Sentiment Analysis on Customer Feedback

A Thesis submitted to

Chhattisgarh Swami Vivekanand Technical University Bhilai(C.G.), India

In partial fulfilment for award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering

By

Shubhi Sahu (301602220009)

CH Sushmita (301602220027)

Rishita Ghosh (301602220055)

Under the Guidance of

Mr. Prakhar Golchha

Assistant Professor

Department of Computer Science & Engineering

Government Engineering College Raipur

Session: 2023–24

DECLARATION BY THE CANDIDATES

We the undersigned, solemnly declare that the thesis of the Project Work entitled ***"Sentiment Analysis on Customer Feedback"***, is based on our own work carried out during our study under the supervision of ***Mr. Prakhar Golchha.***

We assert that the statements made, and conclusions drawn, are an outcome of the project work. We further declare that to the best of our knowledge and belief that the report does not contain any part of any work which has been submitted for the award of any other degree/diploma/certificate in this University/deemed the University of India or any other country. All help received and citations used for the preparation of the Project Work have been duly acknowledged.

(Signature of the candidate)
Name: CH Sushmita
Roll No: 301602220027
Enrolment No: BK0579

(Signature of the candidate)
Name: Rishita Ghosh
Roll No: 301602220055
Enrolment No: BK1449

(Signature of the candidate)
Name: Shubhi Sahu
Roll No: 301602220009
Enrolment No: BK0331

(Signature of the Supervisor)
Mr. Prakhar Golchha
Assistant Professor
Dept. Of Computer Science &
Engineering, Government Engineering College
Sejbahar, Raipur

CERTIFICATE OF THE SUPERVISOR

This is to certify that the thesis entitled ***"Sentiment Analysis on Customer Feedback"*** is a record of bonafide research work carried out by ***CH Sushmita*** bearing Roll no: ***3016022200227*** & Enrolment No.: ***BK0579, Rishita Ghosh*** bearing Roll no: ***301602220055*** & Enrolment No.: ***BK1449*** and ***Shubhi Sahu*** bearing Roll no: ***301602220009*** & Enrolment No.: ***BK0331*** under my guidance and supervision for the award of Degree of Bachelor of Technology in the field of ***Computer Science and Engineering***, of Chhattisgarh Swami Vivekanand Technical University, Bhilai (C.G.), India.

To the best of my knowledge and belief the Project Work

- ❖ Embodies the work of the candidate him/herself,
- ❖ Has duly been completed,
- ❖ Fulfils the requirement of the ordinance relating to the BTech degree of the University.
- ❖ Is up to the desired standard both in respect of contents and language for being referred to the examiners.

Prof. (Dr.) R.H. Talwekar
(Head Of Department)
Dept. Of Computer Science & Engineering
Government Engineering College,
Sejbahar, Raipur (C.G.)

Forwarded to Chhattisgarh Swami Vivekanand Technical University Bhilai

(Signature of Principal)
Government Engineering College, Sejbahar, Raipur

CERTIFICATE BY THE EXAMINERS

The Thesis entitled ***"Sentiment Analysis on Customer Feedback"*** submitted by ***CH Sushmita*** (Roll No.: ***301602220027*** & Enrolment No.: ***BK0579), Rishita Ghosh (***Roll No.: ***301602220055*** & Enrolment No.: ***BK1449)*** and ***Shubhi Sahu (***Roll No.: ***301602220009*** & Enrolment No.: ***BK0331)*** has been examined by the undersigned as a part of the examination and is hereby recommended for the award of the degree of Bachelor of Technology in the field of Computer Science and Engineering of Chhattisgarh Swami Vivekanand Technical University, Bhilai.

Internal Examiner	External Examiner
Date:	Date:

ACKNOWLEDGEMENT

We would like to thank our guide **Mr. Prakhar Golchha**, Assistant Professor, Department of Computer Science and Engineering for his immense support and enlightened guidance for our project which we have developed as B. Tech. students.

We are also very thankful to **Mr. Pushpendra Dhar Dwivedi,** Assistant Professor and Project Coordinator, for his continuous support and guidance. We are very grateful for the inspiring discussions with all our faculties, their valuable support and path-guiding suggestions have helped us to develop this project. Alongside this, we would like to take this opportunity to express our thanks to everyone in the Computer Laboratory of Government Engineering College, Sejbahar, and Raipur (C.G.).

We especially thank our colleagues, batch mates, and friends for the support and help that they offered us during the development of this project.

(Signature of the candidate)
Name: CH Sushmita
Roll No: 301602220027
Enrolment No: BK0579

(Signature of the candidate)
Name: Rishita Ghosh
Roll No: 301602220055
Enrolment No: BK1449

(Signature of the candidate)
Name: Shubhi Sahu
Roll No: 301602220009
Enrolment No: BK0331

ABSTRACT

Sentiment analysis plays a crucial role in understanding and extracting valuable insights from customer feedback on products purchased through e-commerce platforms. This project presents a sentiment analysis model which will in future be used specifically for evaluating customer sentiments expressed in their feedback. The goal is to automatically classify customer sentiments as positive, negative. The trained classical and deep learning based models leverages natural language processing (NLP) techniques, including pre-trained word embeddings and deep learning architectures, to capture the semantic nuances in customer reviews. The dataset used for training and evaluation consists of a diverse set of customer feedback collected from the e-commerce platform.

The evaluation of the sentiment analysis model is conducted using various performance metrics, such as accuracy, precision, recall, and F1 score. Additionally, the model's generalization capabilities are tested on unseen data to assess its robustness in handling new and diverse customer feedback. The results indicate that the proposed sentiment analysis model achieves high accuracy in classifying customer sentiments, thereby providing e-commerce businesses with an effective tool to gain insights into customer satisfaction levels. The model's ability to process large volumes of customer feedback in real-time positions it as a valuable asset for enhancing decision-making processes, product improvements, and overall customer experience in the dynamic landscape of e-commerce.

Table of Contents

List of Figures

List of Tables

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

In this ever-evolving world of technology and business, it has become crucial for businesses to understand and respond to customer feedback which represents the customer sentiment for maintaining the position in the market and elevating its performance. Customer feedback is a valuable part in running a business, it can be expressed through feedback or reviews and by conducting surveys which consequently helps in product improvement, elevating the business profit and enhancement of customer experience.

Sentiment analysis on customer feedback is a project aimed at providing an efficient solution to analyze customer feedback in order to derive insights to elevate user experience and platform excellence. The project leverages advanced deep learning models, including BERT (Bidirectional Encoder Representation of Transformer) and LSTM (Long Short-Term Memory) to analyze whether the feedback is positive or negative.

In the first step, we have implemented various machine learning models to compare the performance of the algorithms for the given dataset. The algorithms include Logistic Regression, Naïve Bayes, Gradient Boosting and LSTM. Following this, the BERT Model is implemented which excels in understanding the contextual nuances and read the data in both direction unlike tradition model which go either right to left or left to right direction.

Sentiment analysis is a natural language processing task, which has emerged as a crucial tool in deciphering and categorizing the sentiments expressed in textual content. In this project, we aim to build an automated system which would help the user to understand the feedbacks of the customer i.e. whether the feedbacks are positive or negative. The model we proposed processes a tokenized data, which undergoes the embedding layer followed by 12 encoders. We have optimized an addition layer of sentiment classifier which will further help the vector to be categorized.

In this project, we have developed a web interface using tools such as HTML, CSS, flask, jinja template engine, python and its libraries. The user interface is designed to be simple and intuitive, allowing users to upload a csv file as input data or a single line of sentence. Further the interface is integrated with the BERT model.

1.2 OBJECTIVE

“Sentiment Analysis on Customer Feedback” project aims to develop an advanced sentiment analysis system with a primary focus on leveraging machine learning algorithms, to classify textual data into positive, negative, or neutral sentiments. The primary objectives is to develop an application that would take customer reviews and analyze it to derive insights on overall customer satisfaction with their respective business to help elevate user experience and platform excellence. encompass the establishment of a robust preprocessing pipeline to clean and standardize textual data, the implementation and training of sentiment analysis models, and a comprehensive evaluation of these models. Furthermore, the project aims to conduct a comparative analysis among various models. This comparative study will involve an exploration of how each algorithm adapts to varying sentiment types, linguistic nuances, and text lengths, providing valuable insights into their practical applicability. In short, Analysing customer feedback on E-commerce platforms to derive insights on overall customer satisfaction to help elevate user experience and platform excellence.

CHAPTER-2

LITERATURE REVIEW

An article by Li Yang and Jing Wang proposes a new sentiment analysis model-SLCABG, which is based on the sentiment lexicon and combines Convolutional Neural Network (CNN) and attention-based Bidirectional Gated Recurrent Unit (BiGRU). In terms of methods, the SLCABG model combines the advantages of sentiment lexicon and deep learning technology, and overcomes the shortcomings of existing sentiment analysis model of product reviews. The SLCABG model combines the advantages of the sentiment lexicon and deep learning techniques. First, the sentiment lexicon is used to enhance the sentiment features in the reviews. Then the CNN and the Gated Recurrent Unit (GRU) network are used to extract the main sentiment features and context features in the reviews and use the attention mechanism to weight. And finally, classify the weighted sentiment features. They take data from real book evaluations of dangdang.com, a famous Chinese e-commerce website, for training and testing, all of which are based on Chinese.

By analyzing the experimental results, it can be found that the model has better classification performance than other sentiment analysis models. By using our model to analyze user reviews, we can help merchants on e-commerce platforms to obtain user feedback in time to improve their service quality and attract more customers to patronize. Besides, with the continuous enrichment of the sentiment lexicon and the increase of the dataset, the classification accuracy of the model will gradually improve. However, the approach proposed in this paper can only divide sentiment into positive and negative categories, which is not suitable in areas with high requirements for sentiment refinement. Therefore, the next step is to study the sentiment fineness classification of text. [1]

An article by Harsheta Pandita; Naveen Kumar Gondhi presents a comprehensive way of sentiment analysis which provides the viewpoints highlighted in various articles regarding the process, required tasks and strategies of sentiment analysis. It also discusses various challenges associated with the sentiment classification process. [2]

The study by Ms. Stuti M. Meshram and Dr. Neeraj Sahu describes an investigation of the sentiment analysis for e-commerce product reviews through machine learning techniques. Machine learning algorithms were employed to classify the reviews as positive, negative, or neutral. The data used were scraped from a popular e-commerce platform and preprocessed to ensure the quality of the dataset. The models trained on the preprocessed dataset achieved promising results, with an accuracy of up to 90%. Moreover, a comparative study of different algorithms was conducted, and the best-performing algorithm was identified. [3]

Bhagat, C.; Mane, in an article proposed a work where they used a hybrid approach of naive Bayes and K-nearest neighbor to divide tweets into three classes: positive, negative, and neutral, and they achieved a better accuracy than the random forest. [4]

Abayomi Bello,Sin-Chun Ng andMan-Fai Leung proposed a work named A BERT Framework to Sentiment Analysis of Tweets with two BERT-based approaches for text classification: BERT-base and cased BERT-base. The study proposes text classification with the use of bidirectional encoder representations from transformers (BERT) for natural language processing with other variants. The experimental findings demonstrate that the combination of BERT with CNN, BERT with RNN, and BERT with BiLSTM performs well in terms of accuracy rate, precision rate, recall rate, and F1-score compared to when it was used with Word2vec and when it was used with no variant.They gathered information from microblogging sites, particularly twitter. Two separate datasets were employed in their experiment, and they were used for sentiment analysis and emotion recognition. emphasized that BERT produces positive outcomes for text classification. The traditional approach of natural language processing (NLP) with the use of Word2Vec, CNN, RNN, and BiLSTM has a few limitations of not capturing the deeper context of the word. The BERT has more understanding than traditional since the encoder process all inputs, which is the whole sentence, simultaneously so when building a context for a word, BERT will take into account the inputs before it and also the inputs after the word, while the Word2Vec restricts the ability to understand a word's meaning across two contexts and used in two different situations which in turn will be reflected on the output vector value for the word.

