations capture the processing steps of a BiLSTM model for emotion detection, where the model learns to predict the probabilities of different emotion classes based on input sequences.

3.10 Proposed Model Architecture

Figure 3.1 presents the suggested system architecture and provides a detailed description of the emotion detection system's organizational structure. The input module uploads the source document into the system based on the architecture. After the input has been retrieved and processed, it is tokenized using a specified LSTM algorithm. Through the output module, the user receives the final emotion recognition that has been produced.

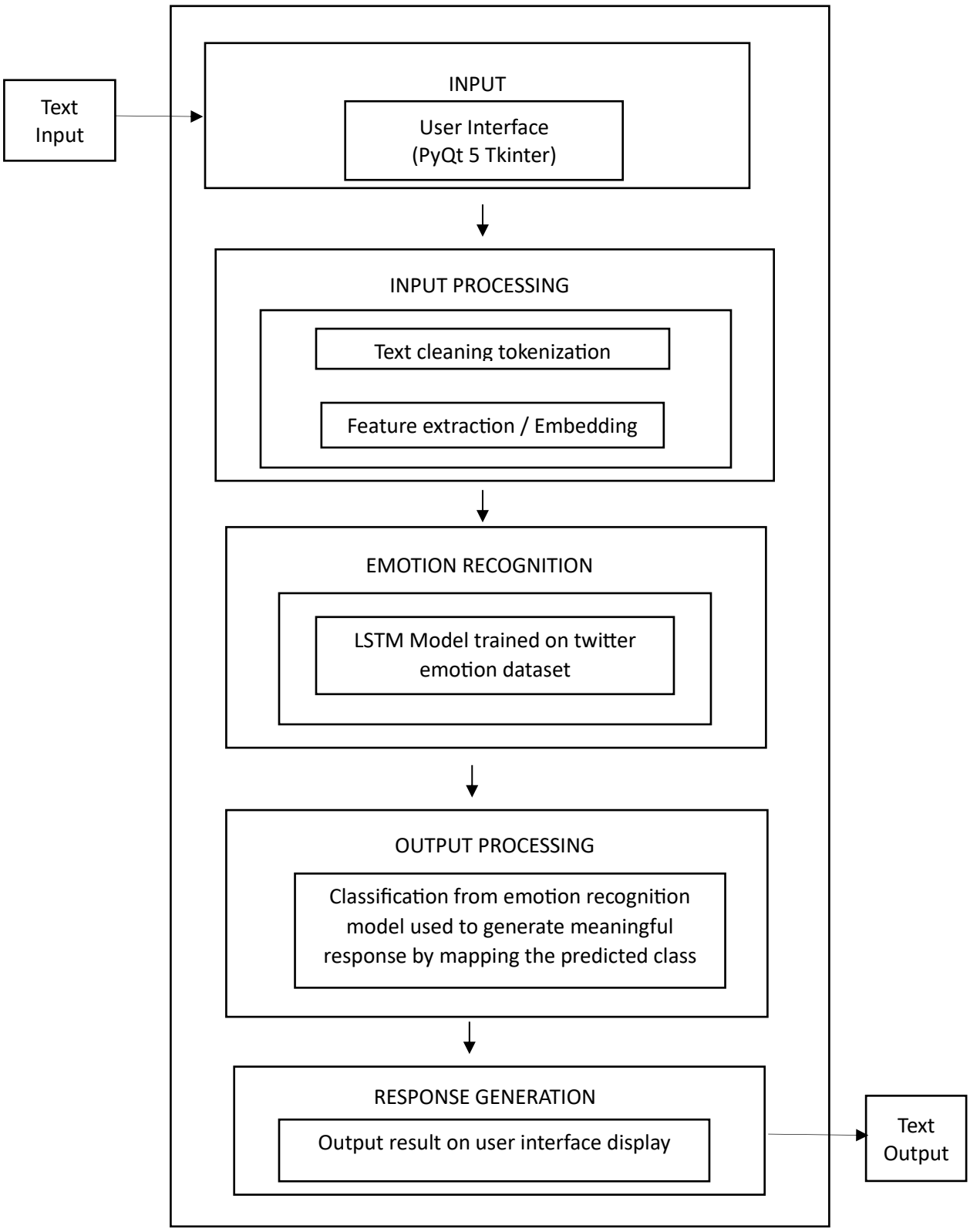


Figure 3.1: Proposed System Architecture

3.11 Methodological flowchart

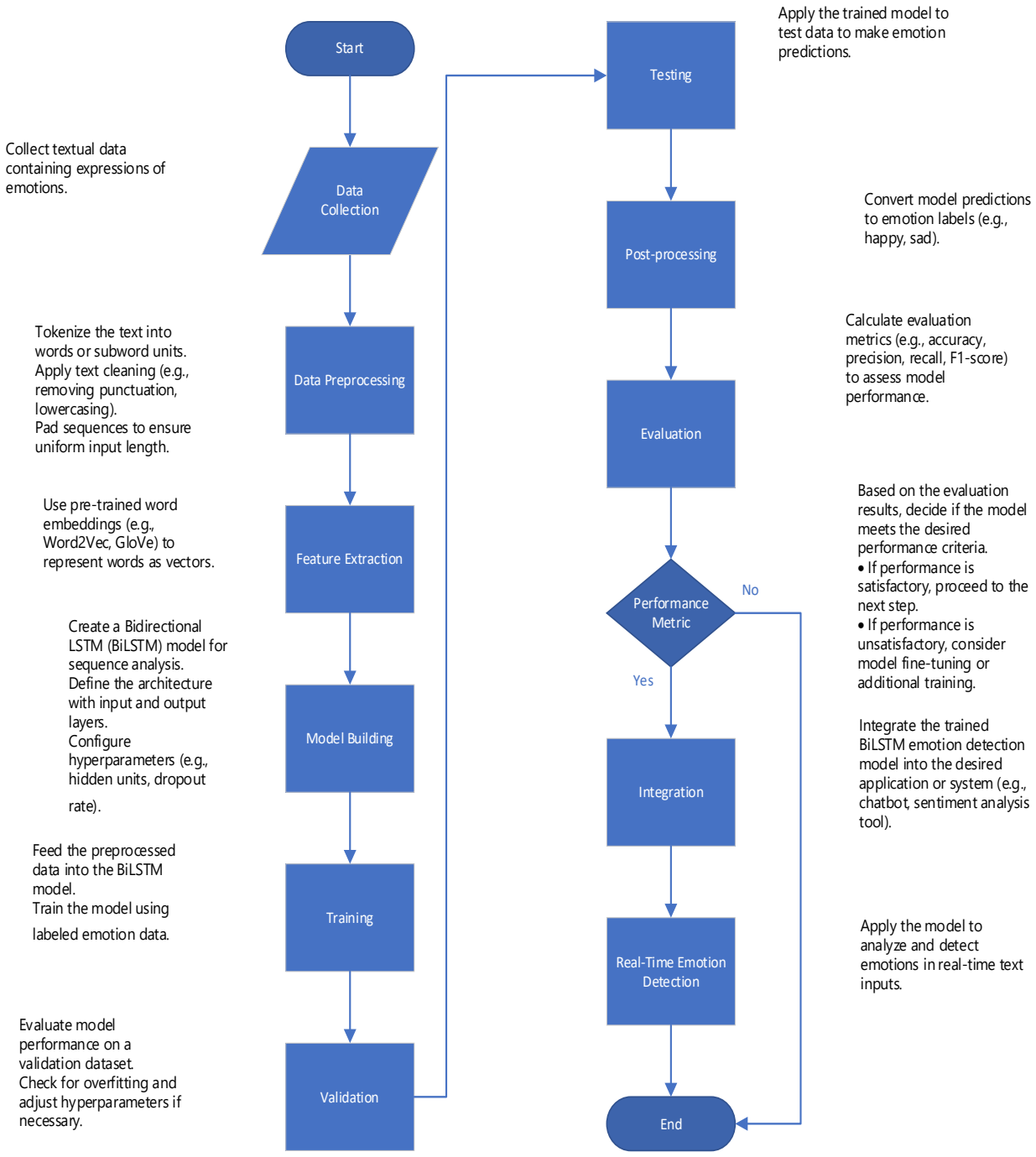


Figure 3.2 Methodical flowchart

3.12 Model Integration with Chatbot Application

A chatbot application framework will incorporate the taught emotion detection model.

3.13 Evaluation Metrics on the chatbot

Metrics like user reaction time and accuracy in identifying user emotions will be used to evaluate the chatbot's performance both before and after the emotion detection model is integrated.

3.14 Web Chat Interface

Steps: 1. Development: Python programming language was used to create the chatbot, and PyTorch frameworks was utilized to detect emotions.

2. Web Interface: A user-friendly web chat interface was be created so that people may communicate with the chatbot. where functions like a text input field for users to write messages and a chat history display area was present.

3. Backend Setup: The chatbot application was hosted on a backend server. The server was set up so that it can process user requests and reply with pertinent messages in accordance with the logic and emotions identified by the chatbot.

4. Integration: The backend code of the chatbot was incorporated into the trained emotion recognition model. In order to forecast the user's emotional state, it ran a message through the emotion detection model after receiving a message from the user. directing the chatbot's response creation based on the anticipated emotion.

5. Deployment: Google Cloud Platform is the web hosting service where the chatbot's application was hosted.

6. Testing: Verify that the implemented chatbot operates properly and can recognize and react to users' emotional states.

3.15 Research Gaps

Class imbalance affects a lot of emotion detection datasets, with some emotion categories being overrepresented and others being underrepresented. This imbalance can impede the detection of minority emotions and result in biased model predictions. Improving the model's performance in all emotion categories may require addressing data imbalance through methods like cost-sensitive learning, sampling tactics, or data augmentation.

Furthermore, there is a dearth of attention on identifying more subtle or complicated emotional states, whereas some research concentrates on identifying basic emotions like happiness, sorrow, anger, and fear. Emotions with low data prevalence, like surprise or disdain, are difficult for models to identify or distinguish from one another.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Results

4.1.1 Exploratory Data Analysis

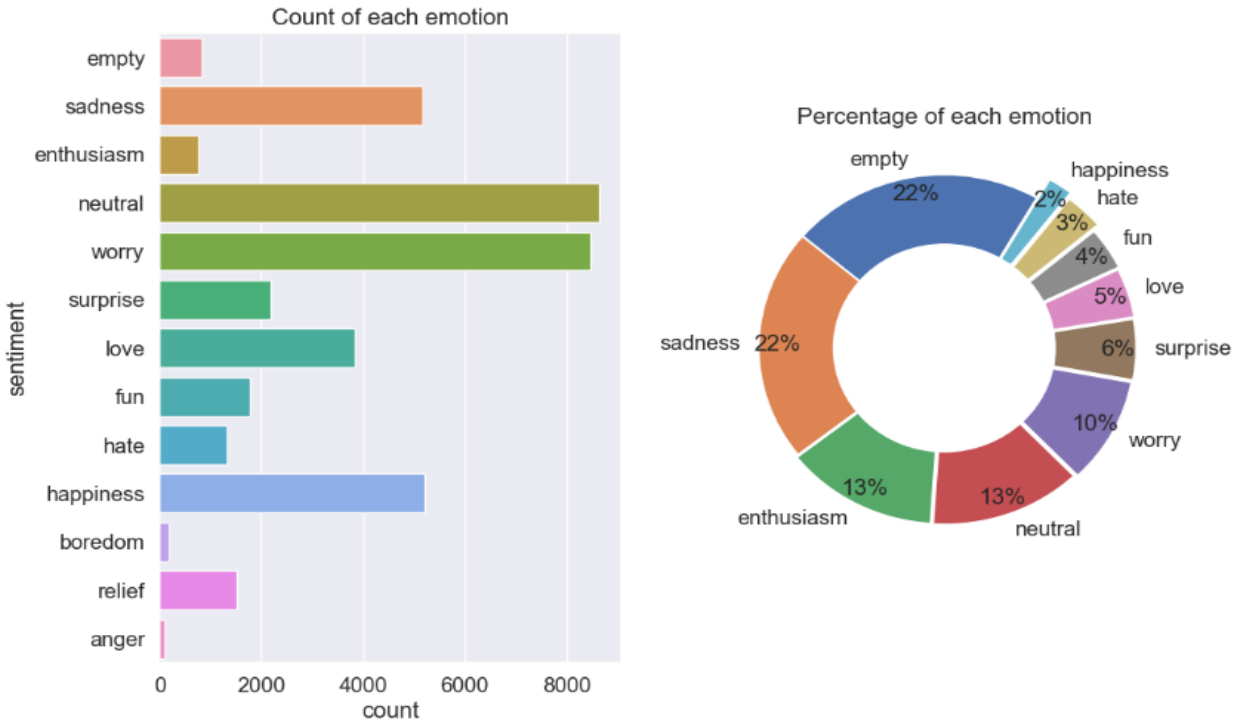


Figure 4.1: Distribution of Sentiments and Checks for Data Imbalance

In Figure 4.1, an imbalance in the data was observed, with certain classes (happy, worry, and neutral) being quite large and others (empty, rage, and boredom) being very little. We then use the data balancing procedure. We must apply a technique known as up-sampling in order to balance the data. To match the largest class in the dataframe (neutral), upsample the remaining classes. Upon achieving data balance, we have Figure 4.2.

