Hybrid Recommendation System of Intelligent Captioning Using Deep Learning Networks

by

Ahamed Al Wasi
21301745
Ezazul Haque Fahim
17101266
Nishat Tasnim Inova
18101112
Abdullah Al Fahim
19101567
Taghrid Tahani Preeti
19301189

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Ahamed Al Wasi
21301745

Ezazul Haque Fahim
17101266

Nishat Tasnim Inova
18101112

Abdullah Al Fahim
19101567

Taghrid Tahani Preeti
19301189

Approval

The thesis titled "Hybrid Recommendation System of Intelligent Captioning Using Deep Learning Network" submitted by

1. Ahamed Al Wasi (21301745)
2. Ezazul Haque Fahim (17101266)
3. Nishat Tasnim Inova (18101112)
4. Abdullah Al Fahim (19101567)
5. Taghrid Tahani Preeti (19301189)

Of Fall 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January, 2023.

Examining Committee:

Supervisor:
(Member)

Moin Mostakim
Senior Lecturer
Department of Computer Science and Engineering
BRAC University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

This research introduces a hybrid recommendation system through sentiment analysis for Bangla long textual sentences. Social media, as a vast source of opinions, can be harnessed through sentiment analysis using deep learning techniques, overcoming language barriers and improving recommendation systems. The paper addresses challenges in Bangla sentiment analysis, such as the scarcity of datasets and linguistic nuances, proposing a model that combines LSTM, Bi-LSTM, and CNN for optimized text sequence classification. The study explores six neural network models (ANN, CNN, LSTM,Bi-LSTM,BERT,RCNN) overcoming obstacles in dataset quality and distribution. Challenges in data collection, model selection, and computational resources are discussed. The paper concludes with the acknowledgment of the evolving frontier of sentiment analysis in Bangla text, emphasizing the transformative potential with continued efforts to expand datasets and refine algorithms.

Keywords: ANN, CNN, LSTM, RCNN, BiLSTM, BERT, Tokenization, Vectorization, Embedding.

Acknowledgement

First and foremost, we want to express our heartfelt gratitude to the Almighty Allah, whose divine guidance and blessings sustained us throughout the journey of this thesis. We are forever indebted to our supervisor , Mr. Moin Mostakim Sir, whose unwavering support and insightful advice were instrumental in navigating the challenges of this research.

To our beloved parents, we extend our deepest love and appreciation.Their sacrifices and constant encouragement provided the strength and motivation we needed to persevere through every obstacle.

As we stand on the verge of graduation, we recognize that this achievement is not ours alone, but a testament to the love, guidance, and support we have received from those who matter most.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

σ Sigma

ANN Artificial Neural Network

$BERT$ Bidirectional Encoder Representations from Transformers

$BiLSTM$ Bidirectional Long Short-Term Memory

CNN Convolutional Neural Network

$LSTM$ Long Short-Term Memory

$RCNN$ Region-based Convolutional Neural Network

Chapter 1

Introduction

Imagine a recommendation system surpassing standard suggestions, crafting personalized user experiences. Achieving this involves a hybrid approach that melds deep learning and natural language processing (NLP) with existing recommendation algorithms. At its core, this approach integrates advanced deep learning models like CNNs and RNNs to analyze extensive user data and multimedia content[14] identifying patterns and learning intricate user-item relationships. NLP techniques, such as word embeddings and sentiment analysis, delve into the meaning and sentiment of associated text, providing context-aware recommendations. The system functions with a recommendation engine generating personalized suggestions, NLP models understanding text[11], and another deep learning model generating informative captions. The benefits include increased accuracy in recommendations, enhanced accessibility through descriptive captions, and improved user engagement, creating meaningful connections between users and the multimedia world.The use of deep learning networks to generate captions also provides an alternative way for visually impaired individuals to understand and access recommended multimedia content. With captions, users can understand the context of the image and make a better-informed decision on whether they want to view the multimedia or not. Additionally, captions can also help to improve the discovery ability of multimedia content, as they provide more information about the content that can be used by search engines and other tools to index and organize the content. Implementing such a hybrid system requires a combination of techniques from the fields of natural language processing, information retrieval, and deep learning. The traditional recommendation algorithm can use techniques such as collaborative filtering or content based filtering, while the deep learning network can be trained on a data set of images and their corresponding captions using an encoder decoder architecture, such as the long short-term memory (LSTM) network and a convolutional neural network (CNN)[8]. One of the challenges in implementing a hybrid recommendation system of this kind is finding a balance between the accuracy and diversity of the recommendations. It's possible to make a recommendation algorithm that is accurate but not diverse, and vice versa. Finding the ideal mix between accuracy and diversity will guarantee the optimum user experience. Another challenge is to make sure that the captions generated by the deep learning network are informative, accurate, and appropriate for the images they are describing. For that measure sentiment analysis can ensure that the results are accurate. Sentiment analysis is a method of NLP that identifies the emotional tone in a piece of text. It is also frequently referred to as opinion mining[3] This is

a wide-used approach by organizations for identifying and group ideas regarding a service or a certain good or concept. Text is mined for information that relies on sentiment and is subjective using data mining, artificial intelligence and machine learning.sentiment analysis systems aid businesses in extracting information from unstructured, unorganized language found in online sources like emails, support tickets, blog posts, forums, web chats, comments and other social media features.

Sentiment analysis can be particularly effective in enhancing the quality of recommendations when only scanty ratings data are available. The fact is that recommendation algorithms choose the things to recommend primarily based on user ratings. These ratings are typically quite low and insufficient. However,The results of sentiments based ratings, which may be produced from comments and evaluations expressed via blogs, social media, online news, and even the recommendation algorithms themselves, seem capable of providing consumers with higher-level suggestions.In order to solve the data-sparsity issue which is plagued traditional recommendation systems and sentiment based approaches have been incorporated into recommendation systems.So,If recommendation system includes sentiment, then it may greatly degrade the quality of the recommendation.
Various of industries including e-commerce, media, finance, and utilities use the applications of recommendation systems.To improve customer pleasure, this type of technology makes tailored suggestions based on a substantial amount of data.All the suggestions are very helpful to client for selecting products meanwhile the organizations can expand the product's consumption.When it comes to social data, sentiment analysis may be quite beneficial in order to gain a better knowledge of a user's behavioral attributes like emotion and opinion, which is ideal to combine in recommendation systems in order to achieve higher suggestion dependability. Additionally, this data can be utilized to reinforce explicit user ratings for products. However, it is thought that the sentiment analysis content that can be found in social media, online news sites, blogs, and the recommendation system itself can give people better suggestions. In sentimental analysis, the three-level extraction feature level, document level, and sentence level can be used. With the help of this procedure, it is possible to learn more about an entity and automatically point out any of its subjectivities. To ascertain whether text is created by humans and also expresses their favorable, unfavorable, or neutral sentiments. Both conventional and deep learning methodologies have been presented as machine learning-based approaches to be used for sentiment analysis. Techniques pertaining to Lexicon are combined with machine learning algorithms in hybrid systems. In comparison to the performance of a single model, the hybrid approaches can improve sentiment analysis accuracy.
In brief, a hybrid recommendation system of intelligent captioning using deep learning networks and sentiment analysis is a promising solution to improving the user experience and accessibility of multimedia content recommendations. Traditional recommendation algorithms can be combined in the right ways to produce a system that is precise, varied, and simple to use.It's an active area of research, and with the advancements in machine learning, we can expect more developments in this field in the near future.

1.1 Problem Statement

When we type something to search for information, we often get suggestions based on the keywords we use. These suggestions help us find what we're looking for. The suggestions are generated using different methods. For English sentences, the system can even detect our sentiment and provide accurate results when we're only halfway through typing. However, when it comes to typing in Bangla, we don't get recommendations based on the sentiment of our sentences. And even if we do get some recommendations, they are not as fast or accurate as the ones for English. Nowadays, typing in Bangla has become easier and more convenient, and many people are using it instead of English or Bang-lish. Additionally, with the increasing availability of smartphones, more people in rural areas of Bangladesh are using them, relying solely on Bangla for communication. They may not have a good understanding of or proficiency in English nut their increasing usage of Bangla language for typing and communication has created a need for an effective recommendation system that analyzes the sentiment of Bangla text. Therefore, if a recommendation system could analyze sentiment in Bangla texts, it would benefit a large number of people. While existing recommendation systems demonstrate accuracy and speed in the English language, the same level of performance is lacking for Bangla sentiment analysis.

The main problem in our project was the lack of enough Bangla datasets for research. It was quite easy to find large datasets in English, but for Bangla, it was much harder. This made it difficult to train and evaluate models and conduct thorough research on sentiment analysis. Another difficulty was the low quality of the Bangla datasets. Duplicate and contradictory data entries were common, adding noise and ambiguity to the study. Similarities between classes, such as "wow" and "surprised," made precise differentiation difficult. Furthermore, the disparity in data distribution across sentiment classes resulted in biased model performance, preventing the development of a credible recommendation system. Bangla and English have significant linguistic differences, which create specific challenges. Bangla has its own phonetics and syntax that require specialized methods for sentiment analysis. Unfortunately, previous studies in this field often overlook these linguistic distinctions, leading to lower accuracy and performance levels. It was difficult to collect Bangla data via online scraping. Many Bangla blogs and internet sources provided disorganized and insufficient information, making it difficult to collect sufficient data for specific sentiment categories. Furthermore, manually enhancing data for categories with very little data proved to be a difficult and time-consuming process. Another challenge in our project was the choosing of models. While we investigated different models, such as Artificial Neural Networks (ANN), mBERT, and Long Short-Term Memory (LSTM), we were unable to identify models that effectively corresponded with our research goals. Furthermore, the huge computational resources required to run these models created major limitations on both time and resources, with each model execution needing at least 8-10 hours of calculation time. In our literature review, we observed that research on Bangla sentiment analysis primarily focused on news corpus and larger blogs, with limited attention given to social media and microblogs. Even within these domains, the existing research lacked the speed and accuracy achieved by sentiment analysis systems in the English language.

1.2 Research Objectives

In this paper we offer a recommendation system of intelligent captioning by sentiment analysis on Bangla long textual sentences. This research paper aims to overcome the obstacles of Bangla language sentiment analysis and supplement existing research. We will develop our model by combining various deep learning techniques such as LSTM, ANN, Bi-LSTM, and CNN to optimize the performance of text sequence classification, taking into account the advantages of using deep learning for text data categorization.
The objective of this research:

1. To take good utilization of social media information, perceive the importance of this data in today's world
2. To develop a model that can understand the sentiment of a Bangla textual sentence
3. To use these results to build a hybrid recommendation system, that can help in captioning
4. To implement the suggested hybrid models and evaluate their performance
5. To address some of the limitations of previous researches
6. To offer a proposed system with better outcomes
7. To make recommendations for further improvement of other models

Chapter 2

Related Work

Text sentiment analysis [5] aims to detect the emotional tone in written messages. One study by Huanhuan Yuan and colleagues from Zhenjiang Analysis InfoTech Ltd. proposed a new approach for this task. This method involves converting text into word vectors using word2vec and then using a LSTM model with attention mechanism for training. The study found that this approach is more effective than traditional machine learning techniques. The LSTM model is implemented to extract features that pertain to the text and the attention technique is employed to generate feature vectors for classifying emotions.

In a research done by Alam and Rahoman [3] it is shown that Convolution Neural Network(CNN) was used to train the model which can analyze Bangla texts. At the time of that particular research, Bangla datasets for analyzing the sentiment were not publicly available. Therefore, to conduct their research, they had to generate their own dataset where they fetched 850 comments which were in Bangla from different platforms. From those comments 500 and 350 comments were positive and negative respectively. In order to make their dataset size large, they had to copy and paste those comments more than 140 times and ensure the same amount of positive and negative comments. Then they applied CNN algorithm on their model to train their model with those comments after the pre-processing phase. After conducting their research they were able to get 99.87% accuracy based on their own dataset. They were able to detect positive and negative sentiment through their model except neutral sentiment. Although their accuracy was high it leaves a room for improvement as there are few Bangla datasets available publicly now and we are not certain how that existing model can work on recent datasets where a small number of comments were not continuously repeated. Additionally, we can aim for detecting neutral sentiment as well.

Another comparable study was carried out by Cheng and Tsai[6], who examined the use of Long Short-Term Memory (LSTM), Bi-directional LSTM, and Gated Recurrent Units (GRU) for sentiment analysis of social media. They utilized web scraping techniques and Python to gather data from platforms such as Facebook and YouTube. The data was then preprocessed and labeled with sentiments to form a balanced dataset for training. Three deep learning models were created for sentiment categorization in the study using word embeddings. Metrics like accuracy, precision, recall, specificity, and F1 score were used to assess the models' performance. The results showed that LSTM achieved the highest performance with 80.83% accuracy, 81.03% precision, 80.32% recall, 74.47% specificity and 80.54% F1 score.. In BiL-

STM the classification performance is 87.17%, 85.80%, 88.89%, 82.88%, 87.29%. In GRU the classification performance is 64.92%, 64.33%, 65.71%, 63.61% ,64.96%.
The Sentimental analysis [4] of posts, product reviews and tweets has become extremely popular. It is used in the intelligence of business and applied in recommender systems. Analyzing through texts on deep learning techniques are becoming better approach to work with. In order to solve few problems in this thesis, deep neural net-works on text polarity analysis and text mining represent a vital role. At first, big sized data sets with properly labeled need to be fed for deep neural networks. Secondly, uncertainties can occur for using vectors that embed words. These are produced from similar sets of data which instructs the model and they are very efficient to route them from popular and big data repositories. Thirdly, it is more convenient to have generic neural network architectures for simplifying model creation which are very effective. It can adjust to numerous texts and summarize the complexity of designs. This thesis provides insights, both practically and methodologically for the above problems for using sentiment analysis along with neural networks of texts and attaining the art results of state. In the initial problem, the persuasiveness of numerous alternatives of crowdsourcing is sought for. Using social tags, two medium, emotion-labeled sets of data are generated. For the next problem, large text collections with a series of experiments of numerous domains and contents were focused on and testing embedded words of several parameters. In the later problem, max-pooling and convolution neural layers with a series of experiments were conducted. Integrating the complexity of bigrams, trigrams and words alongside region-based max-pooling layers in a pair of stacks proved to display the most fa-vorable outcome. The derived architecture performs competitively for business, movie and product reviews when sentiment polarity analysis is performed.
In the paper [6] They have used a varying order of two models, one is CNN and the other is LSTM, The orders are like Bert, CNN, Lstm or BERT, LSTM, CNN.The features vector is generated here using a pre-trained BERT model. Data must ultimately travel via a ReLU activation function after passing through several CNN and LSTM models. They pre-trained BERT since it has several advantages over Word2vec when it comes to converting text data to word embedding. As the BERT generates the features vector for this model, they would feed into the data through the BERT model first. After the output from the BERT model would be used as in the input of the hybrid models they have created. The hybrid models perform the classification. After that they combined CNN and LSTM deep learning models, as they perform really well on sentiment analysis.
In another paper [2] the authors state about sentiment analysis on bangla text.Since bangla is spoken by almost 200 million people worldwide,a large number of people are expressing their opinions and thoughts through social networking websites and microblogging. Moreover,people are demonstrating opinions and thoughts in comments on online news portals.However they also shares their opinions by doing online shopping through online marketplaces.It has become very difficult for businesses to keep track of analyzing the sentiment of comments and reviews from each individual customers.As a result, application of automated Sentiment analysis had been used for enhancing efficiency and productivity.In this paper,author collected a set of Data where there were 10,000 Bangla and Romanized Bangla text samples.Then author pre-processed the data in such a way that researchers can easily read the data.Romanized Bangla texts hold on to 28% where Bangla texts takes 72% of

whole textual data in the dataset.There are 9337 entries in total, Bangla entries is 6698 and Romanized Bangla entries is 2639.They collected 4621 data from Facebook,2610 data from Twitter,801 data from YouTube,1255 data from online news portals,50 data from product review pages.Their dataset were categories in Positive, Negative and Ambiguous.The author used Recurrent Neural Networks more specifically they used LSTM neural network. They obtained the accuracy of 78% with "Ambiguous removed" and 55% accuracy with "Ambiguous converted to 2" in this paper.

In the 1970s, psychologist Paul Eckman presented 6 basic emotions like happiness, disgust, fear, surprise, sadness and anger which claimed to be experienced universally, basically these are domain emotion or primary emotion that to be experienced universally[10]. Later he made the list bigger with emotions like pride, shame, embarrassment, and excitement. A "Wheel of emotions" he called, which can be blended into different feelings like we mixed color red, blue, yellow to create other shades. According to him, basic emotions act like building blocks. For example, basic emotions such as happiness and trust can presented as love together, astonishing and surprise can present as fear, prolonged sadness can turn into depression, etc. this gave us very insight of our dataset while we were struggling with data collection, that choosing a domain of various feeling we can find out the root of the emotion.