The combination of the transformer model BERT with CNN, RNN, and BiLSTM gives a state-of-the-art performance in terms of accuracy, recall, and precision.

Further works can be carried out to analyze sentiment on more data that is not sourced online because some information shared online can be shared by tourists or people with little or no understanding about the subject matter. Moreover, instead of sentiment, emotions such as happy, sad, and surprised can also be studied. Other transformer models such as RoBERTa can also be further investigated in further studies. [5]

CHAPTER-3

PROBLEM IDENTIFICATION

3.1 PROBLEM STATEMENT

Every passing second, thousands of products are purchased, posts on social media are posted or viewed and subsequently reviews, comments on social media and posts are shared. If we had to manually determine and check whether each one was positive or negative, imagine the time it would take for you. This is where sentiment analysis comes into play, helping artificial machines or computers to figure out what emotions are behind all that text. It is important to figure out which sentiment analysis tool works best, since not all of them are equally good.

To tackle and overcome this very challenge, our project seeks to train and build a machine learning model that can accurately understand these sentiments. Whether one is more accurate at judging a positive or negative review or feedback is what we are trying to understand and tackle. Through this approach, we can understand what customers of the e-commerce platforms like Amazon, Flipkart etc feel about the products they are offering, and accordingly adapt some corrective measures, resulting in better growth and performance for these platforms.

3.2 EXISTING SYSTEM

The existing systems of sentiment analysis predominantly rely on rule-based or machine learning approaches to discern sentiment from textual data. Rule-based systems employ predefined rules and lexicons to assign sentiment labels to text, while machine learning models leverage labeled training data to predict sentiment based on learned patterns. However, these systems encounter several challenges that impede their accuracy and reliability in real-world applications.

One significant issue with existing sentiment analysis models is may exhibit disparities in performance across different groups or topics, perpetuating biases in decision-making processes that rely on sentiment insights. Furthermore, the contextual understanding required for accurate sentiment analysis poses a significant challenge. Sentiment interpretation heavily depends on linguistic nuances, cultural contexts, and the presence of sarcasm or irony in text. Existing models struggle to grasp these subtleties, resulting in misclassified sentiment and erroneous conclusions in sentiment analysis tasks.

Another critical limitation is the difficulty in handling negation and sentiment modifiers within text. Negated phrases such as "not bad" or "not happy" can invert sentiment polarity, complicating sentiment analysis accuracy. Similarly, intensifiers (e.g., very, extremely) and mitigators (e.g., somewhat, slightly) can alter sentiment strength, requiring sophisticated text processing techniques to account for these linguistic complexities.

3.3 PROPOSED SYSTEM

Our proposed solution involves the development of a web-based sentiment analysis system designed to leverage cutting-edge deep learning architectures, including Long Short-Term Memory (LSTM), Bidirectional Encoder Representations from Transformers (BERT). This system aims to enhance sentiment prediction accuracy and provide users with a seamless and intuitive interface for analyzing textual data. The web application will feature a user-friendly input interface allowing users to input text data via a text box or file upload. Upon receiving the input, the system will preprocess the text and pass it through integrated LSTM and BERT models for sentiment analysis. The LSTM model will capture sequential patterns and long-range dependencies within the text, while BERT will encode contextual information and handle linguistic nuances. BERT, on the other hand, will generate coherent text representations and facilitate accurate sentiment predictions. The system will display sentiment analysis results, including sentiment labels (positive and negative) and confidence scores, through the web interface, providing users with actionable insights derived from advanced deep learning techniques.

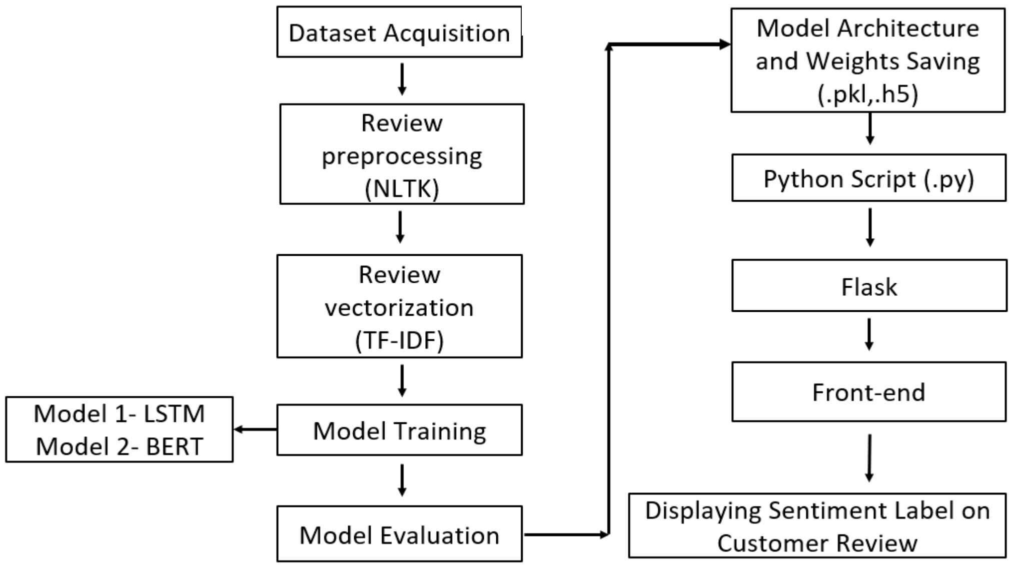
This sentiment analysis system will be trained using labeled sentiment analysis datasets sourced from reputable repositories, ensuring the models' effectiveness across diverse text samples and sentiment categories. The user-friendly interface of the web application will promote accessibility and usability, allowing users to interact with the system intuitively and obtain real-time sentiment analysis results. By integrating state-of-the-art deep learning models with a seamless web-based interface, our proposed solution aims to advance sentiment analysis capabilities and empower users with accurate and actionable insights extracted from textual data analysis. This system holds promise for various applications across industries, facilitating informed decision-making and enhancing understanding of sentiment trends within textual data.

CHAPTER-4

METHODOLOGY

The Flow Chart of our project is as follows:

10



Dataset Acquisition
Review preprocessing (NLTK)
Review vectorization (TF-IDF)
Model Training
Model 1- LSTM
Model 2- BERT
Model Evaluation
Model Architecture and Weights Saving (.pkl,.h5)
Python Script (.py)
Flask
Front-end
Displaying Sentiment Label on Customer Review

Fig 4-a : Flowchart of the Project

4.1 DATASET

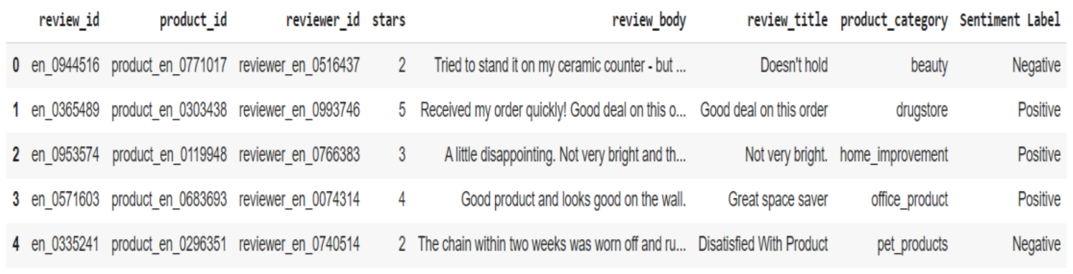
The dataset is gathered from Hugging Face Open Source Library. The dataset contains Amazon product reviews in English. Each record in the dataset contains the review id, the review title, the review body, the star rating, an anonymized reviewer ID, an anonymized product ID and the coarse-grained product category (e.g. ‘books’, ‘appliances’, etc.) The corpus is balanced across stars, so each star rating constitutes 20% of the reviews. The dataset is in CSV format.

Data Fields:

- review_id: A string identifier of the review.
- product_id: A string identifier of the product being reviewed.
- reviewer_id: A string identifier of the reviewer.
- stars: An integer between 1-5 indicating the number of stars.

- review_body: The text body of the review.
- review_title: The text title of the review
- product_category: String representation of the product's category.

Here is a view of the dataset:



	review_id	product_id	reviewer_id	stars	review_body	review_title	product_category	Sentiment Label
0	en_0944516	product_en_0771017	reviewer_en_0516437	2	Tried to stand it on my ceramic counter - but ...	Doesn't hold	beauty	Negative
1	en_0365489	product_en_0303438	reviewer_en_0993746	5	Received my order quickly! Good deal on this o...	Good deal on this order	drugstore	Positive
2	en_0953574	product_en_0119948	reviewer_en_0766383	3	A little disappointing. Not very bright and th...	Not very bright.	home_improvement	Positive
3	en_0571603	product_en_0683693	reviewer_en_0074314	4	Good product and looks good on the wall.	Great space saver	office_product	Positive
4	en_0335241	product_en_0296351	reviewer_en_0740514	2	The chain within two weeks was worn off and ru...	Disatisfied With Product	pet_products	Negative

Fig 4.1-a : Sentiment Labelled Dataset for Sentiment Analysis

4.2 REVIEW PREPROCESSING

In preparing our Amazon reviews dataset accurately for sentiment analysis, a thoughtful and systematic preprocessing function has been implemented. These steps ensure that the text data is refined and conducive to extracting meaningful insights. The intricacies of each step are detailed below:

1. Lowercasing:
Objective: To Achieve uniformity in text by converting all characters to lowercase.
Importance: It Ensures consistency in the analysis, preventing variations in casing from affecting sentiment categorization.

2. Alphanumeric Characters:
Objective: Remove non-alphanumeric characters, retaining only letters and numbers.
Importance: It Eliminates noise and irrelevant symbols, allowing the sentiment analysis to focus on meaningful words and numerical values.

3. URL Removal:
Objective: To Eliminate URLs present in the text data.
Importance: It Prevents the inclusion of web addresses in the analysis, ensuring that sentiment is derived solely from the review content.

4. Plus Symbol Removal:
Objective: To Remove plus symbols from the text.
Importance: It Maintains consistency and readability, preventing plus symbols from influencing sentiment classification.

5. Punctuation Removal:
Objective: To Strip away punctuation marks from the text.
Importance: It Isolates words and aids in identifying sentiment-bearing terms without interference from punctuation.

6. Newline Removal:
Objective: To Eliminate newline characters.
Importance: It Ensures a seamless text processing experience, avoiding disruptions caused by newline characters.

7. Whitespace Reduction:
Objective: To Remove extra whitespace from the text.
Importance: It Maintains clean and consistent formatting, enhancing the accuracy of subsequent analyses.

8. Stopword Removal:
Objective: To Exclude common English stopwords.
Importance: It Emphasizes content-carrying words, reducing noise and enhancing the significance of words contributing to sentiment.

9. Stemming:
Objective: To Reduce words to their root form using the Snowball Stemmer.
Importance: It Facilitates semantic analysis by capturing the core meaning of words, reducing variations and improving sentiment categorization.

By applying above preprocessing steps, our approach ensures that the resulting text data is refined, standardized, and optimized for extracting meaningful sentiment insights from the diverse spectrum of Amazon reviews.

4.3 VECTORIZATION

Vectorization is a process in natural language processing (NLP) and machine learning where textual or categorical data is converted into numerical vectors. This transformation is essential because many machine learning algorithms, including those used for sentiment analysis, text classification, and clustering, require numerical input.
To utilize the power of machine learning models, an essential step is the conversion of textual data into a numerical format. In this project, we have employed TF-IDF (Term Frequency-Inverse Document Frequency) vectorization, a widely used technique for representing text.

4.3.1 TF-IDF VECTORIZATION

TF-IDF, stands for Term Frequency-Inverse Document Frequency, is a numerical statistic that reflects the importance of a word in a document relative to its occurrence across multiple documents. The TF-IDF Vectorizer from the Scikit-learn library facilitates this process. Here's how TF-IDF calculates the importance of a term in a specific document:

1. Term Frequency (TF):
Measures how often a term appears in a specific document (review in this case). If a term appears many times in a document, it is likely to be important for understanding the content of that document.

