A PROJECT REPORT ON

SENTIMENT ANALYSIS AND OPINION MINING USING LARGE LANGUAGE MODEL

Submitted in the fulfillment of the requirement for the

award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By: -

Kashish Gupta (2021021140)

Pragati Yadav (2021021152)

Group-23
---Under the Supervision of---
Prof. P. K. Singh

Department of Computer Science and Engineering

Madan Mohan Malaviya University of Technology, Gorakhpur, (U.P.) – 273010

December, 2024

CERTIFICATE

This is to certify that Kashish Gupta and Pragati Yadav have successfully completed the project titled **"Sentiment Analysis and Opinion Mining using Large Language Model"** as part of the requirements for the Bachelor of Technology degree in Computer Science and Engineering at Madan Mohan Malaviya University of Technology (formerly Madan Mohan Malaviya Engineering College), Gorakhpur (UP). The project was carried out under my supervision and guidance.

The findings and content of this report represent the independent efforts and research of the aforementioned students. Furthermore, this work has not been submitted, in whole or in part, for the award of any other degree or qualification to the candidates or any other individual.

Prof. P. K. Singh
Professor
Department of CSE
MMMUT Gorakhpur

Date:

CANDIDATE’S DECLARATION

I confirm that this contribution is the result of my own effort and original ideas. Wherever I have included ideas or words from other sources, I have properly credited and referenced them. I further confirm that I have upheld the principles of academic honesty and integrity throughout this work, ensuring that no idea, data, fact, or source has been misrepresented or falsified. I acknowledge that any breach of these principles may lead to disciplinary action by the University and could result in legal consequences from improperly cited sources or those from whom necessary permissions were not obtained.

Kashish Gupta (2021021140)
Pragati Yadav (2021021152)

Computer Science & Engineering
Madan Mohan Malaviya University of Technology, Gorakhpur

<u>APPROVAL SHEET</u>

This project report entitled **“Sentiment Analysis and Opinion Mining using Large Language Model”** by **Kashish Gupta and Pragati Yadav** is approved for the degree of Bachelor of Technology in Computer Science and Engineering.

Examiner

____

____

____

Supervisor
Prof. P. K. Singh

Head of Department (CSED)
Dr. Udai Shanker

Dean, Research & Development, or
Other Dean/Professor to be nominated
by the Vice Chancellor in his absence.

.

Date: ___________
Place: Gorakhpur

ACKNOWLEDGEMENT

At the outset of this report, we wish to express our sincere gratitude and deep appreciation to Prof. P. K. Singh from the Department of Computer Science and Engineering, MMMUT, Gorakhpur, for his unwavering support and guidance throughout the course of this project. His insightful intellectual direction, constructive feedback, and valuable suggestions have been instrumental in shaping our work.
We would also want to express our heartfelt gratitude to Prof. P. K. Singh for his kind support, important insights, and expert direction, which were critical in successfully completing the many stages of this endeavor.
Finally, we are profoundly thankful to the Almighty, our parents, and our classmates for their constant encouragement and motivation, without which this endeavor would not have been possible.

Kashish Gupta (2021021140)
Pragati Yadav (2021021152)

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Increase in textual data has led to an increase in demand of tools that can derive meaningful insights from data. Sentiment analysis, is an application of Natural Language Processing (NLP), that identifies emotional tones within text, from user reviews and social media posts to formal documents. This project introduces a sentiment analysis model based on BERT, a model known for its superior contextual understanding.

Traditional approaches of sentiment analysis, such as lexicon-based methods or traditional machine learning models, have significant limitations. These methods struggle to understand the context of natural language, particularly in sentences where context changes the meaning of words. For example, a sentence like *"The movie wasn't bad"* might be misinterpreted as negative, even though it was a positive sentiment. BERT is known to resolve such issues by analyzing text in both directions, that captures the full context of each word in relation to its surroundings.

This project fine tunes the BERT model for sentiment classification tasks. It involves modifying the pre-trained model's weights on a labeled dataset, enabling the model to classify the text as positive, negative, or neutral. This increases the accuracy and adaptability of the model for Sentiment analysis. The model`s architecture consists of two primary components: the user interface (frontend) and the machine learning pipeline (backend).
The **frontend** was developed using Streamlit, that provides interactive and seamless experience for the user. It has two running modes: single-text analysis and batch processing.
The **backend** comprises fine-tuning and functions that process input to produce output.

This project addresses both **real-time and batch processing**, ensuring scalability for dataset. Real-time predictions are for fast response on the input, that allows users to analyze inputs individually within seconds. Batch processing, on the other hand, is designed to handle large datasets efficiently, which makes this tool ideal for businesses, researchers, and other stakeholders managing large textual data. The model is interactive that further enhances the usability of the system by providing clear and intuitive results of sentiments.

A significant challenge that was faced during the development of the model was managing the class imbalance in the dataset used for training and finding the appropriate parameters. The dataset, downloaded from Kaggle, contained a imbalance of high number of positive sentiments compared to neutral and negative sentiments. During Fine-tuning optimal parameters - learning rate, epoch, batch size was found by testing on the dataset. Training was limited to a single epoch, as the pre-trained nature of the model already provided a strong foundation for classification task, and also to prevent overfitting.

Objective

In today's world that is overwhelmed with textual data, it is very important to know the meaning behind it. Users have the platform to give feedback about any new product, book, show, etc. To make this feedback be utilized in the best possible way, companies use tools like sentiment analysis to get a gist of public opinion. Therefore, sentiment analysis can also be referred to as opinion mining. This analysis of the emotion behind reviews or any kind of text helps companies to make better decisions that help in their growth and development. We make use of modern technologies specifically natural language processing and large language models that have already been trained on huge language dataset to refine it for our particular project of sentiment analysis. Below are some specific objectives listed:

1. **Achieving Contextual Precision:** Sentiment analysis tools that make use of traditional technologies such as machine learning and other older models often fail and have degraded performance and outcomes when faced with complex sentence structures or confusing sentences and varied sentiments. Sarcasms, idioms and such things can`t be caught easily by those models. For example, a statement like "The show that was released the day before yesterday was surprisingly not bad" might be misclassified due to the ambiguity it possesses. BERT's bidirectional context understanding allows it to analyze such sentences holistically, ensuring unparalleled accuracy. This project seeks to utilize BERT's deep learning capabilities to provide sentiment predictions that account for subtle linguistic nuances, delivering insights that are both reliable and actionable.

2. **Scalability Across Diverse Use Cases:** The dynamic and constantly changing nature of textual data, from heavy and bulk datasets to single or multiline texts or review, demands a sentiment analysis system that is best in performance, adaptable to circumstances and efficient. This project is designed to adjust and provide results to both real-time sentiment predictions for single inputs and large-scale analysis for datasets containing thousands of records or rows or textual data as reviews and opinions. A robust pipeline was integrated into our sentiment analysis tool using which it effectively gave results and had the capability to scale up as well to more users and more data without effecting or compromising on the performance of it.