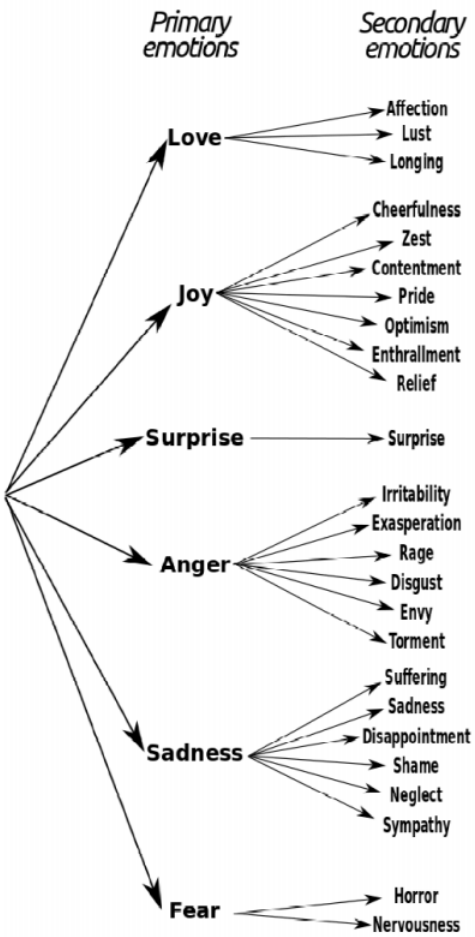

Figure 2.1: Six Types of Emotions

The systematic literature review by Dewi, Ciptayani, and Prayustika[13] delves into hybrid deep learning smart recommendation systems. They analyzed 50 articles from digital libraries and found that textual datasets were the most commonly used, followed by image datasets.These systems were often developed using a Convolutional Neural Network (CNN) and a Long Short-Term Memory (LSTM). Python was the dominant programming language, known for its extensive machine learning libraries. System performance was evaluated using evaluation criteria like accuracy, precision, recall, and F1-Score. In summary, this review provides valuable insights into current trends and practices in hybrid deep learning smart recommendation systems, empowering researchers and practitioners for future work.
Researchers from the University of Salamanca (2021) under the direction of Cach Dang[12] detailed the use of these methods in their recommendations for benchmarking systems for streaming services. In this essay, the usage of regression analysis systems in sentiment analysis is discussed, and novel hybrid deductive learning methods are suggested. The strategy takes into account both item qualities and usage-specific general requirements. Results from experimental studies indicate that the proposed application significantly improves response under system-performance. The difficulties of getting clear feedback and the application of data collection strategies are also discussed in the text. Utilizing a variety of criteria, the performance of the proposed approach is assessed and contrasted with baseline techniques. The results demonstrate the effectiveness of the approach in producing personalized recommendations. There were 889176 users in the analysis. The group propose that the proposed architecture will be tested in other application domains. Aspect sentiment analysis will be addressed using graph convolutional networks to improve user sentiment understanding. The interaction between aspect terms and semantic and syntactic information will be modeled to enhance performance.

Chapter 3

Dataset

3.1 Description of the data

In this paper we are working with long textual Bangla sentences, we will do sentiment analysis of Bangla long textual data using various deep learning model. For this initially we have gathered a dataset of bangla long textual data with 16 classes like, Like, smiley, HaHa, Sad, Skip, Love, Wow, Blush, Consciousness, Rocking, Bad, Angry, Fail, Provocating, shocking, Protesting, Evil, Skeptical from a paper published from SUST on text to speech paper[9]. Going through the data we noticed a lot of repetition of sentences presented with different emotions, which later can cause fatal errors when we will apply the models as they can get confused. Also, as we tried to give the most optimistic model for bangla language sentiment analysis we thought of increasing the classes. So we added extra 40 classes like Very Happy, Extremely Happy, Happy, Agreeable, Moderately Happy, Mixed Positive-Negative, Neutral, Moderately Sad, Sad, Very Sad, Extremely Sad, Love-Hate Mix, Love Dominant, Hate Dominant, Complete Disgust, Safe,Joyful, Surprised, Satisfied, Expected, Alongside Safe, Surprise Dominant, Alongside Happy, Satisfaction Dominant, Generous, Uncertain, Terrifying, Hopeless, Astonishing, Heartbroken, Hurt, Abused, Supportive, Doubtful about Expectations, Anxious, Surprised and Amazed, Alongside Safe with Fear, Disgusted, Not Satisfied, Suspicious. For these classes we started generating data from news websites, bangla blog, various bangla website scraping[1], we faced various problems distinguishing the similar sentiment humanly, it was very tough to do it manually as well, as we did not want any repetition this time. After that we had a dataset of 56k with 56 classes but the problem with it was we had some classes which we could not give enough data, the percentage was so low that it would create a biasedness while running the models, so we decided to eliminate some classes which were almost similar or if we count the domain of the sentiment they will be same, so we decided to remove them, after that finally we ended up with a data of 34 classes with 27 thousand long textual Bangla data. These are, like 3762, sadness 2648, happiness 2358, surprise 1778, safe 1287, mixed 1280, satisfied 1237, supportive 1044, love 990, Smiley 947, agreeable 907, anger 816, expected 784, bad 757, haha 518, hate 493, confused 488, generous 426, fear 424, skip 410, disgust 379, hurt 345, shocking 330, hopeless 316, provocative 308, fail 306, uncertain 300, heartbroken 294, protestant 285, consciousness 283, rocking 267, shyness 244, skeptical 181, unsatisfied 175.

	text	sentiment
0	রুদ্র জাহেদ ভাই খুব ভাল লাগল আপনার মন্তব্যে অন...	love
1	যারা ব্লগ দিয়ে ইন্টারনেট চালানোর দাবি করেছিল স...	like
2	থ্যাংকস সুমন ভাই	like
3	সময়ের নিষ্ঠ প্রতিবাদ	consciousness
4	সমস্যা বটে আর একটা কারন হতে পারে নতুন ব্লগ লিখ...	like
...	...	...
27362	অত্যন্ত বিদর্ধ হয়েছে মন	sadness
27363	অত্যন্ত কষ্টজনক মনে হচ্ছে	sadness
27364	বিপন্ন হৃদয়ের মতো অত্যন্ত দুখিত	sadness
27365	অত্যন্ত বিপর্যয়ে বিষাদিত মন হচ্ছে	sadness
27366	বিপন্নতায় মন অত্যন্ত বিষাদিত হয়েছে	sadness

Figure 3.1: Dataset Sample

3.2 Dataset analysis And Preprocessing

Preprocessing is a crucial phase in readying raw data for analysis or model training. It involves a series of techniques and operations to clean, transform, and organize data into a format suitable for deep learning algorithms or analysis. Here's a breakdown of the various stages encompassed in the provided preprocessing workflow figure:

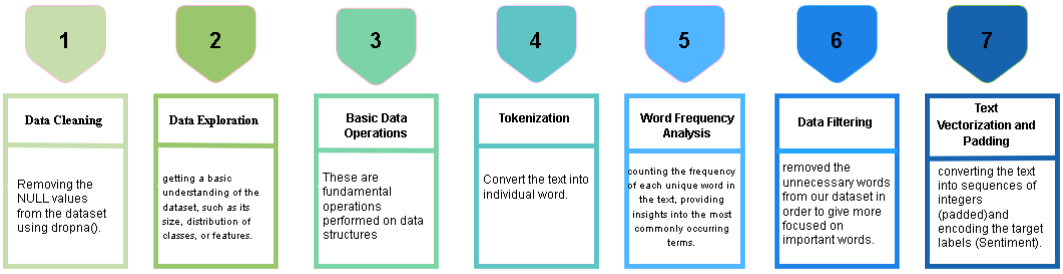

Figure 3.2: Data Pre-processing steps

3.2.1 Data Cleaning

1. **Explanation:** Data cleaning involves handling missing values, inconsistencies, or errors within the dataset to ensure its quality and integrity. Removing missing data ensures that the subsequent analysis or modeling isn't affected by incomplete records.

2. **Code Implementation:** In the code provided, data.dropna() is used to remove rows with missing values, ensuring the dataset's completeness. data.text.isnull().values.any checks if there are any null values in the 'text' column.

3.2.2 Data Exploration

1. **Explanation:** It involves getting a basic understanding of the dataset, such as its size, distribution of classes, or features. This helps in understanding the data's characteristics before proceeding with further analysis or modeling.

2. **Code Implementation:** data.shape provides the size of the dataset, and data.sentiment.value_counts() gives the count of each sentiment category in the 'sentiment'column.

3.2.3 Basic Data Operations

1. **Explanation:** These are fundamental operations performed on data structures, like dictionaries or lists, to organize, manipulate, or retrieve specific information.

2. **Code Implementation:** The code initializes a dictionary x = "a": 1 and extracts the list of keys using list(x.keys()).

3.2.4 Text Preprocessing (Tokenization)

1. **Explanation:** Tokenization is the process of splitting text into smaller units called tokens, typically words or phrases, to facilitate further analysis. It helps in converting unstructured text data into a format suitable for deep learning models.

2. **Code Implementation:** The 'text' data is tokenized by splitting it into words using value.split(" ") in a loop through the 'text' column. The tokens (words) are collected into the 'corpus' list.

3.2.5 Word Frequency Analysis

1. **Explanation:** This step involves counting the frequency of each unique word in the text corpus, providing insights into the most commonly occurring terms.

2. **Code Implementation:** Utilizing np.unique(corpus, return_counts=True), the code computes the unique words and their respective counts.

3.2.6 Data Filtering

1. **Explanation:** Filtering data based on specific criteria, in this case, retaining words with counts exceeding a threshold, to focus on the most relevant or frequent terms.

2. **Code Implementation:** The code filters words with counts above 90 and stores them in a list y.

3.2.7 Text Vectorization and Padding

1. **Explanation:** Vectorization involves converting text into numerical representations (sequences of numbers), while padding ensures uniform sequence lengths, crucial for feeding data into neural networks.

2. **Code Implementation:** The Keras Tokenizer is used to tokenize text (tokenizer.fit_on_texts(text)), generate word indices (tokenizer.word_index), convert text to sequences (tokenizer.texts_to_sequences(text)), and pad sequences (pad_sequences(sequences, padding='post')).

3.2.8 Train-Test Split

1. **Explanation:** Splitting the dataset into training and testing subsets to train a model on one set and validate it on another, ensuring the model's generalizability.

2. **Code Implementation:** Train_Test_Split is used to split the preprocessed data into training and testing sets with an 80-20 ratio.

This comprehensive process transforms raw textual data into a structured format suitable for deep learning algorithms, ensuring cleanliness, uniformity, and readiness for model training or analysis.

3.3 Data Analysis and Visualization

Analyzing the dataset is crucial as we are working with a substantial dataset for our research. We have a dataset of 37287k long Bangla textual sentences. We have 34 classes, These are, like 3762, sadness 2648, happiness 2358, surprise 1778, safe 1287, mixed 1280, satisfied 1237, supportive 1044, love 990, Smiley 947, agreeable 907, anger 816, expected 784, bad 757, haha 518, hate 493, confused 488, generous 426, fear 424, skip 410, disgust 379, hurt 345, shocking 330, hopeless 316, provocative 308, fail 306, uncertain 300, heartbroken 294, protestant 285, consciousness 283, rocking 267, shyness 244, skeptical 181, unsatisfied 175. The dataset comprises an extensive collection of 37,287k long Bangla textual sentences distributed across 34 distinct classes. These classes encompass a variety of emotions and sentiments, with varying counts in each category. The largest categories include 3762 instances of sadness, followed by 2648 instances of happiness, 2358 instances of surprise, 1778 instances of safety, and 1287 instances of mixed emotions. Some categories have relatively fewer instances, such as 181 instances of skepticism and 175 instances of dissatisfaction. This diverse range of emotions provides a comprehensive landscape for analysis and understanding.

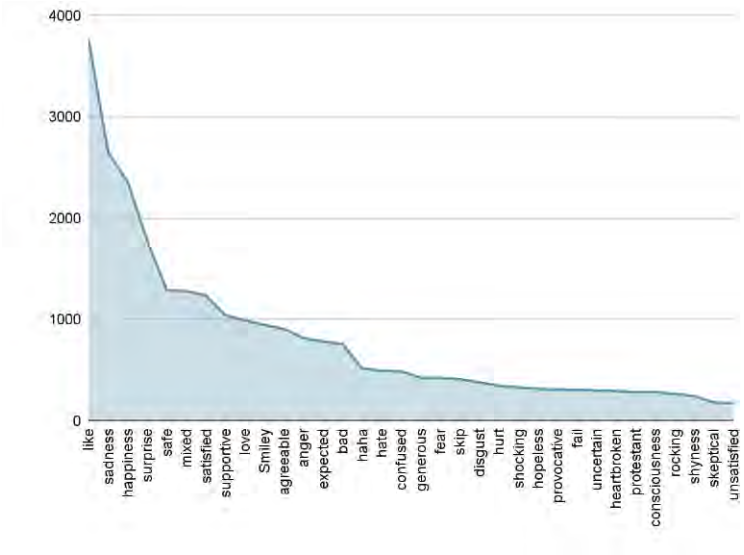

Figure 3.3: Total Data in Descending Order of Each Class

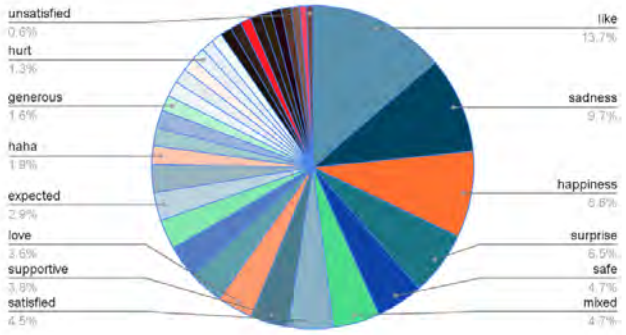

Figure 3.4: Percentage of Data in Each Class

The imbalance in the dataset can significantly impact the training of the model. When a dataset is imbalanced, meaning certain classes have significantly more samples than others, the model tends to be biased towards the majority classes. This imbalance can lead to several issues:

1. **Bias towards majority classes:**The model becomes more inclined to predict the majority classes accurately while neglecting the minority classes. This results in poorer performance on less represented classes.

2. **Reduced generalization:** Models trained on imbalanced data might not generalize well to new, unseen data, particularly for the underrepresented classes. It might fail to recognize or misclassify instances from minority classes.

3. **Misleading evaluation:** Accuracy, the most common metric, might not be a reliable measure for the model's performance, especially in highly imbalanced datasets. A model may achieve high accuracy by merely predicting the majority class.

4. **Loss of valuable information:** The model might not learn enough about the underrepresented classes due to their limited presence in the dataset, leading to a loss of valuable information.

Chapter 4

Model Description

4.1 ANN

Artificial Neural Network (ANN) can be compared to a human brain. It processes information in the similar manner as the biological neural networks in the human brain. Like the human brain ANN also has neurons. Neurons consist of a large number of interconnected processing elements.
Key elements of an ANN:

1. **Neurons:** These are the foundational parts of an ANN. They take in inputs and perform basic operations on the data

2. **Weights:**These are the connection strength parameters between neurons. These weights are given with the inputs in the learning phase to find patterns in inputs.

3. **Activation Function:** An activation function is used to determine the neuron to be activated or not. It is applied on the weighted sum of the inputs. Commonly used activation functions are tanh function and ReLU function. In case of smooth transition activation function is very important because a little change in the input creates a huge impact on the output.

4. **Learning Algorithm:** Learning algorithm works to adjust the weights during the learning phase. It is important to reduce the difference between the actual output and the predicted output.

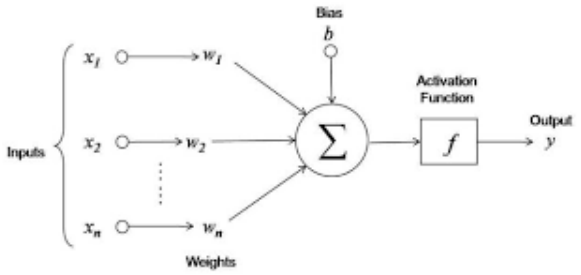

Figure 4.1: ANN Structure

For a basic model of ANN, the total input can be calculated as follows:

$$y_i n = x_1.w_1 + x_2.w_2 + x_3.w_3 x_m.w_m \tag{4.1}$$

where x_i are inputs and w_i are weights.

4.2 Convolutional Neural Network(CNN):

Convolution Neural Network(CNN) is multi-layered artificial neural networks with the ability to detect complex features in data specially extracting features in text data.It is a subset of machine learning.A Convolutional Neural Network on textual data,also known as a 1D CNN, is a specialized type of neural network architecture adapted for processing one-dimensional sequences,such as sentences or documents. While CNNs were originally designed for images but they can be applied to text by treating the input as a one-dimensional grid of word embeddings.A CNN model for text processing applies 1D convolutions over word embeddings to capture local patterns and features. One-dimensional convolution (1D convolution) works on textual data by sliding a filter over the input sequence to detect local patterns and features.