2. Inverse Document Frequency (IDF):
Measures the rarity of a term across all documents. If a term is rare and appears in very few documents, it is likely to be more informative and important.
The product of TF and IDF gives the TF-IDF score for a term in a particular review. Terms with higher TF-IDF scores are considered more important for understanding the content of the document.

Here are the steps for vectorization:

Initialization:
The TF-IDF Vectorizer is initialized, allowing customization of parameters. In this instance, the maximum number of features is set to 1000.

Transformation:
The fit_transform method is applied to the specified text column in the dataset (data[text_column]). This step transforms the textual data into TF-IDF features.

DataFrame Creation:
The resulting TF-IDF features are converted into a DataFrame (tfidf_df), where each column and row corresponds to a unique feature and reviews, respectively.

4.4 FEATURE ENGINEERING

Feature engineering involves transforming and creating new features from the existing data to enhance the performance of machine learning models. In the project, a crucial aspect of feature engineering is the encoding of labels, making the target variable compatible with machine learning algorithms.

4.4.1 LABEL ENCODING

To utilize the power of machine learning models, an essential step is the conversion of textual data into a numerical format. In the context of sentiment analysis, the labels often represent the sentiment categories (e.g., positive, negative). The label encoding function performs the following steps:

Initialization:
A Label Encoder is initialized to transform categorical labels into numerical representations.

Label Transformation:
The fit_transform method is applied to the specified label column in the dataset (data[label_column]). This step assigns a unique numerical label to each category.

Label Mapping:
A dictionary, label_mapping, is created to establish a correspondence between the original categorical labels and their encoded numerical representations. This mapping can be useful for interpreting model outputs.

Return:
The encoded labels (encoded_labels) and the label mapping are returned, providing the transformed target variable and a reference for interpretation.

4.5 LONG SHORT TERM MEMORY (LSTM)

4.5.1 INTRODUCTION

In the realm of artificial intelligence and machine learning, the ability to comprehend and analyze sequential data is paramount. Sequential data, characterized by its ordered nature and dependency on previous elements, is ubiquitous in various domains, including natural language processing, time series analysis, speech recognition, and more. Traditional neural network architectures, while effective for many tasks, struggle to effectively model and capture long-range dependencies in sequential data due to the vanishing gradient problem.

The advent of recurrent neural networks (RNNs) marked a significant advancement in the field, offering a framework for processing sequential data by maintaining a hidden state that evolves over time. However, traditional RNNs suffer from limitations such as difficulty in retaining long-term information and the vanishing gradient problem, which arises when gradients diminish exponentially as they propagate backward through time.

In response to these challenges, Long Short-Term Memory (LSTM) networks were introduced as a specialized variant of RNNs, designed to address the shortcomings of traditional architectures. Developed by Hochreiter and Schmidhuber in 1997, LSTM networks revolutionized the field of sequential modeling by introducing memory cells and gating mechanisms, enabling them to capture and remember long-term dependencies in sequential data.

Explanation of General Architecture of LSTM:

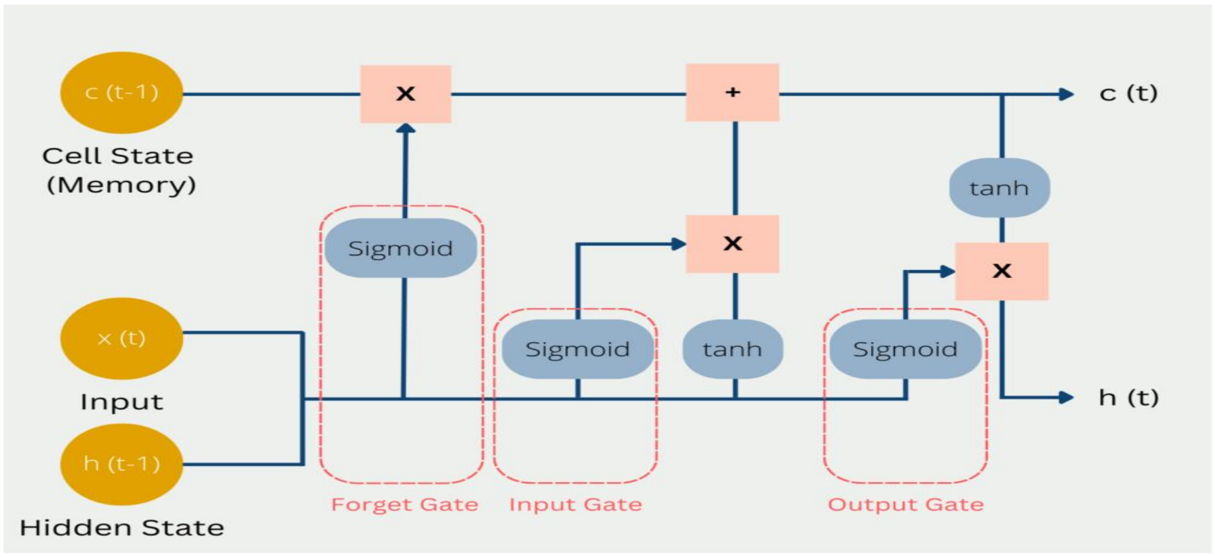


Fig 4.5.1-a : General Architecture of LSTM

In the above diagram,

- **(c_{t-1}) (Cell State at time (t-1)):** This represents the memory cell state at the previous time step, \(t-1 \). It contains information from earlier time steps and serves as the long-term memory of the LSTM unit.

- **(x_t) (Input at time (t)):** This is the input to the LSTM unit at the current time step, \(t). It typically represents the features or data point being processed at the current time step.

- **Forget Gate:** This gate takes the concatenation of the previous hidden state (h_{t-1}) and the current input (x_t) as input. It applies a sigmoid activation function to the input to produce a vector of values between 0 and 1. These values determine how much of each element in the cell state (c_{t-1}) should be retained (1) or forgotten (0).

- **Input Gate:** The input gate also takes the concatenation of the previous hidden state (h_{t-1}) and the current input (x_t) as input. It consists of two parts:

A sigmoid activation function that decides which new information is important to add to the cell state.

A tanh activation function that creates a vector of new candidate values (tilde{c}_t) that could be added to the cell state.

- **(c_t) (Updated Cell State):** This represents the updated cell state at time (t). It is calculated by combining the forget gate's decision about which information to discard from the previous cell state (c_{t-1}) and the input gate's decision about which new information to add to the cell state.

- **Output Gate:** The output gate takes the concatenation of the previous hidden state (h_{t-1}) and the current input (x_t) as input. It consists of two parts:

  A sigmoid activation function that determines which information from the cell state should be output to the next layer.

  A tanh activation function that generates the output based on the cell state, which is then multiplied by the output gate's decision.

- **(h_t) (Hidden State at time (t)):** This represents the output or hidden state of the LSTM unit at time \(t \). It is calculated by applying the output gate's decision to the cell state, after passing through a tanh activation function.

At the heart of the LSTM architecture lies the memory cell, a fundamental building block that distinguishes it from traditional RNNs. The memory cell serves as a storage unit, capable of retaining information over extended periods, thus alleviating the vanishing gradient problem and facilitating the capture of long-range dependencies.

The memory cell is augmented by three gating mechanisms: the input gate, forget gate, and output gate, each responsible for regulating the flow of information through the cell. These gates, implemented as sigmoid and tanh activation functions, allow the LSTM network to selectively retain or discard information based on its relevance to the current context.

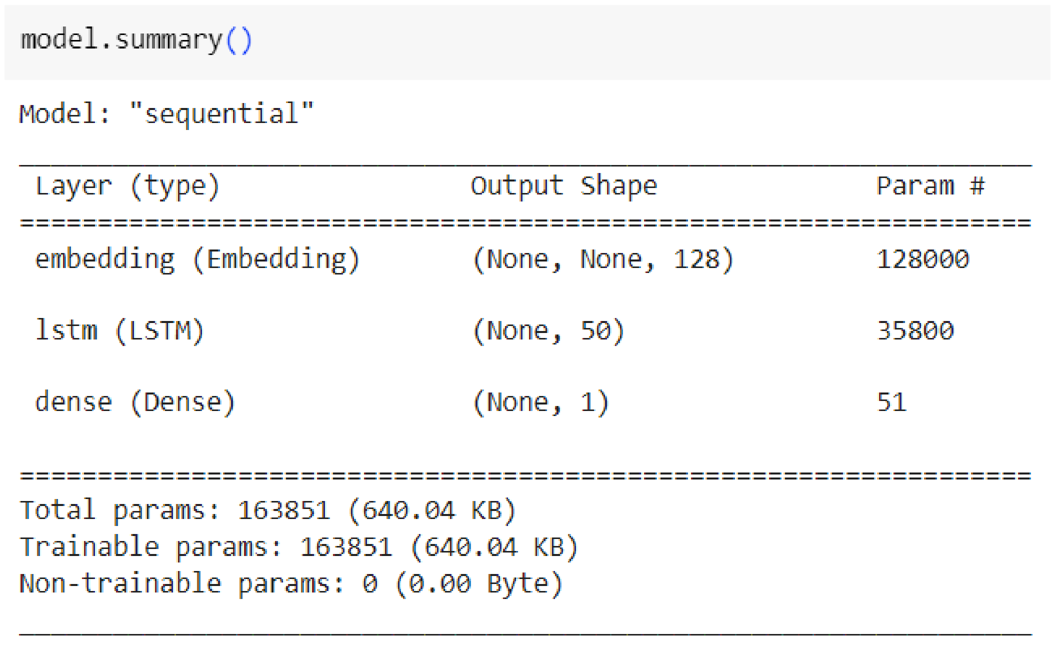
- Forget Gate: The forget gate determines which information from the previous cell state should be discarded or forgotten. It takes as input the concatenation of the previous hidden state and the current input and outputs a vector of values between 0 and 1, representing the degree to which each element of the cell state should be retained.

- Input Gate: The input gate controls the flow of new information into the cell state. It decides which new information is relevant to the current context and should be added to the cell state. Similar to the forget gate, it takes as input the previous hidden state and the current input and produces a vector of values between 0 and 1.

- Output Gate: The output gate determines which information from the cell state should be output to the next layer or timestep. It regulates the flow of information from the cell state to the hidden state, allowing the LSTM network to selectively expose relevant information while suppressing irrelevant details.

By incorporating memory cells and gating mechanisms, LSTM networks excel at capturing and retaining long-term dependencies in sequential data. This makes them particularly well-suited for a wide range of tasks, including natural language processing, time series forecasting, speech recognition, and sentiment analysis, among others. Their versatility and effectiveness have cemented their position as a cornerstone in the field of sequential modelling, driving advancements in artificial intelligence and machine learning.