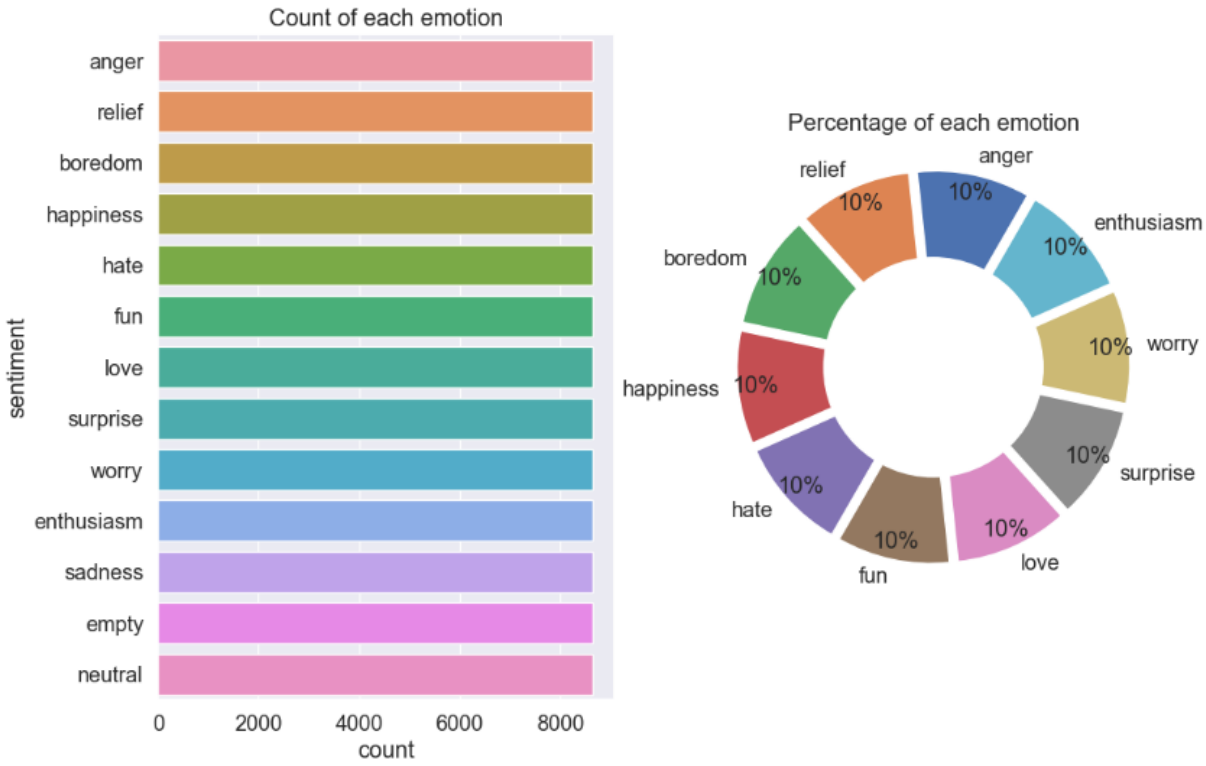


Figure 4.2: Balanced Distribution after upsampling.

4.1.2 Performance Evaluation: Learning Curve

This model takes the data in batches of 702, trains the model, evaluates it, and then goes back to another batch of 702 to repeat the same process; all of this is done to ensure that the model finishes training on time. The total test accuracy that this model achieves is 82.38%. Figure 4.3 depicts the learning curve of 5 epochs, though only 4 epochs were represented.

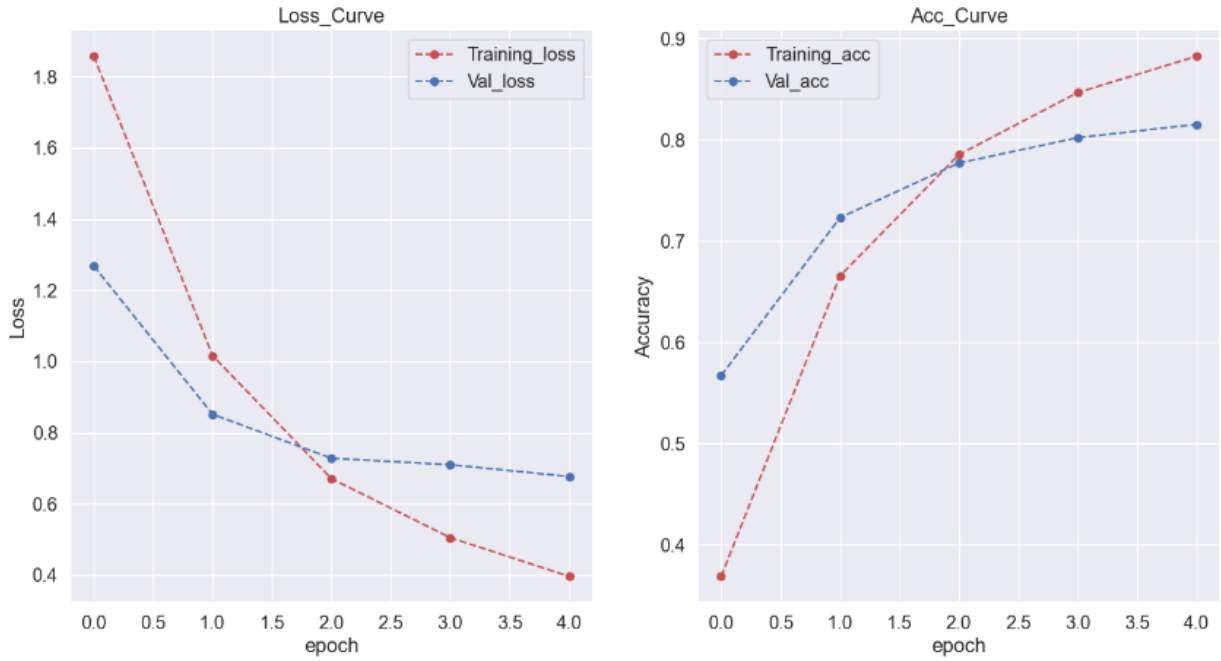


Figure 4.3: Graph showing validation loss/accuracy and training loss/accuracy

There are only four epochs in the learning curve above because, after the fourth epoch, the validation loss stopped declining, training essentially ceased, and the model stopped learning.

4.1.3 Model Architecture

Sentences that have been preprocessed and tokenized are transformed into 100-dimensional Twitter Glove word vectors. We also looked at 300-dimensional vectors, however the memory needs significantly rose and performance was somewhat lower. In order to minimize overfitting problems, a layer trained with dropout was added after the feeding of embedding vectors into a BiLSTM network (Srivastava et al., 2022). Before a final activation layer, the output of the dropout layer was fed into a two-hidden layer network. The two hidden layers were used for the experiments, with the number of neurons varying from 20 to 60 in the first layer and from 100 to

300 in the second. We exclusively concentrate on the 100–64–256–13 architecture, which has the best performance, for the purpose of conciseness (See Figure 4.4). Layers with smaller diameters are

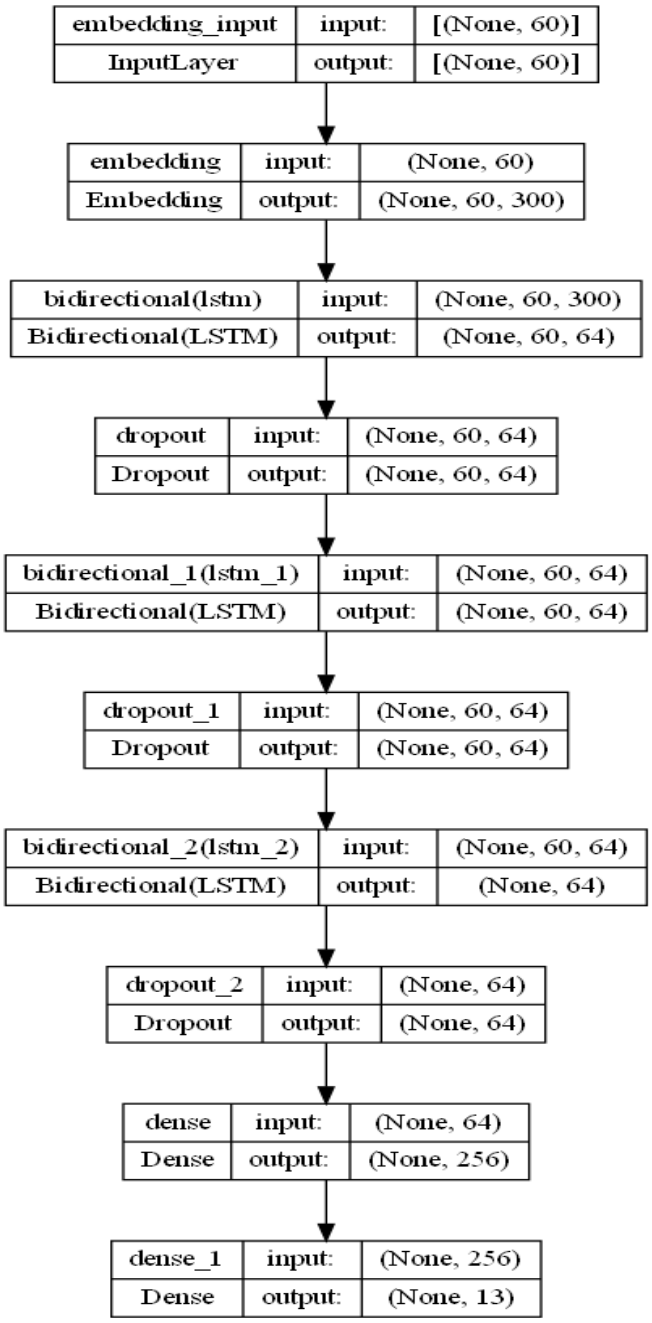


Figure 4.4: Software Architecture

4.1.4 Performance Metrics

On the test dataset, the BiLSTM model used in this study produced an overall accuracy of 82%, with averages for precision, recall, and F1-score of 81%, 81%, and 81%, respectively. These metrics offer a numerical evaluation of the model's ability to categorize emotions, as shown in Table 4.1.

Table 4.1: Evaluation of emotional dataset using a Bi-LSTM model

Classes	**Precision**	**Recall**	**F1-Score**	**Support**
Anger	1.00	1.00	1.00	1748
Relief	0.98	1.00	0.99	1711
Boredom	0.92	0.96	0.94	1739
Happiness	0.92	0.98	0.95	1682
Hate	0.94	0.88	0.91	1755
Fun	0.64	0.70	0.67	1732
Love	0.94	0.97	0.96	1722
Surprise	0.71	0.79	0.75	1738
Worry	0.45	0.39	0.42	1730
Enthusiasm	0.86	0.87	0.86	1727
Sad	0.69	0.69	0.69	1711
Empty	0.87	0.82	0.84	1767
Neutral	0.60	0.56	0.58	1697
Accuracy			**0.82**	**22459**
Macro Avg	**0.81**	**0.81**	**0.81**	**22459**
Weighted Avg	**0.81**	**0.82**	**0.81**	**22459**

4.2 Discussion

4.2.1 Class-Wise Evaluation

a. Anger:

The BiLSTM model achieves remarkable precision, recall, and F1-score averaging around 100% for anger. This indicates that the model accurately identifies instances of anger in textual data without many false positives or false negatives. It suggests that expressions of anger exhibit distinct and recognizable linguistic patterns, making them easily distinguishable for the model.

b. Relief:

Relief demonstrates high precision, recall, and F1-score averaging around 99%. The model effectively captures instances of relief in text data with very few misclassifications. This suggests that expressions of relief are characterized by clear and identifiable features, allowing the model to classify them accurately.

c. Happiness:

Happiness demonstrates high precision, recall, and F1-score values averaging around 95%. The model accurately identifies instances of happiness in text data, suggesting that expressions of happiness are characterized by distinct linguistic features that the model can effectively capture.

d. Boredom:

Boredom exhibits strong precision, recall, and F1-score values averaging around 94%. The model performs well in identifying expressions of boredom in textual data, indicating that boredom-related language possesses discernible patterns that the model can reliably recognize.

e. Hate:

Hate shows strong precision, recall, and F1-score values averaging around 91%. The model performs well in identifying expressions of hate in textual data, indicating that hate-related language possesses identifiable features that the model can accurately classify.

f. Love:

Love demonstrates high precision, recall, and F1-score values averaging around 96%. The model accurately identifies instances of love in textual data, indicating that expressions of love exhibit clear and recognizable linguistic patterns.

g. Enthusiasm:

Enthusiasm demonstrates strong precision, recall, and F1-score values averaging around 86%. The model accurately identifies instances of enthusiasm in textual data, suggesting that expressions of enthusiasm possess clear and identifiable linguistic features.

h. Surprise:

Surprise shows moderate precision, recall, and F1-score values averaging around 75%. The model performs decently in identifying instances of surprise in text data, suggesting that surprise-related language may exhibit some variability that the model struggles to capture consistently.

i. Empty:

Empty sentiments demonstrate strong precision, recall, and F1-score values averaging around 84%. The model effectively identifies instances where the emotional content is minimal or absent in text data, suggesting that such instances possess identifiable linguistic characteristics.

j. Sad:

Sadness exhibits moderate precision, recall, and F1-score values averaging around 69%. The model performs reasonably well in identifying instances of sadness in text data, but there may be room for improvement in capturing the subtleties of sadness-related language.

k. Fun:

Fun exhibits lower precision, recall, and F1-score values averaging around 67%. The model struggles more with accurately identifying instances of fun in text data, suggesting that expressions of fun may be more nuanced or context-dependent, posing challenges for classification.

l. Neutral:

Neutral sentiments show lower precision, recall, and F1-score values averaging around 58%. The model struggles more with accurately identifying instances of neutral sentiments in text data, indicating that neutral expressions may exhibit more variability or ambiguity.

m. Worry:

Worry exhibits lower precision, recall, and F1-score values averaging around 42%. The model faces challenges in accurately identifying instances of worry in textual data, indicating that expressions of worry may be more subtle or context-dependent, making them harder to classify.