3. **Empowering Data-Driven Decision-Making:** In a world where customer and user feedback, opinions on social media, and reviews about products, books,

shows or movies shape or design the strategies of organizations and companies, extracting actionable insights from textual data has become a important part of business intelligence. This project aims to assist companies and businesses, researchers, and analysts with a tool or software application that not only identifies sentiment but also visualizes its trends, which enables stakeholders take the best possible decision given the circumstances. With the help of a tool to get insight behind textual data, companies and organizations can make most effective business strategies and decisions.

4. **Ensuring Accessibility for Non-Technical Users:** A challenge that is tough in deploying advanced and complex machine learning models as solutions is bridging the gap between user understanding/ ability to use model and model complexity and backend. This challenge is overcome by creating a very seamless and easy to use and user-friendly front-end interface using the library Streamlit. The easy and understandable design it offers allows even users without technical background to interact with the sentiment analysis tool effortlessly. The project features such as entering a single or multi line text or uploading bulk texts or reviews in a csv file allow users to easily use the tool and the visualization provided in this project in the form of charts help users understand the statistics of the result better.

5. **Maximizing Model Efficiency Through Optimization:** Even though the power of transformer-based models like the BERT is great and huge, the intense mathematics and complexity behind the model may hinder its performance in real time application. This sentiment analysis tool makes use of a fine-tuned BERT model trained on sentiment-labeled datasets which is then further optimized for better accuracy and improved speed. Techniques such tuning the learning rate and other parameters, minimized number of epochs to prevent overfitting, and effectively handling the data imbalance by visualizing and improving accordingly enhances the performance of our model and tool.

Introduction

Large Language Models (LLMs) has introduced new ways to process and understand the context behind the text. It helps machines to understand human language and process it with great accuracy. The **Transformer architecture** is the reason of this transformation, an innovation that was introduced in the paper *"Attention is All You Need"* by Vaswani et al. Transformers depends on self-attention mechanisms to understand the complex contextual relationships within text unlike traditional models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks that cannot understand the contextual relationship within the text. This shift has led to extensive parallel processing, scalability, and great performance in NLP tasks.

LLM models use Transformers capabilities to achieve great results for tasks like- language translation, summarization, question-answering and sentiment analysis. For these models it is easy to learn and understand patterns in language, it`s semantics, and syntactic structures, since they are already pre-trained on large datasets using un-supervised learning and then fine-tuned for specific tasks. Fine-tuning for specific tasks, like sentiment classification, is a great way to achieve high accuracy when you do not have a massive labeled dataset.

The **Transformer architecture**, consists of two parts that is- encoder and decoder. BERT only uses encoder-part of the transformer model to generate outputs. The encoder uses **multi-head self-attention** and **feed-forward neural networks** to obtain results by processing input. Self-attention is a technique that assigns varying weights to different words that denotes their importance in a sentence with respect to every other word in the sentence. For example, in the sentence *"The movie, although lengthy, was exceptionally engaging,"* a Transformer can understand the relation between "engaging" and "movie," to correctly classify this sentence as positive.

BERT, is a bidirectional Transformer model, which means that it learns the context of a sentence by considering both preceding and succeeding words of every other word. This is totally opposite to unidirectional models, that processes sentences sequentially. This bidirectional nature of BERT makes it the favorite choice for tasks like sentiment analysis, where understanding the full context of a sentence is very important to classify texts.

BERT has been used as core-model in this project of sentiment analysis. The pre-trained model is fine-tuning on sentiment-labeled datasets of average size of 25 thousand, to correctly classify the textual data into positive, negative, or neutral categories with high accuracy. The model is integrated in a user-friendly framework for its practical application.

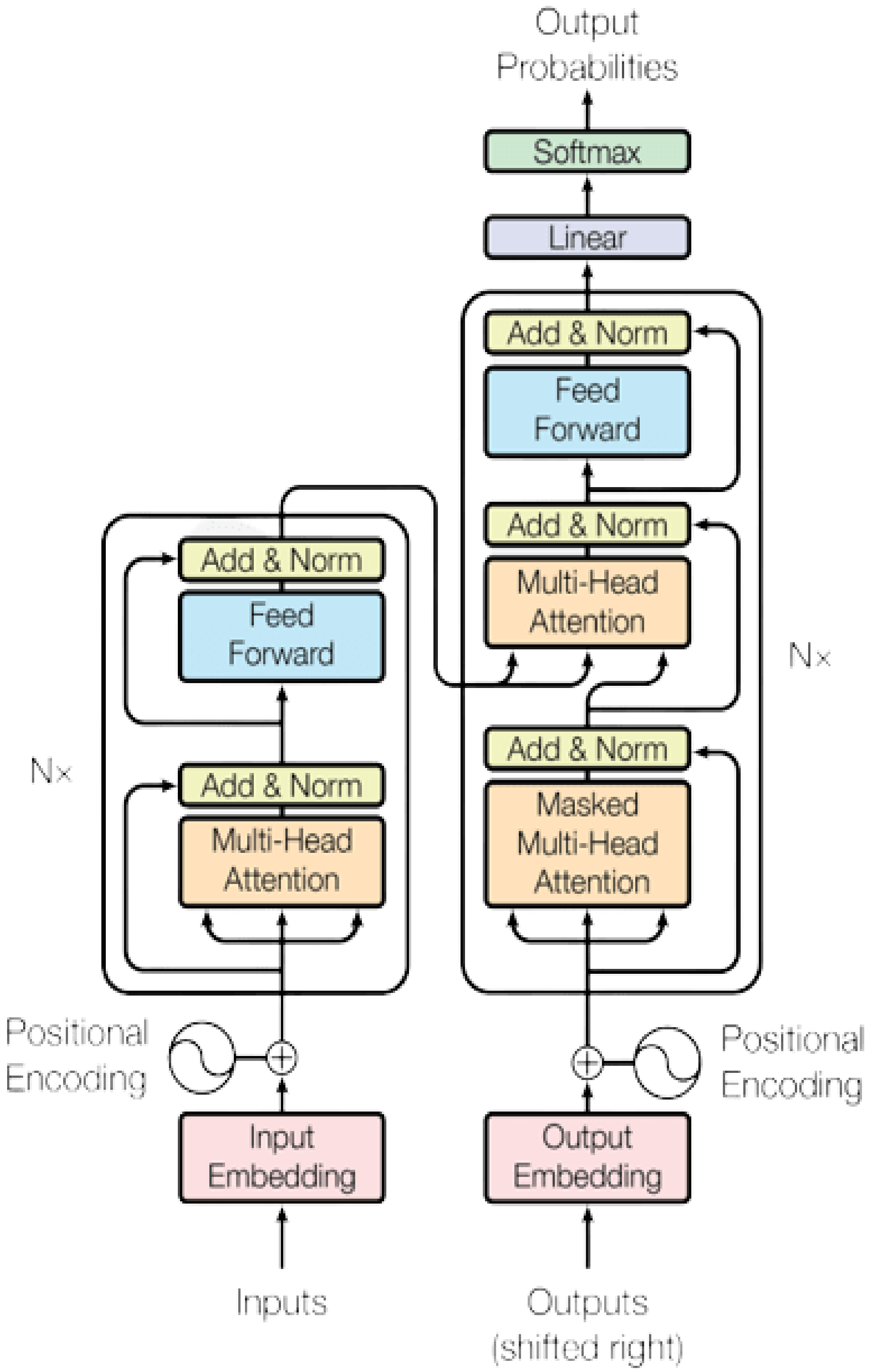

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Figure 3.1

Description of overall software structure

Frontend

The user interface or frontend of this sentiment analysis tool is developed in such a way that even users without any technical background and interact with the tool effectively. This seamless and interactive user interface is designed using a Streamlit library and python language has been used. The front-end has two parts to it. The first one is to do bulk sentiment analysis which has an option to upload a csv file with multiple records. The second is an input field where one can enter a single review of any length text.