4.5.2 MODEL ARCHITECTURE



```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 128)         128000

 lstm (LSTM)                 (None, 50)                35800

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 163851 (640.04 KB)
Trainable params: 163851 (640.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig 4.5.2-a : LSTM Model Architecture

1. **Embedding Layer:**
The embedding layer is the first layer of the LSTM model. It is responsible for converting integer-encoded input data (word indices) into dense vector representations. Each word in the vocabulary is mapped to a unique dense vector of fixed size, called an embedding. These embeddings capture semantic similarities between words and are learned during the training process. The embedding layer allows the model to effectively process textual data by representing words in a continuous vector space.

2. **LSTM Layer:**
The LSTM (Long Short-Term Memory) layer is the core component of the LSTM model. It contains LSTM units, which are specialized cells capable of retaining information over long sequences. The LSTM units consist of three gates - input gate, forget gate, and output gate - and a memory cell, enabling them to selectively retain or discard information based on its relevance to the current context. The LSTM layer processes input sequences, captures temporal dependencies, and generates output representations with enhanced ability to capture long-term dependencies compared to traditional RNNs.

3. **Dense Layer:**
The dense layer is the final layer of the LSTM model. It consists of one or more neurons and applies a linear transformation to the input data, followed by an activation function (in this case, a sigmoid function). In the context of sentiment analysis, the dense layer acts as a classifier, predicting the sentiment label (positive or negative) based on the learned features extracted by the preceding layers. The output of the dense layer represents the probability of each class, and the class with the highest probability is selected as the predicted sentiment label.

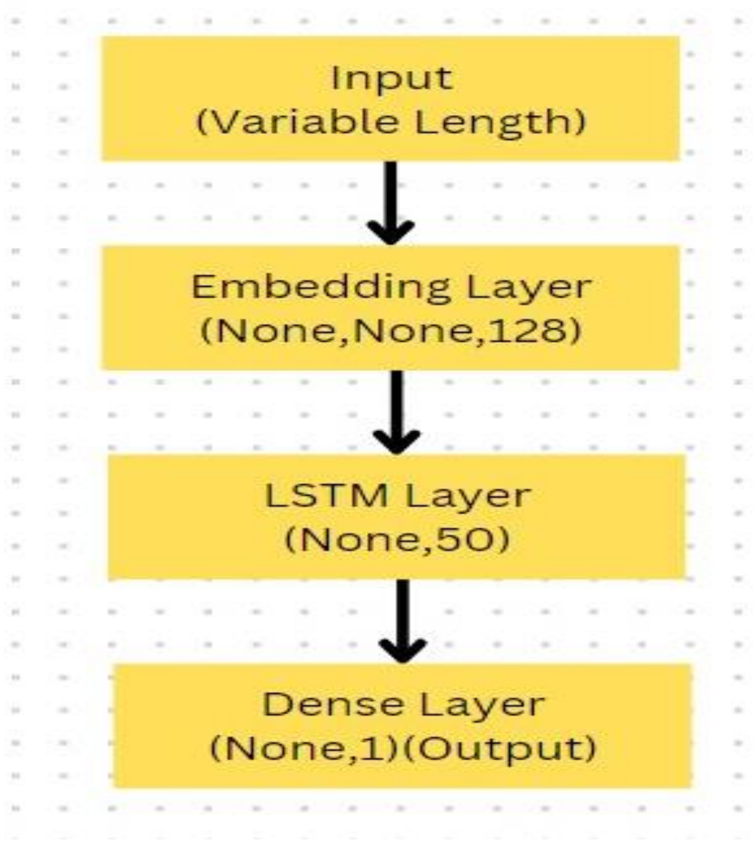


Fig 4.5.2-b : Block Diagram of LSTM Model Arcchitecture

In conclusion, we can say that the embedding layer converts input text data into dense vector representations, the LSTM layer captures sequential patterns and long-term dependencies, and the dense layer performs classification based on the learned features, producing the final sentiment predictions. Together, these layers constitute the architecture of the LSTM model for sentiment analysis.

4.5.3 MODEL EVALUATION

In order to get the most accurate results from LSTM model, we have trained it for different values for the following varibles during the model training phase, they are as follows:

The terms "Number of Epochs" and "Test Size" are commonly used in machine learning, specifically in the context of training and evaluating models. Let's define each term:

1. **Number of Epochs (epochs):**
    - An epoch refers to one complete pass through the entire training dataset during the training of a model. When training a model, the dataset is divided into batches, and the model's parameters are updated based on the gradients computed on each batch.
    - Choosing the appropriate number of epochs is essential for achieving optimal performance without overfitting or underfitting the model. It involves monitoring the model's performance on a separate validation dataset and stopping training when the performance starts to degrade.

2. **Test Size (test_size):**
    - Test size refers to the proportion of the dataset that is reserved for evaluation or testing purposes after training the model.
    - When splitting a dataset into training and testing sets, it is common practice to allocate a certain percentage (test size) of the data to the testing set. The remaining data is used for training the model.
    - The test set is used to evaluate the performance of the trained model on unseen data. It helps assess how well the model generalizes to new data and provides an estimate of its real-world performance.

We got the following findings as a result for the same:

test_size	**epochs**	**Accuracy**
0.2	5	79.52
0.2	7	78.06
0.2	10	76.00

0.2	18	74.98
0.2	24	75.02
0.3	5	79.38
0.4	5	79.26

Table 4.5.3 : Accuracy Score Variation at various epoch values

Visualization of Model Evaluation Phase:

- At epoch=5 and test_size=0.2

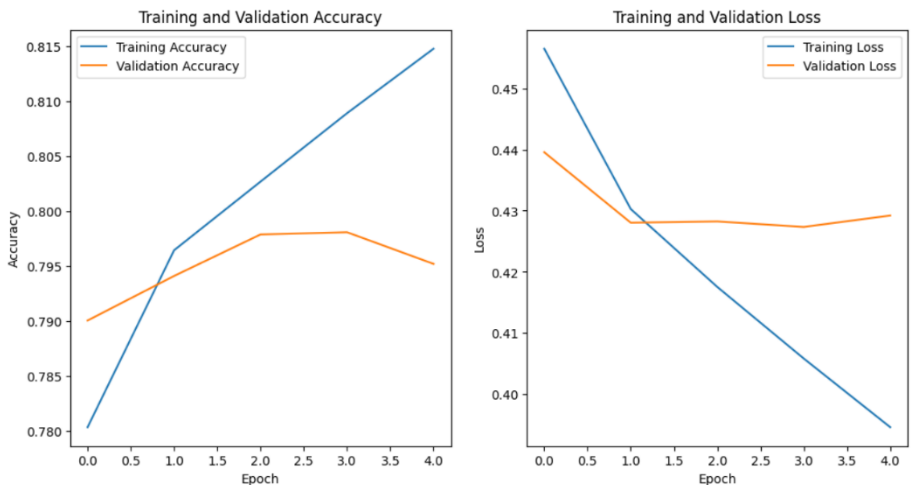


Fig 4.5.3-a : Train Accuracy Vs. Val Accuracy and Train Loss Vs. Val Loss

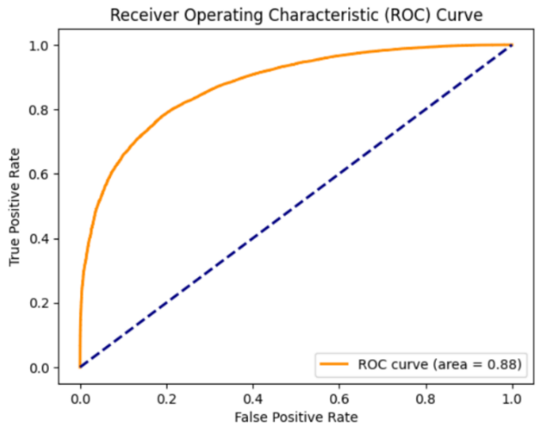


Fig 4.5.3-b : ROC Curve

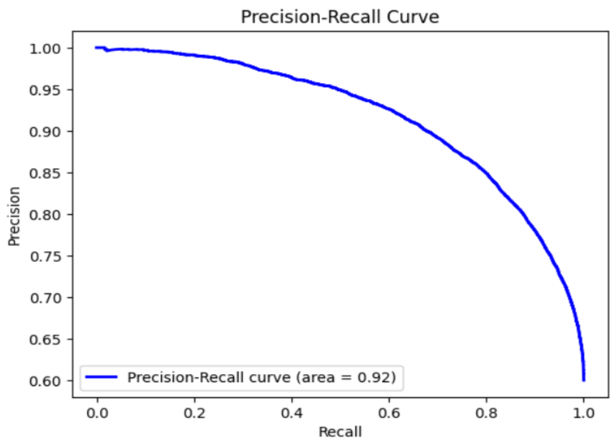


Fig 4.5.3-c : Precision-Recall Curve

- At epoch=7 and test_size=0.2

23

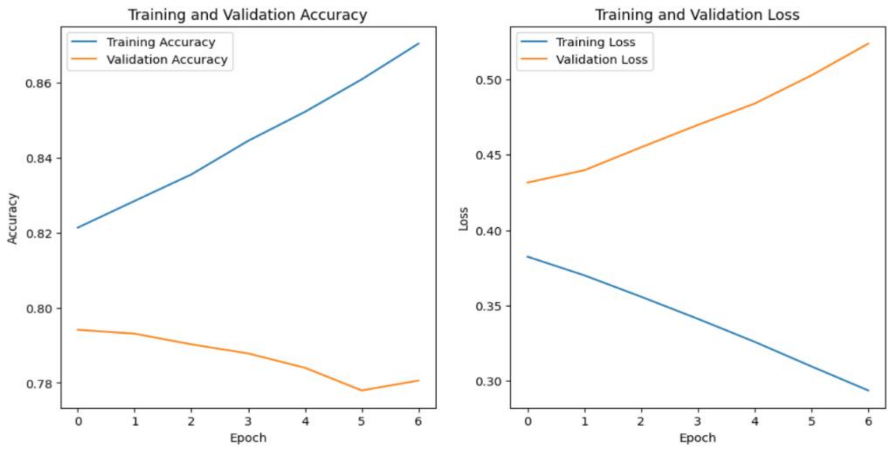


Fig 4.5.3-d : Train Accuracy Vs. Val Accuracy and Train Loss Vs. Val Loss

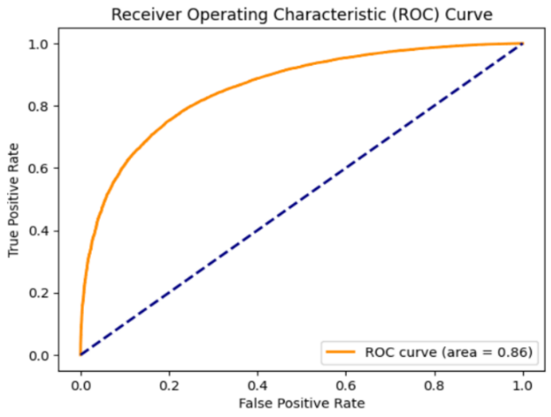


Fig 4.5.3-e : ROC Curve

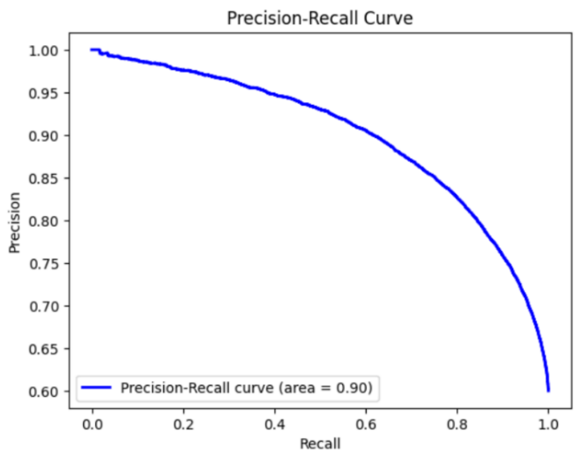


Fig 4.5.3-f : Precision-Recall Curve

- At epoch=10 and test_size=0.2

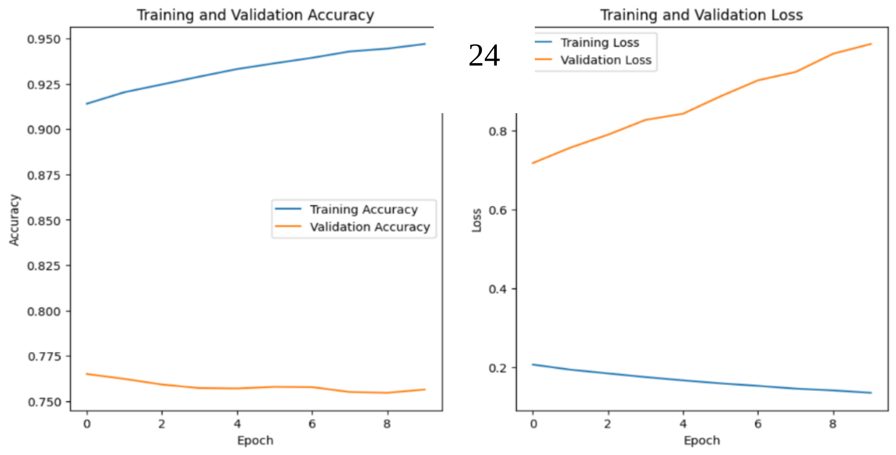


Fig 4.5.3-g : Train Accuracy Vs. Val Accuracy and Train Loss Vs. Val Loss

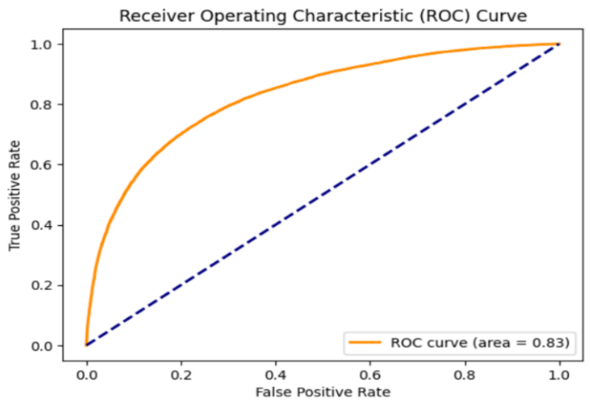


Fig 4.5.3-h : ROC Curve

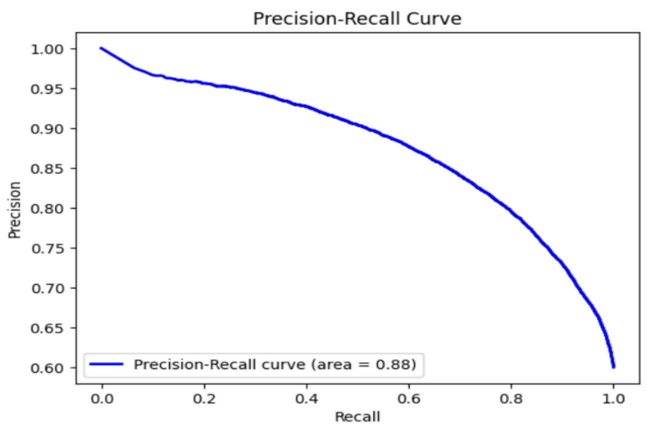


Fig 4.5.3-i : Precision-Recall Curve

- At epoch=18 and test_size=0.2

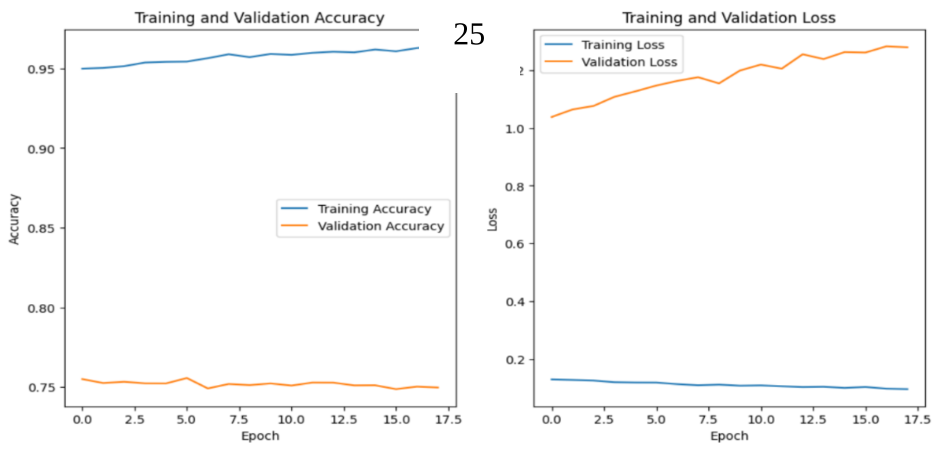


Fig 4.5.3-j : Train Accuracy Vs. Val Accuracy and Train Loss Vs. Val Loss

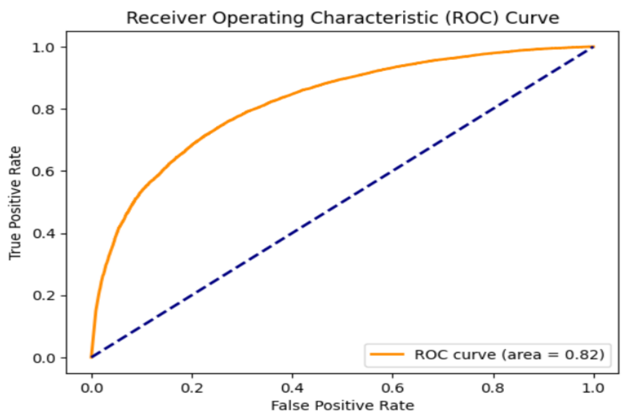


Fig 4.5.3-k : ROC Curve

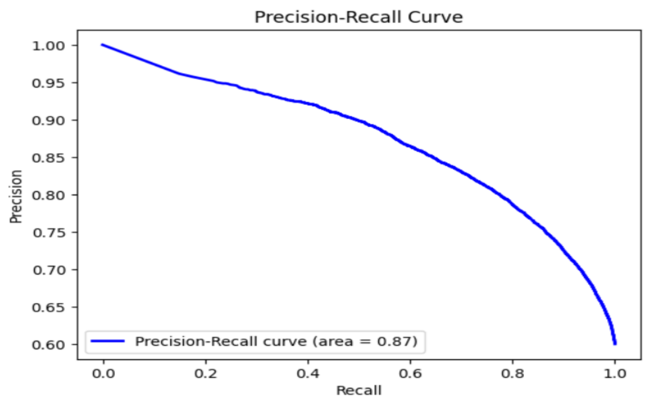


Fig 4.5.3-l : Precision-Recall Curve

- At epoch=24 and test_size=0.2

26

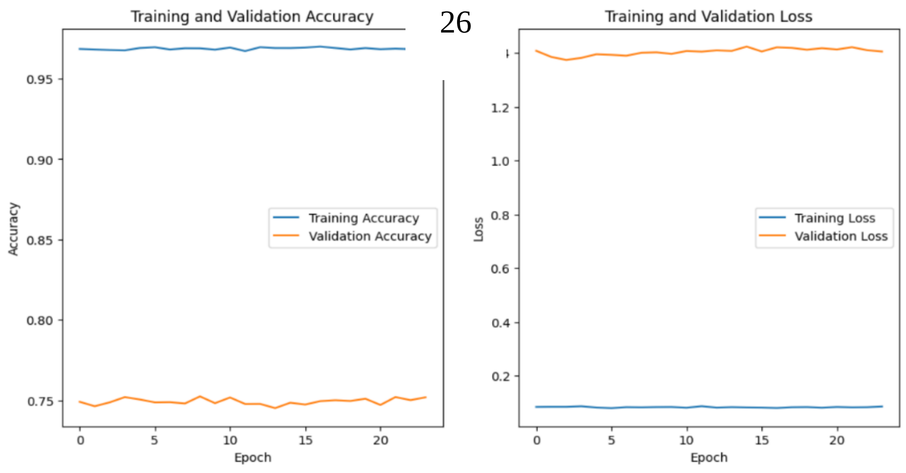


Fig 4.5.3-m : Train Accuracy Vs. Val Accuracy and Train Loss Vs. Val Loss

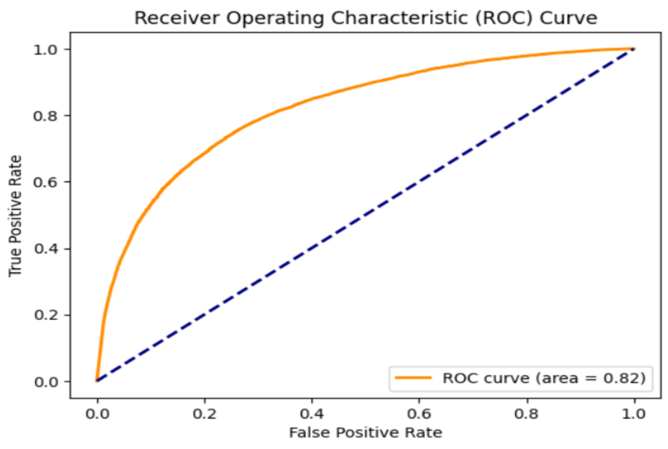


Fig 4.5.3-n : ROC Curve

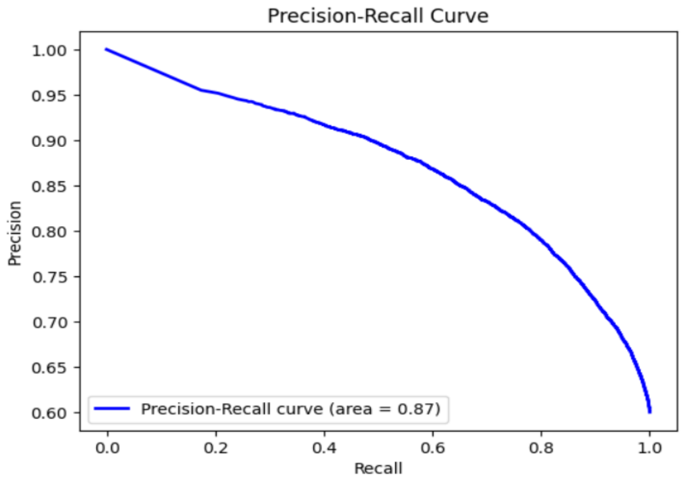


Fig 4.5.3-o : Precision-Recall Curve

4.6 BIDIRECTIONAL ENCODER REPRESENTATION FROM TRANSFORMER (BERT)

4.6.1 INTRODUCTION

BERT stands for Bidirectional Encoded Representation of Transformer. It is an open-source machine learning framework which is used for natural language processing task such as sentiment analysis. It is called bidirectional because it processes the data in both directions which makes it better for contextual embedding. In our proposed model, we have used the Bert-base-cased. The architecture of this model is inspired from transformer, however unlike transformer it only includes the encoder and not the decoder. In this project, we explore the use of BERT model for sentiment analysis and developed a web interface which would aid the business owner to understand the sentiments of the customer represented through their reviews.

Bert is a pre-trained model which is fine tuned using our review dataset. The Bert base has 12 layers of encoder which is used to output a fixed length of vector. The model contains multiple layers that are organized in a defined order. Each layer in the BERT model is responsible for extraction of key features which are passed to the successive layer for further processing. At the tail of model, a sentiment classifier which processes the vector out of the previous layer and classifies it on the basis of the probability that whether the feedback is positive or negative.

Considering the intricacies of the suggested model, we have used the following layers to build it:

1. Input Embedding Layer
2. Multi-Head Self-Attention Layer