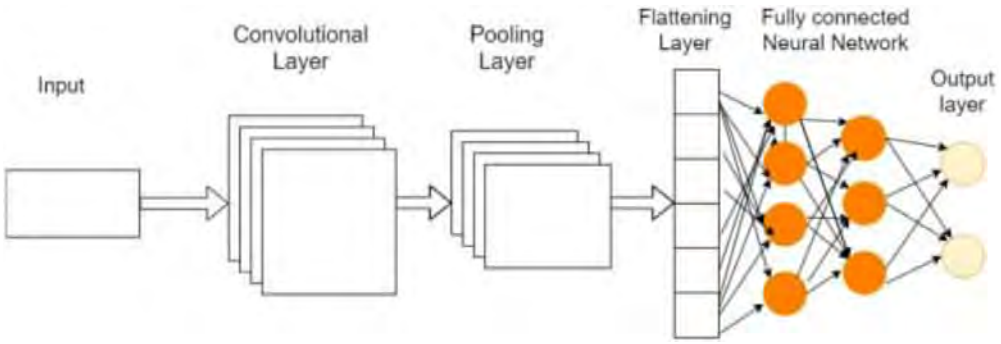

Figure 4.2: CNN Architecture

4.2.1 Input Layer

The input layer takes a sequence of word embeddings as input. The shape of the input tensor would be [sequenceLength, embeddingDimension], where sequenceLength is the length of the input text and embeddingDimension is the dimension of the word embeddings

4.2.2 Word Embeddings

The first layer of the CNN model is typically a Word Embedding layer. This layer converts each word in the input sequence into a high-dimensional vector representation. These vectors are pre-trained (Word2Vec, GloVe) or learned as part of the model training.Text data needs to be converted into numerical form before feeding it to a CNN[7].

4.2.3 Convolutional Layers

We use 1D filters that scan across the input sequence (which can be thought of as a 1D grid). These filters detect specific features or patterns in the text, like n-grams.The convolutional layers apply filters to the input sequence to detect local patterns.A feature map is the output of each convolutional layer. Output Length = (Input Length - Filter Size + 2 * Padding) / Stride + 1.

4.2.4 Feature Maps

The convolutional operation produces feature maps, which are essentially activation maps showing the presence of specific features at different positions in the input sequence. Each feature map is produced by a different filter and represents different learned patterns.

4.2.5 Activation Function

Activation functions play a crucial role in introducing non-linearity into the model. Typically, an activation function, such as ReLU, is applied after the convolution operation to introduce non-linearity. ReLU is commonly used as the activation function in both convolutional and fully connected layers.

4.2.6 Flattening and Fully Connected Layers

After several convolutional layers, the output is typically flattened and passed through one or more fully connected layers for higher-level feature learning. In the context of applying CNN models to text, the data undergoes processing through convolutional and pooling layers, resulting in a multi-dimensional tensor. Flattening is then employed to convert this tensor into a one-dimensional vector.Fully connected layers represent densely connected neural layers, where every neuron in the previous layer is linked to every neuron in the current layer. Consequently, each input from the flattened vector connects to every neuron in the fully connected layer. In the realm of textual data, post flattening the feature map, the one-dimensional array is fed into one or more fully connected layers. These layers are designed to discern complex relationships and higher-level features within the input data.The weights of the connections between neurons in the fully connected layers are learned through the training process. These weights undergo adjustment during backpropagation to minimize the loss function, enabling the network to make accurate predictions or classifications.The output from the fully connected layers is subsequently employed for the final prediction or classification task.

4.2.7 Output Layer

Depending on the task (sentiment analysis, text classification) the output layer could have one or more units with an appropriate activation function (softmax for classification).

4.2.8 Backpropagation and Training

The model undergoes training using backpropagation, wherein the weights are adjusted to minimize a loss function.

4.2.9 Regularization and Optimization

Techniques such as dropout, L2 regularization, and batch normalization can be employed to prevent overfitting. Additionally, various optimization algorithms are used to adjust the weights during training.

4.2.10 Loss Function

The model undergoes training through backpropagation, adjusting the weights to minimize a chosen loss function. The selection of the loss function depends on the specific task at hand.

4.2.11 Transfer Learning

1D CNN can leverage pre-trained models or embeddings for improved performance, especially when the available training data is limited.

However,1D CNN have proven to be highly effective for tasks involving sequential data.Their ability to automatically learn features from input sequences makes them a powerful tool in fields where understanding temporal or sequential patterns is crucial.

4.3 LSTM

The Long Short-Term Memory (LSTM) is a specialised RNN type created to solve the vanishing gradient problem and efficiently describe sequential data. Different from conventional feedforward neural networks. Unlike traditional feedforward neural networks, Tasks involving sequences, time series, and natural language processing are ideally suited for LSTM networks. It has several key elements:

4.3.1 Gates:

The input gate, forget gate, and outout gate are the three different types of gates found in LSTM cells. These gates, in turn, regulate how information enters the cell state, whether it is kept or removed from the cell state, and how information is sent to the output.
Input gate equation:

$$i_t = \sigma\ (\ \mathrm{W}_{\ i} * [h_(t-1), x_t] + b_i) \tag{4.2}$$

Denoted by i_t, has an activation function I_t that is the result of a sigmoid function σ. The formula for i_t is calculated by multiplying the weight matrix for the input gate, W_i, with the concatenation of the previous state, h_(t-1), and current input,

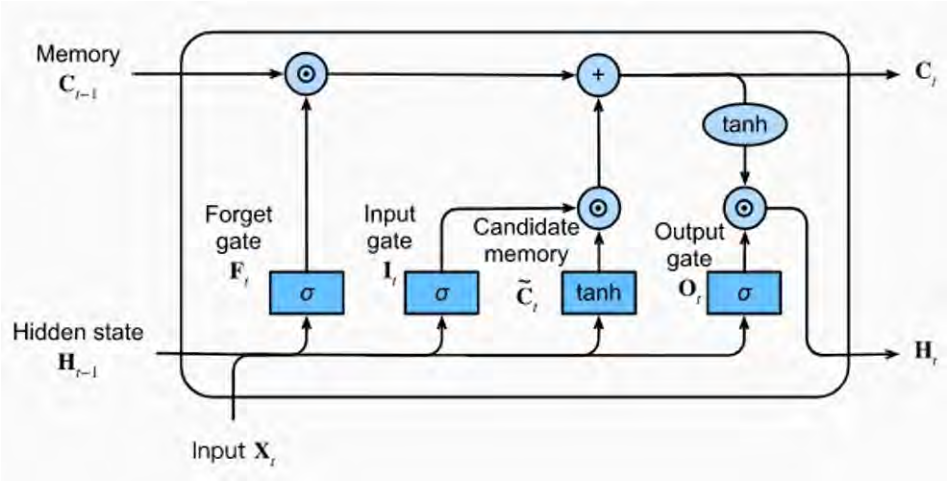

Figure 4.3: LSTM Architecture

x_t. Additional bias, b_i, is included in the equation for the input gate. The input gate controls new input (xt) for the cell state. It uses sigmoid activation (0-1) to decide how much information to incorporate. 0 means no new info, 1 means all info is retained. Forget gate equation:

$$f_t = \sigma\ (\mathrm{W}_{\ f} * [h_(t-1), x_t] + b_f) \tag{4.3}$$

Denoted by f_t, has an activation function f_t that is also the result of a sigmoid function, σ. The formula for f_t is calculated by multiplying the weight matrix for the forget gate, W_f, with the concatenation of the previous hidden state, h_(t-1), and current input, x_t. The forget gate also includes an additional bias, b_f, in its equation.
The forget gate plays a crucial role in determining which information from the previous cell state (C_(t-1)) should be let go. Employing a sigmoid function, it produces values on a scale of 0 to 1, with 0 representing the information being forgotten and 1 representing it being retained.
Output gate equation:

$$o_t = \sigma\ (\ \mathrm{W}_{\ o} * [h_(t-1), x_t] + b_o) \tag{4.4}$$

The output gate (o_ t) updates the output gate activation through the use of a sigmoid activation function σ. This is achieved by multiplying the weight matrix for the output gate (W_o) with a concatenation of the previous hidden state (h_(t-1)) and the current input (x_t), and adding a bias term (b_o).
The output gate serves as a critical component in managing the representation of updated cell state, which is then presented as the hidden state. In order to precisely regulate the flow of information, the gate applies a sigmoid function with a range from 0 to 1. A value of 0 indicates no transmission of information to the output, while a value of 1 allows for complete disclosure.

4.3.2 LSTM Cells

At the core of an LSTM network lies LSTM cells, specialized units designed to capture important long-term relationships within sequential data. These cells process

input sequences piece by piece, all while maintaining an internal state referred to as the cell state or memory cell. In order to tackle the challenge of vanishing gradients, LSTM cells are equipped with gates to control the flow of information.
Cell state equation:

$$C_tilde_t = tanh(W_c * [h_(t-1), x_t] + b_c) \tag{4.5}$$

The cell state, denoted as 'C' and responsible for storing and carrying information across time steps, can be updated through the use of different gates within the LSTM cell. One such gate is the candidate cell state (Cn_t), which is updated through the use of a hyperbolic tangent activation function (tanh). This is achieved by multiplying the weight matrix for the candidate cell state (W_c) with a concatenation of the previous hidden state (h_(t-1)) and the current input (x_t), and adding a bias term (b_c). The candidate cell state (Cn_t) captures new information for the cell state. The hyperbolic tangent activation function limits values from -1 to 1, making it ideal for modeling potential new cell state values. Cell State Update (C_t):

$$C_t = f_t * C_(t-1) + i_t * Cn_t \tag{4.6}$$

When updating the cell state (C_t), we use a combination of the previous state (C_(t-1)) and a candidate state (Cn_t). This is achieved through the forget activation (f_t) and input gate activation (i_t), determining which elements should be kept from the previous state and which new information should be integrated. Essentially, the equation acts as a decision maker, determining the content of the updated cell state. Hidden State (h_t): The hidden state, often denoted as 'h,' is the output of the LSTEM cell at a particular time step. It contains information that the LSTM cell has deemed relevant for making predictions or encoding the input sequence.

$$h_t = o_t * tanh(C_t) \tag{4.7}$$

The hidden state H_t is modified by multiplying it with the output gate activation o_t and running it through the hyperbolic tangent function, denoted by tanh. This results in an updated hidden state that is affected by the updated cell state C_t.
LSTM Weight Parameters: Like traditional neural networks, LSTM models have weight parameters. These weights are learned during training and decide the impact of input data on the cell state and hidden state. Activation Functions:Gates, cell state, and hidden state updates all make use of sigmoid and hyperbolic tangent (tanh) functions. They bring non-linearity into the equation, allowing the model to recognise intricate patterns in sequential data. Learning Process: LSTM models are trained using optimization algorithms like using gradient descent to reduce the gap between expected and actual results.Calculations for updating cell state, hidden state, and gates involve intricate math, considering input data, previous cell state, and weight parameters. These operations are performed at each time step for sequential data processing.

4.4 Bi-LSTM

Bidirectional recurrent neural networks (RNNs) can be thought of as combining two separate RNNs. This unique structure allows for the networks to gather informa-

tion from both forward and backward sequence at each time step, enhancing their understanding of the data.

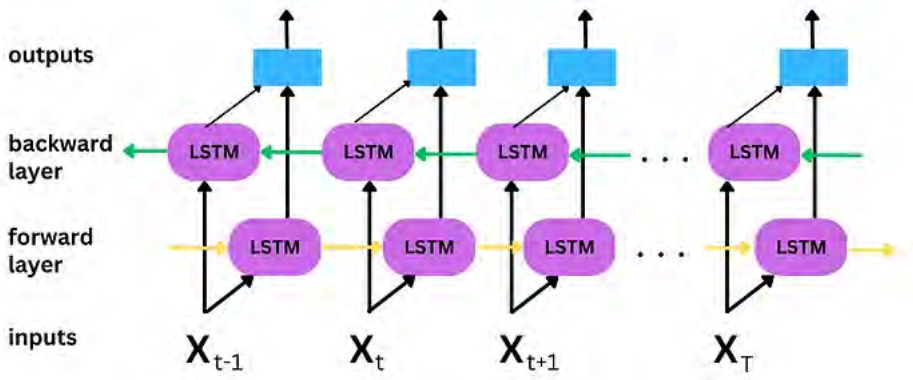

Figure 4.4: Bi-LSTM Architecture

The main functionality of BiLSTM is the additional LSTM layer that works in the reversed direction, allowing for easier backtracking and integration of other algorithms for alternative solutions. It is increasingly popular for solving real-world problems due to its ability to be authenticated and integrated within a shorter time frame, reducing the overall complexity of a machine learning model.

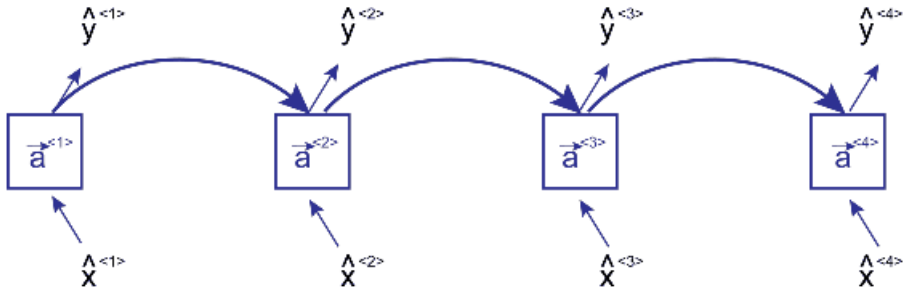

Figure 4.5: Forward RNN(LSTM or GRU) Network

BiLSTM also considers past inputs for classification and analysis of problems. In contrast, standard or unidirectional LSTM only analyzes present inputs, while BiLSTM improves predictions over time as the model is trained with pre-existing values and new inputs provide a better stage for validation according to the model's standards. This results in a more meaningful output for users.

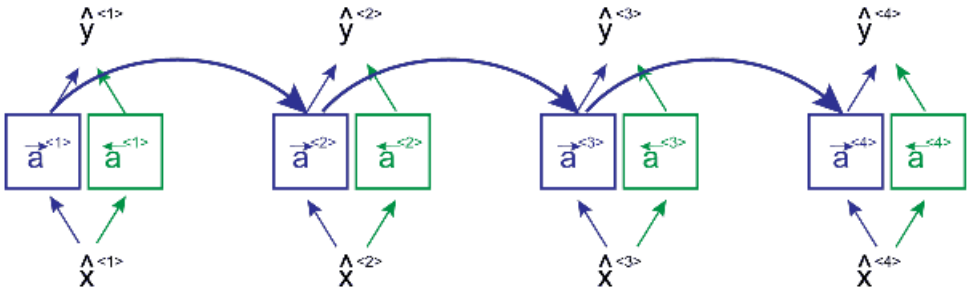

Figure 4.6: Adding Another Cell for Other Direction

BiLSTM also considers past inputs for classification and analysis of problems. In contrast, standard or unidirectional LSTM only analyzes present inputs, while BiLSTM improves predictions over time as the model is trained with pre-existing values and new inputs provide a better stage for validation according to the model's standards. This results in a more meaningful output for users.

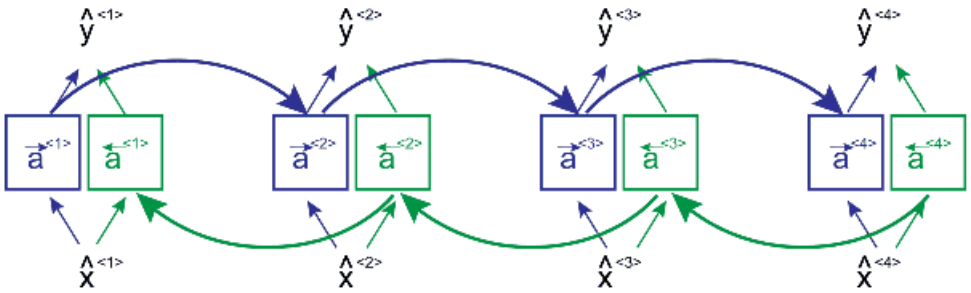

Figure 4.7: Connecting the Backward Cells

Two forward propagations are executed, one for the forward cells and one for the backward cells. This means that both activations, for the forward and backward cells, are taken into consideration when calculating the output $y^{\wedge}$ at time t.

$$\hat{y}^{<t>} = g(W_y [\overrightarrow{a}^{<t>}, \overleftarrow{a}^{<t>}] + b_y)$$

Figure 4.8: Bi-LSTM Backward Concatenation Equation.

Learning Process: Utilizing the bidirectional approach will allow your inputs to be processed in two directions: from past to future and vice versa. What sets this method apart from unidirectional is that, by incorporating a backward-running LSTM, information from the future can be retained. When the two hidden states are merged, information from both past and future can be retained at any given time. Determining their specific strengths is a complex matter, but it is worth noting that BiLSTMs demonstrate impressive performance as they have a heightened ability to grasp the context.

