UNIVERSITY OF LEICESTER

Department of Informatics

University of Leicester

CO7201 Individual Project

Final Report

"MetaSent - Deep Learning for Sentiment Prediction with Meta-Learning Enhancement"

Dinesh Kumar Sammeta

dks18@student.le.ac.uk

229035248

Project Supervisor: Dr. Artur Boronat

Second Marker: Dr Xiao Chen

Word Count: 12135

(exclude Abstract,Table Of Contents and Ref)

17/05/2024

ABSTRACT

A extensive work, the MetaSent project aims to address the growing possibilities and challenges associated with sentiment analysis, especially in the wider context of written data analysis. In the current digital environment, the abundance of textual material offers organisations a great deal of information as well as an immense challenge when trying to understand customer emotion. Businesses may enhance customer service, increase customer perceptions, hone marketing tactics, and make well-informed choices by utilising sentiment research. The main goals of the MetaSent project is to provide sophisticated sentiment analysis techniques that meet the changing demands of commercial enterprises.The research has several goals, including creating a reliable workflow for text preparation, comparing and applying several deep learning architectures—such as CNN, LSTM, Naive Bayes and hybrid mechanisms—to precisely predict emotion labels for reviews. In addition, the project intends to improve label prediction by combining rapid engineering with sophisticated algorithms like Large Language Models (LLMs) to produce logical justifications for sentiment labels that are anticipated. Furthermore, the research investigates meta-learning strategies inside the Langchain Framework, to offer meaningful explanations for sentiment labels, improving interpretability and comprehension. The building of an intuitive interface with Flask on the backend and React on the front end is essential since an intuitive interface is critical to the project's success. In addition to presenting suggestions in an aesthetically pleasant manner to elicit explanations for sentiment classifications, the interface makes it easier to communicate with the sentiment analysis system. The building of deep learning algorithms for sentiment analysis, the improvement of sentiment analysis prompts for descriptive insights are the main needs of the project. The creation of a web-based interface with incorporated prompts, authentication of users, portfolio management to enhance confidence and understanding of model predictions are other suggested features. The MetaSent project aims to transform sentiment analysis and provide organisations with useful insights from textual data through several initiatives.

DECLARATION

All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Sammeta Dinesh Kumar

Date: 17/05/2024

Index

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In the current digital era, sentiment analysis is essential in a variety of fields, including market research and social media analysis. More advanced sentiment analysis methods that can adapt and generalise across many datasets and domains are in high demand as the amount and complexity of data continue to rise—presenting MetaSent, an innovative approach to sentiment prediction that combines deep learning techniques with a brand-new refinement called Meta-Learning. MetaSent is a major advancement in the field of sentiment analysis research. Essentially, MetaSent uses deep learning models to identify complex relationships and subtleties in textual data, which allows it to identify sentiment in various settings reliably. By using Meta-Learning approaches, MetaSent goes beyond simple categorization, in contrast to typical sentiment analysis models.

Meta-learning, or "learning to learn," enables MetaSent to dynamically modify its learning approach according to the properties of any data or text it comes across. With this capacity, MetaSent may more effectively generalise across several domains, increasing its robustness and usefulness in practical applications. This study is to investigate the possibilities of MetaSent in several contexts, such as sentiment analysis on processing consumer feedback. I believe MetaSent can completely transform sentiment analysis by utilising the power of meta-learning to provide more precise, flexible, and scalable solutions. It sets out to rethink sentiment analysis in the age of big data and digital communication, which encourages to exploration of the worlds of deep learning and Meta-Learning through this introduction to MetaSent. To discover the potential and ramifications of MetaSent in influencing sentiment prediction going forward.

1.1. AIM AND OBJECTIVES

This project's goal is to provide a thorough comparison study of several deep learning models for sentiment analysis. A standardised dataset will be used to train and assess these models, and the top-performing model will be chosen to serve as the basis for additional improvement. The chosen model will then be used with a Language Model with Prompt Engineering (LLM) to produce sentiment label explanations. Additionally, to make interacting with the sentiment analysis system

easier, a user-friendly interface will be created on the front end using React and on the back end using Flask.

Objectives:

This project aims to achieve the following objectives:

1. Comparison of Deep Learning Models: Examine how well several deep learning models such as LSTM,LSTM + CNN, Bi-LSTM Convolutional Neural Networks (CNN) and Navie Bayes perform on sentiment analysis tasks.
2. Selection of Best-Performing Model: Based on the findings of the comparison study, choose the deep learning model that performs the best. The chosen model will act as the basis for additional development and integration with cutting-edge algorithms for generating justifications and predicting labels.
3. Integration with LLM for Justification Generation: To produce logical explanations for sentiment labels, combine the chosen deep learning model with the Language Model with Prompt Engineering (LLM). To improve the interpretability and comprehension of the sentiment analysis results, use LLM to offer insightful justifications for the anticipated sentiment labels.
4. Development of User-Friendly Interface: Using Flask for the backend and React for the front end, create a user-friendly interface. This will enable users to enter text data and obtain aesthetically pleasing sentiment forecasts and explanations.

1.2 CONTRIBUTIONS

- Cleaning and Preprocessing of Data Set(AMAZON FOOD REVIEWS)
- Development of Models Architecture(LSTM,CNN,BILSTM,CNN+LSTM AND NAIVE BAYES)
- Setting Hyperparameters like batch size and no of epochs for above models and experimenting models to improve performance

- In order to direct the language model in producing logical and contextually suitable justifications or reasons for sentiment predictions, prompt templates are designed.
- Used React and Flask to Design Web App.

1.3. BACKGROUND

Sentiment analysis, also known as opinion mining, is a field of study that involves the use of computational tools to identify and extract subjective information from text, such as opinions, attitudes, and emotions expressed by individuals (Lin & He, 2009). In recent years, deep learning algorithms have gained significant attention and shown promising results in sentiment analysis tasks. Researchers have explored various models and architectures to enhance the accuracy and efficiency of sentiment analysis systems. Among the popular deep learning models used for sentiment analysis are Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN), Bidirectional LSTM (BiLSTM), and combinations of these models (Alzahrani et al., 2022; Dang et al., 2021; Salur §& Aydin, 2020).