Important features of the front-end include real time feedback activities and mechanism. For sentiment analysis of a single text or review, users can enter the text in the input field box or even paste any text. On clicking the submit button, the frontend part sends the input to the backend part or model for further processing it and then displays the sentiment that the model in the backend outputs immediately. This feature is particularly useful for cases where quick insights are required, such as analyzing social media posts or short customer feedback and reviews for any product.

The bulk or multi review analysis functionality allows users to upload CSV files containing multiple records of reviews or other forms of textual data. This feature is specially designed for businesses and researchers handling large datasets, such as customer feedback analysis or survey responses. Once the user uploads the csv file (which can be up to 300kb as per current design but can be increased for better performance later), the frontend provides visualizations, including interactive pie charts and bar graphs generated and displayed on the user interface using plotly library. These visualizations offer an easier and interactive way to interpret sentiment patterns, making it easier for stakeholders to identify trends and patterns.

Error handling and invalid input detection is also inbuilt in the front-end part of the model. For example, the tool is designed in such a way that it checks for empty inputs, invalid file formats, or corrupted datasets and even the files that are out of allowed range and provides descriptive error messages to guide users so they can correct it. This sentiment analysis tool is scalable as well as adaptable as it can run on mobile phones, tablets, laptops and desktops as well. This user interface enables to cover the gap between easy accessibility of the tool, its usability, understanding and the complexity of the model

Backend

•Preprocessing: Tokenization and Text Preparation

The backend begins with a crucial step—tokenization, which is the process of converting the input text as is or raw text into a format of a particular structure suitable for model to process. In this project, a specific tokenizer, BERT tokenizer is used which breaks the text into small parts and each part is known as a token. These tokens can be individual words, punctuations or sub words. Tokenization is important because like every other transformer model, BERT does not work directly with raw text. What it does is turns it into numerical form and processes that numerical data. For example, the sentence “The book was so great!” will be split into tokens such as "The", "book", "was", “so” and "great". Also, if any word is out of vocabulary, then the tokenizer converts or breaks it into sub words first ensuring that even unfamiliar or not previously seen words can be processed effectively by the model. After tokenization the words are converted into a sequence of integers based on the BERT’s vocabulary. Padding. Since the BERT model expects inputs to be of a fixed length (128 tokens in this case), the sentences are padded with a special token to match this length, or truncated if they exceed the allowed token limit. This step ensures that all input sequences are standardized, making them compatible with the BERT model.

•Encoding Input Data

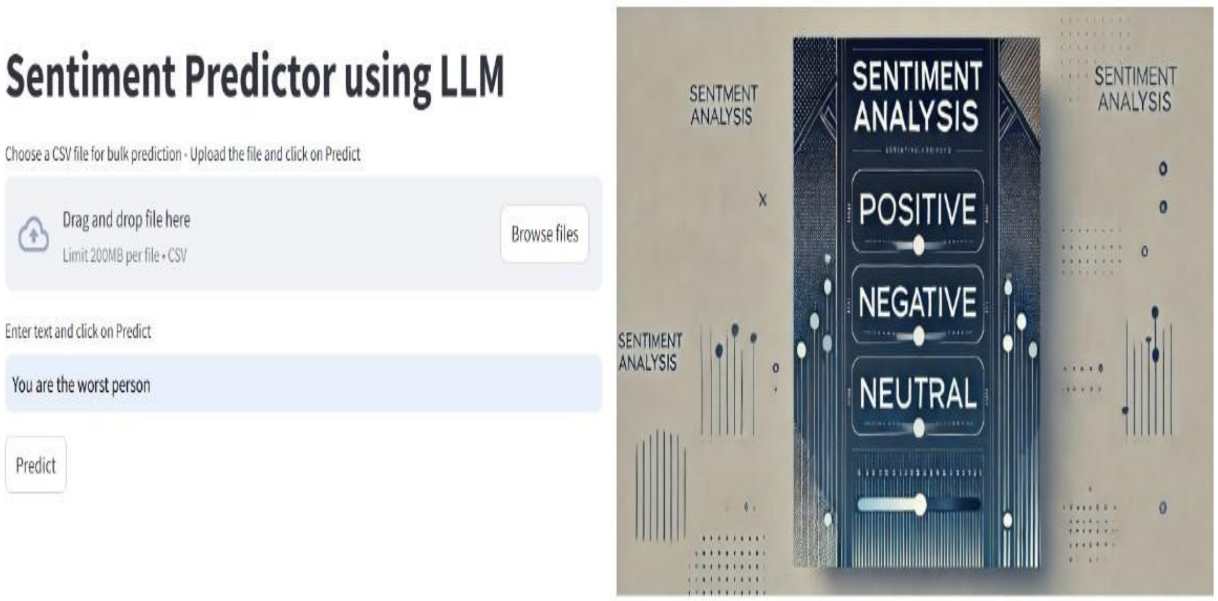
Once the tokens are being generated the next step is to convert it into numbers that can be directly processed. This step is about generating three components: input IDs, attention masks, and token type IDs. Input IDs are the numerical representations of the tokens, which are mapped from BERT's vocabulary. Each token, whether it’s a word or sub-word, is assigned a unique integer ID that the model uses for processing. The attention mask is a binary array indicating which tokens are important for the model to attend to. For example, padding tokens (added to make input sequences of equal length) are assigned a mask value of 0, signaling the model to ignore them. Meanwhile, non-padding tokens are assigned a value of 1 to ensure they are processed. Lastly, token type IDs are used to distinguish between different segments of text in tasks that involve pairs of sentences, such as question answering. For sentiment analysis, this feature may not be as crucial, but it ensures the system can handle more complex tasks in the future. These encoding steps are essential for the model to understand the structure of the input text and the relationships between the tokens.

- **BERT Model Integration**

The most important part for building the sentiment analysis tool is the BERT model, specifically the pre-trained Bert-base-uncased model which is has a transformer-based architecture, and it makes use of a self-attention mechanism that basically processes all tokens in parallel, after it considers both the preceding words and succeeding words to get a very deep understanding of context of that word. This is what makes BERT stand out from previous models like RNNs and machine learning models, which processed text in a sequential manner. The bidirectional nature of BERT means it captures deeper and better contextual relationships between words and their meaning, as it looks at words in both directions (left to right and right to left) to understand meaning. Further fine-tuning is done so that the already trained model can adapt to particular task of sentiment analysis. The model is trained on a sentiment labeled publicly available dataset, where text is classified into three categories namely positive, negative, and neutral. The fine-tuning process changes and adjusts the weights of the pre-trained model, allowing it to specialize in the task of sentiment analysis, leveraging its general language understanding while adapting to the nuances of sentiment classification.