4.5 BERT

BERT (Bidirectional Encoder Representations from Transformers) model has gained significant acclaim for its exceptional capabilities in natural language processing (NLP). Developed by Google, this pre-trained NLP model is built on the transformer architecture, allowing it to effectively understand the contextual meaning of words within a sentence by incorporating both left and right context. BERT is considered a transformer-based model, which leverages self-attention mechanisms and is a type of neural network architecture.BERT was thoughtfully crafted for the sole purpose of tackling natural language processing (NLP) tasks with prowess. It achieves this through its exceptional ability to capture both forward and backward context, resulting in contextualized embeddings for words within sentences. By combining these contextual embeddings with powerful transfer learning capabilities, BERT excels in hybrid recommendation and sentiment analysis on textual data. This is thanks to its advanced understanding of context, allowing it to effectively comprehend textual data and produce superior results in both recommendation and sentiment analysis. BERT's foundation is built upon the cutting-edge Transformer architecture, a deep learning model where each output element is intricately linked to every input element. These links are dynamically determined based on their connections, strengthening BERT's capacity to accurately process and analyze text.

4.5.1 Input Layer

The input layer, complete with token embeddings, positional encodings, and segment embeddings, is passed through the transformer architecture of the BERT model. Comprised of multiple layers, the model utilizes attention mechanisms to capture the contextual relationships and bi-directional dependencies between words as it processes the input in parallel.

4.5.2 Pre-Training

During pre-training, BERT is exposed to vast amounts of text data and achieves fluency through practicing with a technique called the masked language model (MLM). This method involves randomly masking certain words in a sentence and challenging the model to accurately fill in the blanks based on the context provided by the words around them.

4.5.3 Bidirectional Context

BERT differs from traditional language models as it takes into consideration context from both directions, allowing for a more thorough understanding of the intricate relationships between words and their surroundings.

4.5.4 Tokenization

Tokenization in the BERT model involves breaking down input text into smaller units, typically subwords or word pieces, to create a vocabulary that the model can process. WordPiece Tokenization:BERT uses a tokenization method called WordPiece, where words are broken down into smaller subword units. This allows BERT

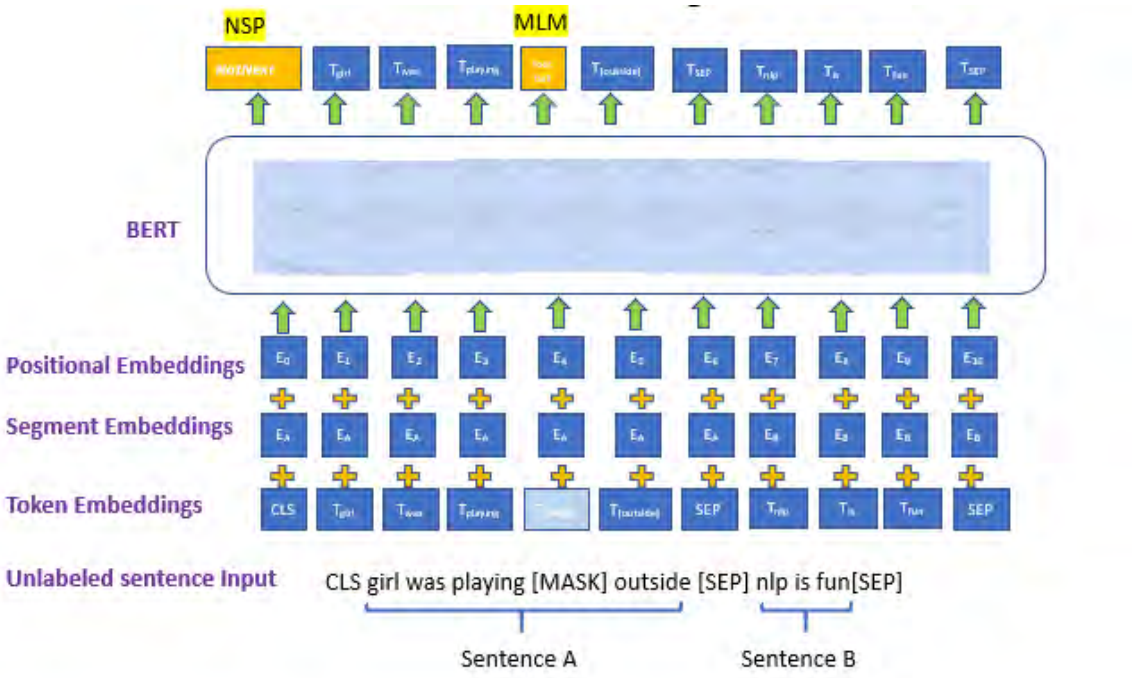

Figure 4.9: Pre-Training(BERT)

to handle a vast and diverse vocabulary efficiently. For example, the word "running" might be tokenized into ["run", "##ning"].

1. **Vocabulary Creation:** BERT has a pre-defined vocabulary that includes both complete words and subword units. The vocabulary is created during pre-training on a large corpus of text. Subword units help capture morphological variations and handle out-of-vocabulary words.

2. **Token IDs:** Each token in the input text is mapped to a unique token ID based on the BERT vocabulary. These token IDs serve as input to the model.

3. **Special Tokens:**Special tokens are added to the tokenized input to convey additional information to the model:

   (a) **[CLS] (Classification):** Added at the beginning of the input to represent the start of the sequence. It is used when the model is tasked with sequence classification.

   (b) **[SEP] (Separator):** Added between pairs of sentences or at the end of a single sentence. It indicates the separation between different segments of text.

4. **Segment IDs (for Sentence Pairs):** If the input consists of pairs of sentences, segment IDs are assigned to each token to indicate to which sentence it belongs. This helps the model distinguish between different segments of text.

5. **Masking for MLM (Masked Language Model):**During pre-training, certain tokens in the input sequence are randomly chosen and masked. The model is subsequently trained to predict the masked tokens based on the context provided by the surrounding tokens. This process encourages the model to comprehend bidirectional context.

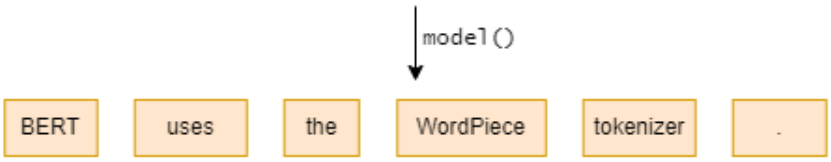

Figure 4.10: BERT Using WordPiece Tokenizer

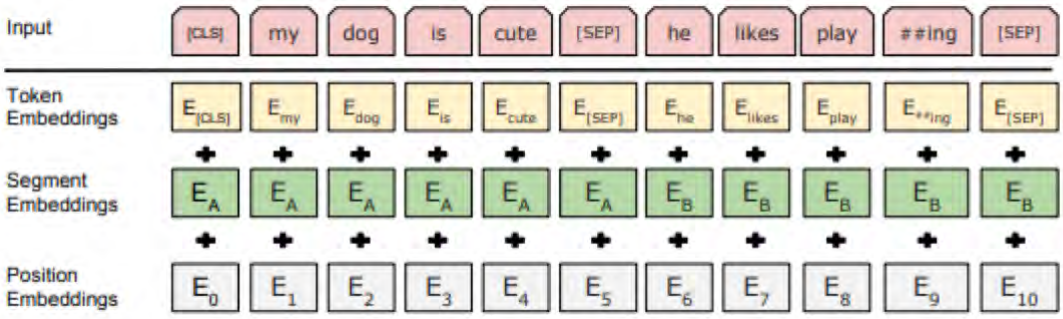

Figure 4.11: BERT Input Representation

4.5.5 Transformer Architecture

BERT uses a transformer architecture, which allows it to efficiently process and analyze sequences of tokens in parallel. The transformer's attention mechanism enables the model to focus on different parts of the input sequence, capturing dependencies between words.

4.5.6 Attention Mechanism

BERT utilizes self-attention mechanisms to assess the importance of different words in a sequence when processing each token. This attention mechanism aids the model in comprehending the relationships between words in a sentence.

4.5.7 Layers and Stacking

BERT consists of multiple layers and each layer refines the representation of the input text. The model's depth allows it to capture complex hierarchical patterns in language.

4.5.8 Fine-Tuning for Specific Tasks

Once the pre-training phase is complete, BERT can be fine-tuned for various downstream tasks, such as text classification, named entity recognition, and question answering. This involves adding task-specific layers and training the model on a smaller dataset tailored to the specific task.

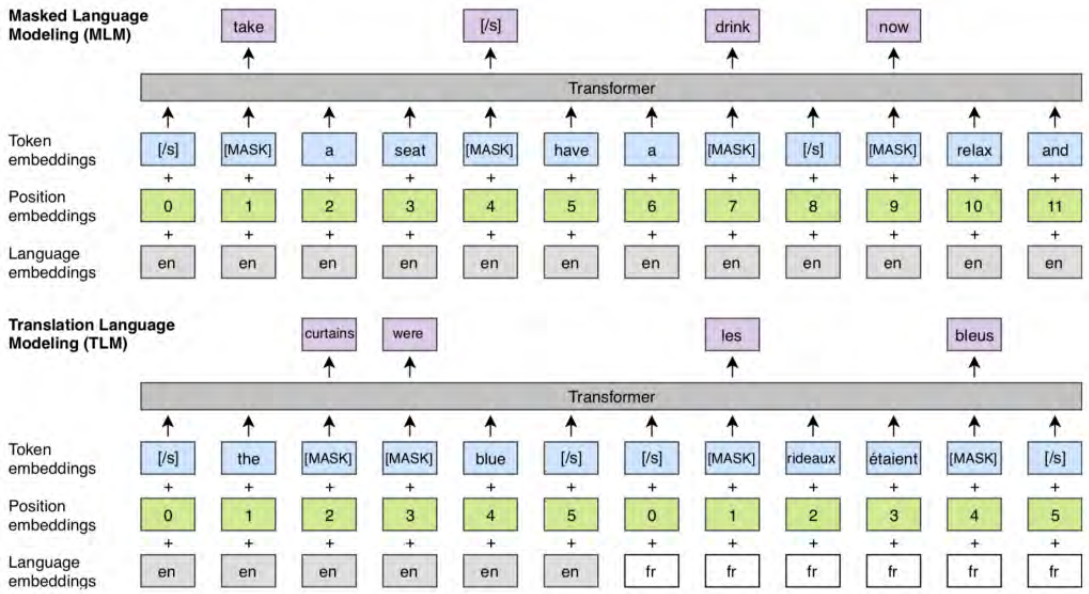

Figure 4.12: MLM and TLM

4.5.9 Contextualized Embeddings

BERT generates contextualized embeddings for each token in a sequence, capturing the nuances of meaning based on its context within the entire sentence.

4.5.10 Pooling Operation

To obtain a fixed-size representation for the entire sequence, a pooling operation is applied to the contextualized embeddings. Common pooling methods include max pooling, average pooling, or a combination of both. For instance, max pooling involves selecting the maximum value along each dimension across all tokens.

4.5.11 Fixed-Size Representation

The outcome of the pooling operation is a fixed-size representation that effectively summarizes the information from the entire sequence. This representation is designed to capture the most salient features of the input for downstream tasks.

4.5.12 Task-Specific Prediction Layers

The fixed-size representation obtained from the pooling operation is then fed into task-specific prediction layers. These layers are typically composed of fully connected (dense) layers that transform the representation to make predictions for the specific task.

4.5.13 Transfer Learning

BERT is pre-trained on a large corpus, allowing it to capture general language patterns. Fine-tuning tailors the model to specific recommendation or sentiment analysis tasks.

4.5.14 Versatility

BERT's architecture is versatile and it can be fine-tuned for various NLP applications, including text classification, named entity recognition, sentiment analysis, question answering, and more. This adaptability makes it a go-to choice for diverse language understanding tasks.

4.5.15 Multilingual Support

BERT models exist for multiple languages, making it applicable to a diverse range of linguistic contexts.

4.5.16 Community and Adoption

BERT has gained widespread adoption in both academia and industry. Its success has spurred research and developments in the field of pre-trained language representations, leading to the creation of various BERT-based models.

BERT has the ability to capture bidirectional context and contextualized word embeddings has led to its widespread adoption and success in various natural language processing tasks.BERT excels in hybrid recommendation and sentiment analysis for textual data through its utilization of contextualized embeddings and transfer learning. Its capability to capture context-rich embeddings significantly enhances the model's understanding of textual data, leading to improved results in both recommendations and sentiment analysis.While BERT has demonstrated remarkable performance, it is worth noting that the model's large size and computational requirements can be challenging for deployment in resource-constrained environments. As a result, there have been efforts to create smaller, more efficient variants of BERT while maintaining its effectiveness for certain applications.

4.6 RCNN

The RCNN (Recurrent Convolutional Neural Network) is a hybrid deep learning architecture that merges the capabilities of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). It's primarily designed to handle sequential data effectively while leveraging the spatial understanding of CNNs.The model starts with a convolutional layer, commonly used in CNNs, to extract spatial features from the input data. These layers can capture patterns, edges, and spatial relationships within the data.Unlike standard CNNs, RCNNs incorporate RNN components like LSTMs (Long Short-Term Memory) or GRUs (Gated Recurrent Units) to process sequential data. These RNN layers help capture temporal dependencies and patterns within sequences. One of the unique aspects of the RCNN architecture is its ability to merge spatial features from CNNs and sequential understanding from RNNs.

This integration enables the model to learn from both local features and long-term dependencies in the data. By combining the strengths of CNNs in understanding spatial features and RNNs in understanding sequential data, RCNNs become adept at tasks that require understanding both spatial and temporal contexts, such as object recognition in videos, sentiment analysis in texts, and action recognition in sequences. RCNNs are particularly effective in Natural Language Processing tasks where understanding context and sequential patterns in text data is crucial. They're used in sentiment analysis, named entity recognition, text classification, and machine translation. The advantage of RCNN lies in its ability to effectively capture spatial features through the CNN layers while also preserving sequential dependencies using LSTM units. By combining these capabilities, RCNN models are particularly suitable for tasks involving sequential data, where understanding both local features and long-term dependencies in the input text is crucial for accurate predictions or classifications.

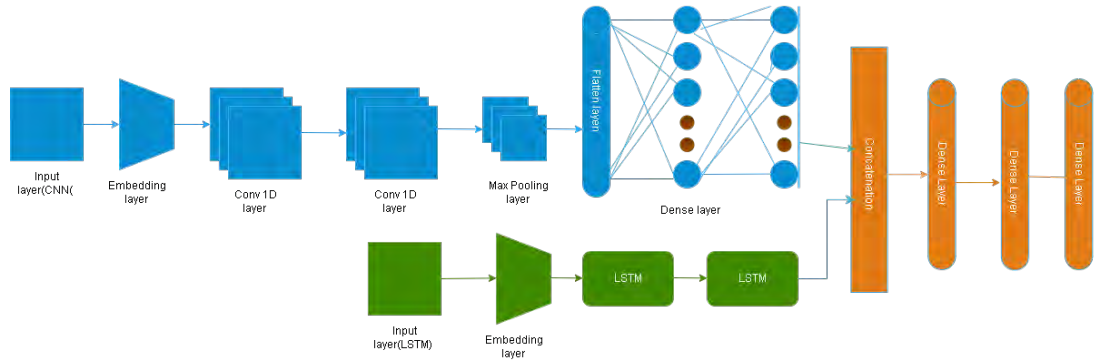

Figure 4.13: Hybrid model RCNN

Input Layers: Two separate input layers (input_cnn and input_lstm) are defined for the CNN and LSTM branches, respectively.

4.6.1 Convolutional Layers (CNN)

1. Feature Extraction:The initial part of the RCNN model comprises Convolutional Neural Network (CNN) layers, which are excellent at capturing spatial information in data.

2. Embedding: The input text data is transformed into fixed-sized dense vectors using an Embedding layer.

3. Convolution and Pooling: Sequential Conv1D layers with ReLU activation apply filters across the embedded sequences to extract important features. MaxPooling1D layers reduce the dimensionality, retaining the most relevant features.

4.6.2 Recurrent Layers (LSTM)

1. Sequential Learning: The RCNN incorporates Recurrent Neural Network (RNN) layers, specifically LSTM units, to process sequential information present in the data.

2. Embedding: Similar to the CNN branch, the input data is embedded for the LSTM branch.

3. Sequential Learning: LSTM layers capture the long-range dependencies and sequential patterns within the data. The first LSTM layer returns sequences, while the second LSTM layer captures overall patterns.

4.6.3 Concatenation

Combining Spatial and Sequential Information: The outputs from the CNN (after pooling) and LSTM branches are concatenated. This step merges the spatially extracted features (from CNN) with the learned sequential patterns (from LSTM).

4.6.4 Dense Layers

Further Feature Learning: Additional Dense layers are added after concatenation to learn complex representations from the combined features.

4.6.5 Output Layer

Classification: The final Dense layer with a softmax activation function generates the output probabilities for the different classes in the dataset.

4.6.6 Advantages

1. Long-term Dependencies: RCNNs address the vanishing gradient problem better than traditional RNNs, enabling them to capture long-term dependencies effectively.

2. Feature Learning: The model learns hierarchical features from both spatial and sequential domains, improving the representation of complex patterns in the data.

4.6.7 Limitations

Complexity: Combining CNNs and RNNs can make the model complex and computationally intensive, which might lead to longer training times and increased resource requirements.