LSTM, a type of recurrent neural network (RNN), is well-suited for processing and analyzing sequential data due to its ability to retain long-term dependencies. In sentiment analysis, LSTM has been utilized to capture the context and temporal information of text data, enabling the model to make more informed predictions about the sentiment expressed in the text (Omara et al., 2022). CNNs, on the other hand, are effective in capturing local patterns and features in data, making them suitable for tasks like sentiment analysis where identifying key features in text is crucial (Hardjita et al., 2022). Combining LSTM and CNN, as seen in models like CNN-LSTM, allows for leveraging the strengths of both architectures to improve the overall performance of sentiment analysis systems (Zhang & Qin, 2022).

Moreover, the integration of LSTM with CNN in models like CNN-LSTM has shown enhanced capabilities in sentiment analysis tasks, particularly in domains like e-commerce product reviews and social media sentiment classification (Zhang & Qin, 2022). These hybrid models benefit from the ability of LSTM to capture long-term dependencies and the feature extraction prowess of CNNs, resulting in more accurate sentiment classification outcomes (Zhang & Qin, 2022). Additionally, the

use of Bayesian neural networks (BNN) in sentiment analysis models has been proposed to address uncertainty in predictions, thereby improving the reliability of sentiment analysis systems (Mrhar et al., 2021).

In the context of sentiment analysis for label generation (e.g., positive or negative sentiment classification), the combination of LSTM and CNN has been particularly effective. Models like LSTM+CNN have demonstrated success in classifying text data into sentiment categories, enabling the automated identification of positive or negative sentiments expressed in reviews, tweets, or other textual data (Nawaz et al., 2021). These models leverage the strengths of both LSTM and CNN to extract relevant features and contextual information from text, leading to more accurate sentiment labeling (Nawaz et al., 2021).

Furthermore, for reason generation associated with sentiment labels, Language Model (LM) approaches like GPT (Generative Pre-trained Transformer) have been employed. GPT models, known for their ability to generate coherent and contextually relevant text, can be utilized in sentiment analysis tasks to provide explanations or reasons behind the assigned sentiment labels (Lin & He, 2009). By integrating GPT with sentiment analysis models, researchers aim to not only classify sentiments but also generate explanations for the predicted sentiment, enhancing the interpretability and transparency of sentiment analysis systems (Lin & He, 2009).

1.4. RESEARCH SCOPE & LIMITATIONS

This project's scope includes developing a user-friendly interface with React and Flask, integrating Language Model with Prompt Engineering (LLM) for justification generation, and thoroughly exploring deep learning models for sentiment analysis. This research attempts to determine which architecture" LSTM, CNN, Bi-LSTM, LSTM + CNN,Naive Bayes" performs best for sentiment prediction tasks for a given Data(Amazon Food Reviews) by comparing and contrasting them. Additionally, the creation of cogent explanations is made possible by the integration of LLM with the chosen model, which improves the interpretability and comprehension of sentiment analysis data.

It is important to recognise specific limitations that fall within the purview of this study.

First off, many aspects including dataset properties, hyperparameter settings, and preprocessing methods might affect how successful deep learning models are. Despite the efforts to standardise these elements, performance variances could still happen. Furthermore, there is complexity and computational cost added by integrating LLM with the chosen model, which might affect the scalability and real-time performance of the system.

Additionally, the particular dataset that was employed for training and assessment may have reduced the findings' generalizability. For consistency, a standardised dataset will be used, although when applied to datasets from various domains or with different levels of complexity, the models' performance may vary. Furthermore, time and budget constraints may limit the extent of the user interface development, which might prevent the inclusion of additional features or customisation possibilities.

Despite these drawbacks, the goal of this study is to offer insightful information on the efficiency of deep learning models in sentiment analysis as well as the possible advantages of including LLM for the creation of justifications. The goal of this project is to improve sentiment analysis techniques and provide more approachable text data analysis tools by tackling these obstacles and constraints.

2. LITERATURE REVIEW

2.1. CURRENT TRENDS AND GAPS

Utilising Deep Learning Models for Sentiment Analysis:

For sentiment analysis usage, deep learning models—particularly RNNs (recurrent neural networks) and convolutional neural network models (CNNs)—have proven to be quite successful. In order to classify sentences, Kim (2014) presented a CNN architecture and showed how well it could extract both local and global contextual information from text data. According to Tang et al. (2015), LSTM-based models have demonstrated encouraging outcomes when it comes to identifying sequential patterns and long-range relationships in sentiment analysis tasks. Moreover, it has been suggested to use hybrid architectures that combine CNN and LSTM to capitalise on their respective advantages for better sentiment prediction (Severyn & Moschitti, 2015).

Interpretability:

Even while neural network models have demonstrated impressive results in sentiment analysis, interpretability is sometimes hampered by their black-box design. Gaining confidence and faith in a system, particularly in situations where choices influence people's lives or enterprises, requires an understanding of why a model generates a certain forecast.

Integration of Language Models with Prompt Engineering:

Due to its capacity to produce cogent explanations or arguments for model predictions, Language Models with Prompt Engineering (LLM) has drawn interest. GPT-3, a cutting-edge autoregressive language model, was presented by Brown et al. (2020) and showed remarkable results in a range of natural language processing tasks. In order to produce perceptive explanations for sentiment predictions, recent studies have investigated the integration of LLM with deep learning models for sentiment analysis (Kapoor et al., 2021). The objective of researchers is to improve the interpretability and transparency of model predictions by integrating LLM into sentiment analysis workflows.

Creation of User Interfaces for Text Analysis Systems:

Performance:

Sentiment analysis models are frequently evaluated using conventional measures such as precision, recall, accuracy, and F1-score. However these measurements could not adequately convey the subtleties of sentiment analysis assignments, particularly when misclassification costs or class imbalances are substantial.

By allowing users to enter text data, view findings, and understand model predictions, user interfaces are essential to enable engagement with text analysis systems. Popular frameworks for creating frontend and backend elements of web applications include Flask and React, respectively. To improve the user experience, the interface design placed a strong emphasis on responsiveness, interaction, and simplicity.