3. Layer Normalization
4. Feedforward Neural Network
5. Output Layer
6. Optimizer

7. Loss Function
8. Training Loop

Input Embedding Layer

This is the initial layer of our model which is responsible for converting the tokens into dense vector also called as embedding that helps in understanding the semantic of the words. Tokens are the smallest unit of a text and it can be a word, sentence or paragraph. The different types of tokens used in our model are cls token, sep token, pad token and unk token.
The input text is tokenized using the named tokens, which converts words into token ID. Further these token IDs are converted into embeddings using the embedding matrix. Each token ID is represented using high dimensional vectors representation which helps the model to capture the semantic meaning

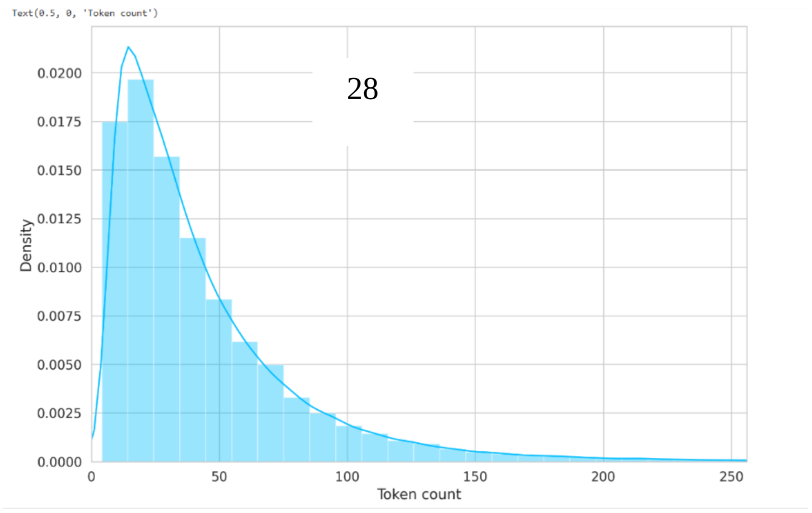


Fig 4.6.1-a : Input Embedding Layer

Multi-Head Self-Attention Layer

The main purpose of this layer is to make the model understand the contextual meaning of the words in the sentences. Self-attention mechanism is used to weigh the importance of different words in a sequence and also talks about the relationship between different words. It assigns an attention score to each word determining how much focus should be put on each word when encoding the input sequence.

Multi-head attention applies the self-attention mechanism multiple times in parallel, each with a different set of learned parameters. Each word is assigned a key vector, query vector and value vector. These vectors are used to calculate the attention score of each word. The scaled dot product of key vector and query vector is called the attention score. To prevent the gradient from becoming too large, the dot product is scaled by the square root of key vector.

The attention score is passed through a SoftMax function to convert them into weights that sum up to 1. The weights are responsible for determining how much each word contribute to the final representation of each word. The value vector is multiplied with the calculated vector. In final step, it sums ups the weighted value vector which provides the self-attention output for a given word.

Layer Normalization

It is a layer which is responsible for normalizing the output of self-attention layer. In BERT model, layer normalization is applied after each sub layer because it stabilizes the training process and ensures that activation in each layer has consistent distribution.
It normalizes the activations of a layer by subtracting the mean and dividing by the standard deviation across the feature dimension. Mathematically, for a tensor X with shape (N, feature_dim) , the normalized output Norm(X) is calculated as

$$\mathrm{Norm}(X) = \frac{X - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

where μ is the mean of X, σ is the stand 29 ation of X, and ϵ is a small constant added for numerical stability.

Layer normalization typically includes learnable scale and shift parameters ($\gamma\gamma$ and $\beta\beta$) that allow the model to adaptively scale and shift the normalized activations:

$$\mathrm{LN}(X) = \gamma \cdot \mathrm{Norm}(X) + \beta \mathrm{LN}(X) = \gamma \cdot \mathrm{Norm}(X) + \beta$$

These parameters are learned during training and help the model to better capture the underlying distribution of the data. After normalization, the activations are passed through a residual connection (i.e., the original activations are added back to the normalized activations) to facilitate the flow of gradients during training.

Feedforward Neural Network

Here, feedforward neural network is used as a part of transformer architecture which is the BERT model. It is the final layer for sentiment classification which is responsible for mapping the contextual embedding to sentiment classes. It helps the model to learn more robust and generalized features by randomly deactivating some of the neurons. It is specifically used in the sentiment classifier and includes two layers :

Linear Transformation layer (Fully-Connected Layer) – Once the contextual embedding is done and attention vectors are generated then the output is passed through a fully connected layer (self.out) without an activation function.

Dropout layer – Before passing the output of the fully connected layer to linear layer, it is passed through dropout layer. The function (self.drop) is applied which is a regularization technique that sets fraction of unit values to zero in order to avoid the case of overfitting

Output layer

The input to the output layer is the output of the feed forward layer. The output layer is a fully connected dense layer in which every neuron of the feed forward neural network is connected to the neurons of the output layer. The number of neurons in the output layer corresponds to the number of classes or labels in the sentiment analysis task.

The activation function used in this layer is a softmax function, which converts the raw outputs into probabilities. The formula of softmax function is:

$$\mathrm{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Where: zi is the ith element of the input [illegible]

K is the total number of classes.
e is Euler's number (approximately equal [illegible]328).

Each neuron in the output layer represents a sentiment class like positive or negative, the output of each neuron represents the probability that the output belongs to that class.

LOSS FUNCTION

The loss function used in our model is cross entropy loss function. In sentiment analysis, it measures how well does the model predicts the sentiment label as compared to actual sentiment label for each input text.

Cross-entropy measures the difference between predicted probability distribution output by the model and actual distribution of the target variable. It is defined as:

$$\mathrm{CrossEntropyLoss}(\mathbf{p}, \mathbf{q}) = -\sum_i p_i \log(q_i)$$

Where p is the true probability distribution, q is the predicted probability distribution and pi and qi are the probabilities on the ith class in true and predicted one, respectively.

OPTIMIZER

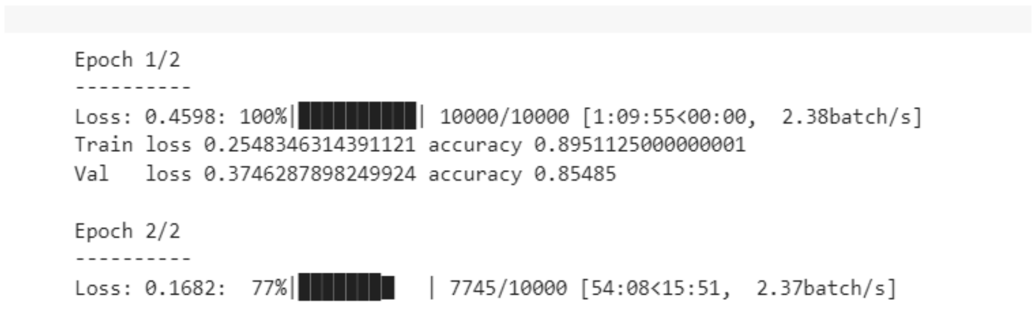
In our proposed model, we have used the AdamW optimizer which is provided by the hugging face. It corrects the weight decay which is a technique employed to prevent overfitting. It adds a regularization term which is square of weights, encouraging the model to use smaller weights. We have also used linear scheduler with no warmups.

In our model, AdamW optimizer is initialized with learning rate of 2/100000 and the correct_bias is set to be False. Bias is introduced by first and second moment estimates and the bias correction is used to correct this bias.

TRAINING LOOP

Training loop can be defined as a code which is used in our model for iteratively updating the parameters based on training data to minimize the loss function. It is initialized with several parameters like number of epoch and best_accuracy which is 1 and 0 respectively in our model.
For each epoch, the loop iterates over the training dataset in a fixed size batch. Here the model is set to training model which allows the layers like dropout to behave differently during training compared to evaluation.

The loop iterates through each batch and gives output like train loss, accuracy, val loss and accuracy.
In our model we have also included the progress bar which tells the progress of loop through the data in percentage.