Chapter 5

Result Analysis and Discussion

Precision, F1, and Recall:
Outputs of the models can be displayed in a variety of ways. Among them classification model's F1 score, precision, and recall are crucial. They help in assessing the model's performance in particular scenarios and offer insightful information on a variety of aspects of the model's success. Incase of finding precision, recall and F1 score we need to know TP,TN,FP and FN.

True Positives (TP): Number of occurrences which are correctly classified as positive. These circumstances include instances where the model correctly categorizes outcomes as positive.

True Negatives (TN): Number of occurrences which are correctly classified as negative. These circumstances include instances where the model correctly categorizes outcomes as negative.

False Positives (FP): Instances that the model incorrectly interprets as positive.

False Negatives (FN): Instances that the model incorrectly interprets as positive.

Precision: A model's ability to identify the positive occurrences is measured by precision. To determine the precision we take True Positives and divide it by True Positives and False Positives combined. Higher precision means our model's possibility of categorizing negative occurrences as positive is very low. Precision formula:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{5.1}$$

Recall: Through recall rate we can determine how well our model has been trained

to detect the thing that we are actually looking for. To measure recall we take True Positives and divide it by True Positives and False Negatives combined. Higher recall rate means our model is very accurate to identify our goal. Recall formula:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{5.2}$$

F1 Score: Comparable to a grade, the F1 Score indicates the effectiveness of a machine learning software. To determine the F1 score we have to take both precision and recall in account. Models with high precision and good recall get recognized by the F1 score, which does not prefer one over another.Therefore, a high F1 Score indicates that a software is performing well in terms of both accuracy and object detection. F1 score formula:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.3}$$

5.1 ANN Result

In the embedded layer we have converted integer-encoded words into 16 sized dense vectors. We set 500 as the size of the vocabulary so the words that are converted into dense vectors will be represented as integers 1 to 499. Then we used an average pooling layer to reduce the model complexity and to reduce the spatial complexity. After that three fully connected dense layers have been used. There are 32, 64 and 34 neurons respectively where for the first two layers we used ReLU function and for the last layer we used softmax activation because of its ability to handle multi-class classification. In the model architecture an embedding dimension of 380 is used in the first layer. The embedding layer's weights are determined by the pre-trained embedding matrix. The largest input sequences have a maximum length of 380 tokens. To train the model we took 100 epochs and set the batch size to 64. We kept 20% of the data to validate the performance. In order to enhance the model's predictive performance on the training set of data, the weights are updated during the training phase using the computed loss and gradients. After performing model evaluation we get the test data accuracy and the loss of our model.

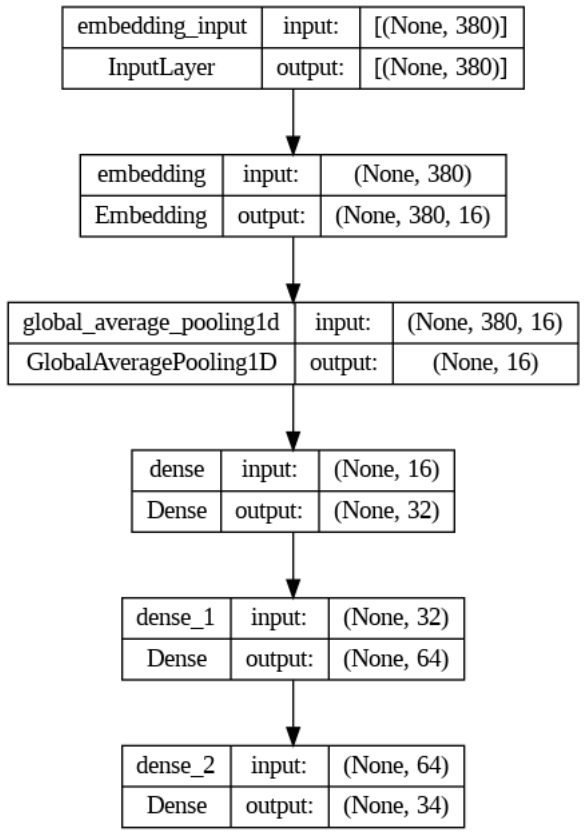

Figure 5.1: ANN Model Layer Architecture

Test Accuracy	0.6036
Test Loss	1.4842
Precision	0.4562
Recall	0.4246
F1 Score	0.4939

Table 5.1: ANN classification accuracy table

After testing the model we get an accuracy of 0.6036. With an accuracy of 0.6036, the model was able to correctly predict around 60.36% of all observations.However, we also have other parameters to determine the accuracy of our selected model. We got our precision 0.4562 which indicates that when the model predicts a class with an accuracy of roughly 45.62% for that particular class. Our ANN model achieved a recall score of 0.4246, indicating that it correctly identified around 42.46% of all actual positive instances. Lastly, our F1 Score for this model is 0.4939 which suggests the overall balance between precision and recall for this model.

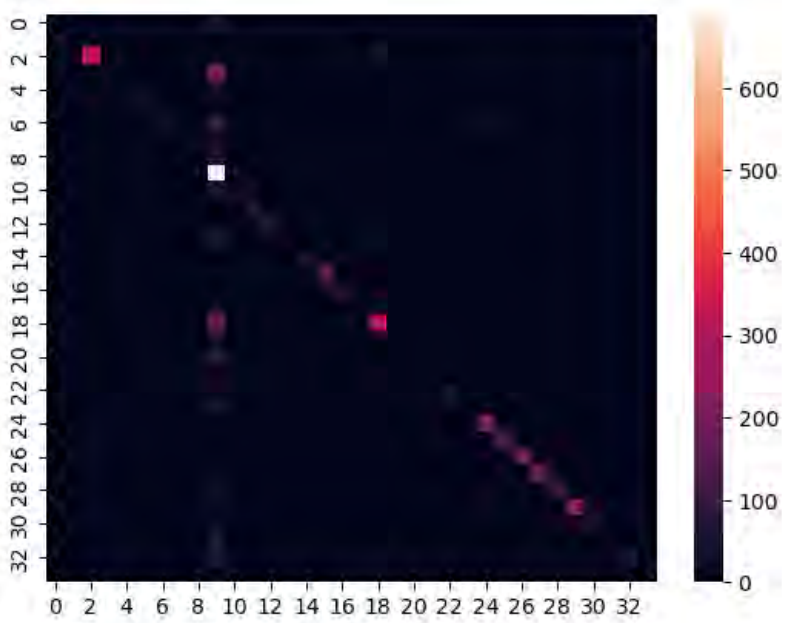

Figure 5.2: ANN Confusion Matrix

In this heatmap, in the x-axis we have the predicted values, and in the y axis we have our test data. We can see in our 2nd class(Sadness) we made about 380 right predictions. In our 18th(generous) class we have 350 right predictions, also in 29th(Protetstant)class we have 250 right predictions. The diagonal of the heatmap shows the true positive result and as 15/16 of the classes are showing the true positive result. So, we can assume the model diagonal shows the accuracy of 57%. Also, we have a highest right prediction in 9th class(Love). It raises some doubts as in the 9th class(Love) we are also seeing a straight line. This is occurring because 9th class(Love) is falsely predicted as various emotions as an anomaly.

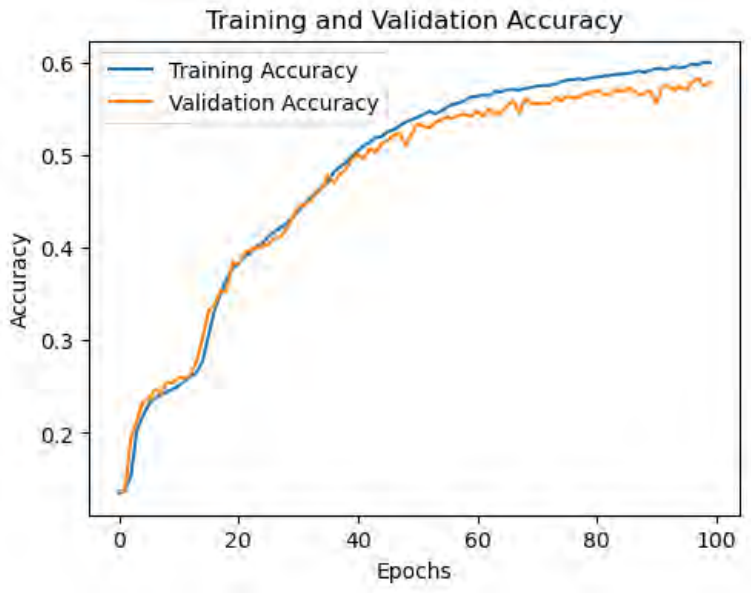

Figure 5.3: Training and Validation Accuracy for ANN

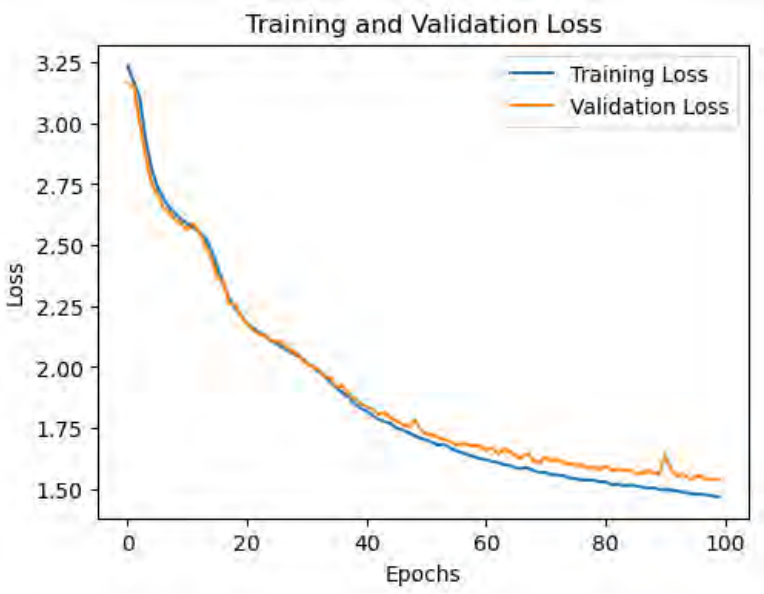

Figure 5.4: Training and Validation Loss for ANN

5.2 CNN result

In text data, CNNs process word embeddings as a 2D matrix, treating word sequences like image rows. Convolutional layers slide filters over these matrices, capturing local patterns. Pooling layers downsample the output, retaining essential information. Flattening and dense layers transform the representation for classification tasks.CNNs are primarily designed for image-related tasks. However, they can be used for sequence data as well. The accuracy we have achieved might be due to factors such as network architecture, hyperparameters or the nature of our dataset.If our dataset contains images or data with a spatial structure, CNNs are a better choice than ANNs.However,we want to improve performance,so we might need to fine-tune the CNN architecture and hyperparameters to better suit your specific data. Accuracy is slightly below ANN.

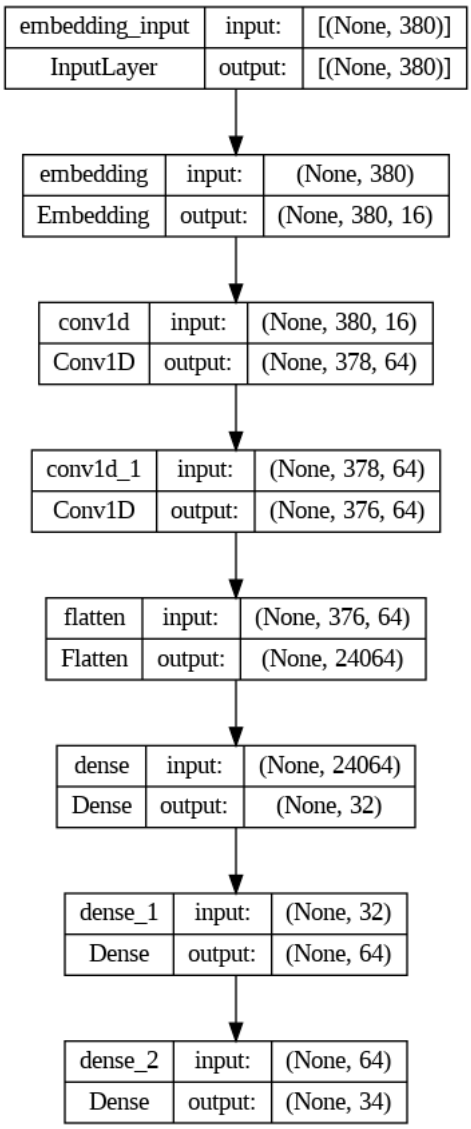

Figure 5.5: CNN Model Layer Architecture

CNNs are designed for images but can be adapted for sequences. The accuracy may depend on factors like architecture and hyperparameters. CNNs perform well when spatial features are important.In CNN,we have the accuracy of 59.68%.Accuracy alone may not offer a comprehensive understanding, particularly when dealing with an imbalanced dataset.

Test Accuracy	0.5968
Test Loss	2.2018
Precision	0.505164
Recall	0.4761695
F1 Score	0.48567

Table 5.2: CNN classification accuracy table

Our precision is 50.516%.Precision measures the accuracy of positive predictions made by the model. It is defined as the ratio of true positive predictions to the sum of true positives and false positives. Our recall is 47.617%. Recall is known as

sensitivity or the true positive rate, gauges the model's ability to correctly identify all relevant instances of a particular class. It is defined as the ratio of true positives to the sum of true positives and false negatives. Our F1 score is 48.567%.The F1 score is a metric that combines precision and recall into a single value, offering a balanced measure of a model's performance. In the context of Convolutional Neural Networks (CNNs) applied to textual data, the F1 score is particularly useful when seeking a balance between precision and recall.

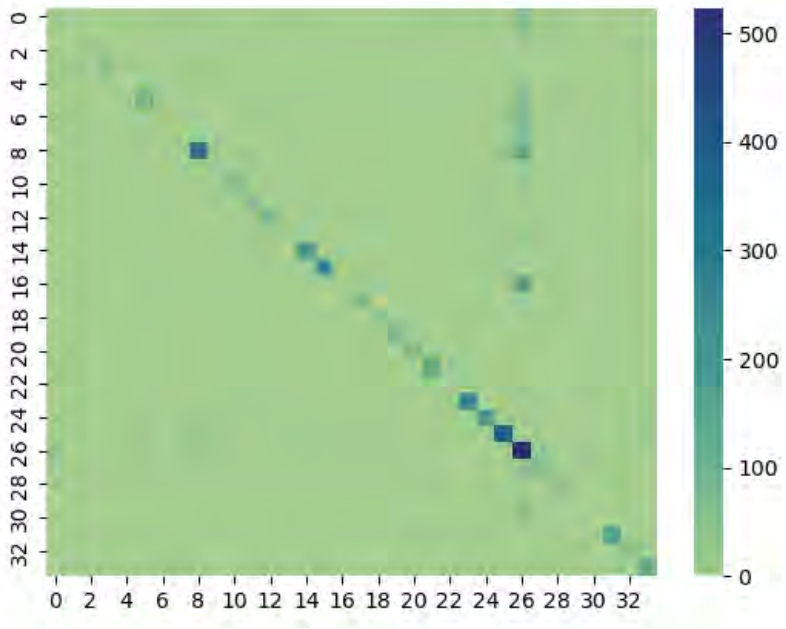

Figure 5.6: CNN Confusion Matrix

In this heatmap, in the x-axis we have the predicted values, and in the y axis we have our test data. As we can see, almost 11/12 classes have right predictions in the range from 200-400. In the 25th class(hopeless) we have almost 400+ right predictions. Also, the 8th class, 14th class, 15th class and 23th class have shown a good number of right predictions. Although, 26th class predicted 100% correctly, but, it also falsely predicted various emotions as an anomaly, so we are getting a straight line in the 26th class(fail). This confusion matrix shows the 59% accuracy through the diagonal.