- **Model Inference**

After the input text is processed and tokenized, it is passed through the BERT model for inference. During inference, the model produces something named logits which are raw and direct scores for each of the three-sentiment class (positive, negative, neutral). These logits tell the models prediction but are not normalized and converted into probabilities yet. It is still to be done. To convert these raw scores into probabilities, we make use of a SoftMax function and apply it in the models result. The SoftMax function makes transition of the logits into a probability distribution, which ensures that the sum of probabilities for all three classes when added equals to 1. This step allows the model to interpret the logits and output the likelihood of each sentiment. For example, if the model predicts a 0.8 probability for "positive" and a 0.1 probability for "neutral" and 0.1 for negative, such results indicates a strong chance and confidence that the sentiment behind the text is positive. Finally, the highest probability class is selected to be shown as output as the predicted sentiment, and the result is displayed on the user interface or returned to the frontend. This process makes sure that each piece of text is classified with best possible accuracy based on its actual and contextual meaning, which shows the ability of model to understand and find the subtle changes in text and underlying hidden meaning as well.



Sentiment Predictor using LLM

Choose a CSV file for bulk prediction - Upload the file and click on Predict

Enter text and click on Predict

You are the worst person

Figure 4.1

Basic Data Analysis

- **Dataset Preprocessing**

The first step in fine-tuning the model is to collect and perform cleaning on a suitable dataset that can be used for training and testing. Dataset for this sentiment analysis model was taken from Kaggle, which is an open-source resource for downloading datasets. The downloaded dataset had text reviews labeled as- **positive**, **negative**, and **neutral**. The dataset obtained was then cleaned to remove any useless notations like HTML tags, missing values, or any special or binary characters that are of no use in training. This step of cleaning the dataset ensures that only useful information is left in the dataset, to make training of the model easy and less complicated. The dataset after cleaning was then split into two parts- training and testing datasets. The dataset used to train the model is training dataset and the dataset used to find the accuracy of the model on the data it has not seen is testing data. This testing dataset also helps us to see if the model is overfitting or underfitting.

- **Handling Class Imbalance**

During fine-tuning the model class imbalance was found, which means dataset size of one class was more than the other classes. In this project`s dataset, maximum of the reviews was "positive," while the "neutral" and "negative" categories had comparatively few examples. Class imbalance can lead the model to favor one class over another causing biased results as output of the model. Weighted loss function was used to remove the effect of class imbalance during training. The model was given more penalty for making predicting mistake for minority classes, such as "neutral" or "negative" that helped the model to learn in a better way. This method ensures correct outputs for the given inputs.

- **Exploratory Data Analysis (EDA)**

Exploratory Data Analysis (EDA) is the next step that analyzes the data`s characteristics before feeding it to the model for fine-tuning. EDA analyses and visualizes the distribution of classes across the dataset to find if there is any special relation among the data. **Pie charts** were used to show the proportion of classes. EDA provides us with valuable insights about the data, that can be used understand the data and further process it. This exploration helps with knowing about the dataset in detail.

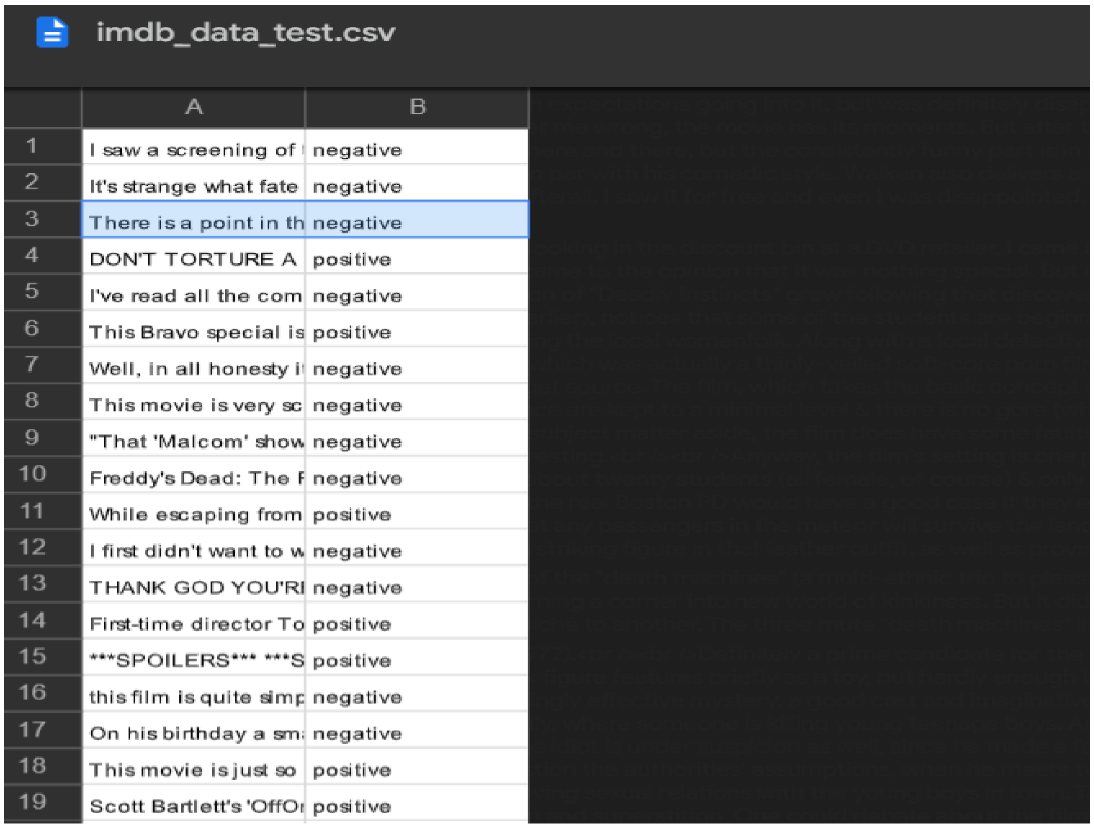
 Token Distribution and Vocabulary Analysis
Token-level analysis is crucial for understanding how the model interprets the text. Each word or sub-word in the dataset is tokenized using BertTokenizer, and the distribution of tokens across the entire dataset is analyzed. This analysis allows us to identify the most common tokens and the frequency distribution of words. By examining the **token frequency**, we can identify potential issues such as rare words that may require further preprocessing or handling.

Dataset

- Multiclass Dataset was downloaded from Kaggle, an open-source platform for downloading datasets.
- “imdb” dataset was used for Binary Sentiment classification

- “imdb” dataset contains 25,000 rows with two columns of input. Input columns had one column as review and the other as the label- positive or negative. After fine-tuning, giving any input to the model displayed the corresponding sentiment associated with it.