4.2.2 Analysis and Interpretation

Advantages and disadvantages

An in-depth comprehension of the advantages and disadvantages of the BiLSTM model's emotion identification across several emotion classes is shown by the evaluation of the system. Although the model does a great job of correctly detecting some emotions, like love, relief, and rage, there is always room for improvement, especially when it comes to picking up on more nuanced emotions and neutral feelings.

4.2.3 Confusion Matrix

This confusion matrix offers a thorough understanding of the model's efficacy in identifying emotions for our investigation.

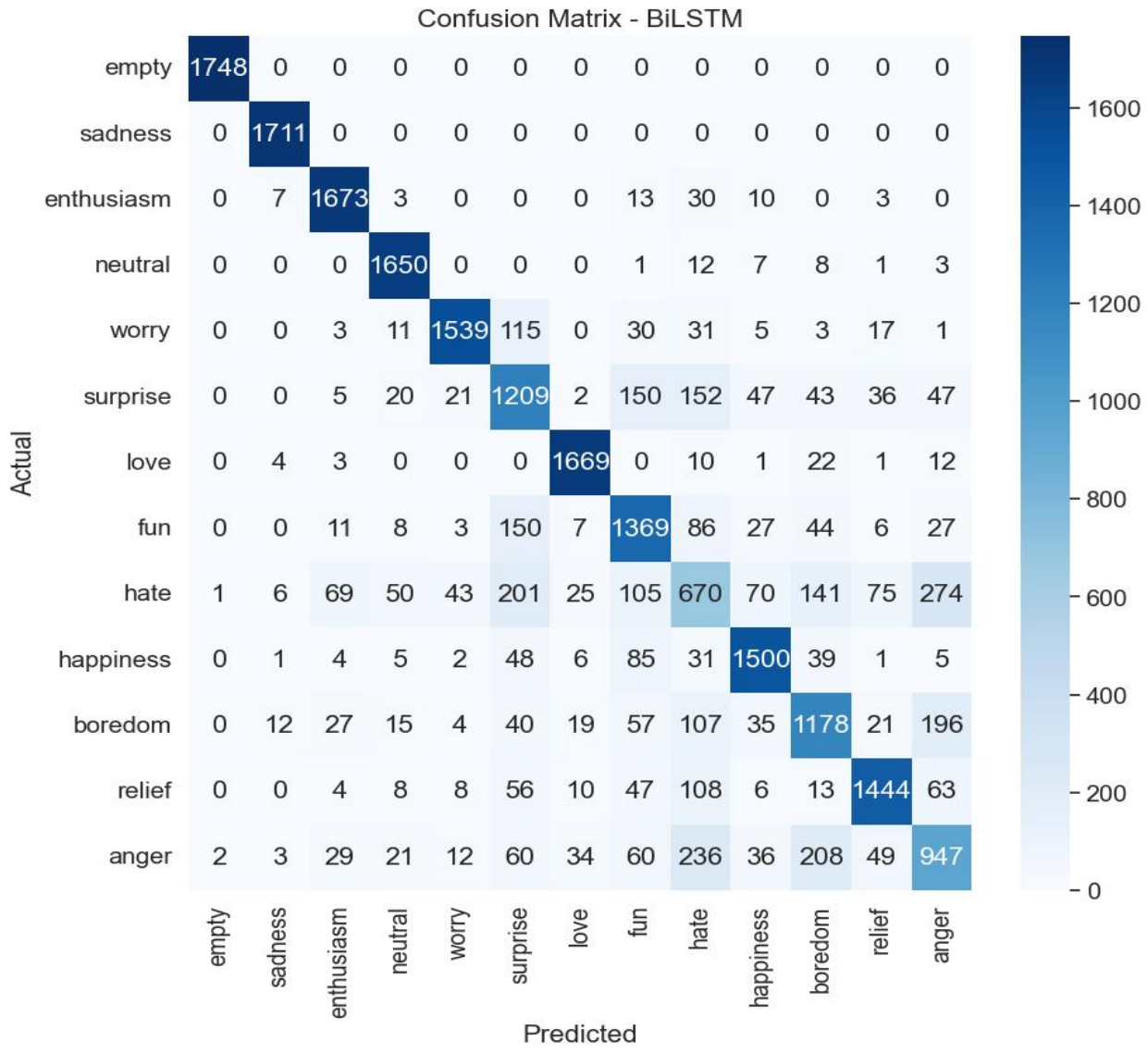


Figure 4.5: Confusion Matrix

4.2.4 Model Integration into Chatbot

Once the BiLSTM model has been successfully integrated into a chatbot, Figure 4.6 displays a chatbot that converses with a user and correctly identifies the user's emotion as "happy" by applying a BiLSTM model. The interface has several elements, such as an input box and a send button. Also, figure 4.7 shows user sad mood predicted in real-time.

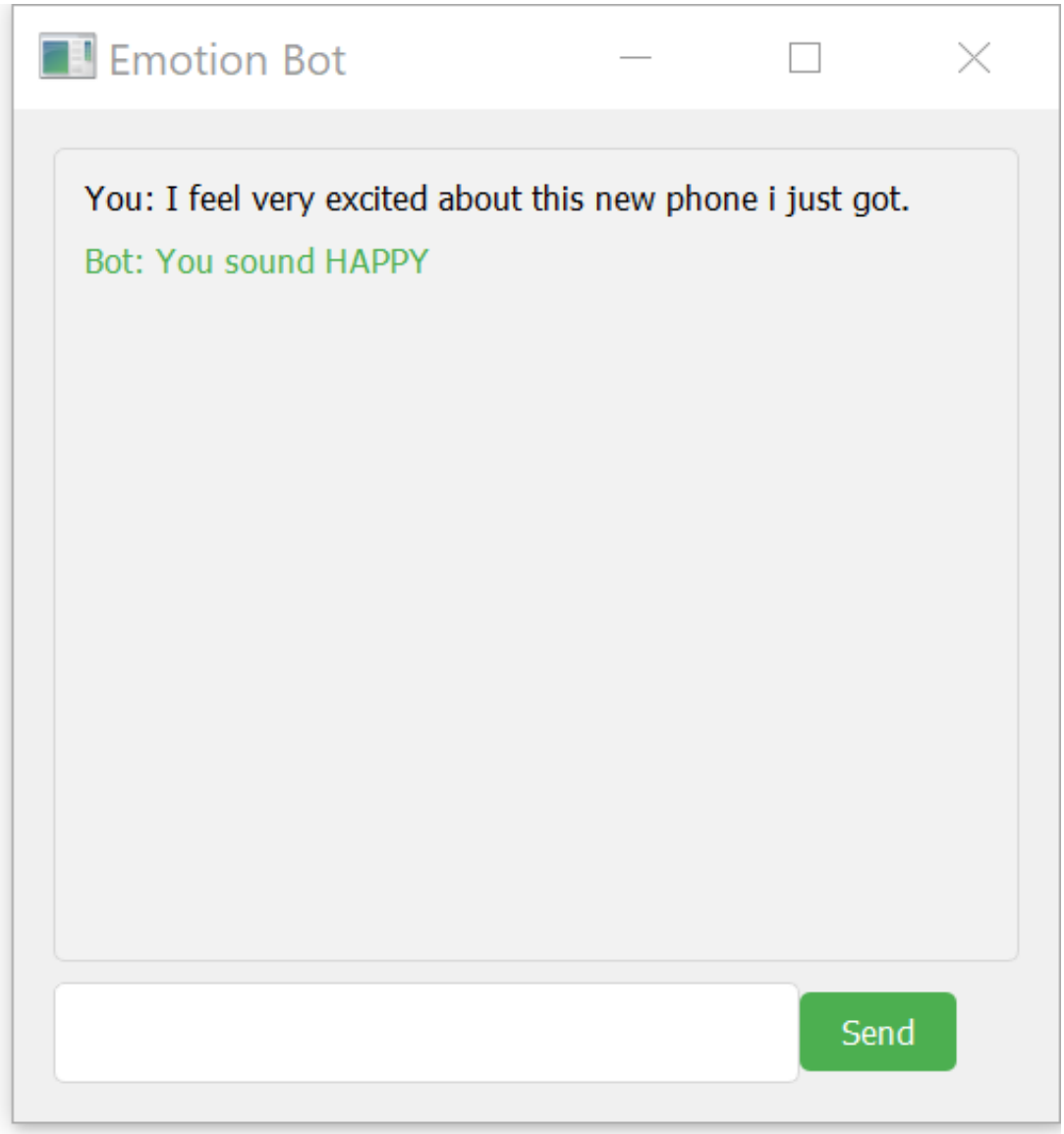


Figure 4.6: User Happy Mood Predicted in Real Time

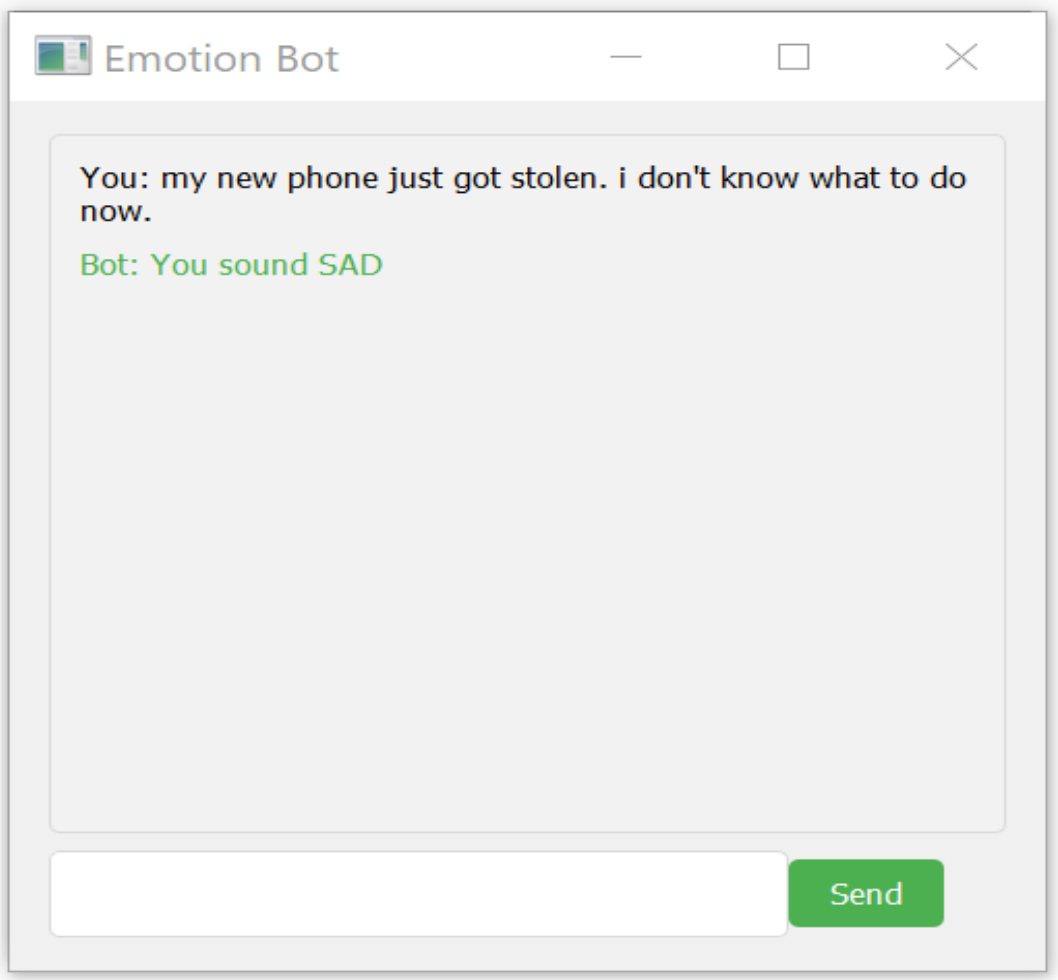


Figure 4.7: User Sad Mood Predicted in Real-Time

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In summary, the incorporation of a Bidirectional Long Short-Term Memory (BiLSTM) model into a chatbot architecture shows promise as a method for conversational emotion identification. The BiLSTM architecture demonstrates a significant ability to comprehend and react to the subtle emotional expressions made by users during interactions, thanks to its capacity to extract contextual information from both past and future sequences.

Our experiments with the BiLSTM model in a chatbot environment have shown promising results in correctly recognizing and reacting to different user emotional states. Because of the model's skill in capturing the temporal dynamics of talks, it can identify minute changes in emotion, which enables the chatbot to respond in a more customized and sympathetic manner.

In addition, the integration of the BiLSTM model improves the user experience in general by allowing the chatbot to modify its tone and replies in accordance with the identified emotions, resulting in a more captivating and fulfilling exchange for users.

It's crucial to recognize that despite the BiLSTM model's enhancement of chatbots' emotional intelligence, there are still issues and room for development. These include managing computing resources for real-time inference in large-scale deployment scenarios, eliminating biases in training data, and handling out-of-context replies.

5.2 Recommendations

This thesis suggests using Bidirectional Long Short-Term Memory (BiLSTM) models in conjunction with continuous training, domain-specific fine-tuning, context-aware features, bias mitigation strategies, computational efficiency enhancements, and the incorporation of user feedback mechanisms in order to maximize the efficacy of emotion detection within chatbots.