2.2. REVIEW ON EXISTING SYSTEMS

Approach	Description	Advantages	Challenges
Rule-based Approaches	Sorting text according to sentiment categories by applying preset guidelines or patterns. frequently make use of human rule construction or sentiment lexicons.	Transparent and comprehensible Can be customised for certain domains and application cases	Having problems expressing emotions in a context-dependent manner The challenge of establishing and implementing norms
Lexicon-based Methods	Pairing words with items in dictionaries or sentiment lexicons that have been annotated with sentiment polarity ratings to analyse textual data.	Computing effectiveness Interpretable and transparent- Easy to put into practice	Restricted coverage for emotion expressions that are context- or domain-specificUpdating and maintaining lexicons may be challenging.
Machine Learning Algorithms	Text may be classified into sentiment categories by using supervised learning algorithms like Support Vector Machine, Naive Bayes, and Logistic Regression that have been trained on labelled datasets.	Capability to record intricate links in data With enough training, may generalise to unknown data	Large volumes of tagged data are needed for training. Limited capacity to adjust to new environments and domains Difficulty in selecting models and feature engineering
Deep Learning Neural Networks	For sentiment analysis, use neural network designs like LSTM and Bi-LSTM, as well as Convolutional and Recurrent neural networks (RNNs).	Capable of identifying complex correlations and patterns in text data. May perform better than conventional machine learning models.	Large volumes of data are needed for training. Prone to overfitting in the absence of appropriate regularisation methods. Complexity of computation and resource needs

Table 1 Table comparing Existing Systems

Compared to conventional methods like lexicon-based techniques, rule-based methods, and algorithms that use machine learning for sentiment analysis, neural networks trained using deep learning provide a number of benefits. Deep learning networks, exhibit superior effectiveness when

capturing complex connections and patterns within textual data, despite the fact that each approach has pros and cons of its own.

It's crucial to recognise the drawbacks of deep learning networks, though, including the need for a lot of training data, the possibility of overfitting, and computationally demanding requirements. Notwithstanding these obstacles, developments in hardware acceleration, model topologies, and regularisation strategies have made deep learning algorithms more useful and reachable for sentiment analysis applications.

Therefore, choosing deep learning neural systems as the main emphasis seems reasonable given the project's objective of offering an in-depth comparative study of many models used in deep learning for sentiment analysis. The project intends to determine the best model for analysing sentiment tasks by investigating and assessing several deep learning designs and methodologies, taking into consideration aspects like accuracy, comprehension, and scalability.

3. CASE STUDY:Machine Learning Models for Sentiment Analysis Research

Title: Improving Sentiment Evaluation for Label Development and Interpretation using LSTM, CNN, Bi-LSTM, & LLM-GPT

Intoduction:

The analysis of sentiment, which focuses on removing subjective details from text data such as views, views, and feelings. is an essential part of natural language processing. This case study explores the application of sophisticated deep learning models for sentiment detection tasks, including GPT, long-short-term memory (LSTM), Convolutional Neural Network (The CNN network), Bidirectional Long Short-Term Memory (Bi-LSTM), and Language Models (LM). The project's main goal is to

classify textual information into positive and negative sentiment categories and offer justifications for the labels that have been assigned.

Methodology:

- LSTM and CNN Integration: In this research, a hybrid LSTM+CNN model is constructed to improve sentiment evaluation accuracy by incorporating temporal dependencies with local characteristics in text data. This builds on the work of (Salur &Aydin 2020), who demonstrated the usefulness of LSTM and CNN in emotion classification.
- Dual-LSTM for Enhanced Sentiment Categorization: For sentiment analysis, a Stacked Bidirectional LSTM approach is used, taking influence in (Zhou et al., 2019). By helping to extract sentiment characteristics using Chinese microblog data, this approach enhances the ability to categorise feelings into positive and negative groups.
- Reasoning Generation using LLM-GPT: The work highlights the reasoning powers of language models such as GPT-4 by incorporating ideas from (Truhn, 2023). The project attempts to improve the understanding of the sentiment analysis method by providing reasons for the sentiment labels provided by incorporating GPT for reason generation.

Results:

- The hybrid LSTM+CNN model outperforms conventional techniques in terms of improved sentiment classification accuracy.
- The Bi-LSTM model improves sentiment analysis performance by efficiently capturing long-distance relationships in text data.
- The incorporation of LLM-GPT into reason generation yields significant justifications for the anticipated sentiment labels, thereby augmenting the system's transparency in sentiment analysis.

Conclusion:

The ramifications of these results imply that sentiment analysis performance and transparency may be greatly enhanced by using a comprehensive method that combines several deep learning architectures and language models. Subsequent investigations may delve into enhancing these models, examine supplementary integration tactics, and examine their suitability for practical situations in other fields. Furthermore, there exists the possibility to investigate the scalability and effectiveness of these models in managing extensive sentiment analysis assignments and customising them to particular business or domain demands.

4. TECHNICAL EXPLAINATION AND IMPROVEMENTS FOR DNN MODELS.

4.1 LSTM

4.1.1 LSTM STRUCTURE

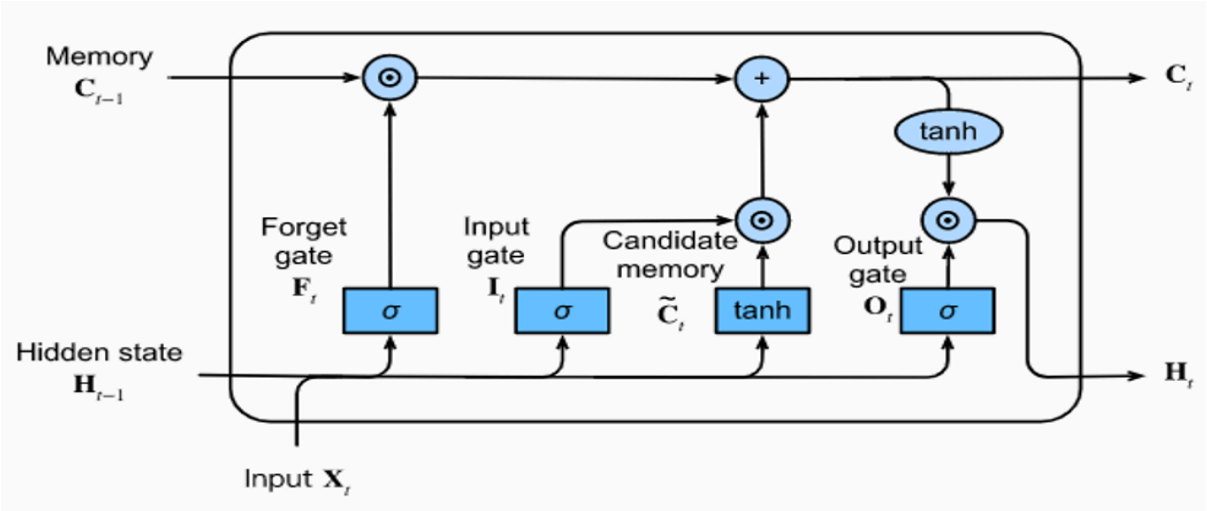


Figure1 Structure of the LSTM Model

LSTM networks, a particular kind of RNN, were created to solve the issue of disappearing gradients and better represent long-distance interactions in sequences. A more advanced storage cell that can