- The public dataset had 27,482 rows and two columns. Input columns had any text which was an opinion or a review and its corresponding label as positive, neutral or negative. Then after training, giving any input to the model displayed the corresponding sentiment associated with it.


imdb_data_test.csv

	A	B
1	I saw a screening of	negative
2	It's strange what fate	negative
3	There is a point in th	negative
4	DON'T TORTURE A	positive
5	I've read all the com	negative
6	This Bravo special is	positive
7	Well, in all honesty i	negative
8	This movie is very sc	negative
9	"That 'Malcom' show	negative
10	Freddy's Dead: The F	negative
11	While escaping from	positive
12	I first didn't want to w	negative
13	THANK GOD YOU'RI	negative
14	First-time director To	positive
15	***SPOILERS*** ***S	positive
16	this film is quite simp	negative
17	On his birthday a sm	negative
18	This movie is just so	positive
19	Scott Bartlett's 'OffOr	positive

Figure 5.1

Optimizing The Performance and Feature Scaling

- **Fine-Tuning the BERT Model**

  A very important step in maximizing and upscaling the effectiveness of an already trained model such as BERT (bidirectional encoder representation of transformer model) is fine-tuning. BERT needs to be adjusted for a specific goal or a particular task of our need, like sentiment classification or summary generation or transforming text of one language to any other language in order to adjust its weights for the best results in that specific task or area, even though it has already been trained on large datasets and is aware of general language trends and patterns and understands it. Training the model on a sentiment-labeled dataset—where reviews were categorized as neutral, negative, or positive—was how the model was fine tuned for this particular project. By backpropagating the error through the network or the feed forward neural network, the model's parameters can be changed, enabling it to learn task-specific characteristics and perform a specific language task. To guarantee steady convergence and prevent it from going away or beyond the ideal weights, a learning rate of 1e-4 was chosen which was the most optimal. This learning rate is not too low, but it is small enough to allow the learning to happen gradually. Fine-tuning is done for a limited number of epochs or cycles (one in this project) to avoid overfitting, especially since BERT already has pre-trained weights that gives it a very rigid and strong foundation.

- **Early Stopping to Prevent Overfitting**

  When we are training machine learning models, especially deep learning models with many parameters like BERT does, overfitting is an obvious problem. Overfitting decreases the model's capacity to generalize to new data that it has never seen before by learning the training set too specifically and learning the noise in the data as well, including other noise or irrelevant patterns. Early stopping was integrated into the training procedure to reduce the chances of overfitting. During training, early stopping means keeping an eye on the model's performance on a validation set and stopping when the validation performance begins to deteriorate, a sign that the model is starting to overfit. By doing this, the model is kept from learning noise in the training data and its capacity to generalize is maintained.

- **Hyperparameter Tuning**

  The number of epochs, learning rate, and batch size are examples of hyperparameters that have a big impact on deep learning model performance. This project involved adjusting a number of hyperparameters to determine the best values for sentiment analysis tasks. The batch size, or how many training examples were used in a single forward/backward pass, was one important hyperparameter to consider. Because it enables the model to update weights more often, a smaller batch size (e.g., 16) was shown to yield better outcomes in this research, leading to better gradient updates. Larger batch sizes may result in less accurate models in some situations even when they reduce the number of updates. Because the pre-trained BERT model already had a strong grasp of language, the number of epochs was kept to one in order to avoid overfitting. The model's optimal performance without overtraining was attained through experimentation with various batch sizes and learning rates.

Technology Used

- **Transformers and BERT**

BERT is a transformer model that processes text bidirectionally, which is done for improving sentiment classification accuracy. The library we used here to get access to an already trained model is Hugging Face Transformers library which provides a pre-trained TFBertForSequenceClassification model, which was fine-tuned for sentiment analysis tasks.

- **Hugging Face Transformers Library**

The Hugging Face Transformers library helps us ease the process of working with BERT by allowing us to use pre-trained models, tools for tokenization, and integration with the sentiment analysis pipeline as well, which in turn speeds up development and model fine-tuning.

- **TensorFlow**

TensorFlow that is renowned was used for the purpose of fine-tuning the BERT model, providing tools for efficient computation and GPU acceleration to optimize performance during model training and inference.

- **Streamlit**

Streamlit was installed and used to build the user-friendly and extremely interactive web interface, allowing real-time input of text or CSV uploads and providing interactive visualizations of sentiment analysis results.

- **Plotly**

Plotly which is a visualization library was used for creating interactive charts, pie charts specifically and multiple visualizations, such as pie charts and bar graphs, to display sentiment distributions and allow users to better understand the results of the model.

• **GitHub**

GitHub provided us with version control and eased the process of deployment and working together as a team ensuring that the project's code was organized, easy to maintain and track, and accessible to everyone for future updates and improvements as well as maintenance.

• **Google Colab**

Google Colab gave us access to a cloud-based environment for training the BERT (bidirectional encoder representation of transformer) model, that offered and gave us access to free GPU resources that saved us the training and testing time and helped us utilize our system resources.

• **Jupyter Notebook**

Jupyter Notebook was used for data pre-processing, where we extracted only the important columns of the multi column dataset and also used it to clean the data. It gave us a platform to balance our data well by visually looking at using various libraries. It allowed us to interactively develop our model and project.

• **Kaggle**

Dataset for this sentiment analysis model was taken from Kaggle, which is an open-source resource for downloading datasets. The downloaded dataset had text reviews labeled as- **positive**, **negative**, and **neutral.**

Working

Data Collection and Preprocessing

We begin the process by collecting a sentiment-labeled dataset from Kaggle for multiclass classification whereas binary class classification dataset used is the one

available publicly by the name of "imdb" dataset. The text data is then carefully cleaned and balanced as well to remove irrelevant characters, special symbols, and stop words. This step makes sure that only the most meaningful, important and relevant information is remained and left for sentiment analysis. After the data is cleaned, the text is tokenized using the BERT tokenizer. This tokenizer that we import and use breaks down the input sentences into smaller units, called tokens, which are easier for the BERT model to process and understand as compared to words directly. Tokenization is the process that converts the words into numerical representations of them, which enables the model to work with the data effectively.

Model Training and Fine-Tuning

Once preprocessing and data balancing is complete, the tokenized text is given to into a pre-trained BERT model using the Hugging Face Transformers library that directly gives access of model. The model is already trained and fine-tuned by us according to our use on the sentiment-labeled dataset that we processed to adjust its already trained weights, making it specialized for the task of binary as well as multi-class sentiment classification. During the process of fine-tuning, the model is trained using appropriate hyperparameters such as a appropriate learning rate and batch size and epochs. Techniques and methods like early stopping are used to avoid overfitting. The entire training process is handled by TensorFlow, that makes advantage of GPU acceleration to speed up computations.