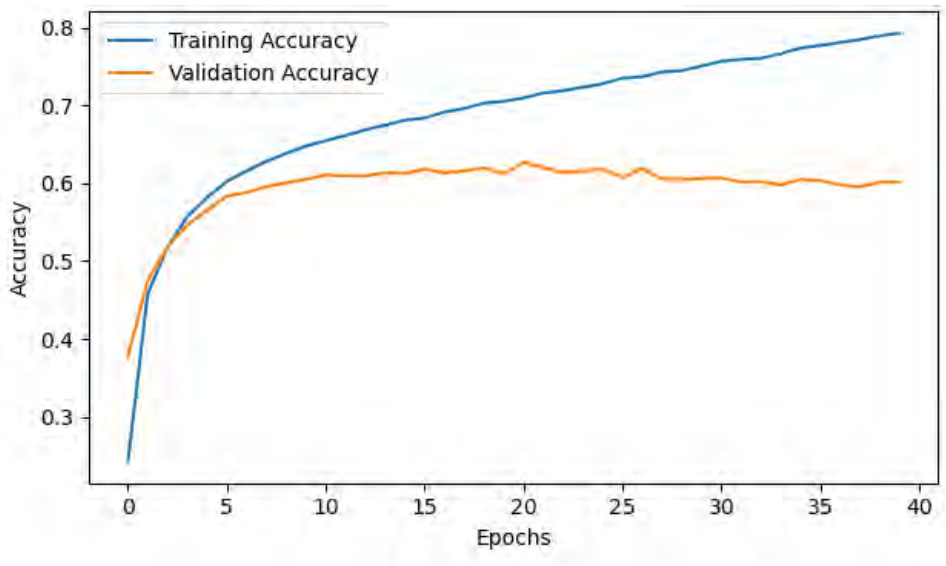

Figure 5.7: Training and Validation Accuracy for CNN

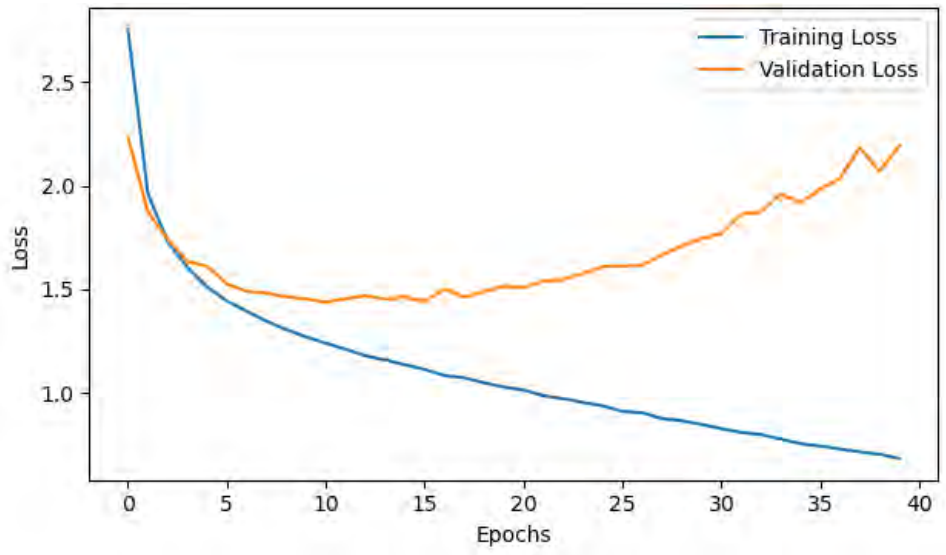

Figure 5.8: Training and Validation Loss for CNN

5.3 LSTM Result

LSTMs (Long Short-Term Memory) are recurrent neural networks designed to capture long-range dependencies in sequential data, making them well-suited for sentiment analysis tasks.They excel at understanding context and the relationships between words within a text, which is crucial for accurately classifying sentiment.

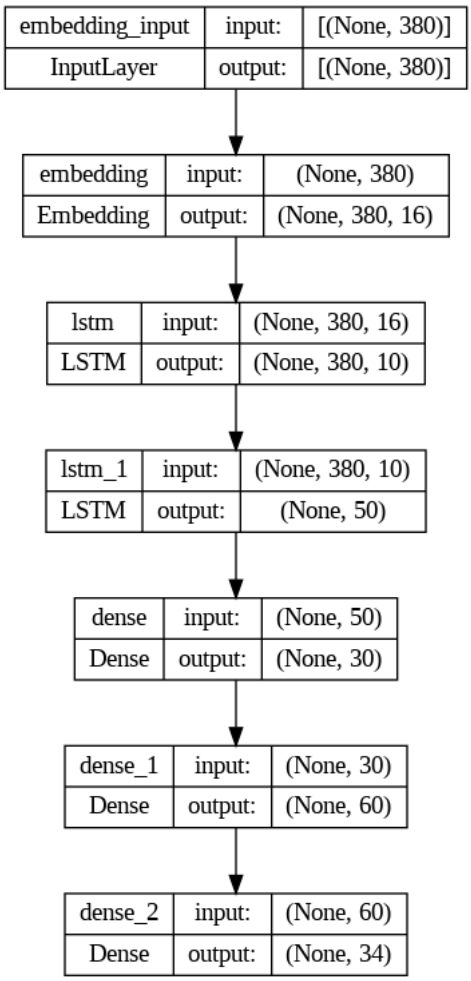

Figure 5.9: LSTM Model Layer Architecture

The model's sentiment prediction accuracy is 13.79%, highlighting the need for further evaluation, particularly for imbalanced datasets. During testing, a loss of 3.1801 was recorded, indicating the model's deviation from actual values. A lower loss value is desired. The precision score of 0.405% reflects the model's ability to accurately predict positive sentiments, which is currently low. This could be due to a high number of false positives in the model's predictions.

Test Accuracy	0.1379
Test Loss	3.1801
Precision	0.00405
Recall	0.02941
F1 Score	0.00713

Table 5.3: LSTM classification accuracy table

Let's talk about Recall (2.941%), an important metric in evaluating a model's ability to accurately identify relevant sentiments. Also known as sensitivity or true positive rate, this measure signifies the model's capability to correctly identify all relevant instances of a specific sentiment class. The low recall rate highlights a significant number of false negatives in the predictions, indicating room for improvement.

Along with Recall, the F-1 score (0.713%) also plays a crucial role in assessing a model's performance. It combines both precision and recall to provide a balanced measure of its overall performance. In this case, the low F-1 score suggests a need for improvement and draws attention to the model's potential to do better.

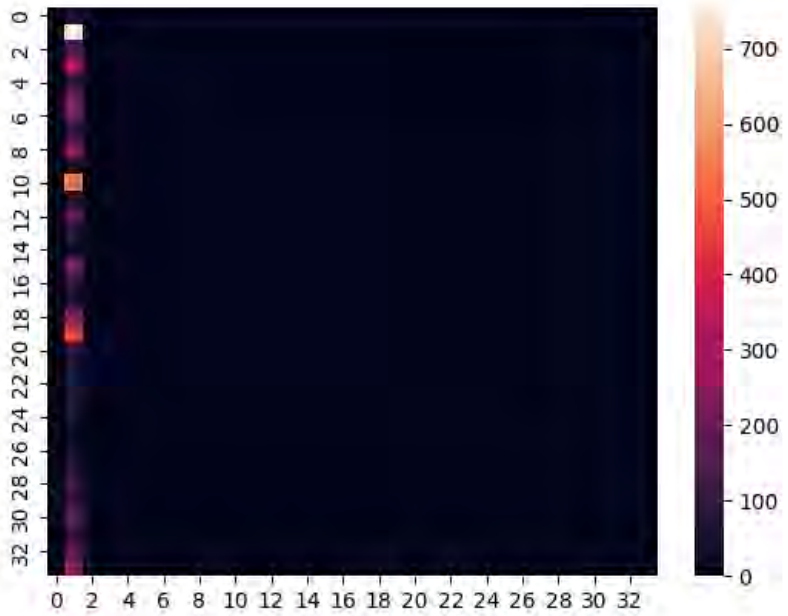

Figure 5.10: LSTM Confusion Matrix

Both matrices, with 34 rows and 34 columns each, correspond to the number of sentiment classes. Upon examining them, we have observed a noticeable concentration of accurate classifications along the diagonals. However, there is also a significant number of off-diagonal elements, indicative of misclassifications. This suggests that the performance of the model falls short of our expectations. Without sufficient data, it is challenging to identify specific patterns. However, we have noticed a higher misclassification rate for certain classes, such as "sadness," "angry," "happy," and "surprise." These classes seem to be frequently mistaken for each other. Overall, based on the confusion matrices, it is clear that the LSTM model requires improvement.

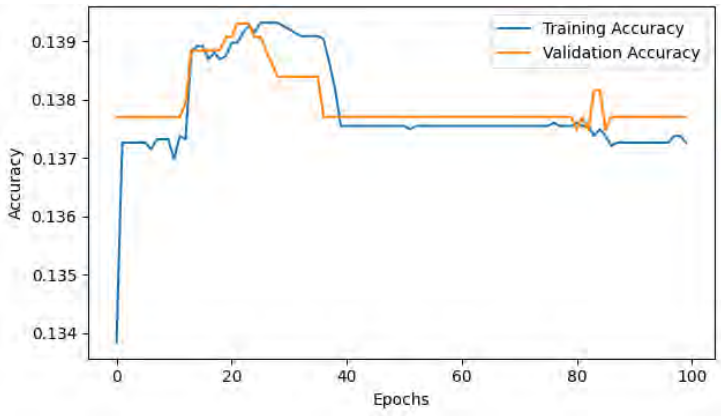

Figure 5.11: Training and Validation Accuracy for LSTM

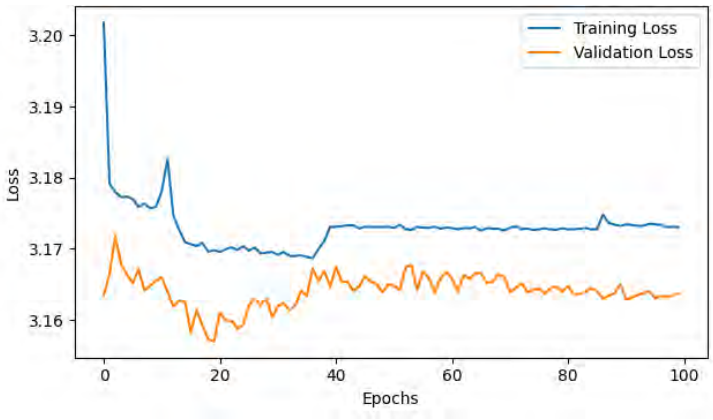

Figure 5.12: Training and Validation Loss for LSTM

5.4 Bi-LSTM Result

With its strong ability to capture long-range dependencies and context in text sequences, Bi-LSTM networks prove to be an ideal choice for sentiment analysis tasks. By processing text bidirectionally, these networks can effectively leverage both past and future information, resulting in more accurate predictions.

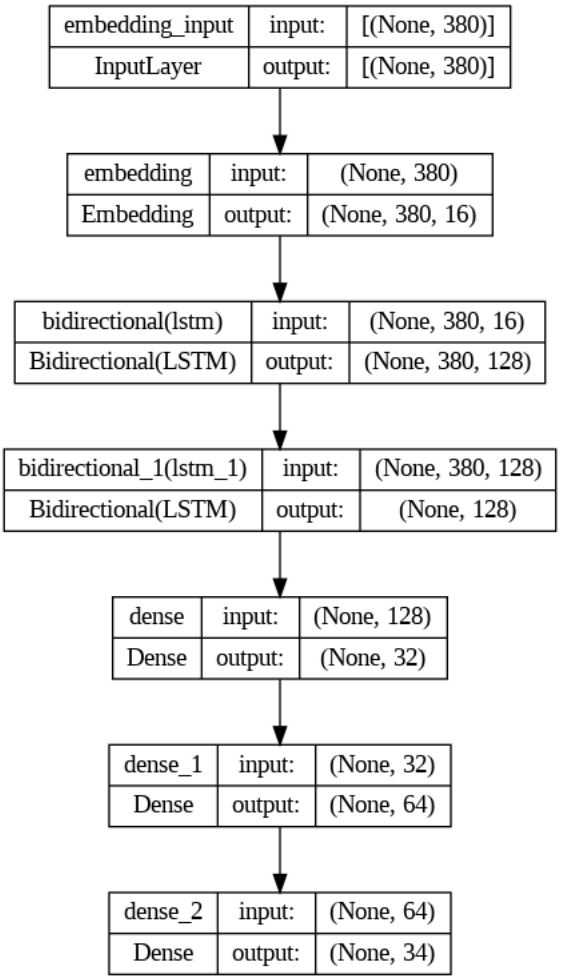

Figure 5.13: BiLSTM Model Layer Architecture

With a precision rate of 56.61%, the model demonstrates a considerable number of precise positive predictions among the total predictions made. The model showcases a considerable recall rate of 53.88%, displaying its ability to accurately identify sentiments within the dataset.Our F-1 score, which takes into account both precision and recall, stands at 54.78%. This metric is crucial in achieving a balance between accurately identifying positive cases while capturing all relevant instances.The results for test accuracy and loss exhibit promise for the model's ability to predict sentiment, with a solid 64.60% accuracy. However, the accompanying loss score of 1.8430 suggests potential for improvement and highlights the potential benefit of fine-tuning or refining our data set. Upon analyzing the overall performance of the Bi-LSTM model, we see well-balanced precision, recall, and F-1 scores. Upon closer inspection, we note specific instances where the model excels and others that require attention. For instance, its accuracy in identifying sentiments classified as "fail" (class 26) is notably high, but it also erroneously predicts anomalies, causing some confusion.

Test Accuracy	0.64609
Test Loss	1.8430
Precision	0.5661
Recall	0.5388
F1 Score	0.5498

Table 5.4: BiLSTM classification accuracy table

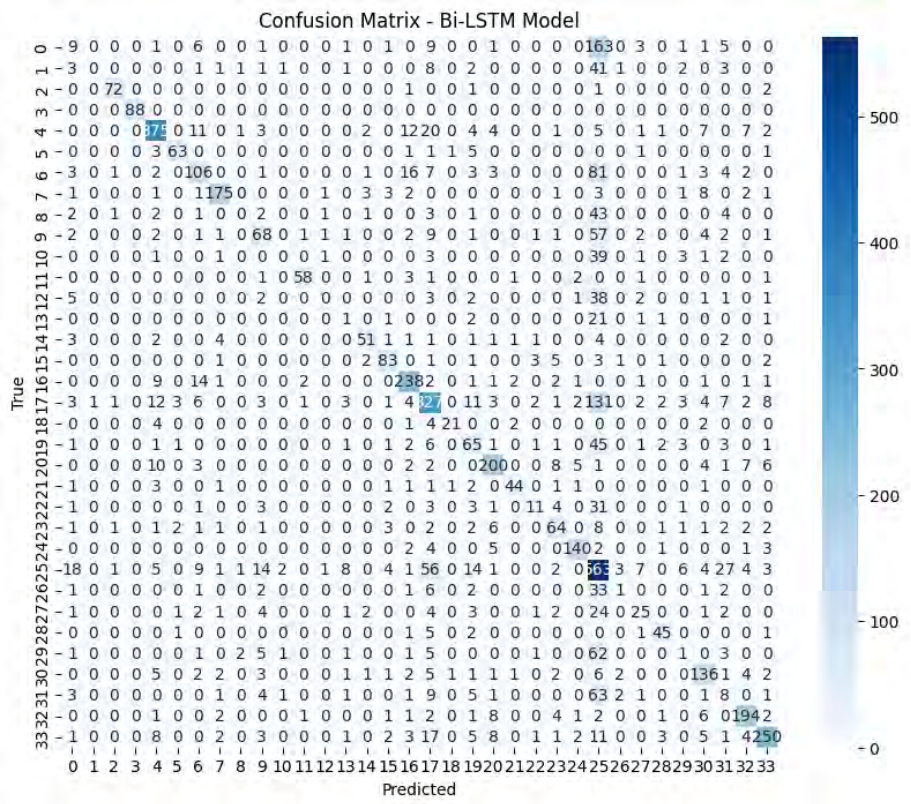

Figure 5.14: BiLSTM Confusion Matrix

In the heatmap, in the x-axis we have the predicted values, and in the y axis we

have our test data. In this heat map we can see 2nd class, 18th class, 24th to 29 class have shown good predictions especially the 2nd and the 18th class. Though the 8th class has a 100% right prediction but it also has falsely predicted various emotions as an anomaly. Here, in our dataset almost all models have predicted a class falsely positive, that something we noticed and trying to remove. After doing so we are confident to have a better accuracy than 64% in Bi-LSTM, as it has shown a better performance as we can see from heat map than CNN and LSTM.

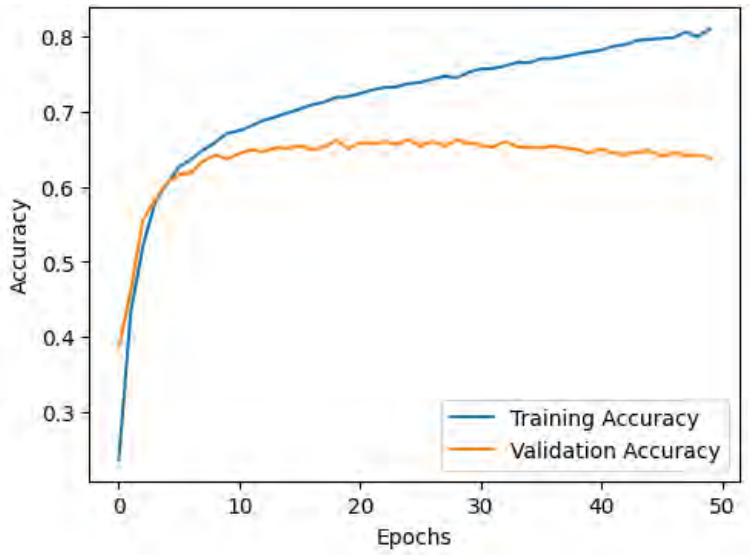

Figure 5.15: Training and Validation Accuracy for BiLSTM

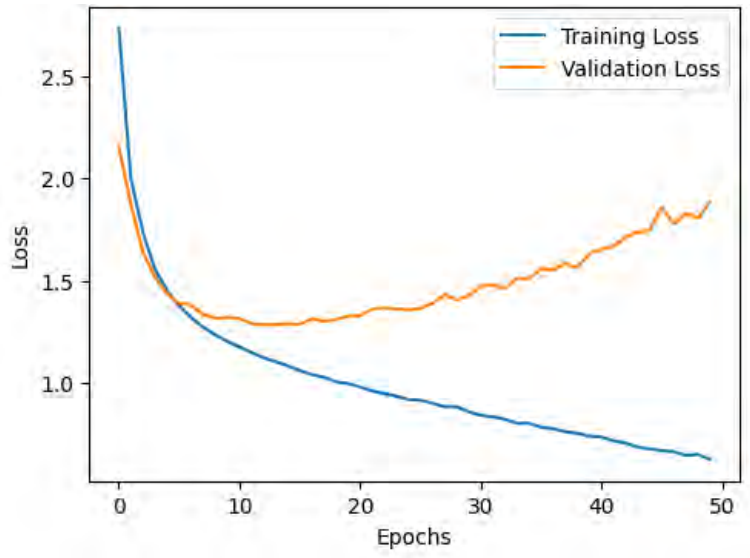

Figure 5.16: Training and Validation Loss for BiLSTM

The Bi-LSTM sentiment analysis model exhibits promising performance, yet there's room for enhancement. Focusing on hyperparameter tuning, dataset analysis, and potential model refinements could lead to substantial improvements in accuracy and overall performance.