select what information to learn and discard across time steps is offered by LSTMs. The entry gate, recall gate and output gate are the three main parts of an LSTM cell. These gates regulate the amount of data flow and the amount of prior information that is lost or kept at each time step. Because of their design, LSTMs are especially well-suited for jobs requiring situational and memory recall as they can record pertinent information over extended periods.

4.1.1 MODIFICATIONS FOR EXISTING MODEL:

Adding a layer that embeds and dropout layers to improve the deep neural network model's performance for sentiment analysis. By learning dense representations of the words in the input vocabulary, the embedding layer enables the model to efficiently capture the semantic links between words. An LSTM layer is placed after the embedding layer in order to extract sequential dependencies from the input text. Dropout regularisation is used in both the LSTM layer and a later dropout layer to prevent overfitting. During training, dropout randomly removes a portion of the output from the LSTM units, encouraging generalisation by lowering dependence on particular features or patterns. Together, these changes strengthen the model's capacity to represent and comprehend the underlying sentiment in textual input, which leads to improved results in tasks involving sentiment analysis.

4.1.2. LSTM FOR SENTIMENT ANALYSIS:

Because they can handle sequential data, record long-term relationships, automatically extract features, manage sequences of different lengths, and give contextual comprehension, LSTM (Long Short-Term Memory) models are essential in sentiment analysis. Based on empirical data, it appears that LSTM-based models perform better in sentiment evaluation than more conventional methods and less complex neural network designs.

4.1.3. COMPLEXITY IN LSTM :

The magnitude of the input data, the LSTM model's architecture, and the difficulty of the sentiment analysis are some of the variables that affect how complicated Long Short-Term Memory (LSTM) networks are in sentiment analysis. Below are main factors need to be considered

Model Architecture: An LSTM model's architecture, which includes the amount of LSTM layers, dimensions of the LSTM units (i.e., the quantity of memory cells) can affect how complicated the model is. Larger parameter counts and more intricate structures typically have more computational complexity.

Input Data:The computational cost of LSTM operations is influenced by the amount of the input data, which is expressed in terms of words or tokens. Longer input sequences demand more calculations and memory since LSTMs process input sequences sequentially.

Training and Inference Times: An LSTM model's training time is influenced by several elements, including the size of the training dataset, the architecture of the model, and the optimisation technique (such as Adam) that was employed during training.

Memory and Processing Power: Using deep learning frameworks like TensorFlow, training intricate LSTM models on sizable datasets may demand a substantial amount of memory and processing power.

4.2.BI-LSTM

4.2.1 BI-LSTM STRUCTURE

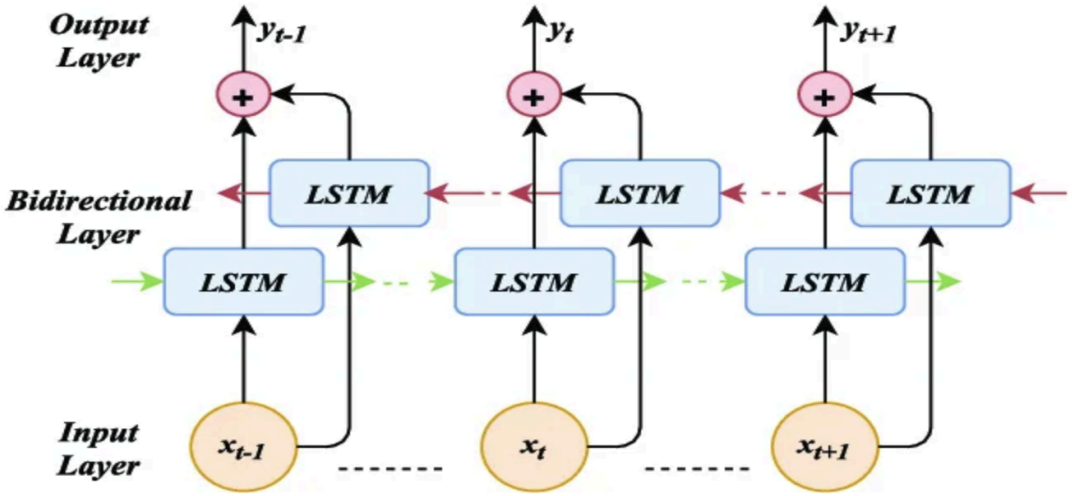


Figure 2 Structure of BI-LSTM Model

The Bidirectionally LSTM is an advanced architecture designed to process text or time series. LSTM networks use memory cells and gating methods similar to LSTM. The input sequence is processed in both forward and backward directions. This involves two different LSTM networks: a forward LSTM that moves through the sequence from beginning to end, and a reverse LSTM that moves through it from beginning to end. Both networks keep their own hidden and cell states at each time step, and the bidirectional layer's final output is often produced by concatenating or combining their outputs.

4.2.2. MODIFICATION ON BI-LSTM

The embedding and dropout layers that have been added to the Bidirectional LSTM (Bi-LSTM) model greatly improve its performance in sentiment analysis tasks. By using an embedding layer, the model can learn dense word representations and efficiently extract semantic associations from the input text. This enables the model to encode word meaning more thoroughly, improving its comprehension of textual material. In addition, the use of dropout layers presents a regularisation method designed to alleviate overfitting. Dropout enhances the model's capacity to generalise to new data by preventing it from being overly dependent on certain characteristics or patterns in the training set by randomly deactivating a portion of the units during training. These changes essentially increase the Bi-LSTM model's capacity to identify significant features from input text while resolving any overfitting issues, which eventually results in better sentiment analysis job performance.