Prediction and Inference

When we finally fine-tune the model, it is used to predict the sentiment of new text which may be unseen by the model. When a user enters input text in the given input box available, the model generates and produces logits for each sentiment category (positive, negative, neutral). These logits are passed through an activation function, the one we used is the softmax function to convert them logits into equivalent probabilities. The sentiment of all, with the highest probability is selected as the final prediction or output label given by model. The results are displayed on the front-end and shown to the user in real-time through Streamlit’s web interface, making the predictions easy to access and interpret and visualize as well.

 Visualization of Results

The sentiment predictions are visualized effectively by us using Plotly library made available to provide clear insights. Interactive pie charts and bar graphs display the distribution of sentiments, whether for a single prediction or bulk analysis. These visualizations make it easier for users to understand the sentiment breakdown at a glance. During development, Jupyter Notebook was used to prototype and test the visualizations, allowing for quick adjustments and debugging before integrating them into the Streamlit interface.

 Bulk Data Processing

Users can not only check for single sentences but also do bulk data processing. For bulk analysis, users can upload CSV files containing multiple text entries, such as reviews. The backend processes each entry by generating sentiment predictions for every text and aggregating the results. This feature enables efficient analysis of large datasets, making it useful for businesses or researchers seeking insights from customer feedback or social media posts. The results are presented interactively through Streamlit, showing both individual predictions and the overall sentiment distribution for the dataset.

Code Design

Model Fine-Tuning Code:-

```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from transformers import TFBertForSequenceClassification
import tensorflow_datasets as tfds
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
```

Preprocessing data:

```
import pandas as pd
new_reviews = pd.read_csv('/kaggle/input/train-new2/train_new.csv')
label_map = {'negative': 0, 'neutral': 1, 'positive': 2}
new_reviews['sentiment'] = new_reviews['sentiment'].map(label_map)

train_new, test_new = train_test_split(
    new_reviews[['text', 'sentiment']],
    test_size=0.2,
    random_state=42
)

def create_tf_dataset(df):
  # Create separate tensors for reviews and feedback
     reviews = df['text'].astype(str).to_numpy()
     feedback = df['sentiment'].astype(int).to_numpy()
     return tf.data.Dataset.from_tensor_slices((reviews, feedback))

train_new_dataset = create_tf_dataset(train_new)
test_new_dataset = create_tf_dataset(test_new)
```

```
 merged_train = train_new_dataset
 merged_test = test_new_dataset
```

Training Model:

```
from transformers import BertTokenizer
tokenizer  =  BertTokenizer.from_pretrained("bert-base-uncased",  do_lower_case  =
True)
 max_length = 512
 batch_size = 4
 def convert_example_to_feature(review):
             return tokenizer.encode_plus(review,
                          add_special_tokens = True,
                          max_length = max_length,
                          pad_to_max_length = True,
                         return_attention_mask = True)

 def map_example_to_dict(input_ids, attention_mask, token_type_ids, label):
       return {
         "input_ids": input_ids,
         "token_type_ids": token_type_ids,
         "attention_mask": attention_mask
   }, label


 def remove_stopwords(text):
       stop_words = set(stopwords.words('english'))  # Use appropriate language
       words = text.split()
       filtered_words = [word for word in words if word.lower() not in stop_words]
       filtered_text = " ".join(filtered_words)
       return filtered_text

 def encode_examples(ds, limit=-1):
       input_ids_list = []
       token_type_ids_list = []
```

```
        attention_mask_list = []
        label_list = []
        if(limit > 0):
          ds = ds.take(limit)
        for review, label in tfds.as_numpy(ds):
          review = remove_stopwords(review.decode())
          bert_input = convert_example_to_feature(review)
          input_ids_list.append(bert_input['input_ids'])
          token_type_ids_list.append(bert_input['token_type_ids'])
          attention_mask_list.append(bert_input['attention_mask'])
          label_list.append([label])

      return tf.data.Dataset.from_tensor_slices((input_ids_list, attention_mask_list,
token_type_ids_list, label_list)).map(map_example_to_dict)



train_encoded = encode_examples(merged_train).shuffle(5000).batch(batch_size)
test_encoded  = encode_examples(merged_test).batch(batch_size)

train_encoded.save('train_encoded.tfrecord')
test_encoded.save('test_encoded.tfrecord')
loaded_train = tf.data.Dataset.load('train_encoded.tfrecord')
loaded_test = tf.data.Dataset.load('test_encoded.tfrecord')

 import tensorflow as tf



 learning_rate = 1e-5
 epochs = 2
 model = TFBertForSequenceClassification.from_pretrained("bert-base-uncased",
 num_labels = 3)
 optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate, epsilon=1e-08)
 loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits = True)
 metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
 tf.config.set_soft_device_placement(False)
```

```
model.compile(optimizer = optimizer, loss = loss, metrics = [metric])

bert_history = model.fit(loaded_train, epochs = epochs, validation_data =
loaded_test)

model.save_weights('my_model_weights_epochs2.h5')
```