5.3 Contribution to Knowledge

Our research contributes to the field in several ways:

1. This method demonstrates strong results in emotion recognition tasks, particularly because BiLSTMs handle temporal and contextual information effectively, making them suitable for sequential text analysis.

2. This research contributes to applications in human-computer interaction, where understanding user emotions can enhance responsiveness and empathy in chatbots or virtual assistants.

REFERENCES

Abad, A., Ortega, A., Teixeira, A., Mateo, C. G., Hinarejos, C. D. M., Perdigão, F., ... & Mamede, N. (Eds.). (2021). Advances in Speech and Language Technologies for Iberian Languages: Third International Conference, IberSPEECH 2021, Lisbon, Portugal, November 23-25, 2021, Proceedings (Vol. 10077). Springer.

Acheampong, F. A., Wenyu, C., & Nunoo-Mensah, H. (2020). Text-based emotion detection: Advances, challenges, and opportunities. Engineering Reports, e12189.

Al-Omari, H., Abdullah, M. A., & Shaikh, S. (2020). Emodet2: Emotion detection in English textual dialogue using BERT and BiLSTM models. In 2020 11th International Conference on Information and Communication Systems (ICICS) (pp. 226–232). IEEE.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Batbaatar, E., Li, M., & Ryu, K. H. (2019). Semantic-emotion neural network for emotion recognition from text. IEEE Access, 7, 866–878.

Bengio, Y., Courville, A., & Vincent, P. (2020). Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), 1798-1828.

Binali, H., Wu, C., & Potdar, V. (2010). "A new significant area: Emotion detection in e-learning using opinion mining techniques." 2010 IEEE International Conference on Digital Ecosystems and Technologies (DEST).

Binali, H., Wu, C., & Potdar, V. (2020). Computational approaches for emotion detection in text. In 4th IEEE International Conference on Digital Ecosystems and Technologies (pp. 172-177). IEEE.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2020). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5, 135–146.

Canales, L., & Martínez-Barco, P. (2020). Emotion detection from text: A survey. In Proceedings of the Workshop on Natural Language Processing in the 5th Information Systems Research Working Days (JISIC) (pp. 37-43).

Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C. et al. (2021). Universal sentence encoder. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing.

Chatterjee, A., Narahari, K. N., Joshi, M., & Agrawal, P. (2021). Semeval-2021 task 3: Emocontext contextual emotion detection in text. In Proceedings of the 13th International Workshop on Semantic Evaluation (pp. 39–48).

Chen, S., Hou, Y., Cui, Y., Che, W., Liu, T., & Yu, X. (2020). Recall and learn: Fine-tuning deep pretrained language models with less forgetting. arXiv preprint arXiv:2004.12651.

Dai, A. M., & Le, Q. V. (2020). Semi-supervised sequence learning. In Advances in Neural Information Processing Systems (pp. 3079–3087).

De Leersnyder, J., Mesquita, B., & Kim, H. S. (2020). Where do my emotions belong? A study of immigrants' emotional acculturation. Personality and Social Psychology Bulletin, 37(4), 451-463.

Deng, Y. (2020). Using deep learning method for Chinese microblog sentiment analysis research. [D]

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2022). BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2020). Decaf: A deep convolutional activation feature for generic visual recognition. In International Conference on Machine Learning (pp. 647–655).

Eckman, P. (2021). Basic Emotions. In Handbook of Cognition and Emotion.

Fadhil, A., & Gabrielli, S. (2017). Addressing Challenges in Promoting Healthy Lifestyles: The AI Chatbot Approach. In Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare (pp. 261–265). ACM.

Felbo, B., Mislove, A., Søgaard, A., Rahwan, I., & Lehmann, S. (2022). Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion, and sarcasm. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (pp. 1615–1625).

Garcıa, M., Maldonado, S., & Vairetti, C. (2021). Efficient n-gram construction for text categorization using feature selection techniques. Intelligent Data Analysis, 25(3), 509–525.

Gers, F. A., Jürgen, S., & Cummins, F. (2020). Learning to forget: Continual prediction with LSTM. Advances in Neural Information Processing Systems, 850-855.

Ghazi, D., Inkpen, D., & Szpakowicz, S. (2021). Detecting emotion stimuli in emotion-bearing sentences. In International Conference on Intelligent Text Processing and Computational Linguistics (pp. 152–165). Springer.

Goodfellow, I., Bengio, Y., & Courville, A. (2020). Deep learning (Vol. 1). Cambridge: MIT Press, 2020, 367-415.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2020). Deep learning. MIT Press.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, L., Wang, G., & Cai, J. (2020). Recent advances in convolutional neural networks. arXiv preprint arXiv:1512.07108.

Gupta, U., Chatterjee, A., Srikanth, R., & Agrawal, P. (2020). A sentiment-and-semantics-based approach for emotion detection in textual conversations. arXiv preprint arXiv:1707.06996.

Hasan, M., Agu, E., & Rundensteiner, E. (2014). Using hashtags as labels for supervised learning of emotions in twitter messages. In Proceedings of the ACM SIGKDD Workshop on Healthcare Informatics, HI-KDD.

Hasan, M., Rundensteiner, E., & Agu, E. (2020). Automatic emotion detection in text streams by analyzing twitter data. *International Journal of Data Science and Analytics*, 7(1), 35–51.

Hasan, M., Rundensteiner, E., & Agu, E. (2021). Emotex: Detecting emotions in twitter messages. In Proceedings of the Sixth ASE International Conference on Social Computing (SocialCom 2021). Academy of Science and Engineering (ASE), USA.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., & Gelly, S. (2019). Parameter-efficient transfer learning for nlp. arXiv preprint arXiv:1902.00751.

Howard, J., & Ruder, S. (2020). Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146.

Imran, A. S., Daudpota, S. M., Kastrati, Z., & Batra, R. (2020). Cross-cultural polarity and emotion detection using sentiment analysis and deep learning on COVID-19 related tweets. IEEE Access, 8, 181074–181090.

Katsigiannis, S., & Ramzan, N. (2022). DREAMER: A database for emotion recognition through EEG and ECG signals from wireless low-cost off-the-shelf devices. *IEEE Journal of Biomedical and Health Informatics,* 22(1), 98-107.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., & Fidler, S. (2022). Skip-thought vectors. In Advances in Neural Information Processing Systems, pp. 3294–3302.

Koper, M., Kim, E., & Klinger, R. (2020). Ims at emoint-2020: Emotion intensity prediction with affective norms, automatically extended resources and deep learning. In Proceedings of

the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pp. 50–57.

Kosti, R., Alvarez, J., Recasens, A., & Lapedriza, A. (2019). Context-based emotion recognition using emotic dataset. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Kratzwald, B., Ilic, S., Kraus, M., Feuerriegel, S., & Prendinger, H. (2020). Deep learning for affective computing: Text-based emotion recognition in decision support. Decision Support Systems, 115, 24–35.

Kruger, J., Epley, N., Parker, J., & Ng, Z. W. (2020). Egocentrism over e-mail: Can we communicate as well as we think? Journal of Personality and Social Psychology, 89(6), 925.

Kumar, A., & Jaiswal, A. (2020). Systematic literature review of sentiment analysis on twitter using soft computing techniques. Concurrency and Computation: Practice and Experience, 32(1).

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). ALBERT: A lite BERT for self-supervised learning of language representations. CoRR, abs/1909.11942.

Li, F., Huang, M., Yang, Y., & Zhu, X. (2018). Emotion classification of microblogs using Bi-LSTM. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 131-136).

Li, Z. Research and system realization of automatic emotion recognition based on multi-domain features of EEG time-space frequency and frequency [D].

Luo, L., & Wang, Y. (2019). Emotionx-hsu: Adopting pre-trained bert for emotion classification. arXiv preprint arXiv:1907.09669.

Mehrabian, A. (2020). Framework for a comprehensive description and measurement of emotional states. Genetic, Social, and General Psychology Monographs.

Mikolov, T., Chen K., Corrado G., et al. (2020). Efficient estimation of word representations in vector space. Computer Science.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2020). Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems (pp. 3111-3119).

Mohammad, S. M., & Bravo-Marquez, F. (2021). WASSA-2021 shared task on emotion intensity. In Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA), Copenhagen, Denmark.

Mohammad, S., & Bravo-Marquez, F. (2020). Emotion intensities in tweets. In Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (SEM 2020). Vancouver, Canada: Association for Computational Linguistics, pp. 65–77.

Mohammad, S., Bravo-Marquez, F., Salameh, M., & Kiritchenko, S. (2020). Semeval-2018 task 1: Affect in tweets. In Proceedings of the 12th International Workshop on Semantic Evaluation, pp. 1–17.

Mollahosseini, A., Hasani, B., & Mahoor, M. H. (2021). Affectnet: A database for facial expression, valence, and arousal computing in the wild. IEEE Transactions on Affective Computing, 10(1), 18-31.

Olaleye, O., Arogundade, O. T., Abayomi-Alli, A., & Adesemowo, A. K. (2021). An ensemble predictive analytics of COVID-19 infodemic tweets using bag of words. In Data Science for COVID-19, Elsevier, pp. 365–380.

Pan, S. J., & Yang, Q. (2020). A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 22(10), 1345–1359.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., & Cournapeau, D. (2021). Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research, 12, 2825–2830.

Pennington, J., Socher, R., & Manning, C. D. (2020). Glove: Global vectors for word representation. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1532–1543).

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2020). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

Peters, M., Ruder, S., & Smith, N. A. (2019). To tune or not to tune? Adapting pretrained representations to diverse tasks, pp. 7–14.

Phang, J., Fevry, T., & Bowman, S. R. (2021). Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. arXiv preprint arXiv:1811.01088.

Plutchik, R. (2021). A general psychoevolutionary theory of emotion. In Theories of Emotion (pp. 3-33). Academic Press.

Polignano, M., Basile, P., de Gemmis, M., & Semeraro, G. (2019). A comparison of word-embeddings in emotion detection from text using bilstm, cnn and self-attention. In Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization, pp. 63–68.

Poria, S., Cambria, E., Bajpai, R., & Hussain, A. (2017). A review of affective computing: From unimodal analysis to multimodal fusion. Information Fusion, 37, 98-125.

Posner, J., Russell, J. A., & Peterson, B. S. (2020). The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. Development and Psychopathology, 17(3), 715.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140), 1–67.

Ragheb, W., Aze, J., Bringay, S., & Servajean, M. (2019). Attention-based modeling for emotion detection and classification in textual conversations. arXiv preprint arXiv:1906.07020.

Ren, Y., Zhang, Y., Zhang, M., & Ji, D. (2021). Context-sensitive Twitter sentiment classification using neural network. In AAAI (pp. 215-221).

Ruder, S., Peters, M. E., Swayamdipta, S., & Wolf, T. (2019). Transfer learning in natural language processing. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials, pp. 15–18.

Russell, J. A. (2021). A circumplex model of affect. Journal of Personality and Social Psychology, 39, 1161–1178.

Saravia, E., Liu, H. C. T., Huang, Y. H., Wu, J., & Chen, Y. S. (2020). Carer: Contextualized affect representations for emotion recognition. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 3687-3697.

Schuster, M., & Paliwal, K. K. (2019). Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, 45(11), 2673–2681.

Sharma, S., Joshi, A., & Bhattacharyya, P. (2020). Attention visualization for sentiment analysis in Twitter. Information Processing & Management, 57(1), 102119.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2021). Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034.

Strapparava, C., & Mihalcea, R. (2021, March). Learning to identify emotions in text. In Proceedings of the 2021 ACM Symposium on Applied Computing, pp. 1556-1560.

Subramanian, R., Wache, J., Abadi, M. K., Vieriu, R. L., Winkler, S., & Sebe, N. (2021). ASCERTAIN: Emotion and personality recognition using commercial sensors. IEEE Transactions on Affective Computing, 9(2), 147-160.

Tao, J., & Tan, T. (2021). Affective computing: A review. In Affective Computing and Intelligent Interaction (pp. 981-995). Springer. doi:10.1007/11573548.

Teng, Z., Vo, D. T., & Zhang, Y. (2020). Context-sensitive lexicon features for neural sentiment analysis. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (pp. 1629-1638).

Trinh, T. H., Dai, A. M., Luong, M. T., & Le, Q. V. (2020). Learning longer-term dependencies in RNNs with auxiliary losses. arXiv preprint arXiv:1803.00144.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2020). Attention is all you need. In Advances in Neural Information Processing Systems (pp. 5998–6008).

Wang, J., Yu, L.-C., Lai, K. R., & Zhang, X. (2021). Dimensional sentiment analysis using a regional cnn-lstm model. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), (Vol. 2, pp. 225–230).

Wang, W., Huang, K., Li, C., & Li, X. (2020). Emotion classification of social media texts using bidirectional LSTM. IEEE Access, 8, 40299-40307.

Wei, J., Liu, A., & Tang, J. (2020). Research on alarm model of digital TV monitoring platform based on deep learning neural network technology. Cable TV Technology, 2020(7).

Xu, H., Liu, B., Shu, L., & Yu, P. S. (2019). BERT post-training for review reading comprehension and aspect-based sentiment analysis. arXiv preprint arXiv:1904.02232.

Xu, T., Ma, C., & Zhang, J. (2019). LAMOST spectral classification based on deep learning. Acta Astronomica Sinica, 60(02).

Yang, M., & Chen, N. (2022). Covering song recognition model based on deep learning and hand-designed feature fusion. Journal of East China University of Science and Technology (Natural Science), 44(05), 138-145.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2019). Learning to attend, copy, and generate for session-based query suggestion. In Proceedings of the 42nd International

ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 165-174).