```
Epoch 1/2
----------
Loss: 0.4598: 100%|██████████| 10000/10000 [1:09:55<00:00,  2.38batch/s]
Train loss 0.2548346314391121 accuracy 0.8951125000000001
Val   loss 0.3746287898249924 accuracy 0.85485

Epoch 2/2
----------
Loss: 0.1682:  77%|███████▋  | 7745/10000 [54:08<15:51,  2.37batch/s]
```

Fig: 4.6.1-b : Train and Val Data logs during Training Phase

4.6.2 MODEL ARCHITECTURE

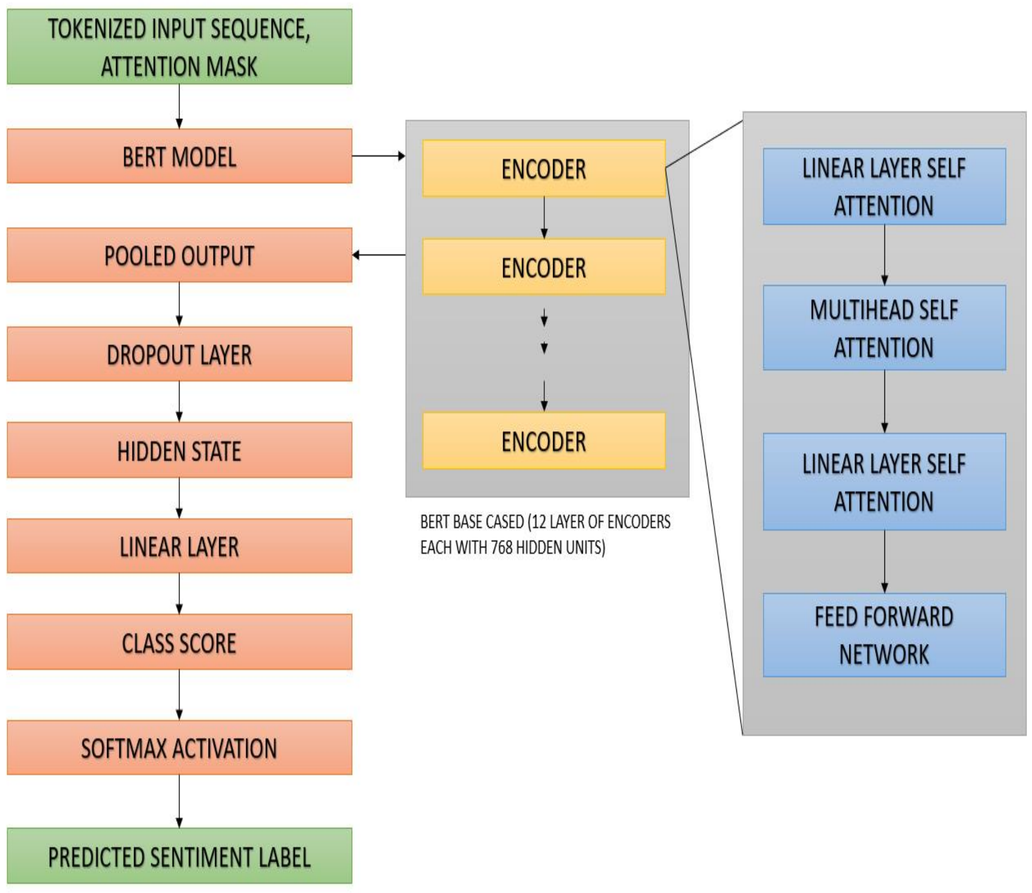

TOKENIZED INPUT SEQUENCE, ATTENTION MASK
BERT MODEL
POOLED OUTPUT
DROPOUT LAYER
HIDDEN STATE
LINEAR LAYER
CLASS SCORE
SOFTMAX ACTIVATION
PREDICTED SENTIMENT LABEL
ENCODER
ENCODER
ENCODER
BERT BASE CASED (12 LAYER OF ENCODERS EACH WITH 768 HIDDEN UNITS)
LINEAR LAYER SELF ATTENTION
MULTIHEAD SELF ATTENTION
LINEAR LAYER SELF ATTENTION
FEED FORWARD NETWORK

Fig 4.6.2-b : Block Diagram BERT Model Architecture

4.7 MODEL INTEGRATION WITH FRONTEND

Integrating the back end with the front end remains a crucial step. This process involves connecting the server-side components (backend), which include models such as LSTM and BERT, with the user interface (frontend) to enable smooth communication and data exchange.

The integration of the models is fundamental in our project “Sentiment Analysis on Customer Feedback”. The objective is to combine the output generated by our best performing model with comparatively higher accuracy than the other, to enhance the way of predicting sentiment label for customer feedback present on E-Commerce platforms for different product across different product categories.

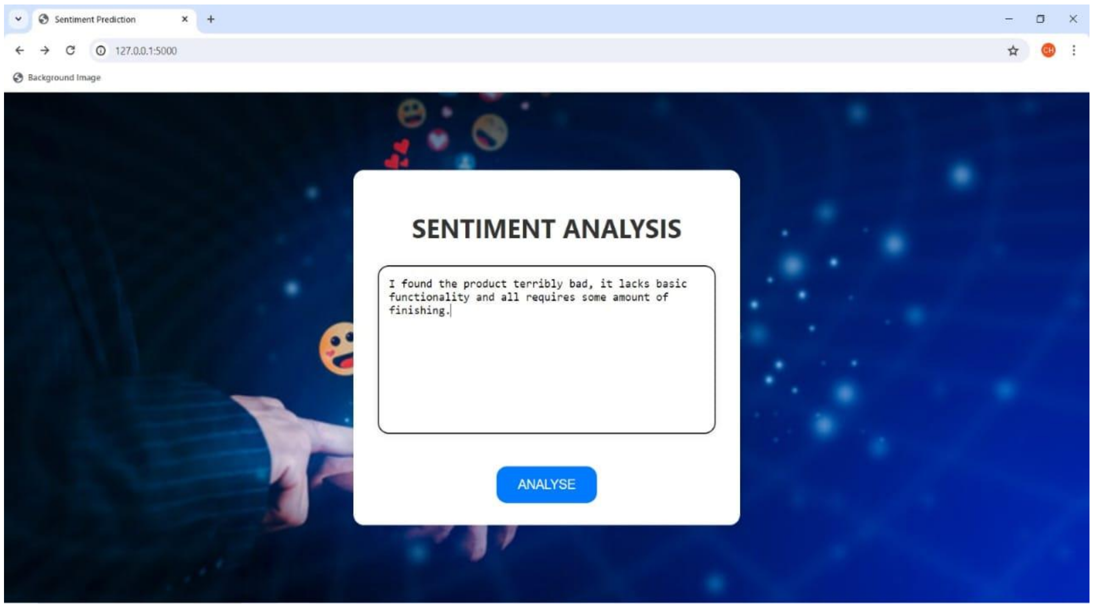
This integration process involves merging the result of the model to produce a comprehensive aggregate result in terms of Sentiment Label and Confidence/ Probability Score for the review entered by user of our web application in the text box provided, indicating the sentiment expressed in the customer feedback.

1. **Model Saving (.h5 extension for HDF5):**
    - After training the LSTM and BERT models for sentiment analysis, it's crucial to save these models for later use to prevent retraining.
    - The models are serialized using the HDF5 format, commonly represented by the .h5 extension, using libraries such as Keras for LSTM and TensorFlow for BERT.
    - Saving the models in HDF5 format ensures that they can be easily loaded and used in other scripts or applications without the need for retraining.
    - For example, the LSTM model trained using Keras can be saved using `model.save('lstm_model.h5')`, and the BERT model trained using TensorFlow `tf.keras.models.save_model(bert_model, 'bert_model.h5')`.

2. **Tokenizer Saving (.pkl file):**
    - Along with saving the models, it's essential to save the tokenizers used for preprocessing text data.
    - Tokenizers are serialized using the pickle module in Python, allowing them to be saved as .pkl files.
    - The tokenizer pickle files contain the vocabulary and configuration necessary for tokenizing new text data consistently.
    - For instance, the tokenizer for LSTM model can be saved using `pickle.dump(tokenizer, open('lstm_tokenizer.pkl', 'wb'))`, and similarly for the BERT model tokenizer.

3. **Flask Integration:**
    - Integrate the Python sentiment analysis script with Flask within app.py python script for model linking, model loading, and inferencing, flask is a micro web framework for Python.
    - Flask handles HTTP requests and routes them to appropriate endpoints defined in the Flask application.
    - Define routes for receiving new customer reviews from the frontend, processing them using the sentiment analysis script, and returning the predicted sentiment labels.
    - Flask provides a lightweight and flexible framework for building web applications, making it suitable for integrating with machine learning models for sentiment analysis.

4. **Front-end Development:**
    - Develop the frontend interface using HTML, CSS, and Jinja templates to create a user-friendly interface for interacting with the sentiment analysis web application.
    - Design the frontend to allow users to input their customer feedback through a form or text input field.
    - Use CSS to style the frontend interface and make it visually appealing and responsive across different devices.
    - Jinja templates are used to dynamically generate HTML content and incorporate Python code into HTML templates, enabling seamless integration with the Flask backend.
    - We have an “Analyze” button which upon entering the customer feedback, need to be clicked and it redirects the user to a page which shows the result.