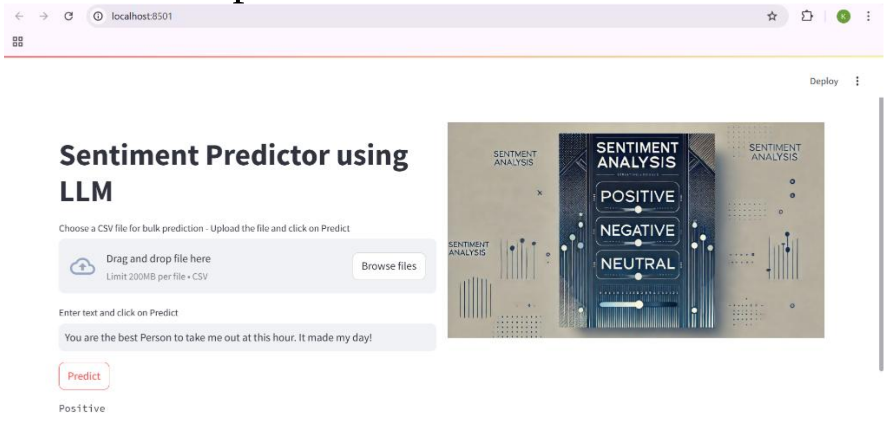
Front-end Development Code: -

```
import streamlit as st
import pandas as pd
import requests
from io import BytesIO
from transformers import TFBertForSequenceClassification
from transformers import BertTokenizer
import tensorflow as tf
import plotly.express as px
import plotly.figure_factory as ff



tokenizer = BertTokenizer.from_pretrained("bert-base-uncased", do_lower_case =
True)
model = TFBertForSequenceClassification.from_pretrained("bert-base-uncased")

model.load_weights('my_model_weights (2).h5')

learning_rate = 2e-5
epochs =
optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate, epsilon=1e-08)

loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits = True)
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
```

```
model.compile(optimizer = optimizer, loss = loss, metrics = [metric])

def make_prediction(test_sentence, model):

    if test_sentence.strip() == "":  # Changed: Check if input is empty or only spaces
        return "Please enter a valid sentence."
    encoded_sent = tokenizer.encode(test_sentence, truncation = True, padding =
'max_length', return_tensors = "tf", max_length = 128)
    tf_output = model.predict(encoded_sent)[0]
    tf_prediction = tf.nn.softmax(tf_output, axis = 1)
    labels = ['Negative', 'Positive']
    label = tf.argmax(tf_prediction, axis = 1)
    label = label.numpy()
    return labels[label[0]]



def process_file(uploaded_file, model):
    try:
        file = pd.read_csv(uploaded_file, header=None)
        if file.empty:
            st.write("Error: File is empty.")
            return
        positive = 0
        negative = 0
        neutral = 0

        length = file.shape[0]
        rpositive = 0
        rnegative = 0
        rneutral = 0
        for index, review in file.iterrows():
            senti = review[1]
            if senti == "Positive" or senti == "positive":
                rpositive += 1
            elif senti == "Negative" or senti == "negative":
                rnegative += 1
            else:
                rneutral += 1
            prediction = make_prediction(review[0], model)
            if prediction == "Positive":
                positive += 1
            elif prediction == "Negative":
                negative += 1
            else:
                neutral += 1
        df = pd.DataFrame({
            'Sentiment': ['Positive', 'Negative', 'Neutral'],
            'Count': [positive, negative, neutral]
        })
        rdf = pd.DataFrame({
            'Sentiment': ['Positive', 'Negative', 'Neutral'],
            'Count': [rpositive, rnegative, rneutral]
        })
        st.write("Positive % = ", positive/length * 100 )
        st.write("Negative % = ", negative/length * 100 )
        st.write("Neutral % = ", neutral/length * 100 )

        st.subheader('Model Prediction')
        fig = px.pie(df, values='Count', names='Sentiment')
        st.plotly_chart(fig)
        st.subheader('Real Prediction')
        fig = px.pie(rdf, values='Count', names='Sentiment')
        st.plotly_chart(fig)
    except Exception as e:
        st.write("Error processing file:", str(e))
st.set_page_config(layout="wide")
col1, col2 = st.columns(2)

with col1:
    st.title("Sentiment Predictor using LLM")
    uploaded_file = st.file_uploader("Choose a CSV file for bulk prediction - Upload
the file and click on Predict", type="csv",
    )

    user__inputt = st.text_input("Enter text and click on Predict", "")
    if st.button("Predict"):
        if uploaded_file is not None:
            process_file(uploaded_file, model)

        else:
            st.text(make_prediction(user__inputt, model))

with col2:
    st.image("image.webp")
```

Outputs from Front-end

Positive output:


localhost:8501

Deploy

Sentiment Predictor using LLM

Choose a CSV file for bulk prediction - Upload the file and click on Predict

Drag and drop file here
Limit 200MB per file • CSV
Browse files

Enter text and click on Predict

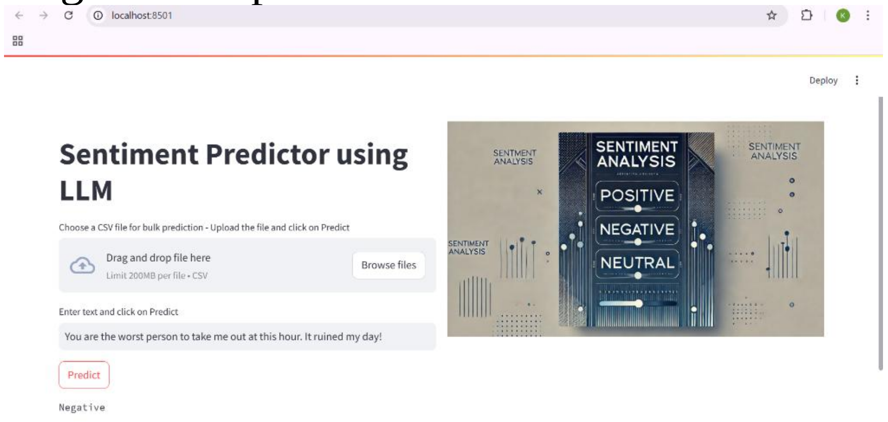
You are the best Person to take me out at this hour. It made my day!

Predict

```
Positive
```

Fig 6.1

Negative output:


localhost:8501

Deploy

Sentiment Predictor using LLM

Choose a CSV file for bulk prediction - Upload the file and click on Predict

Drag and drop file here
Limit 200MB per file • CSV
Browse files

Enter text and click on Predict

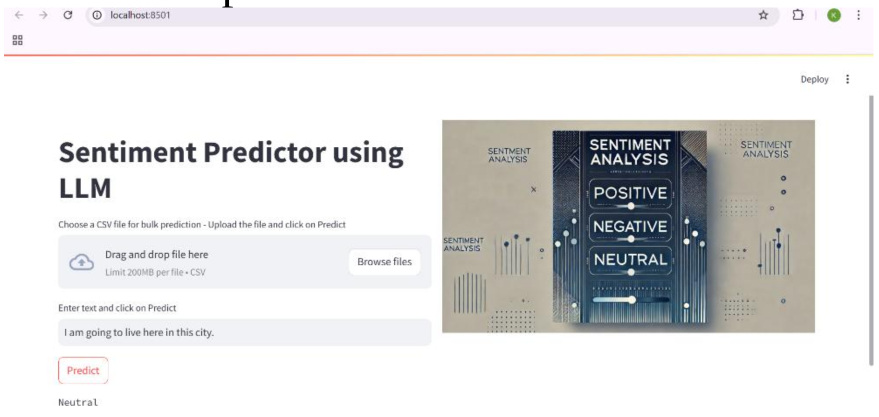
You are the worst person to take me out at this hour. It ruined my day!

Predict

```
Negative
```

Fig 6.2

Neutral output:


localhost:8501

Deploy

Sentiment Predictor using LLM

Choose a CSV file for bulk prediction - Upload the file and click on Predict

Drag and drop file here
Limit 200MB per file • CSV
Browse files

Enter text and click on Predict

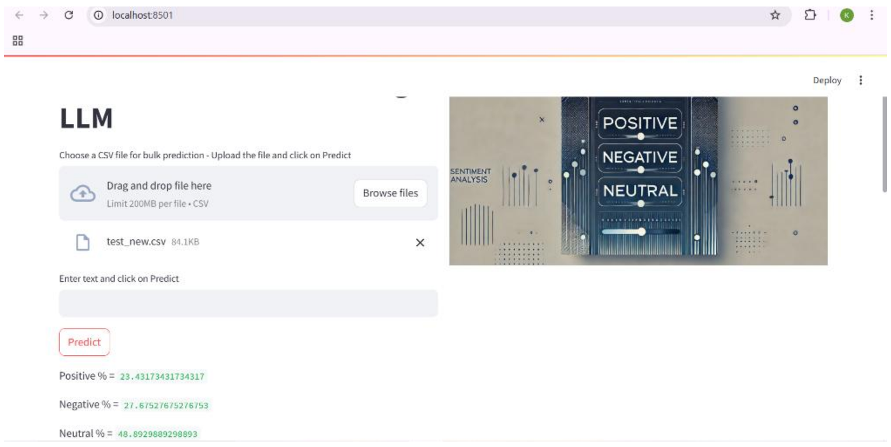
I am going to live here in this city.