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2020). Recent trends in deep learning-based natural language processing. IEEE Computational Intelligence Magazine, 13(3), 55-75.

Yu, J., Marujo, L., Jiang, J., Karuturi, P., & Brendel, W. (2020). Improving multi-label emotion classification via sentiment classification with dual attention transfer network. ACL.

Zhang, L., Wang, S., & Liu, B. (2020). Deep learning for sentiment analysis: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(4), e1253.

Zhang, Y., Yuan, H., Wang, J., & Zhang, X. (2017). YNU-HPCC at Emoint2022: Using a CNN-LSTM model for sentiment intensity prediction. In Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment, and Social Media Analysis (pp. 200-204).

Zhuang, Y. (2020). Interpretation of research progress and trends of big data from the perspective of database. Electronic Technology and Software Engineering, 2020(17), 206-206.

APPENDIX A

SOURCE CODE LISTING

Import Requirements

In [1]:

```
import re

import nltk

import string

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from nltk.corpus import stopwords

from nltk.stem import SnowballStemmer, WordNetLemmatizer

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

from tensorflow.keras.utils import to_categorical

from tensorflow.keras.preprocessing.text import Tokenizer
```

```
from tensorflow.keras.preprocessing.sequence import pad_sequences


from tensorflow.keras.optimizers import Adam

from tensorflow.keras.models import Sequential

from tensorflow.keras.callbacks import EarlyStopping

from tensorflow.keras.layers import Dense, LSTM, Embedding, Bidirectional


#nltk.download("stopwords")

stop_words = set(stopwords.words("english"))

lemmatizer= WordNetLemmatizer()


# Modelling

from sklearn.model_selection import train_test_split,KFold, GridSearchCV

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score,confusion_matrix, classification_report

from sklearn.pipeline import Pipeline

from sklearn.metrics import f1_score
```

```
#Lime

from lime import lime_text

from lime.lime_text import LimeTextExplainer

from lime.lime_text import IndexedString,IndexedCharacters

from lime.lime_base import LimeBase

from lime.lime_text import explanation

sns.set(font_scale=1.3)

nltk.download('omw-1.4')



# Logging

import logging

logging.basicConfig(level=logging.INFO)

[nltk_data] Error loading omw-1.4: <urlopen error [Errno 11001]

[nltk_data]     getaddrinfo failed>
```

Read Data

```
In [2]:

df = pd.read_csv('./data/tweet_emotions.csv', delimiter=',')

df.head()
```

Out[2]:

	tweet_id	sentiment	content
0	1956967341	empty	@tiffanylue i know i was listenin to bad habi...
1	1956967666	sadness	Layin n bed with a headache ughhhh...waitin o...
2	1956967696	sadness	Funeral ceremony...gloomy friday...
3	1956967789	enthusiasm	wants to hang out with friends SOON!
4	1956968416	neutral	@dannycastillo We want to trade with someone w...

Exploratory Data Analysis (EDA)

Counting number of tweets in Data

In [3]:

```
print('No. of Tweets: ', df.shape[0])
```

No. of Tweets: 40000

In [4]:

```
print()
print("-- Number of null values in the columns --")
print(df.isnull().sum())
```

```
print()

print('-- Sentiment-Count --')

print()

print(df.sentiment.value_counts())
```

-- Number of null values in the columns --

```
tweet_id     0

sentiment    0

content      0

dtype: int64



Name: sentiment, dtype: int64
```

In [5]:

```
all_classes = df.sentiment.unique().tolist()

print(all_classes)
```

['empty', 'sadness', 'enthusiasm', 'neutral', 'worry', 'surprise', 'love', 'fun', 'hate', 'happiness', 'boredom', 'relief', 'anger']

Distribution of Sentiments and Checks for Data Imbalance

In [6]:

```
col = 'sentiment'

fig, (ax1, ax2)  = plt.subplots(nrows=1, ncols=2, figsize=(12,8))

explode =

list((np.array(list(df[col].dropna().value_counts()))/sum(list(df[col].dropna().value_counts())))[

::-1])[:10]

labels = list(df[col].dropna().unique())[:10]

sizes = df[col].value_counts()[:10]

#ax.pie(sizes, explode=explode, colors=bo, startangle=60, labels=labels,autopct='%1.0f%%',

pctdistance=0.9)

ax2.pie(sizes,  explode=explode, startangle=60, labels=labels,autopct='%1.0f%%',

pctdistance=0.9)

ax2.add_artist(plt.Circle((0,0),0.6,fc='white'))

sns.countplot(y =col, data = df, ax=ax1)

ax1.set_title("Count of each emotion")

ax2.set_title("Percentage of each emotion")

plt.show()
```

Data Balancing by Upsampling

upsample the other classes to match the largest class in dataframe (neural)

In [7]:

```
from sklearn.utils import resample


maxx = 3

df_majority = df[df.sentiment==all_classes[maxx]]


for cl in range(13):

    df_minority = df[df.sentiment==all_classes[cl]]

    df_minority_upsampled = resample(df_minority, replace=True,
n_samples=len(df_majority), random_state=123)

    if cl == 0:

        df_upsampled = pd.concat([df_minority_upsampled, df_majority])

    if cl>0 and cl!=maxx:

        df_upsampled = pd.concat([df_minority_upsampled, df_upsampled])


df_upsampled['sentiment'].value_counts()
```

View Balanced Distribution

In [9]:

```
col = 'sentiment'
```

```
fig, (ax1, ax2)  = plt.subplots(nrows=1, ncols=2, figsize=(12,8))

explode =

list((np.array(list(df_upsampled[col].dropna().value_counts()))/sum(list(df_upsampled[col].dro

pna().value_counts())))[::-1])[:10]

labels = list(df_upsampled[col].dropna().unique())[:10]

sizes = df_upsampled[col].value_counts()[:10]

#ax.pie(sizes, explode=explode, colors=bo, startangle=60, labels=labels,autopct='%1.0f%%',

pctdistance=0.9)

ax2.pie(sizes,  explode=explode, startangle=60, labels=labels,autopct='%1.0f%%',

pctdistance=0.9)

ax2.add_artist(plt.Circle((0,0),0.6,fc='white'))

sns.countplot(y =col, data = df_upsampled, ax=ax1)

ax1.set_title("Count of each emotion")

ax2.set_title("Percentage of each emotion")

plt.show()
```

Distribution of character length and token length overall

In [10]:

```
df_upsampled_copy = df_upsampled.copy()
```

In [11]:

```
df_upsampled_copy['char_length'] = df_upsampled_copy['content'].apply(lambda x : len(x))

df_upsampled_copy['token_length'] = df_upsampled_copy['content'].apply(lambda x :
len(x.split(" ")))
```

In [12]:

```
fig, (ax1, ax2)  = plt.subplots(nrows=1, ncols=2, figsize=(12,6))

sns.histplot(df_upsampled_copy['char_length'], ax=ax1)

sns.histplot(df_upsampled_copy['token_length'], ax=ax2)

ax1.set_title('Number of characters in the tweet')

ax2.set_title('Number of token(words) in the tweet')

plt.show()
```

Distribution of character length sentiment-wise [Top 6 sentiments]

In [13]:

```
fig, ax = plt.subplots(figsize=(16,8))

for sentiment in df_upsampled_copy['sentiment'].value_counts().sort_values()[-
6:].index.tolist():

    #print(sentiment)


sns.kdeplot(df_upsampled_copy[df_upsampled_copy['sentiment']==sentiment]['char_length'],a
x=ax, label=sentiment)
```

```
ax.legend()

ax.set_title("Distribution of character length sentiment-wise [Top 6 sentiments]")

plt.show()
```

Distribution of token length sentiment-wise [Top 6 sentiments]

In [14]:

```
fig, ax = plt.subplots(figsize=(8,6))

for sentiment in df_upsampled_copy['sentiment'].value_counts().sort_values()[-
6:].index.tolist():

    #print(sentiment)


sns.kdeplot(df_upsampled_copy[df_upsampled_copy['sentiment']==sentiment]['token_length'],
ax=ax, label=sentiment)

ax.legend()

ax.set_title("Distribution of token length sentiment-wise [Top 6 sentiments]")

plt.show()
```

Average Character and Token Length for each emotion class

In [15]:

```
avg_df = df_upsampled_copy.groupby('sentiment').agg({'char_length':'mean',
'token_length':'mean'})
```

In [16]:

```
fig, (ax1, ax2)  = plt.subplots(nrows=1, ncols=2, figsize=(20,10))

ax1.bar(avg_df.index, avg_df['char_length'])

ax2.bar(avg_df.index, avg_df['token_length'], color='green')

ax1.set_title('Avg number of characters')

ax2.set_title('Avg number of token(words)')

ax1.set_xticklabels(avg_df.index, rotation = 45)

ax2.set_xticklabels(avg_df.index, rotation = 45)

plt.show()
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_6928\2512640700.py:6: UserWarning:

FixedFormatter should only be used together with FixedLocator

  ax1.set_xticklabels(avg_df.index, rotation = 45)

C:\Users\hp\AppData\Local\Temp\ipykernel_6928\2512640700.py:7: UserWarning:

FixedFormatter should only be used together with FixedLocator

  ax2.set_xticklabels(avg_df.index, rotation = 45)
```

Data Preprocessing

In [17]:

```
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.model_selection import train_test_split
import nltk
#from num2words import num2words
import inflect
import contractions
from bs4 import BeautifulSoup
import re, string, unicodedata
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords


nltk.download('punkt')
nltk.download('wordnet')
from nltk.stem import LancasterStemmer, WordNetLemmatizer
[nltk_data] Error loading punkt: <urlopen error [Errno 11001]
[nltk_data]     getaddrinfo failed>
[nltk_data] Error loading wordnet: <urlopen error [Errno 11001]
[nltk_data]     getaddrinfo failed>
```

Text normalization function

In [18]:

```
def text_preprocessing_platform(df, text_col, remove_stopwords=True):


    def denoise_text(text):
        # Strip html if any. For ex. removing <html>, <p> tags
        soup = BeautifulSoup(text, "html.parser")
        text = soup.get_text()
        # Replace contractions in the text. For ex. didn't -> did not
        text = contractions.fix(text)
        return text
    def remove_non_ascii(words):
        """Remove non-ASCII characters from list of tokenized words"""
        new_words = []
        for word in words:
            new_word = unicodedata.normalize('NFKD', word).encode('ascii', 'ignore').decode('utf-
8', 'ignore')
            new_words.append(new_word)
        return new_words
```

```
def to_lowercase(words):
    """Convert all characters to lowercase from list of tokenized words"""
    new_words = []
    for word in words:
        new_word = word.lower()
        new_words.append(new_word)
    return new_words


def remove_punctuation(words):
    """Remove punctuation from list of tokenized words"""
    new_words = []
    for word in words:
        new_word = re.sub(r'[^\w\s]', '', word)
        if new_word != '':
            new_words.append(new_word)
    return new_words
```

```
    def replace_numbers(words):
        """Replace all interger occurrences in list of tokenized words with textual
representation"""
        p = inflect.engine()
        new_words = []
        for word in words:
            if word.isdigit():
                new_word = p.number_to_words(word)
                new_words.append(new_word)
            else:
                new_words.append(word)
        return new_words


    def remove_stopwords(words):
        """Remove stop words from list of tokenized words"""
        new_words = []
        for word in words:
            if word not in stopwords.words('english'):
```

```
            new_words.append(word)
    return new_words


def stem_words(words):
    """Stem words in list of tokenized words"""
    stemmer = LancasterStemmer()
    stems = []
    for word in words:
        stem = stemmer.stem(word)
        stems.append(stem)
    return stems
def lemmatize_verbs(words):
    """Lemmatize verbs in list of tokenized words"""
    lemmatizer = WordNetLemmatizer()
    lemmas = []
    for word in words:
        lemma = lemmatizer.lemmatize(word, pos='v')
        lemmas.append(lemma)
```

```
    return lemmas


### A wrap-up function for normalization
def normalize_text(words, remove_stopwords):
    words = remove_non_ascii(words)
    words = to_lowercase(words)
    words = remove_punctuation(words)
    words = replace_numbers(words)
    if remove_stopwords:
        words = remove_stopwords(words)
    #words = stem_words(words)
    words = lemmatize_verbs(words)
    return words


# Tokenize tweet into words
def tokenize(text):
    return nltk.word_tokenize(text)
# A overall wrap-up function
```

```
    def text_prepare(text):
        text = denoise_text(text)
        text = ' '.join([x for x in normalize_text(tokenize(text), remove_stopwords)])
        return text

    # run every-step
    df[text_col] = [text_prepare(x) for x in df[text_col]]

    # return processed df
    return df
```

In [19]:

```
print("Before Text Preprocessing")
display(df_upsampled.head()[['content']])
processed_df = text_preprocessing_platform(df_upsampled, 'content',
remove_stopwords=False)
print("After Text Preprocessing")
display(processed_df.head()[['content']])
```

Before Text Preprocessing

	content
39222	@johncmayer you are one of my favorite musicia...
15506	Had a shower. it's 5:55 PM. Triple 5's! Crap, ...
24880	wakey wakey lemon shakeyyyy! haha, goin' 2 sc...
32244	@Buddy021193 i hear you.. it pisses me off haha
33361	@roberto121 that's some serious shit steve. wh...

c:\Users\hp\miniconda3\envs\nlp\lib\site-packages\bs4__init__.py:435: MarkupResemblesLocatorWarning: The input looks more like a filename than markup. You may want to open this file and pass the filehandle into Beautiful Soup.

warnings.warn(

After Text Preprocessing

	content
39222	johncmayer one favorite musiciansartists ever ...
15506	shower five hundred and fifty-five pm triple f...
24880	wakey wakey lemon shakeyyyy haha go two school...

	content
32244	buddy021193 hear piss haha
33361	roberto121 serious shit steve send picture cal...