```
I found the product terribly bad, it lacks basic
functionality and all requires some amount of
finishing.
```

Fig 4.7-a : Web Interface Home Page – Test with Negative Review

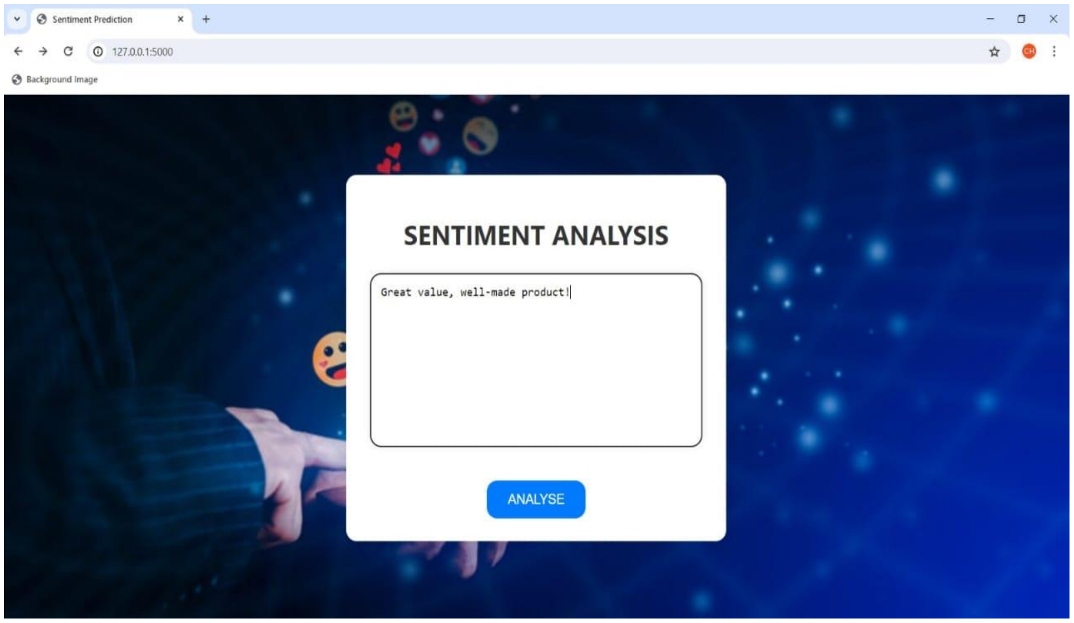


Fig 4.7-b : Web Interface Home Page– Test with Positive Review

5. **Displaying Sentiment Label**
    - After performing sentiment analysis on the customer review entered by user, the Flask backend stores the sentiment label.
    - The frontend interface can display the Sentiment Label alongside Probability Score which shows the probability of the review being either “Positive” or “Negative” providing users with insights into the sentiment expressed by the review.
    - It also has a “Home” button that can directly take us back to the page where the review can be entered.

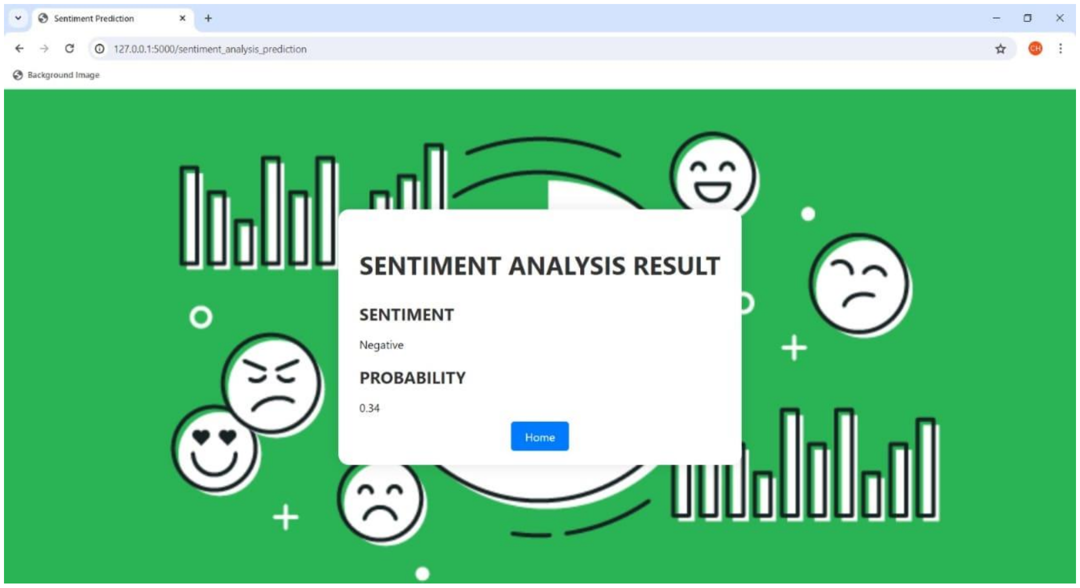


Fig 4.7-c : Web Interface Result Page – For Negative Review as Input

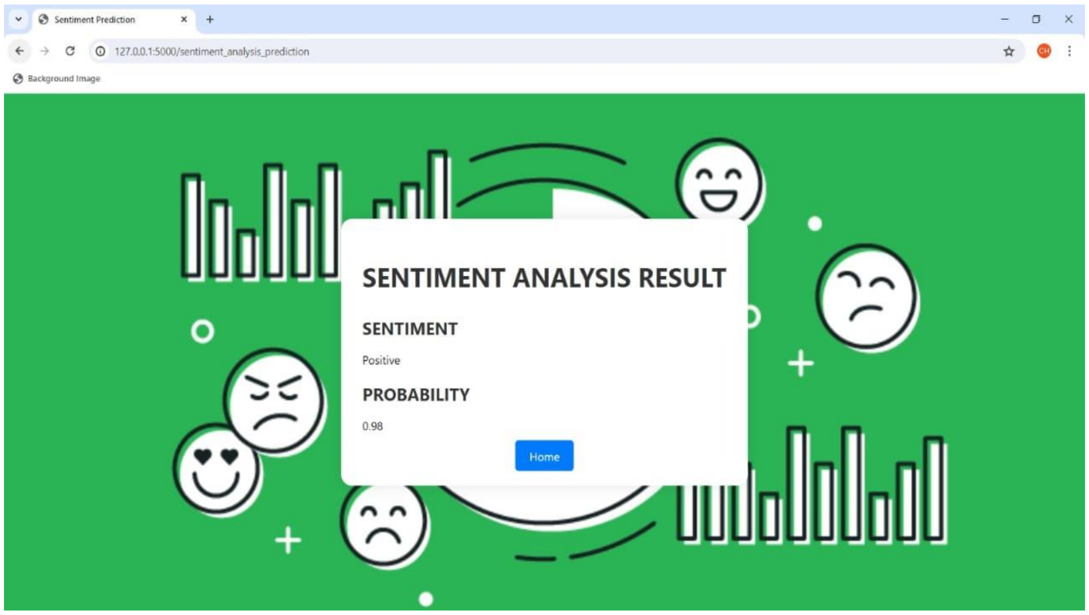


Fig 4.7-d : Web Interface Result Page – For Positive Review as Input

CHAPTER-5

RESULT AND CONCLUSION

5.1 RESULT

For effective probability prediction and robust false-negatives detection we have incorporated 2 Machine Learning based models which work together seamlessly.

The first model incorporated in this project uses advanced deep learning techniques, leveraging LSTM neural networks, renowned for their ability to process sequential data like text. Unlike simpler models, LSTMs excel at capturing long-range dependencies in text, making them ideal for tasks such as sentiment analysis.

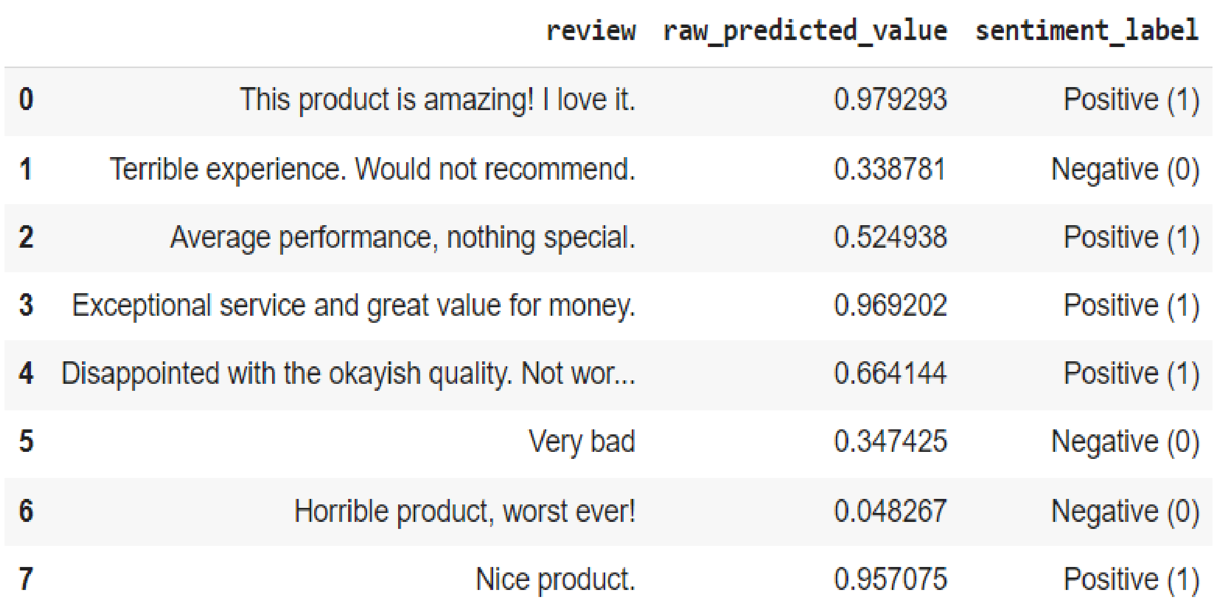
The model has been trained on a dataset of 200000 rows of customer review. It is a labelled dataset containing information of different products, purchases, person id which helps to better understand the data. Due to the computational intensity of LSTM architectures, our training regimen was optimized for efficiency. While constraints on computational resources limited the number of training epochs, the LSTM model demonstrated impressive accuracy in predicting sentiment polarity.

The next model used is BERT which is used for contextual embedding because unlike other models it analyzes the text data in both the direction. It has achieved an accuracy of 80.34% on testing data. While testing it with random data it gives a good accuracy result.

The second model was trained on the same dataset which provides a transformer-based architecture and works bidirectionally for text data. Because of being a very heavy model, only a smaller number of epochs could be run, however it has successfully predicted the positive and negative sentiment.

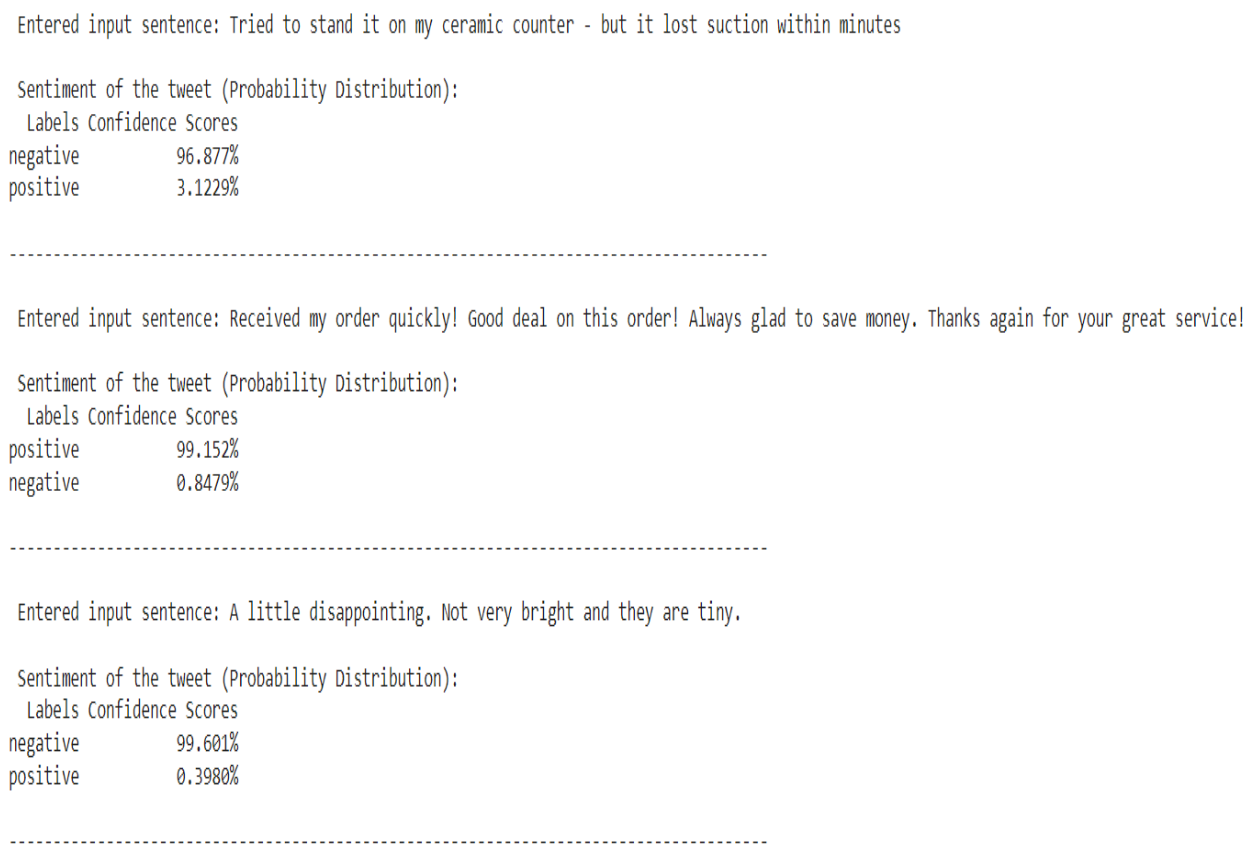
Since, the project implements two models, the overall time to get the prediction for an image is about 5-8 seconds, which depends on the size of the data like the user can put either a single sentence of review or a csv file.

We made this project readily accessible by the means of a user friendly and easy to use web application. It is designed to be minimal, distraction-free, and visually appealing, keeping in view the requirements of business professionals. The web application allows users to interact with the model via a frontend, using which they can upload the csv file and get the result of the prediction.



	review	raw_predicted_value	sentiment_label
0	This product is amazing! I love it.	0.979293	Positive (1)
1	Terrible experience. Would not recommend.	0.338781	Negative (0)
2	Average performance, nothing special.	0.524938	Positive (1)
3	Exceptional service and great value for money.	0.969202	Positive (1)
4	Disappointed with the okayish quality. Not wor...	0.664144	Positive (1)
5	Very bad	0.347425	Negative (0)
6	Horrible product, worst ever!	0.048267	Negative (0)
7	Nice product.	0.957075	Positive (1)

Fig 5.1-a : Output of LSTM