4.2.3. BI-LSTM FOR SENTIMENT ANALYSIS

First of all, since the sentiment of a phrase can be impacted by its surrounding words, sentiment analysis frequently necessitates a comprehension of the text's sequential structure. By taking into account both past and future situations, they provide a contextual awareness that helps individuals to appreciate the subtle meanings of terms and phrases within their present context. Additionally, in order to extract features, Bi-LSTM models automatically learn a hierarchy of input sequences. This enables them to extract both high-level and low-level features that are pertinent to sentiment analysis. Finally, Bi-LSTM models ensure higher generalisation performance, especially with big datasets, by successfully addressing overfitting problems using approaches like dropout regularisation.

4.2.4. COMPLEXITY OF BI-LSTM

In sentiment analysis, the time and space complexity of a Bidirectional LSTM (Bi-LSTM) may be used to understand its complexity. The length of the input sequence determines the time complexity, which is linear but can also rise with the number of LSTM units and network depth. Due to its forward and backward processing, bidirectional LSTMs have twice as many parameters than unidirectional LSTMs.

Space complexity is based on the number of parameters in the model. Bi-LSTMs provide better contextual information capture performance in spite of their computational expense. Their implementation for jobs involving sentiment analysis is now possible thanks to advancements in technology and optimisation techniques.

4.3 CNN

4.3.1 CNN STRUCTURE

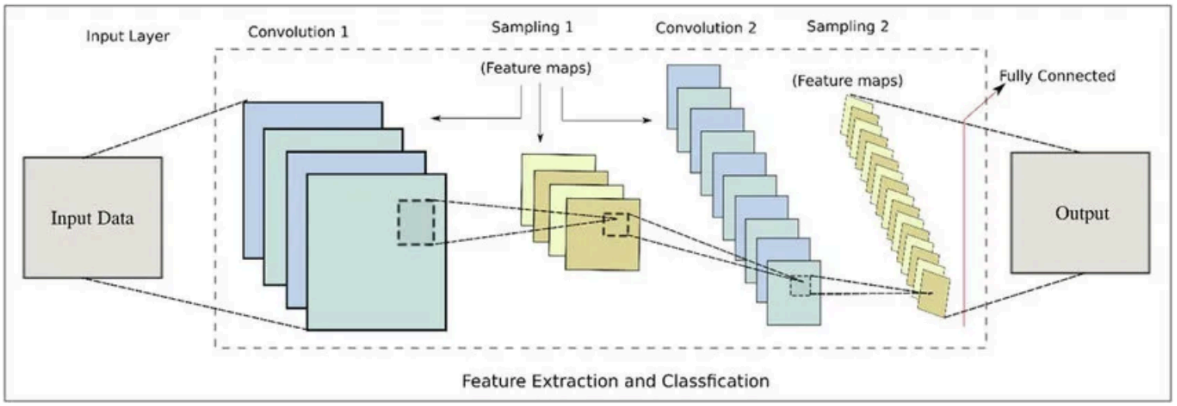


Figure 3 Structure of CNN Model

Convolutional layers are used by CNNs to extract local characteristics like combinations of words or phrases by scanning the input text with tiny windows or filters that move over the input sequence. Each filter creates a feature map that indicates the existence of particular patterns or structures in the input text by applying a convolution operation to a window of words. After that, these feature maps are run through activation functions, including Rectified Linear Unit (ReLU), to add non-linearities

and improve the model's capacity to identify intricate links in the data.Pooling layers pool the retrieved features by downsampling them, lowering the dimension of the feature maps while preserving the most pertinent information. They are frequently employed after convolutional layers. The output of the layers for convolution and pooling is often sent through any number of fully connected layers after its flattening into a one-dimensional vector. These layers then take up higher-level visualisations of the input text features. These layers combine the retrieved information and forecast the text's sentiment, frequently generating probability distributions across sentiment classes (positive, negative) using softmax activation.

In order to train CNNs for sentiment analysis, labelled datasets are frequently used. Through gradient-based optimisation techniques like Adam , the model acquires the ability to relate input text to matching sentiment labels.

4.3.2 MODIFICATION ON CNN MODEL

By adding more layers like pooling and dropout layers, a Convolutional Neural Network (CNN) model's sentiment analysis ability may be greatly enhanced. Convolutional layers are followed by pooling layers, such as MaxPooling or AveragePooling, which downsample the feature maps, extract dominating features.. This reduces computational complexity and overfitting while allowing the model to concentrate on the most pertinent data.

In order to regularise the model, dropout layers are included after convolution or fully connected layers. During training, dropout layers randomly deactivate neurons to provide robust and generalised learning. The model can efficiently extract hierarchical illustrations from the input text by deliberately combining pooling and dropping layers into the CNN architecture. This allows the model to capture both local and global characteristics that are crucial for sentiment analysis. These changes increase the model's ability to learn distinctive characteristics while limiting overfitting, which eventually improves sentiment analysis task performance and generalisation ability.

4.3.3 CNN FOR SENTIMENTAL ANALYSIS

In the field of sentiment analysis, convolutional neural network (CNN) models have become crucial, especially in the field of natural language processing (NLP). Their importance is derived from many major benefits. First off, CNNs are highly skilled at seeing local dependencies in sequential data—text, for example—which enables them to discover critical combination of words or phrases that are vital for sentiment analysis. Furthermore, these models generate features—such as word frequency, certain phrases, or syntactic patterns—from the input data to represent different language aspects expressing sentiment. CNNs may also be trained to learn text input in a hierarchical representation structure, whereby they first learn low-level properties and then progressively capture higher-level abstractions in deeper layers, such sentiment markers. Additionally, CNNs are better suited for sentiment analysis tasks—especially when dealing with huge datasets—than classic recurrent neural networks (RNNs) since they process text input more computationally efficient.

4.3.4 COMPLEXITY OF CNN MODEL

The following is a summary of Convolutional Neural Network (CNN) model complexity in sentiment analysis. The quantity of input data, filter sizes, and number of convolutional layers all affect how much a CNN costs to compute in terms of time complexity. Although CNNs are capable of processing vast amounts of text input effectively, deeper topologies or bigger filter sizes may result in an increase in time complexity.