Train-test Split

Encode sentiment

apply one-hot encoding to the target variable, "sentiment"

In [20]:

```
# Label encoding target column

le = LabelEncoder()

df_upsampled['sentiment'] = le.fit_transform(df_upsampled['sentiment'])



## df for training and prediction:

df = df_upsampled
```

In [21]:

```
preprocess = True

text = 'content'

target = 'sentiment'
```

```
MAX_SEQUENCE_LENGTH = 60
```

In [22]:

```
X = df[text]

y = df[target]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

In [23]:

```
y.tail()
```

Out[23]:

```
39990    8

39991    8

39992    8

39993    8

39995    8

Name: sentiment, dtype: int32
```

In [24]:

```
X.head()
```

Out[24]:

```
39222    johncmayer one favorite musiciansartists ever ...
```

15506 shower five hundred and fifty-five pm triple f...

24880 wakey wakey lemon shakeyyyy haha go two school...

32244 buddy021193 hear piss haha

33361 roberto121 serious shit steve send picture cal...

Name: content, dtype: object

Embedding

Selected embedding:

- **GloVe**: Global Vector for Word Representation

In [25]:

```
def loadData_Tokenizer(X_train, X_test, MAX_NB_WORDS=75000,
MAX_SEQUENCE_LENGTH=60):
    np.random.seed(7)
    text = np.concatenate((X_train, X_test), axis=0)
    text = np.array(text)
    tokenizer = Tokenizer(num_words=MAX_NB_WORDS)
    tokenizer.fit_on_texts(text)
    sequences = tokenizer.texts_to_sequences(text)
    word_index = tokenizer.word_index
```

```
text = pad_sequences(sequences, maxlen=MAX_SEQUENCE_LENGTH)

print('Found %s unique tokens.' % len(word_index))

indices = np.arange(text.shape[0])

# np.random.shuffle(indices)

text = text[indices]

print(text.shape)

X_train = text[0:len(X_train), ]

X_test = text[len(X_train):, ]

embeddings_index = {}

f = open("./glove.42B.300d/glove.42B.300d.txt", encoding='utf-8')

for line in f:

    try:

        values = line.split()

        word = values[0]

        try:

            coefs = np.asarray(values[1:], dtype='float32')

        except:

            pass
```

```
            embeddings_index[word] = coefs
        except UnicodeDecodeError:
            pass
    f.close()
    print('Total %s word vectors.' % len(embeddings_index))
    return (X_train, X_test, word_index,embeddings_index, tokenizer)
```

Model Evaluation Functions

In [26]:

```
from sklearn.metrics import matthews_corrcoef, confusion_matrix
from sklearn import metrics
from sklearn.utils import shuffle
```

In [27]:

```
def get_eval_report(labels, preds):
    mcc = matthews_corrcoef(labels, preds)
    tn, fp, fn, tp = confusion_matrix(labels, preds).ravel()
    precision = (tp)/(tp+fp)
    recall = (tp)/(tp+fn)
    f1 = (2*(precision*recall))/(precision+recall)
```

```
    return {
        "mcc": mcc,
        "tp": tp,
        "tn": tn,
        "fp": fp,
        "fn": fn,
        "precision" : precision,
        "recall" : recall,
        "F1" : f1,
        "accuracy": (tp+tn)/(tp+tn+fp+fn)
    }
def compute_metrics(labels, preds):
    assert len(preds) == len(labels)
    return get_eval_report(labels, preds)


def plot_graphs(history, string):
  plt.plot(history.history[string])
  plt.plot(history.history['val_'+string], '')
```

```
  plt.xlabel("Epochs")
  plt.ylabel(string)
  plt.legend([string, 'val_'+string])
  plt.show()


def class_balance(df, target):
  cls = df[target].value_counts()
  cls.plot(kind='bar')
  plt.show()
```

Confusion Matrix

In [28]:

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns


def plot_confusion_matrix(confusion_mtx, model_name, classes):
    plt.figure(figsize=(10, 10))
```

```
    sns.heatmap(confusion_mtx, annot=True, fmt='d', cmap='Blues', xticklabels=classes,
yticklabels=classes)
    plt.title(f'Confusion Matrix - {model_name}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
```

Model Creation & Training | *Bidirectional LSTM*

In [29]:

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dropout, Dense, Bidirectional


def Build_Model_RNN_Text(word_index, embeddings_index, nclasses,
                         MAX_SEQUENCE_LENGTH=60, EMBEDDING_DIM=300, dropout=0.5,
lstm_node=32,
                         optimizer='adam'):
    """Builds a Bidirectional LSTM-based neural network for text classification using word
embeddings."""
```

```
# Model building

model = Sequential()



# Create an embedding matrix using pre-trained word embeddings

embedding_matrix = np.zeros((len(word_index) + 1, EMBEDDING_DIM))

for word, i in word_index.items():

    embedding_vector = embeddings_index.get(word)

    if embedding_vector is not None:

        embedding_matrix[i] = embedding_vector



# Add the Embedding layer

model.add(Embedding(len(word_index) + 1,

                    EMBEDDING_DIM,

                    weights=[embedding_matrix],

                    input_length=MAX_SEQUENCE_LENGTH,

                    trainable=True))
```

```
    # Add multiple Bidirectional LSTM layers with dropout
    for i in range(2):
        model.add(Bidirectional(LSTM(lstm_node, return_sequences=True,
recurrent_dropout=0.5)))
        model.add(Dropout(dropout))

    # Add the final Bidirectional LSTM layer with dropout
    model.add(Bidirectional(LSTM(lstm_node, recurrent_dropout=0.5)))
    model.add(Dropout(dropout))

    # Add a Dense hidden layer with ReLU activation
    model.add(Dense(256, activation='relu'))

    # Add the output Dense layer with softmax activation for multi-class classification
    model.add(Dense(nclasses, activation='softmax'))

    # Compile the model
    model.compile(loss='sparse_categorical_crossentropy',
```

```
            optimizer=optimizer,
            metrics=['accuracy'])

    return model
```

In [30]:

```
print("Generating Glove Embeddings...")
X_train_Glove,X_test_Glove, word_index,embeddings_index, tokenizer =
loadData_Tokenizer(X_train,X_test,
MAX_SEQUENCE_LENGTH=MAX_SEQUENCE_LENGTH)
```

```
Generating Glove Embeddings...
Found 40818 unique tokens.
(112294, 60)
Total 1917495 word vectors.
```

In [31]:

```
print(X_train_Glove.shape, X_test_Glove.shape)
```

```
(89835, 60) (22459, 60)
```

Model Training

In [32]:

```
import warnings
```

```
import tensorflow as tf


nclasses = len(y_train.unique())


with warnings.catch_warnings():
    print("Building Model ...")
    model_RNN = Build_Model_RNN_Text(word_index,embeddings_index, nclasses)
    model_RNN.summary()
    tf.keras.utils.plot_model(model_RNN, show_shapes = True)
    print("\n Starting Training ... \n")
    RNN_history = model_RNN.fit(X_train_Glove, y_train,
                                validation_data=(X_test_Glove, y_test),
                                epochs=5,
                                batch_size=128,
                                verbose=1)
    warnings.simplefilter("ignore")
Building Model ...
Model: "sequential"
```

Starting Training ...

Epoch 1/5

702/702 [==============================] - 759s 1s/step - loss: 1.8604 - accuracy: 0.3634 - val_loss: 1.1988 - val_accuracy: 0.6007

Epoch 2/5

702/702 [==============================] - 669s 953ms/step - loss: 0.9717 - accuracy: 0.6803 - val_loss: 0.8174 - val_accuracy: 0.7385

Epoch 3/5

702/702 [==============================] - 687s 978ms/step - loss: 0.6374 - accuracy: 0.7952 - val_loss: 0.7117 - val_accuracy: 0.7881

Epoch 4/5

702/702 [==============================] - 850s 1s/step - loss: 0.4652 - accuracy: 0.8591 - val_loss: 0.6401 - val_accuracy: 0.8139

Epoch 5/5

702/702 [==============================] - 1247s 2s/step - loss: 0.3612 - accuracy: 0.8920 - val_loss: 0.6641 - val_accuracy: 0.8236

In [33]:

```
print("\n Plotting results ... \n")


RNN_history = RNN_history.history


fig, axs = plt.subplots(1,2, figsize=(16,8))


axs[0].set_title('Loss_Curve')

ep = range(len(RNN_history['loss']))

axs[0].plot(ep, RNN_history['loss'],'o--r',label = 'Training_loss')

axs[0].plot(ep, RNN_history['val_loss'],'o--b',label = 'Val_loss')

axs[0].set_xlabel('epoch')

axs[0].set_ylabel('Loss')

axs[0].legend()

axs[1].set_title('Acc_Curve')

ep = range(len(RNN_history['loss']))

axs[1].plot(ep, RNN_history['accuracy'],'o--r',label = 'Training_acc')

axs[1].plot(ep, RNN_history['val_accuracy'],'o--b',label = 'Val_acc')

axs[1].set_xlabel('epoch')
```

```
axs[1].set_ylabel('Accuracy')

axs[1].legend()

plt.show()
```

In [34]:

```
# Evaluating Model

predicted_probabilities = model_RNN.predict(X_test_Glove)

predicted_classes = np.argmax(predicted_probabilities, axis=1)