```
 Entered input sentence: Tried to stand it on my ceramic counter - but it lost suction within minutes

 Sentiment of the tweet (Probability Distribution):
  Labels Confidence Scores
negative          96.877%
positive          3.1229%

---------------------------------------------------------------------------------

 Entered input sentence: Received my order quickly! Good deal on this order! Always glad to save money. Thanks again for your great service!

 Sentiment of the tweet (Probability Distribution):
  Labels Confidence Scores
positive          99.152%
negative          0.8479%

---------------------------------------------------------------------------------

 Entered input sentence: A little disappointing. Not very bright and they are tiny.

 Sentiment of the tweet (Probability Distribution):
  Labels Confidence Scores
negative          99.601%
positive          0.3980%

---------------------------------------------------------------------------------
```

Fig 5.1-b : Output of BERT

5.2 CONCLUSION

In conclusion, our thesis has focused on sentiment analysis, utilizing both LSTM (Long Short-Term Memory) and BERT (Bidirectional Encoder Representations from Transformers) models. These models have shown remarkable performance in capturing the contextual and semantic information present in textual data, enabling us to accurately classify sentiments in various text inputs.

The LSTM model, with its ability to remember long-term dependencies, proved effective in capturing sequential patterns in text data. It demonstrated strong performance in sentiment classification tasks, especially with shorter texts where context is key.

On the other hand, the BERT model, with its bidirectional attention mechanism, surpassed LSTM in understanding the context and semantics of longer texts. BERT's pre-trained embeddings and fine-tuning capabilities allowed us to achieve state-of-the-art results in sentiment analysis tasks, particularly for longer texts and datasets with complex sentiments.

Overall, our experiments and analyses have highlighted the strengths and weaknesses of both LSTM and BERT models in sentiment analysis. While LSTM excels in capturing sequential patterns and short-term dependencies, BERT surpasses it in understanding context and semantics, especially in longer texts. The choice of model depends on the specific requirements of the sentiment analysis task, such as the length of texts, complexity of sentiments, and available computational resources.

Our research contributes to the growing body of knowledge in sentiment analysis and provides valuable insights for practitioners and researchers seeking to utilize LSTM and BERT models for sentiment analysis tasks.

CHAPTER-6

FUTURE SCOPE

The field of sentiment analysis is continuously evolving, and there are several avenues for future research and improvement in this area, particularly concerning LSTM and BERT models:

1. Model Enhancements: Researchers can explore ways to further enhance LSTM and BERT models for sentiment analysis. This could involve experimenting with different architectures, hyperparameters, and training strategies to improve performance and efficiency.

2. Multimodal Sentiment Analysis: Integrating LSTM and BERT models with other modalities, such as images, videos, and audio, could lead to more comprehensive and accurate sentiment analysis systems, especially in multimedia content analysis.

3. Fine-Tuning Strategies: Fine-tuning BERT models on domain-specific datasets can improve their performance on specialized sentiment analysis tasks. Research in this area can explore effective fine-tuning strategies for different domains and languages.

4. Interpretable Models: Developing methods to make LSTM and BERT models more interpretable can help users understand the reasoning behind model predictions, increasing trust and usability.

5. Efficiency Improvements: Finding ways to make LSTM and BERT models more computationally efficient can expand their applicability to resource-constrained environments, such as mobile devices and edge devices.

6. Domain Adaptation: Investigating methods for adapting LSTM and BERT models to new domains with limited labeled data can improve their usability in real-world applications.

7. Cross-Lingual Sentiment Analysis: Expanding LSTM and BERT models to handle multiple languages can enable sentiment analysis in multilingual contexts, benefiting global applications.

By addressing these areas of improvement and exploring new avenues, the sentiment analysis models can be further refined and advanced, leading to more accurate and efficient prediction of customer reviews, thereby helping in improving the business profits. Ultimately, this can contribute to the development of automated systems that aid business growth and helps the industry manufacture better good for user satisfaction

CHAPTER-7

REFRENCES

[1] Yang, Li, et al. "Sentiment analysis for E-commerce product reviews in Chinese based on sentiment lexicon and deep learning." *IEEE access* 8 (2020): 23522-23530.

[2] Liu, Yi, et al. "Sentiment analysis for e-commerce product reviews by deep learning model of Bert-BiGRU-Softmax." *Mathematical Biosciences and Engineering* 17.6 (2020): 7819-7837.

[3] Pandey, Prashant, and Nitasha Soni. "Sentiment analysis on customer feedback data: Amazon product reviews." *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)*. IEEE, 2019.

[4] Moghaddam, Samaneh. "Beyond sentiment analysis: mining defects and improvements from customer feedback." *Advances in Information Retrieval: 37th European Conference on IR Research, ECIR 2015, Vienna, Austria, March 29-April 2, 2015. Proceedings 37*. Springer International Publishing, 2015.

[5] Kim, Rae Yule. "Using online reviews for customer sentiment analysis." *IEEE Engineering Management Review* 49.4 (2021): 162-168.