Predict

```
Neutral
```

Fig 6.3

Output from processing csv file:



LLM

Choose a CSV file for bulk prediction - Upload the file and click on Predict

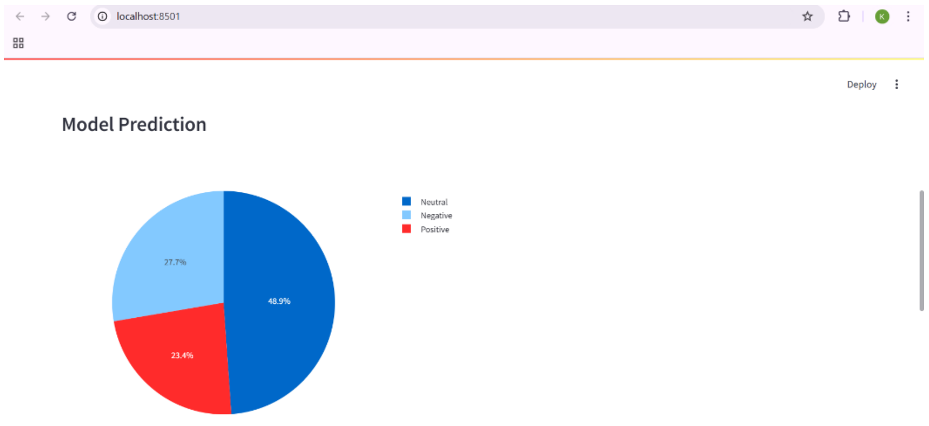
Enter text and click on Predict

Positive % = `23.43173431734317`

Negative % = `27.67527675276753`

Neutral % = `48.8929889298893`

Fig 6.4



Model Prediction

Fig 6.5

Conclusion

In today's world that is overwhelmed with textual data, it is very important to know the meaning behind it. Users have the platform to give feedback about any new product, book, show, etc. To make this feedback be utilized in the best possible way, companies use tools like sentiment analysis to get a gist of public opinion. Therefore, sentiment analysis can also be referred to as opinion mining. This analysis of the emotion behind reviews or any kind of text helps companies to make better decisions that help in their growth and development.

The best and amazing part about this project is not just its accuracy in predicting sentiment behind plain and simple reviews and texts but its ability to handle and understand sarcasms, idioms and even subtle changes. The traditional methods based on machine learning model fail to give such results because they do not have the special capability to process text in a bidirectional way like BERT. Every token and every word, is thoroughly examined within the greater context and wider picture, making sure that no sentiment is being misunderstood at all by the model.

The journey from data preprocessing to fine-tuning, model training, and prediction was not just about achieving a very high accuracy but was about **creating a tool** that empowers its users who have no knowledge about the internals of the model to make better, data-driven decisions. Whether it's a brand using the tool for analyzing customer feedback, a researcher sifting through survey responses, or a business monitoring social media. The inclusion of interactive visualizations in the tool and real-time predictions by the model ensures the satisfaction of the user catering their need in a better way.

Language has become quantifiable, and the emotions within it, now more comprehensible than ever before. This is not the end of the journey but merely the beginning. As new datasets emerge, and as language evolves, the potential for continual improvement and adaptation remains boundless. The fusion of cutting-edge machine learning, user-centered design, and data-driven insights lays the foundation for a future where sentiment analysis is not just a task but a crucial tool for understanding the pulse of human interaction with technology.

References

- **Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł., & Polosukhin, I. (2017).** Attention is all you need. *Proceedings of NeurIPS 2017*. Retrieved from https://arxiv.org/abs/1706.03762

- **Hugging Face. (2021).** Transformers Documentation. *Hugging Face*. Retrieved from https://huggingface.co/docs/transformers

- **TensorFlow. (2021).** TensorFlow 2.0 Documentation. *TensorFlow*. Retrieved from https://www.tensorflow.org/

- **Streamlit. (2021).** Streamlit Documentation. *Streamlit*. Retrieved from https://docs.streamlit.io/

- **Plotly. (2021).** Plotly Python Graphing Library. *Plotly*. Retrieved from https://plotly.com/python/

- **Google Colab. (2021).** Google Colab Documentation. *Google Colab*. Retrieved from https://colab.research.google.com/

Kanchan Lata

paper

10 word
PhD Scholar
Madan Mohan Malaviya University of Technology

Document Details

Submission ID
trn:oid:::1:3111028104

Submission Date
Dec 11, 2024, 10:23 AM GMT+5:30

Download Date
Dec 11, 2024, 10:49 AM GMT+5:30

File Name
report_tocheck.pdf

File Size
693.4 KB

31 Pages

6,316 Words

35,188 Characters

12% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

59 Not Cited or Quoted 12%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

8%	Internet sources
8%	Publications
7%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

59 Not Cited or Quoted 12%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

8%	Internet sources
8%	Publications
7%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	www.analyticsvidhya.com	2%
2	Student papers	University of North Texas	2%
3	Publication	Shivam R Solanki, Drupad K Khublani. "Generative Artificial Intelligence", Springe...	1%
4	Internet	stackoverflow.com	1%
5	Publication	Mehdi Ghayoumi. "Generative Adversarial Networks in Practice", CRC Press, 2023	1%
6	Student papers	University of Hull	0%
7	Internet	fastercapital.com	0%
8	Internet	medium.com	0%
9	Internet	www.mdpi.com	0%
10	Publication	Arvind Dagur, Dhirendra Kumar Shukla, Nazarov Fayzullo Makhmadiyarovich, Ak...	0%

11	Internet	
ebin.pub		0%
12	Student papers	
Sunway Education Group		0%
13	Internet	
gpttutorpro.com		0%
14	Student papers	
Royal Holloway and Bedford New College		0%
15	Publication	
Glass, Grant. "Fidelity, Remix, and the Adaptive Potential: Leveraging AI and ML T...		0%
16	Internet	
arxiv.org		0%
17	Internet	
www.coursehero.com		0%
18	Student papers	
Liverpool John Moores University		0%
19	Student papers	
Sheffield Hallam University		0%
20	Student papers	
Toronto Business College		0%
21	Publication	
Dipanjan Sarkar. "Text Analytics with Python", Springer Science and Business Me...		0%
22	Publication	
Nogueira, Fábio Miguel Pereira. "Identifying References to Legal Literature in Por...		0%
23	Publication	
Sharon Chiang, Vikram R. Rao, Marina Vannucci. "Statistical Methods in Epilepsy",...		0%
24	Publication	
Shrestha, Neeraj. "A Novel Spam Email Detection Mechanism Based on XLNet", T...		0%

25	Student papers	
University of San Diego		0%
26	Internet	
atheros.ai		0%
27	Internet	
benjad.github.io		0%
28	Publication	
Rudolf Debelak, Matthias Aßenmacher, Timo Koch, Clemens Stachl. "From Embed...		0%
29	Internet	
fg2019.org		0%
30	Internet	
medium-parser-seven.vercel.app		0%
31	Internet	
www.gushiciku.cn		0%