print(metrics.classification_report(y_test, predicted_classes))
```

Plot Confusion Matrix for BiLSTM

In [35]:

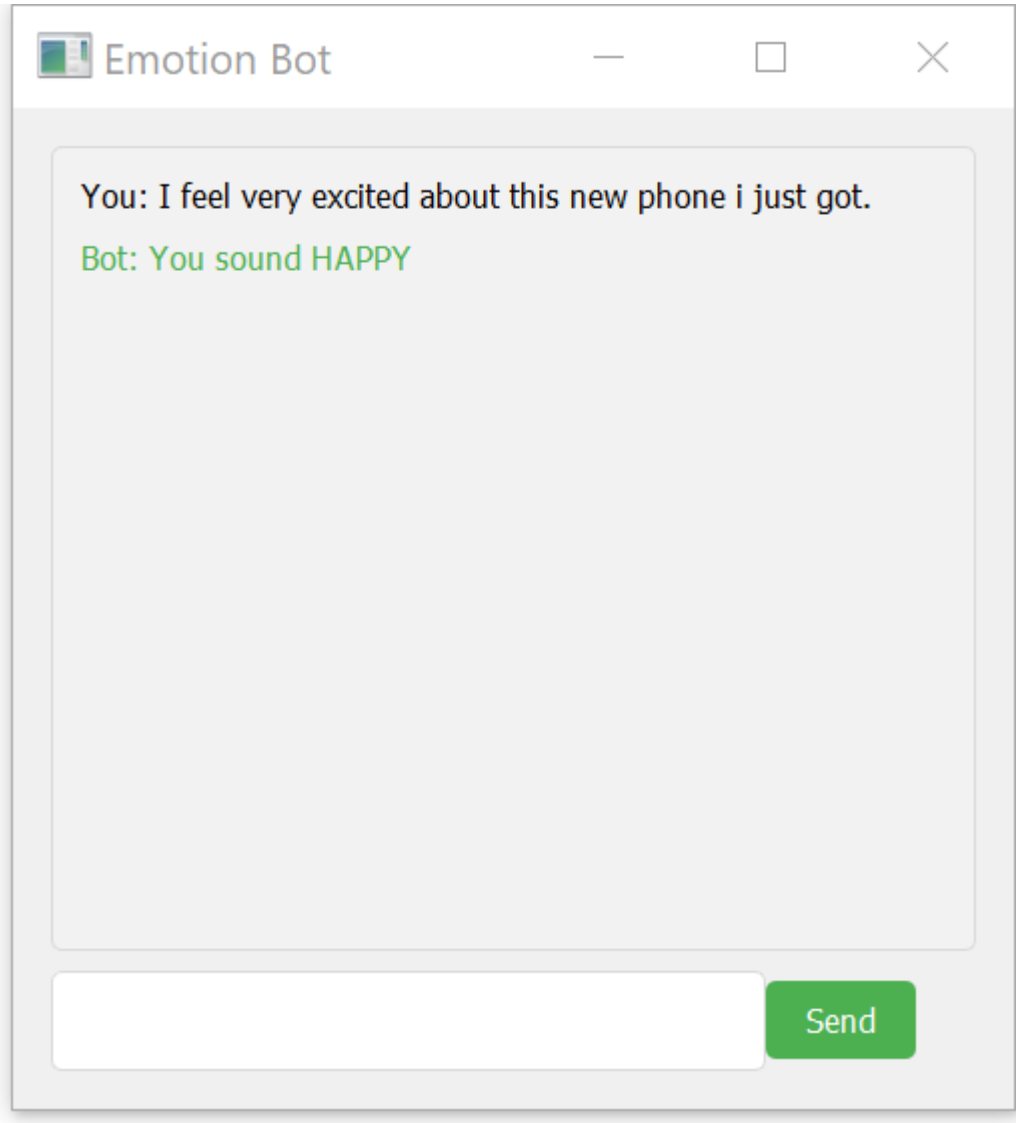
```
confusion_matrix_bilstm = confusion_matrix(y_test, predicted_classes)
```

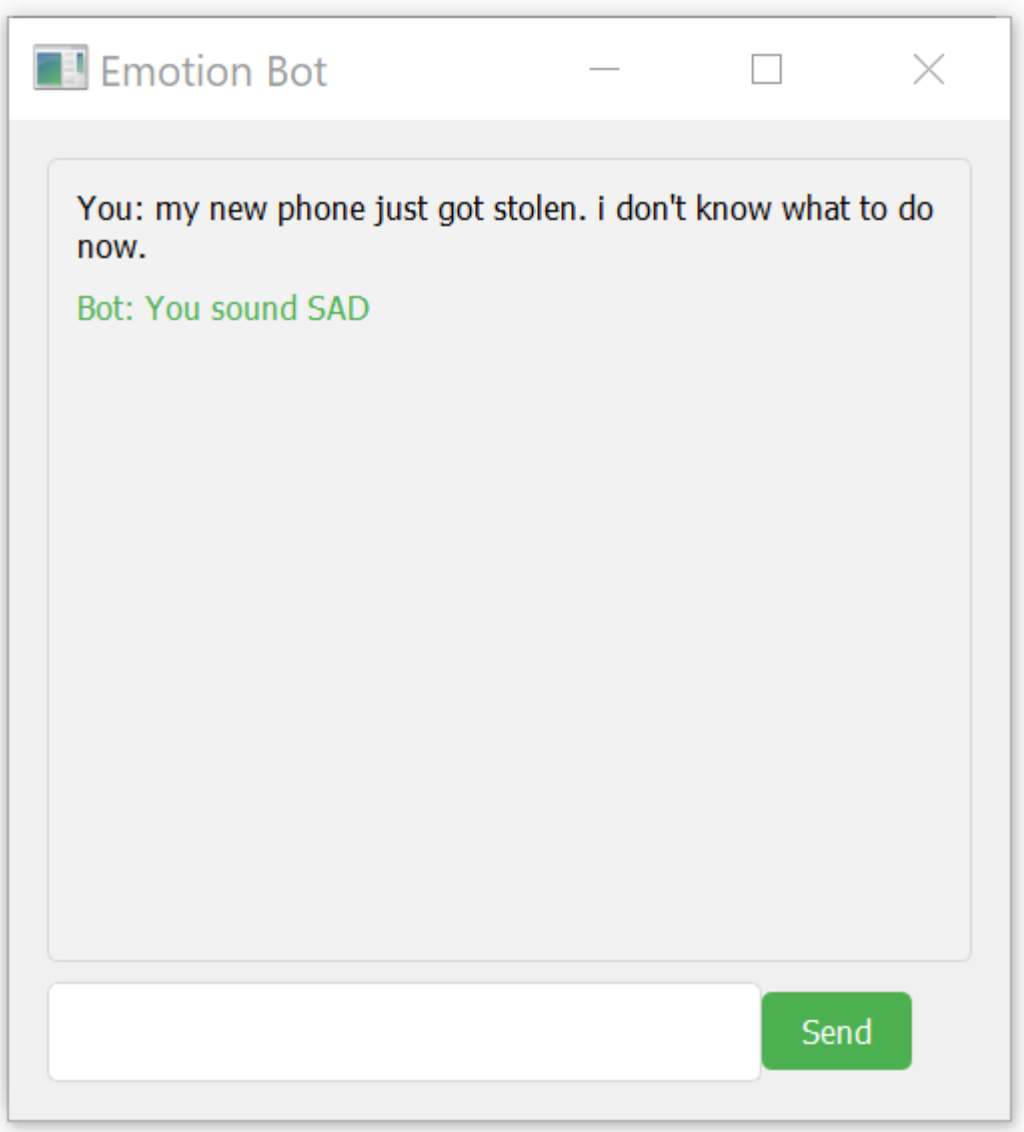
In [36]:

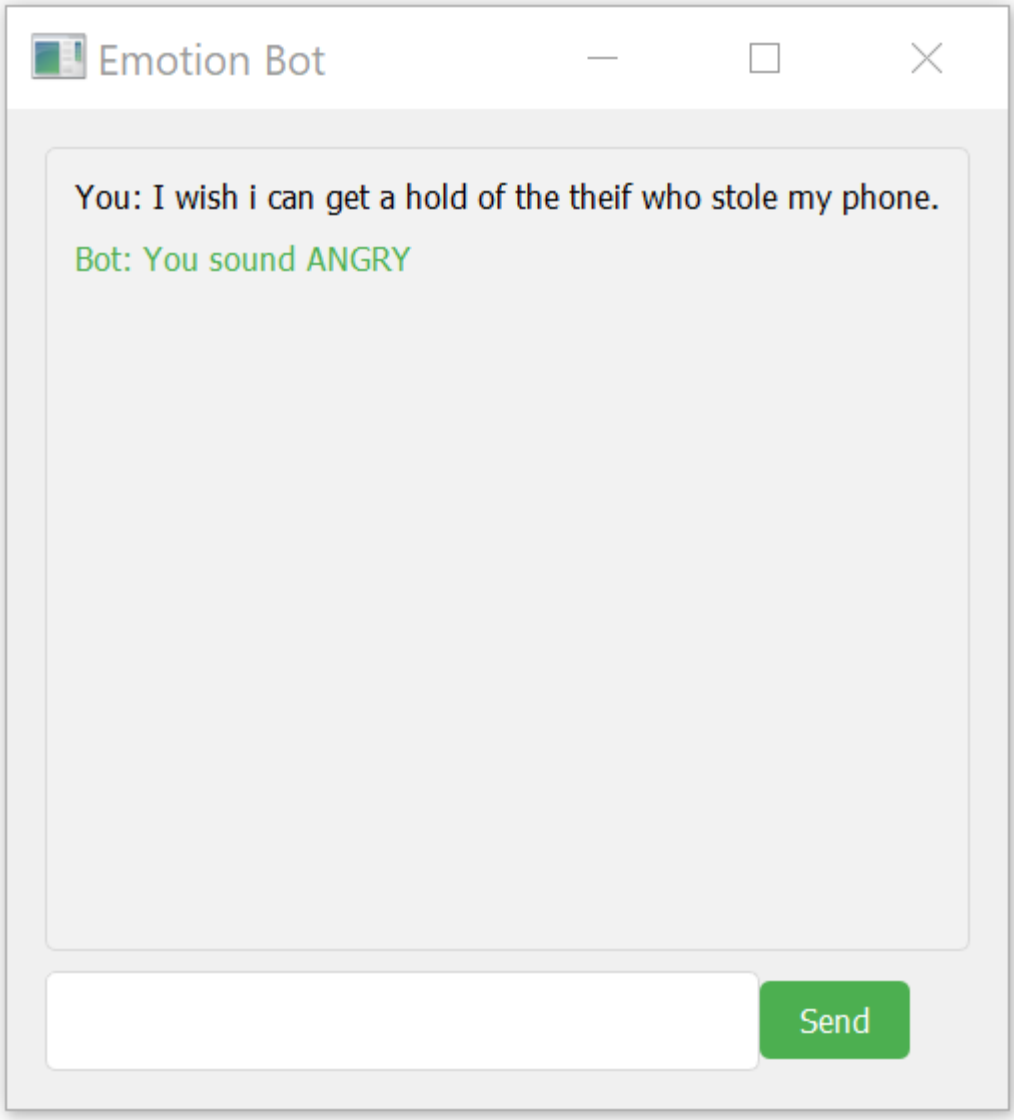
```
plot_confusion_matrix(confusion_matrix_bilstm, "BiLSTM", all_classes)
```

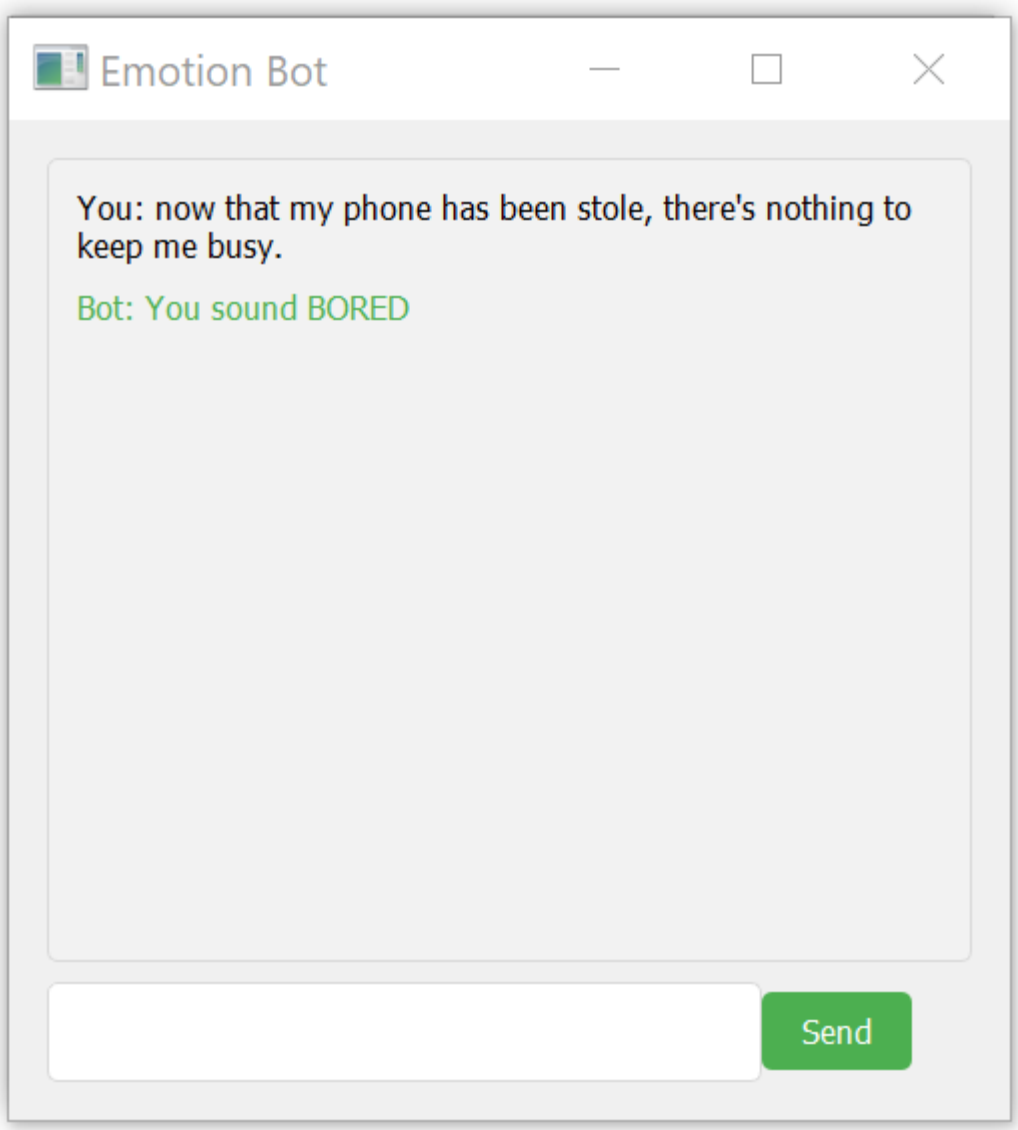
APPENDIX B

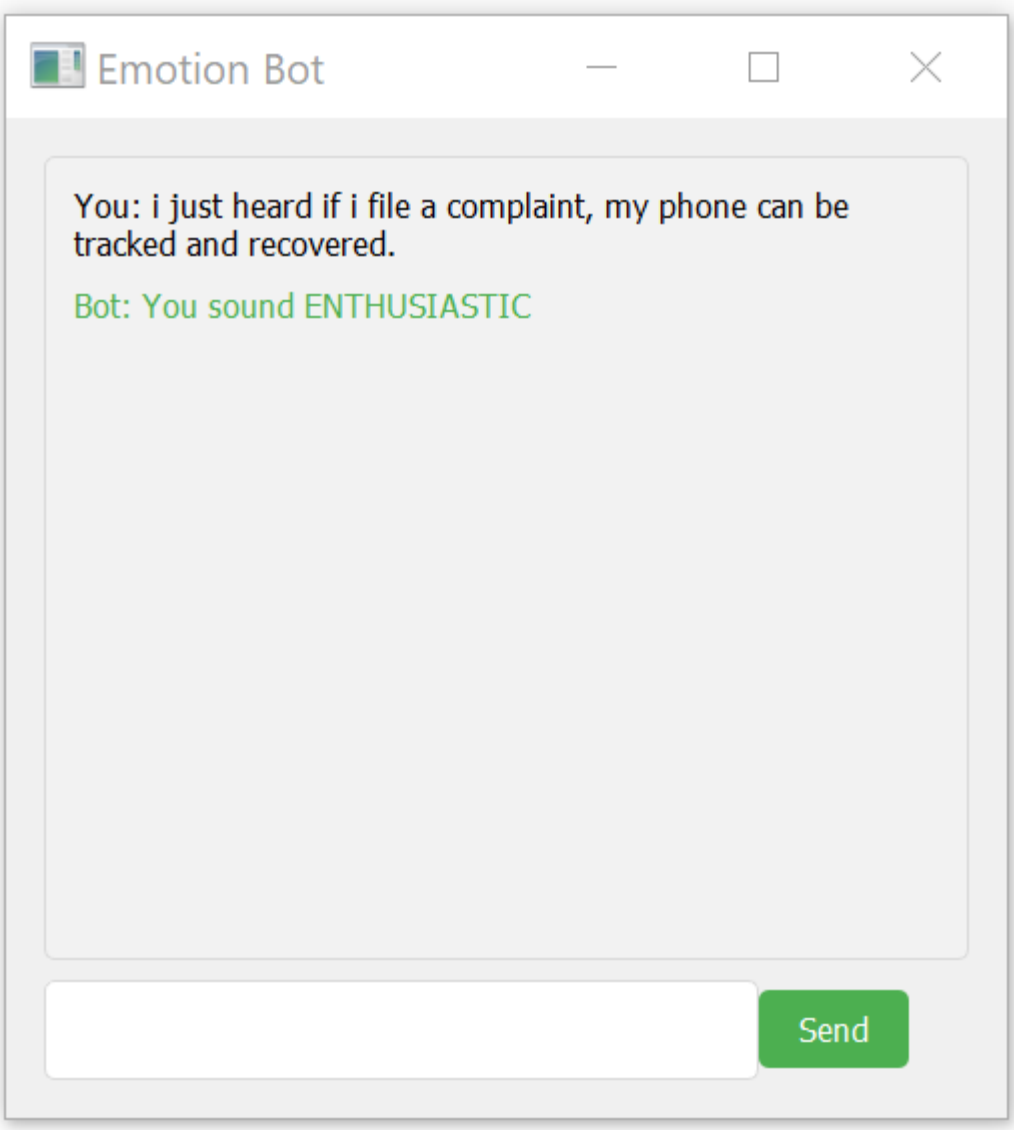
SAMPLE OUTPUTS

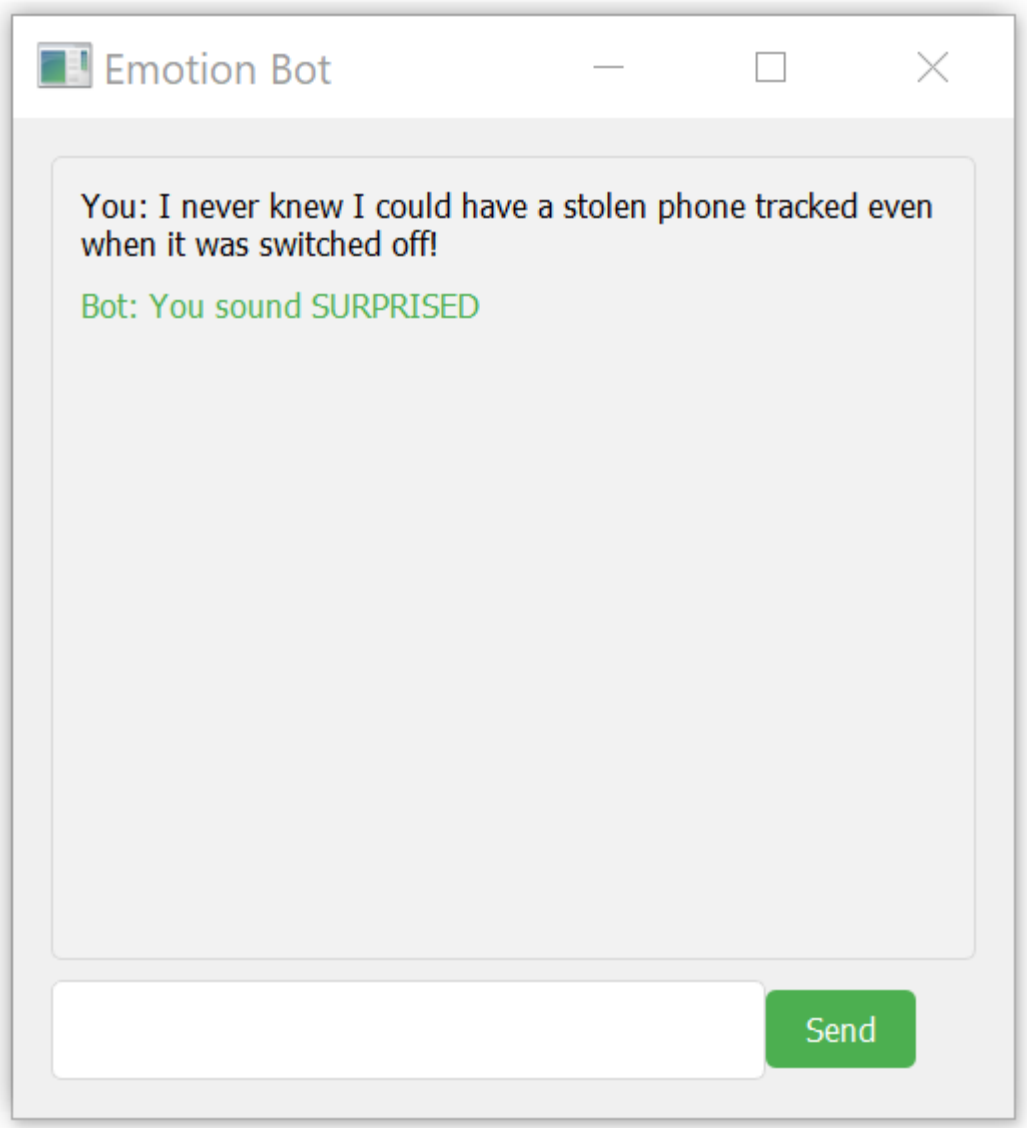


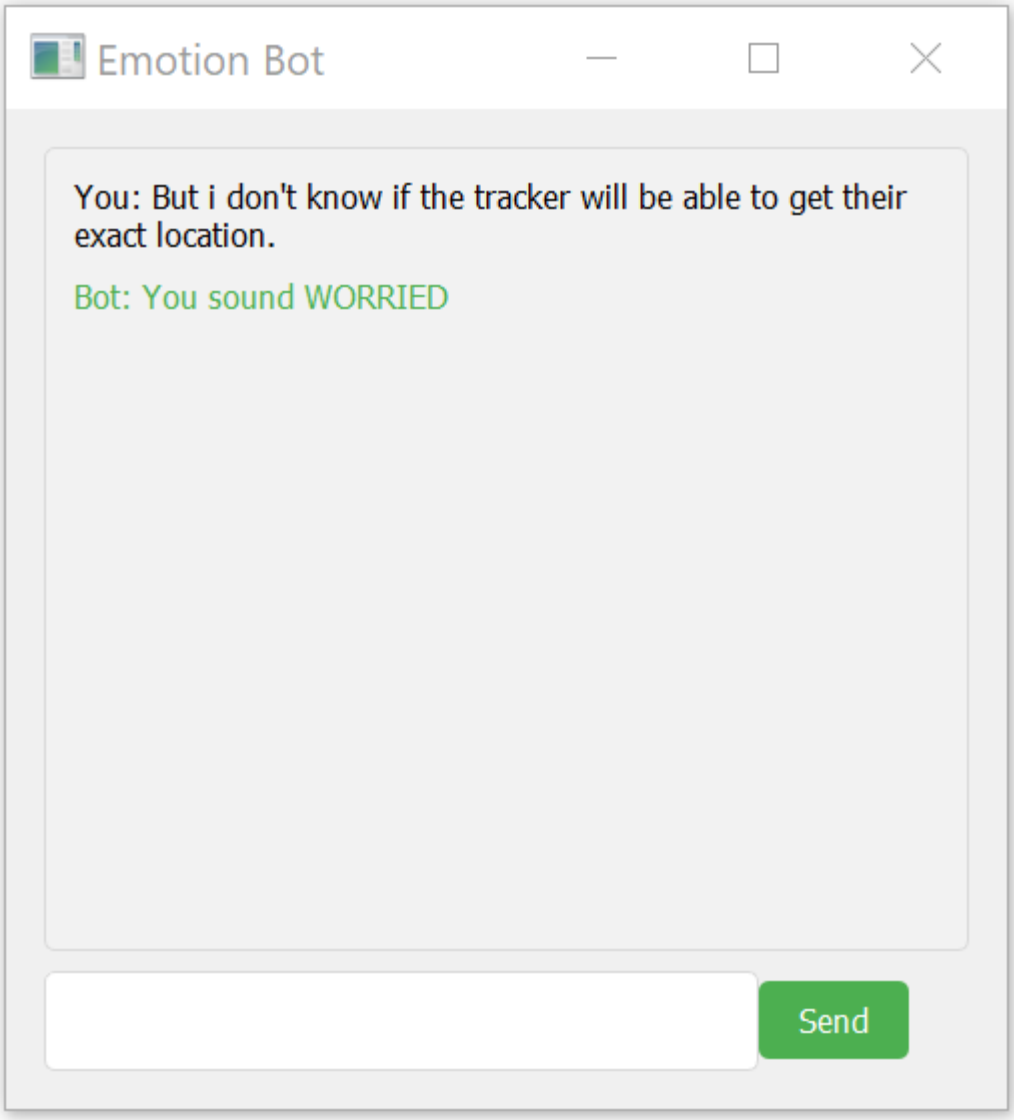

Emotion Bot
You: my new phone just got stolen. i don't know what to do now.
Bot: You sound SAD
Send