When it comes to space complexity, CNNs need memory to hold various parameters, such as bias and weight variables, which can increase in size as the model architecture gets bigger. But when compared with recurrent neural networks (RNNs) like LSTMs, CNNs usually have a smaller memory footprint, which makes them more memory-efficient.

4.4.1 LSTM+CNN STRUCTURE

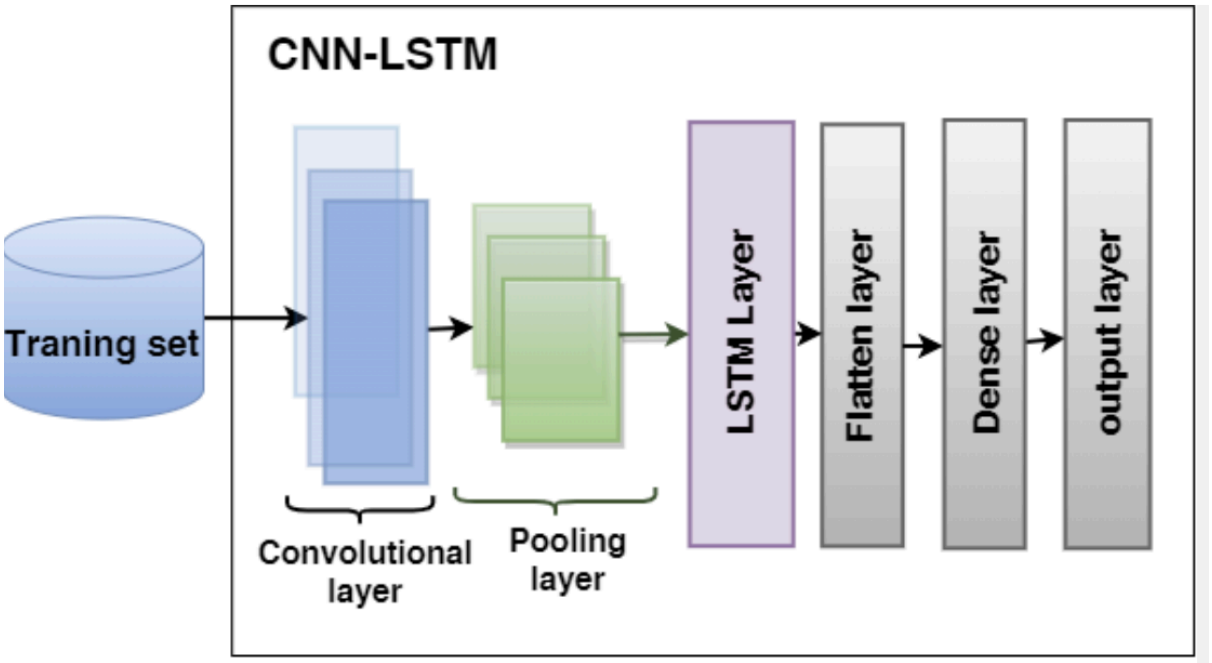


Figure 4 Structure of LSTM+CNN Model

In order to maximise performance in sentiment assessment and other tasks related to natural language processing, LSTM networks (Long Short-Term Memory) and Convolutional Neural Networks, can be combined to create the potent LSTM+CNN model structure. The CNN layer in this architecture is where the model starts, and it is in charge of feature extraction. Convolutional filters are used by this layer to scan the text being input, find local patterns, and extract relevant characteristics. By identifying certain patterns, such as combination of words or n-grams, each filter helps to create feature maps which capture various facets of the input text. For sequential modelling, an LSTM layer is incorporated after the CNN layer. The long-term dependencies in the text are captured by the LSTM as it successively analyses the retrieved characteristics, taking word order into account.

4.4.2 MODIFICATIONS ON LSTM+CNN MODEL

An LSTM+CNN model can perform much better in sentiment analysis assignments if it has extra layers, such as layers that pool data, dropouts layers, and dense layers. Pooling layers—like max pooling—play a crucial role in lowering the dimensionality of feature maps produced by convolutional layers, keeping the most important features and removing the rest. By collecting the most crucial elements of the input data, this enhances the generalisation and efficiency of the model. Dropout layers also function as a regularisation strategy, reducing the possibility of overfitting by arbitrarily deactivating a portion of units during training. This keeps the model from learning the training set and improves generalisation to new data. Furthermore, by include thick layers, complicated decision boundaries and non-linear transformations may be integrated, allowing the model to learn more complex patterns in the data and provide predictions that are more accurate.

4.4.3 LSTM+CNN MODEL FOR SENTIMENT ANALYSIS

Key elements of neural network models used for sentiment analysis include CNN and LSTM, each of which brings special advantages to the job at hand. CNNs are very good at spotting minute patterns and characteristics in textual data, which makes them ideal for sentiment analysis tasks like text classification. They effectively extract essential textual features, such word or sentence combinations that convey emotion. LSTMs acquire text representations that take into account word order and reflect the temporal dynamics of language—both of which are essential for comprehending sentiment.CNNs and LSTMs work together to create a potent sentiment analysis model. Together, these enable the model to use hierarchical representation to understand sentiment at different granularities.

4.4.4 COMPLEXITY OF LSTM+CNN

Addressing several technological challenges is necessary while building an LSTM+CNN model. First, to balance model complexity and computational efficiency, carefully examine the hyperparameters to choose, such as the quantity of LSTM and CNN layers, the size of the convolutional filter, and the

degree of detail of the LSTM hidden states. Tokenization, padding sequences, and maybe embedding word vectors are additional computationally demanding preparation steps for text data, particularly for big datasets. Furthermore, complex choices about the extraction of features, hierarchical representation, and and information flow across layers must be made when developing the architecture to successfully combine the CNN and LSTM components. In order to avoid overfitting, regularisation strategies like dropout and batch normalisation must be used in order to strike a balance between model capacity and generalisation performance. Additionally, choosing the best optimisation algorithms, learning rates, and batch sizes is essential to optimising the model's training process. It also entails keeping an eye on training dynamics like gradient stability and convergence. Finally, scalability, latency, and resource limits must be taken into account when implementing the model in real-world applications. This calls for optimisation strategies such model compression and hardware deployment on specialised platforms. In order to provide a reliable and efficient solution for sentiment assessment along with various natural language processing tasks, developing an LSTM+CNN model often entails overcoming a wide range of technological obstacles.