5.5 Bert

5.5.1 Data Preparation:

1. Label Encoding: The sentiment labels in your dataset are converted into numerical labels.

2. Parameters Setup: Some settings such as the maximum sequence length, number of classes, learning rate, epochs, batch size, and the BERT model used.

5.5.2 Dataset Setup:

1. A custom dataset is created (TextClassificationDataset) to process text data for the BERT model. This dataset handles the tokenization (splitting text into words or tokens) using the BERT tokenizer.

5.5.3 BERT Classifier

1. A model called BERTClassifier is defined. This model is based on the BERT architecture and incorporates a neural network for text classification. It uses BERT's pre-trained layers and adds a layer for classification.

5.5.4 Training the Model

1. The train function trains the BERT Classifier model using the training dataset (train_dataloader). It updates the model's parameters by adjusting them based on the error it makes in its predictions.

5.5.5 Model Evaluation:

1. The evaluate function assesses how well the trained model performs on data. It computes metrics like accuracy, precision, recall, and generates a confusion matrix to show the model's performance.

5.5.6 Making Predictions:

1. The predict_sentiment function uses the trained model to predict the sentiment of a given piece of text.

5.5.7 Data Preparation and Splitting:

1. The available data is divided into two parts: a training set and a validation set using train_test_split.

2. Tokenization is performed to convert text inputs into numerical tokens that the BERT model can understand.

3. Data loaders (train_dataloader and val_dataloader) are prepared to feed the data into the model during training and evaluation.

5.5.8 Training Loop:

1. The model is trained over multiple cycles or epochs, with each cycle or epoch consisting of a complete run of the training data set. During each epoch, the model learns from the training data and adjusts parameters to improve predictions.

Test Accuracy	0.7324
Precision	0.64
Recall	0.64
F1 Score	0.63

Table 5.5: BERT classification accuracy table

We have got an accuracy of 73% in our dataset. So far it is the highest accuracy we have. An accuracy score of 0.7324 indicates that approximately 73.24% of the model's predictions were correct across all classes. We have got a precision value of 0.64. With a value of 0.64, on average, around 64% of the samples predicted as positive by the model are indeed correct across all classes.With a F1 Score of 0.63, our model shows impressive overall performance in accurately identifying positive cases while keeping false positives low across all classes.

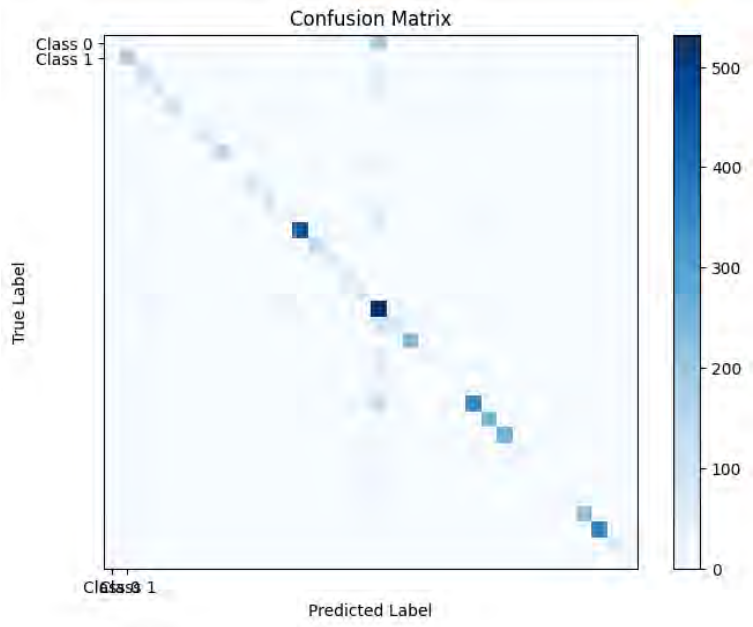

Figure 5.17: BERT Confusion Matrix

These represent the correctly classified instances for each class. Higher values along this diagonal indicate better performance in correctly predicting those specific classes. Higher values outside the diagonal represent instances where the model predicted a different class than the true class.Usually, a color gradient is used to visualize the intensity of the values in the confusion matrix. Higher values might be represented by darker shades or different colors, helping to identify areas where the model is making more mistakes. For instance, the number 178 in the second

row and second column shows that 178 instances of the second class were correctly classified as the second class. On the other hand, the number 3 in the first row and third column means that three instances of the first class were predicted as the third class. Looking at the counts in each row, it appears that some classes have very few predictions for instance, the first class has only 4 predictions, while others, like the second class (178), have significantly more. An imbalance in the distribution of classes might impact the model's performance. Classes with fewer instances might have lower prediction accuracy due to limited training data.

5.6 RCNN Result

The RNN model analyzes text in a sequential manner, processing data token by token and capturing text-based details in its hidden layers. However, its limitations lie in its focus on only the most frequently used words in a sentence, often leading to sentiment misinterpretation. To address this, a CNN model was introduced with a max-pooling layer, culminating in the development of the RCNN model, combining RNN and CNN. CNN's pooling layer serves as a selector, identifying crucial words within text or sentences. Nevertheless, CNN's reliance on a fixed-sized search window impacts its learning speed. Unlike CNN, which mainly focuses on local features, RCNN integrates contextual information through its recurrent layers. This allows it to capture not just individual patterns but also the sequence-based dependencies within the data.The CNN component of RCNN excels in feature extraction. The combination of convolutional layers and max-pooling aids in identifying significant patterns or features within the input data. RNNs often encounter vanishing or exploding gradient problems, which can hinder learning in deeper networks. By merging CNN's ability for feature extraction with RNN's sequence modeling, RCNN mitigates these issues to some extent. RCNN generates more informative representations of input data by integrating both spatial and sequential information, resulting in richer feature representations compared to standalone CNN or RNN models.It's noteworthy that RCNN, while efficient, demands higher computational resources for both training and testing the data.

In our model the embedding layer Maps integer indices (representing words) to dense vectors. It transforms each index into a fixed-size dense representation. After the two conv 1D layer we have a max pooling layer which Reduces the dimensionality of the CNN output while retaining important information.Fully connected dense layers applying nonlinear transformations to the flattened CNN output.We have also Stacked LSTM layers that capture sequential information in the input data.Then finally we have merged the output from CNN and LSTM model. Additional fully connected layers applied to the concatenated output, enabling the model to learn high-level features. At last the output layer Produces the final output predictions with a softmax activation function for multiclass classification (34 classes).

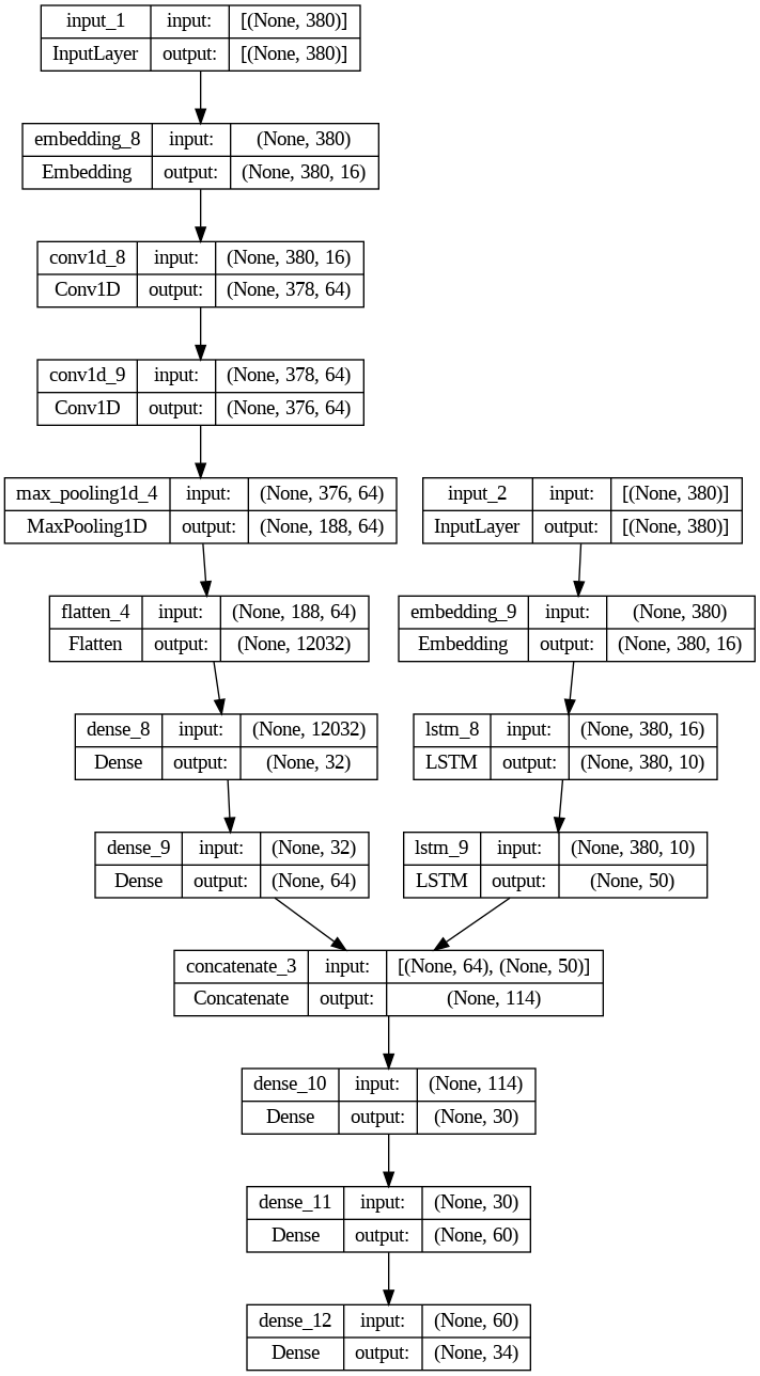

Figure 5.18: RCNN(CNN+LSTM) Model Layer Architecture

Test Accuracy	0.5797
Test Loss	2.6519
Precision	0.4878
Recall	0.4923
F1 Score	0.4839

Table 5.6: RCNN(CNN+LSTM) classification accuracy table

We have got an accuracy of 57.96%. In our case, the accuracy of 0.5797 indicates that around 57.97% of all predictions made by the model were correct. However, accuracy

alone might not provide the full picture, especially in an imbalanced dataset. Our precision is 0.4878. A precision score of 0.4878 suggests that when the model predicts a class, it is correct about 48.78% of the time for that specific class. Our RCNN model achieved a recall score of 0.4923, indicating that it correctly identified around 49.23% of all actual positive instances. F1 Score is 0.4839. The F1 score is the harmonic mean of precision and recall. It considers both false positives and false negatives. The obtained F1 score of 0.4839 suggests the overall balance between precision and recall for this model.

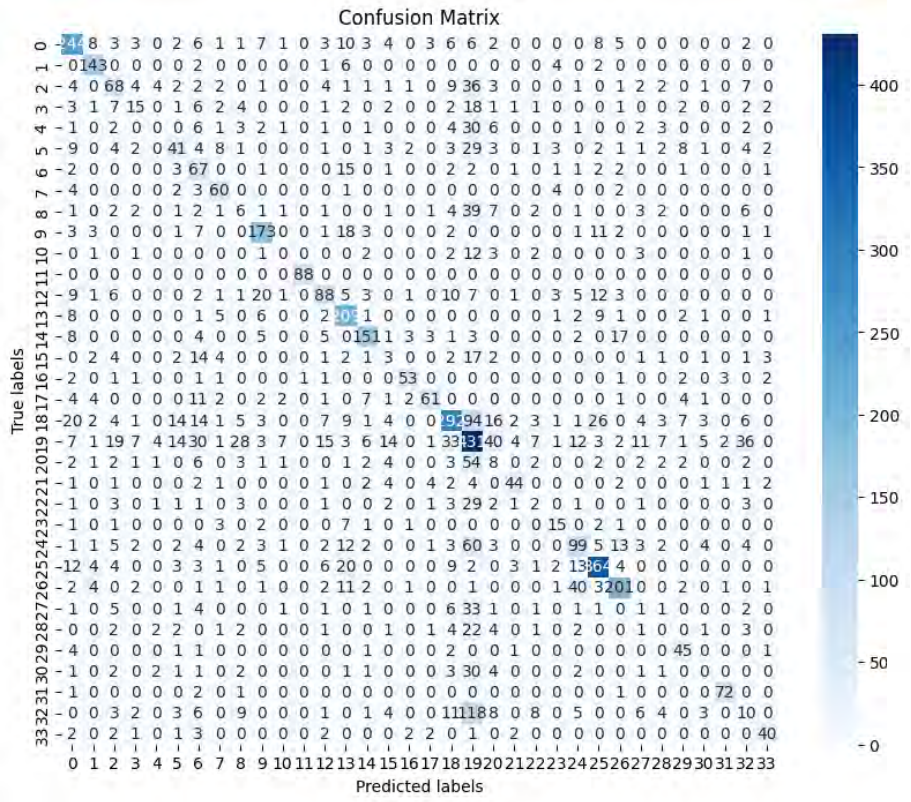

Figure 5.19: RCNN(CNN+LSTM) Confusion Matrix

In this heatmap, in the x-axis we have the predicted values, and in the y axis we have our test data. We can see in our 1st class(Like) we made 244 right predictions. In our 25th(provocative) class we have 364 right predictions, also in 13th(expected)class we have 209 right predictions. The diagonal of the heatmap shows the true positive result and as 10/11 of the classes are showing the true positive results. So, we can assume the model diagonal shows the accuracy of 57%. Also, we have a highest right prediction in class 19th(fear), which is a little questionable as in the 19th class(fear) we are also seeing a straight line, however 19th class(Fear) is falsely predicted as various emotions as an anomaly.

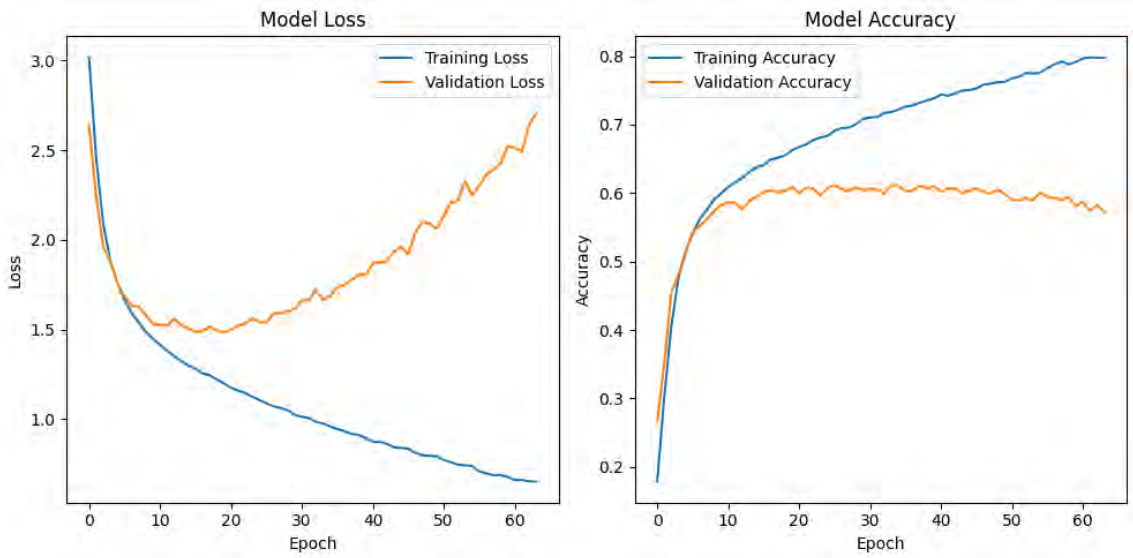

Figure 5.20: Training and Validation Accuracy and Loss for RCNN(CNN+LSTM)

Interpretation:

1. The accuracy of 57.97% indicates that the model is performing better than random guessing but might not be sufficiently high for certain applications.