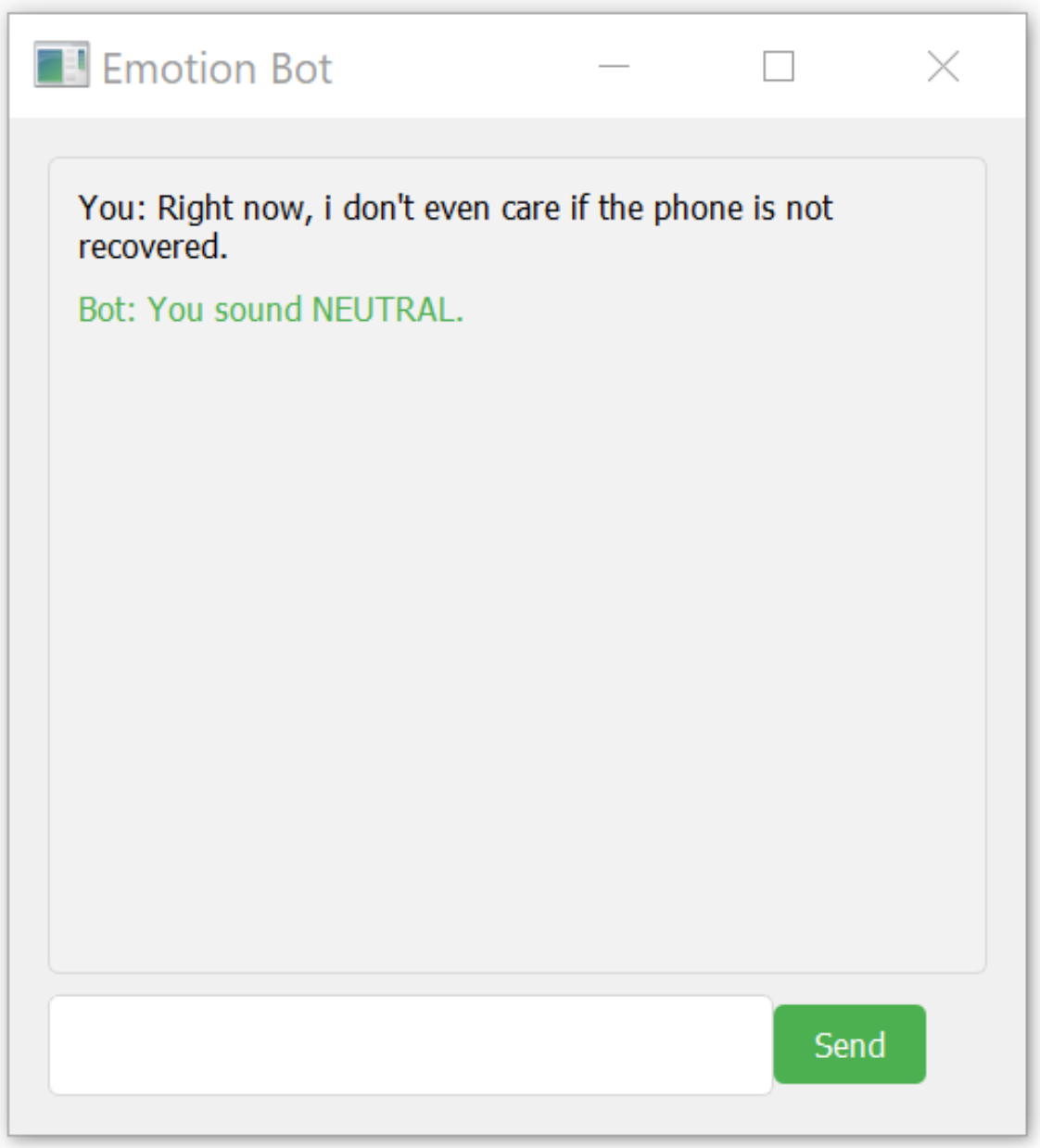

Emotion Bot
You: I wish i can get a hold of the theif who stole my phone.
Bot: You sound ANGRY
Send


Emotion Bot
You: now that my phone has been stole, there's nothing to keep me busy.
Bot: You sound BORED
Send


Emotion Bot
You: i just heard if i file a complaint, my phone can be tracked and recovered.
Bot: You sound ENTHUSIASTIC
Send


Emotion Bot
You: I never knew I could have a stolen phone tracked even when it was switched off!
Bot: You sound SURPRISED
Send


Emotion Bot
You: But i don't know if the tracker will be able to get their exact location.
Bot: You sound WORRIED
Send


Emotion Bot
You: Right now, i don't even care if the phone is not recovered.
Bot: You sound NEUTRAL.
Send