4.5 COMPARISION OF DEEP NN MODELS

Model	Description	Advantages	Challenges
LSTM	Because short- and long-term memory networks can recognise sequential relationships in textual material, they are a good fit for jobs involving sentiment analysis.	Capable of capturing long-term relationships Good at handling sequential material, such sentences and paragraphs	May have trouble identifying intricate links in textual data- Requiring a lot of computation, particularly for big datasets
Bi-LSTM	Bidirectional Long Short-Term Memory networks improve the model's comprehension of sequential input and its sentiment analysis applications by taking into account both past	Effectively capture sequential dependencies Contextual awareness is enhanced when historical and future settings are taken into account.	More computational complexity with comparison to models of unidirectional LSTMs Due to its bidirectional nature, training could need more data.

	and future contexts.		
CNN	Convolutional neural network models are a good fit for sentiment analysis because they can use convolutional filters to identify local patterns and correlations in text input.	Capture local patterns and relationships in text data efficiently. Architecture that is computationally efficient, especially for big datasets	Possible difficulty identifying long-term relationships in text data Requires meticulous adjustment of filter sizes and hyperparameters for best results.
LSTM+CNN	By using the advantages of both models, the combination of LSTM and CNN architectures improves sentiment analysis by identifying both short- and long-term correlations in text data.	Effectively captures both short-term and long-term dependence Integrates CNN and LSTM architectures' best features for thorough text data analysis	Enhanced complexity in the model design and training procedure demands meticulous hyperparameter adjustment and optimisation in order to strike a balance between computational economy and performance.

Table 2 Table comparing DNN Models

4.6 NAIVE BAYES

Naive Bayes is made to deal with datasets that are unbalanced and have a predominance of some classes. For jobs involving text categorization, it is very helpful.

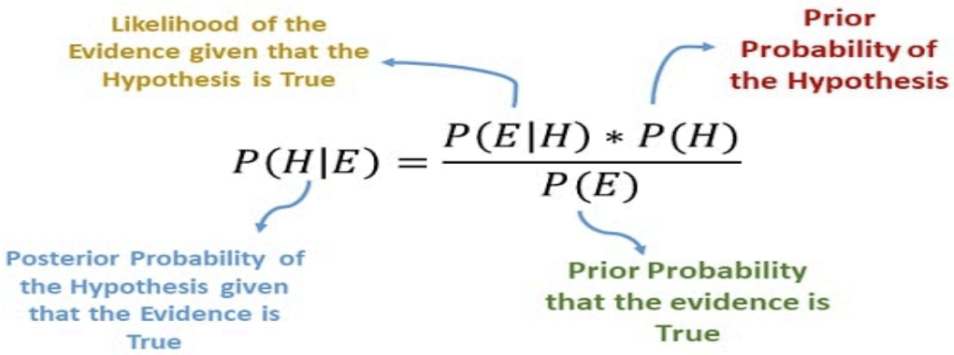


Figure 5 Formulae for Naive Bayes

P(H| E) = Chance of event C occurring in the case that event E occurs P(E | C) = Chance of event E occurring in the case that event H occurs P(H) = Chance of event H occurring P(x) is the likelihood that event E will occur.Thus, the Bayes Theorem provides a method for determining the Conditional Probability. The core of the Naive Bayes hypothesis is the Bayes theorem.

Variants of Naive Bayes Models :

Feature	ComplementNB	MultinomialNB	BernoulliNB
Type of Features	Designed for imbalanced datasets and works well with text classification tasks where features are frequency counts	Suitable for text classification tasks where features represent the frequency of words or tokens	Suitable for binary feature vectors, where features represent whether a particular term appears in the document or not
Probability Estimation	Estimates the conditional probability of observing the document features given each class by considering the complement of the class frequencies	Estimates the probability of each feature occurring in documents of each class using the frequency of terms	Estimates the probability of each feature occurring in documents of each class using binary features (presence or absence of terms)
Handling Imbalanced Datasets	Explicitly addresses imbalanced datasets by considering the complement of the class frequencies during training	Does not explicitly handle imbalanced datasets but can be affected by class imbalance	Can handle imbalanced datasets to some extent but may not perform as well as ComplementNB for highly imbalanced datasets
Model Assumptions	Assumes that features are conditionally independent given the class, similar to other Naive Bayes models, but adjusts the probability estimation to address class imbalances	Assumes that features are multinomially distributed given the class, which is suitable for text classification tasks	Assumes that features are Bernoulli distributed given the class, which is suitable for binary feature vectors
Performance	Tends to perform well on imbalanced datasets and text classification tasks with skewed class distributions	Commonly used for text classification tasks and performs well when the features represent the frequency of words or tokens	Useful when dealing with binary feature vectors and performs well in certain text classification tasks, especially when considering the presence or absence of terms

Table 3 Table comparing Variants of Naive Bayes Models

4.7 LLM ARCHITECTURE

This project leverages lessons from relevant research to construct a Language Model (LLM) for explanation generation according to labels given by an LSTM model that is used for Amazon food reviews, utilising the LangChain framework. The reasoning capabilities of pre-trained languages models (LLMs) in complicated tasks were investigated by Chen et al. (2022). This research may be expanded to see whether LLMs are useful in producing justifications for emotion tags in Amazon food reviews. Furthermore, Saparov & He (2022) provided a simulated question-answering dataset to methodically assess LLMs' reasoning abilities; this dataset may be used to produce justifications for sentiment labelling in reviews.Rajasekharan et al. (2023) provided evidence of how well LLMs extract information from language, a technique that may be applied to deduce the rationale behind sentiment labelling in Amazon food reviews. This framework may be used to provide a sequence of steps for reasoning that can be understood in relation to the sentiment labels that the LSTM algorithm in the Amazon food review dataset assigned.

4.8 EVALUATION METHODS

Different assessment techniques and metrics may be applied to evaluate the sentiment analysis performance of LSTM, CNN, Bi-LSTM, and LSTM+CNN models. Using information from reliable sources, we may outline each model's assessment procedures and justifications as follows:

Accuracy:

The total accuracy of the algorithm's predictions is measured by accuracy. It determines the proportion of accurately predicted cases to all instances.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Precision:

Precision evaluates how well the model predicts favourable outcomes. It determines the proportion of successfully predicted positive cases to all positive cases that were expected.