2. The precision, recall, and F1 score values, around 0.48-0.49, suggest that the model's performance is moderate, but it may struggle to handle certain classes effectively.

3. Considering the presence of 34 classes, imbalanced classes might influence these metrics. For instance, classes with fewer instances might have a lower impact on overall metrics.

4. Further investigation into class-wise metrics or strategies to address class imbalance could enhance the model's performance, especially for minority classes.

In summary, while the RCNN (CNN+LSTM) model exhibits moderate performance across multiple evaluation metrics, class imbalance or specific class difficulties could benefit from additional model tuning or data preprocessing techniques to improve overall effectiveness.

BiLSTM (Bidirectional Long Short-Term Memory networks) often outperform LSTM in certain tasks due to their enhanced ability to capture context. BiLSTMs process data in both forward and backward directions, allowing them to capture context from both past and future sequences. This bidirectional approach enables the model to understand dependencies more comprehensively, especially when the information available in both directions is crucial for accurate predictions. BiLSTMs offer a more nuanced understanding of the context within sequences. They can better capture long-term dependencies and relationships between words or tokens in both directions, improving the model's ability to understand and interpret sequences effectively. In RCNN, where a comprehensive understanding of both local features

and long-term dependencies within sequential data is crucial, the bidirectional aspect of BiLSTMs can offer a more robust and detailed comprehension of the text, leading to higher accuracy compared to unidirectional LSTM models.

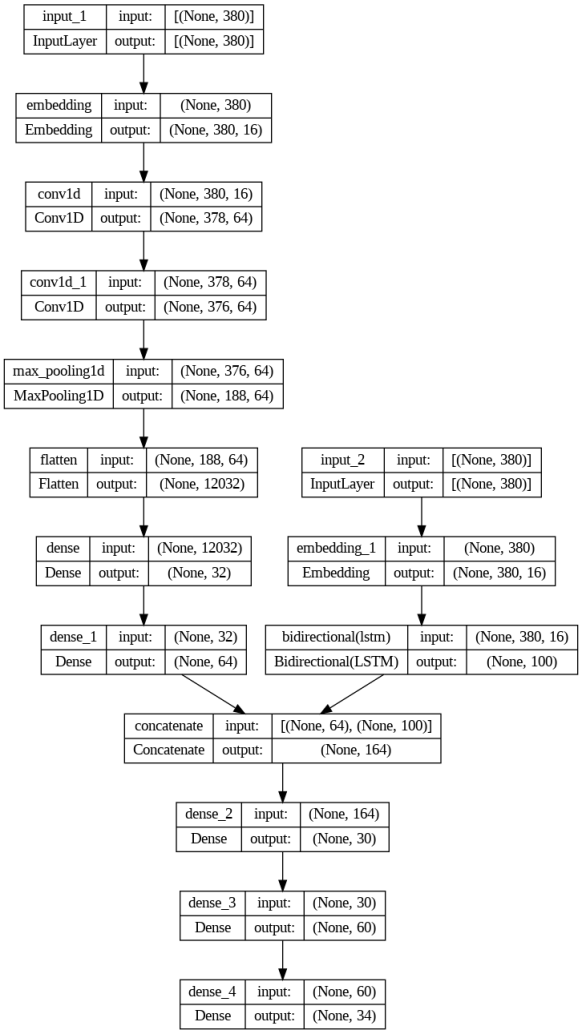

Figure 5.21: RCNN(CNN+BiLSTM) Model Layer Architecture

So, considering the strengths of Bi-LSTM we considered modifying our hybrid model, we replaced the LSTM model with Bi_Lstm, as it has shown good accuracy in our dataset. We have separate input layers for both CNN and Bi_Lstm. Then in CNN model we have a embedding layer which has vectorized the input layer, following than we have 2 conv 1D layers, than a pooling layer, in the Bi_lstm model we have an embedding layer following that one Bi_Lstm layer than lastly we have concatenated these two layers for a better output.

Test Accuracy	0.5891
Test Loss	3.0679
Precision	0.5077
Recall	0.5143
F1 Score	0.5088

Table 5.7: RCNN(CNN+BiLSTM) classification accuracy table

With an accuracy of 58.91%, the model correctly predicts approximately 58.91% of the instances in our dataset. The precision of 50.77% suggests that when it predicts an instance as positive, it's correct around 50.77% of the time. The recall of 51.43% indicates that it captures about 51.43% of all positive instances. The F1 Score of 50.88% considers both precision and recall.

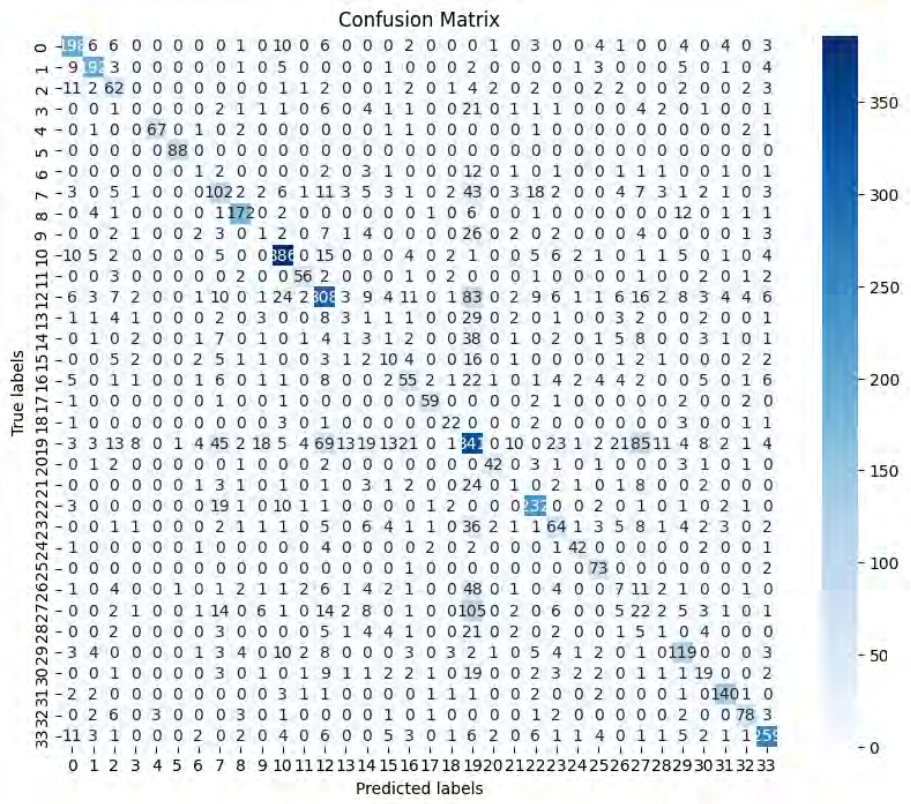

Figure 5.22: RCNN(CNN+BiLSTM) Confusion Matrix

From this heat map we can see that some classes have better true positive predictions than we did with Lstm. Some classes like 10th class, 12th class, 33th class, 22th class have better right predictions, though the problem with the 19th class seems to be remaining.This model seems to perform reasonably well but may benefit from further improvements in precision, recall, and the F1 score for a more balanced and accurate prediction across all classes. So, since it has better accuracy, we hope to work with this model when we overcome the limitations of our Dataset.

The range of models used to process the dataset revealed varying degrees of performance. The Artificial Neural Network (ANN) model and Convolutional Neural Network (CNN) exhibited moderately competitive accuracy of 60% and 59%, signifying their capability to capture certain patterns within the data. ANN might be capturing some patterns but might not handle complexities or sequence data as effectively as other models.. Though CNN is Good at capturing spatial features; useful for image data. However, for sequential data like text, it might not capture dependencies well.

Model Name	Test Accuracy	Precision	Recall	F1 Score
ANN	0.6036	0.4551	0.4046	0.4058
CNN	0.5968	0.5052	0.4761	0.48567
LSTM	0.1379	0.00405	0.02941	0.00713
Bi-LSTM	0.64609	0.5661	0.5388	0.5498
BERT	0.7324	0.64	0.64	0.63
RCNN with Lstm	0.5797	0.4878	0.4923	0.4839
RCNN with Bi-Lstm	0.5891	0.5077	0.5143	0.5088

Table 5.8: Accuracy Comparison Between Our Applied Models

However, the LSTM model surprisingly yielded notably low accuracy of 13%, suggesting potential issues in training, data preparation, or the model's ability to comprehend sequential patterns effectively. This could indicate issues with model training, data preprocessing, or the model architecture not effectively capturing sequence patterns. Conversely, the Bi-LSTM model showed improved accuracy of 64%, benefitting from its bidirectional nature. Better performance than the traditional LSTM, as bidirectional LSTM captures patterns from both past and future contexts, which might be helpful in understanding the sequence.

The BERT model, known for its context-awareness, emerged as the top performer, showcasing a high accuracy level. It has an accuracy of 73%. High accuracy, indicating strong performance. BERT, being a transformer-based model, is excellent at understanding context in sequential data. The RCNN architectures, combining convolutional and recurrent networks, displayed moderate accuracy rates of 57%, with the RCNN featuring Bi-LSTM slightly surpassing its LSTM counterpart. The combination of CNN for spatial features and LSTM for sequence understanding might not have synergized well in this particular architecture or dataset. Whereas RCNN with Bi-lstm as an accuracy slightly improved as 58%. Slightly improved from RCNN with LSTM, indicating that bidirectional LSTM might capture more complex patterns than a unidirectional LSTM within the RCNN architecture.

Overall, models capable of comprehending contextual information, like BERT and Bi-LSTM, exhibited superior performance, highlighting the importance of context understanding in sequential data analysis, while emphasizing the need for further exploration and optimization in the RCNN architectures.

5.7 Discussion

Our primary objective is to build a recommendation system for intelligent captioning using sentiment analysis.To accomplish this goal, we initiated our efforts by thoroughly reviewing relevant research papers that could provide insights for our ongoing research.This comprehensive review has granted us a clear understanding of the challenges and intricacies associated with the topic. We delved into 15 papers that presented diverse approaches to addressing the problem.Our focus extended to the exploration of sentiment analysis in social networks, encompassing platforms

such as Twitter, Facebook and review sentiments with an emphasis on integrating them into a recommendation system through textual inspection. During this exploration, we encountered the latest technologies employed in recent works for analysis, as well as less efficient methods previously applied to the same problem.Our investigation unearthed various techniques for data preprocessing and shed light on the diverse methodologies people have employed to tackle this particular problem. Our

initial target was to obtain reliable data to train our model.Due to the scarcity of available datasets in Bangla, we generated 27,368 data manually.We conducted data preprocessing and developed ANN,CNN,LSTM,Bi-LSTM,RCNN and BERT models to address our specific problem. Our efforts included mitigating challenges such as overfitting and underfitting, aiming to train and test for effectively resolving our problem.Moreover,We have started by amalgamating the data and models we have created. Following a training session conducted here,we proceed to test our data, assessing the accuracy of the results. Subsequently, we have calculated the error or loss functions. Our evaluation process encompasses examining the training results, testing outcomes, and overall model performance.We encountered numerous challenges during the development of the model, particularly in achieving satisfactory accuracy due to the presence of an imbalanced dataset. Furthermore, we intend to compare our model outputs with the findings from prior research conducted by others.

5.8 Future Work

Due to the imbalanced dataset, we are unable to achieve the expected level of accuracy. In our future work, we plan to address the imbalance in the dataset, aiming to enhance it. This improvement is anticipated to lead to higher accuracies across all models.However,we will try to create models that can adapt and perform well in different domains without extensive retraining. This involves techniques that can generalize sentiments across various industries or specific topics.Additionally, we will improve the ability of sentiment analysis models to interpret colloquial language, abbreviations, slang, and emojis, which are common in informal communication.We will build systems that can interact with users to refine sentiment analysis results based on user feedback, thus improving accuracy and user satisfaction.Lastly,in our upcoming work, we intend to employ additional hybrid models with the aim of achieving improved accuracy.

Chapter 6

Conclusion

The widespread access to social media is leading to a vast amount of opinions, which can be analyzed through sentiment analysis.This method of deep learning is used to determine the emotional tone of text and can be used to improve recommendation systems. Sentiment analysis proves particularly valuable in scenarios where only sparse ratings data are available.This approach can also help to overcome language barriers.For instance,many people who use Bangla as their first language may encounter challenges in comprehending and composing English text.A recommendation system that analyzes sentiment in Bangla writings could prove beneficial for our population.However,there is room for improvement, given the current limitations in publicly available Bangla datasets.This thesis aims to use sentiment analysis in conjunction with neural networks to improve text analysis.Different crowdsourcing options will be investigated and two medium-sized and emotion-labeled song data sets will be constructed using social tagging. Additionally, various text collections will be used in studies with different contents and domains, as well as word embeddings.Therefore, we tried our best to generate dataset manually due to the absence of sufficient Bangla dataset.In this paper, we explored six distinct neural network models(ANN,LSTM,CNN,Bi-LSTM,RCNN,BERT) and successfully navigated numerous challenges with our dataset.Thus far,we have diligently undertaken data preprocessing efforts to achieve an optimistic level of accuracy, aspiring to attain a noteworthy accuracy level with this dataset.Sentiment analysis applied to Bangla text represents a burgeoning frontier with immense potential. The exploration of emotions embedded in Bangla writings, facilitated by sophisticated algorithms, not only enriches our understanding of linguistic nuances but also opens avenues for tailored applications. Despite the current challenges posed by limited publicly available Bangla datasets, the significance of sentiment analysis in bridging language barriers and enhancing user experiences cannot be overstated. Continued efforts to expand datasets, refine algorithms, and adapt methodologies to the intricacies of the Bangla language will undoubtedly propel sentiment analysis into a transformative tool, offering profound insights and applications across various domains for the Bangla-speaking population.

Bibliography

[1] S. Chowdhury and W. Chowdhury, "Performing sentiment analysis in bangla microblog posts," 2014. DOI: 10.1109/ICIEV.2014.6850712.

[2] A. Hassan, M. R. Amin, A. K. A. Azad, and N. Mohammed, "Sentiment analysis on bangla and romanized bangla text using deep recurrent models," in *2016 International Workshop on Computational Intelligence (IWCI)*, IEEE, Dec. 2016, pp. 51–56.

[3] M. H. Alam, M. M. Rahoman, and M. A. K. Azad, "Sentiment analysis for bangla sentences using convolutional neural network," in *2017 20th International Conference of Computer and Information Technology (ICCIT)*, IEEE, Dec. 2017, pp. 1–6.

[4] E. Çano, "Text-based sentiment analysis and music emotion recognition," 2018, arXiv preprint arXiv:1810.03031.

[5] H. Yuan, Y. Wang, X. Feng, and S. Sun, "Sentiment analysis based on weighted word2vec and att-lstm," in *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence*, Dec. 2018, pp. 420–424.

[6] L.-C. Cheng and S.-L. Tsai, "Deep learning for automated sentiment analysis of social media," 2019.

[7] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019, arXiv preprint arXiv:1908.10084.

[8] A. Khamparia, B. Pandey, S. Tiwari, D. Gupta, A. Khanna, and J. J. Rodrigues, "An integrated hybrid cnn-rnn model for visual description and generation of captions," *Circuits, Systems, and Signal Processing*, vol. 39, no. 2, pp. 776–788, 2020.

[9] A. Ahmad *et al.*, "Sust tts corpus: A phonetically-balanced corpus for bangla text-to-speech synthesis," *Acoustical Science and Technology*, vol. 42, no. 6, E2126, 2021, Accessed: 22 January 2024. [Online]. Available: https://www.jstage.jst.go.jp/article/ast/42/6/42_E2126/_article/-char/ja/.

[10] K. Cherry. "The 6 types of basic emotions and their effect on human behavior." (2021), [Online]. Available: https://www.verywellmind.com/an-overview-of-the-types-of-emotions-4163976.

[11] C. N. Dang, M. N. Moreno-García, and F. D. la Prieta, "An approach to integrating sentiment analysis into recommender systems," *Sensors*, vol. 21, no. 16, p. 5666, 2021. DOI: 10.3390/s21165666.

[12] C. N. Dang, M. N. Moreno-García, and F. D. la Prieta, "Hybrid deep learning models for sentiment analysis," *Complexity*, vol. 2021, 2021.

[13] K. Dewi, P. Ciptayani, and P. Prayustika, "A systematic literature review on hybrid deep learning smart recommendation systems," in *Proceedings of the 4th International Conference on Applied Science and Technology on Engineering Science*, 2021. DOI: 10.5220/0010942000003260.

[14] S. Munuswamy, M. S. Saranya, S. Ganapathy, S. Muthurajkumar, and A. Kannan, "Sentiment analysis techniques for social media-based recommendation systems," *Journal Name*, 2021.