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

Recall:

The model's recall quantifies its capacity to accurately distinguish positive examples from all real positive examples. The ratio between the overall amount of actual positive outcomes and the properly projected positive instances is computed.

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

F1 score:

The F1-Score provides a balance among accuracy and recall by taking the harmonic median of the two criteria. It shows how accurate the model is in terms of recall and precision.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

These measures indicate the models' overall ability to reliably categorise attitudes into either favourable or adverse categories in LSTM (Nurrohmat & Sn, 2019). In a similar vein, CNN models are evaluated for their efficacy in sentiment classification tasks employing above mentioned metrics; specifically, precision and recall measure the model's capacity to accurately identify positive sentiments and recall them from all real positive sentiments, accordingly (Arunabala et al., 2019).As compared to this, Bi-LSTM models are assessed using a variety of measures, including as metrics, which offer a thorough evaluation of their efficacy in sentiment analysis tasks (Nawaz et al., 2021). Finally, the efficacy of the hybrid LSTM+CNN model in sentiment classification is assessed using measures mentioned (Salur & Aydin, 2020). These evaluation techniques make sure that every model's effectiveness for sentiment analysis is thoroughly evaluated, which helps to clarify the models' strengths and weaknesses in this area.

5. REQUIREMENTS SPECIFICATION

Essential:

1.Developing Deep Learning Models for Sentiment Analysis (CNN, LSTM, LSTM + CNN) & Naive Bayes Model:

- Data Preparation: Preprocess the dataset to ensure consistency and quality.
- Word Representation: Represent words as dense vectors using techniques like Tokenizer to capture semantic associations.
- Model Architectures: Develop CNN, LSTM, and LSTM + CNN, Naive Bayes architectures for sentiment analysis.
- Training and Evaluation: Train the models on a dataset divided into test, validation, and training sets to evaluate their performance.

2.Langchain for Sentiment Analysis:

- Utilize Langchain, a framework for natural language processing tasks, to train Language Models with Prompt Engineering (LLM) for sentiment analysis.
- Implement meta-learning approaches to enhance the performance of LLMs over time, ensuring more precise sentiment analysis outcomes.

3.Enhancing Sentiment Analysis Prompts for Explanatory Insights:

- Improve sentiment analysis prompts to generate explanations for sentiment classifications in addition to categorizing them.
- Enhance the queries to elicit justifications or explanations for the sentiment labels, providing deeper insights into the model's decisions.

Recommended:

Web Interface:

- Design a user interface for the sentiment analysis project with integrated prompts for obtaining reasons behind sentiment labels.
- Ensure the interface is user-friendly, facilitating easy interaction with the sentiment analysis system.
- User Authentication and Portfolio Management :Implement user authentication to provide secure access to the sentiment analysis system through customized accounts.
- Develop a portfolio feature where users can track their sentiment analysis tasks and results for better management and analysis.

Optional:

- Performance Comparison of Different Models:Conduct a comprehensive comparison of various models based on performance measures such as accuracy, precision, recall, F1-score, and computing efficiency.
- Improve Trust and Comprehension of Model Predictions:Implement interpretability approaches, such as feature importance analysis, to enhance trust and comprehension of the model's predictions by shedding light on the decision-making process.

By fulfilling these requirements, the sentiment analysis project aims to develop robust deep learning models, enhance the interpretability of sentiment analysis results, and provide users with a user-friendly interface for efficient interaction with the system.

5.1. SYSTEM REQUIREMENTS

Hardware specifications:

CPU: Modern multi-core CPUs, such the Intel Core i5 or above, are capable of handling data processing and modelling tasks with ease. Running the Flask web application, managing data, and training models all require at least 8 GB of RAM.

Storage: A minimum of 10GB should be set aside for the storage of datasets, trained models, and web app resources.

(Details optional) Graphic Card With a dedicated GPU, the training and evaluation processes of the LSTM model may be accelerated. For software creation and delivery, the possible operating systems are Windows, macOS, or Linux (Ubuntu is recommended).

Python is the primary programming language used in data analysis, modelling, and online app development. Version 3.7 or later is required.Kera 2.12.0 is needed.

Integrated Development Environments (IDEs): IDEs for coding and debugging, such as Jupiter Notebook,Kaggle Notebook, Visual Studio Code.

6. SYSTEM REQUIREMENTS

6.1 TECHNICAL SPECIFICATION

Technical specification is provided below

TYPE	NAME	VERSION	SUMMARY
Programming Language	Python	3.11.x	Python is a popular and flexible language that's often utilised in machine learning applications. It provides a robust framework for implementing various machine-learning techniques and algorithms.
	REACT	17.0.2	Used for developing Interactive Web Application.
Framework	Flask	2.3.2	Flask is a lightweight, adaptable Python micro web framework

	Langchain	0.1.x	designed to make building web applications simple and rapid. It's a fantastic choice for developing web apps, APIs, and prototypes. Integration of LLM into local application.
OS(Operating System)	Mac	M1	Latest Version
Python Packages(Package Installer)	pip	3	For Python package installation.
IDE	VS Code, Kaggle Notebook	1.81	Microsoft's IDE is the best for creating Flask and Kaggle notebooks for creating and refining machine learning models.
Other tools used	draw.io, Lucid	-	draw.io is used for developing related diagrams

Table 4 Technical Specification

6.2. LIST OF LIBRARIES USED

Category	Tools	Purpose
Data Retrieval	Kaggle	Publicly Available Data is used for training Deep Neural Network Model

Data Preparation	Pandas, Numpy,bs4(beautifulsoap),nltk	The Python data analysis and modification toolkits provide flexible structures for information and tools to enable efficient work with organised data.
Data Visualization	Matlab Plot	A well-known Python package for creating interactive, static plots and publication-quality visualisations.
LSTM ,CNN,BI-LSTM and CNN+LSTM Models	TensorFlow/Keras 2.12.0	The models for deep learning and neural networks may be more easily created with the help of the well-known open-source TensorFlow/Keras machine learning framework.
LLM Integration	LangChain	integrates GPT into personal Application.
Web App Development	flask (Flask , Flask-login , SQLAlchemy)	Package to develop web app

Table 5 List of Libraries

7.METHODOLGY AND PROCESS FLOW

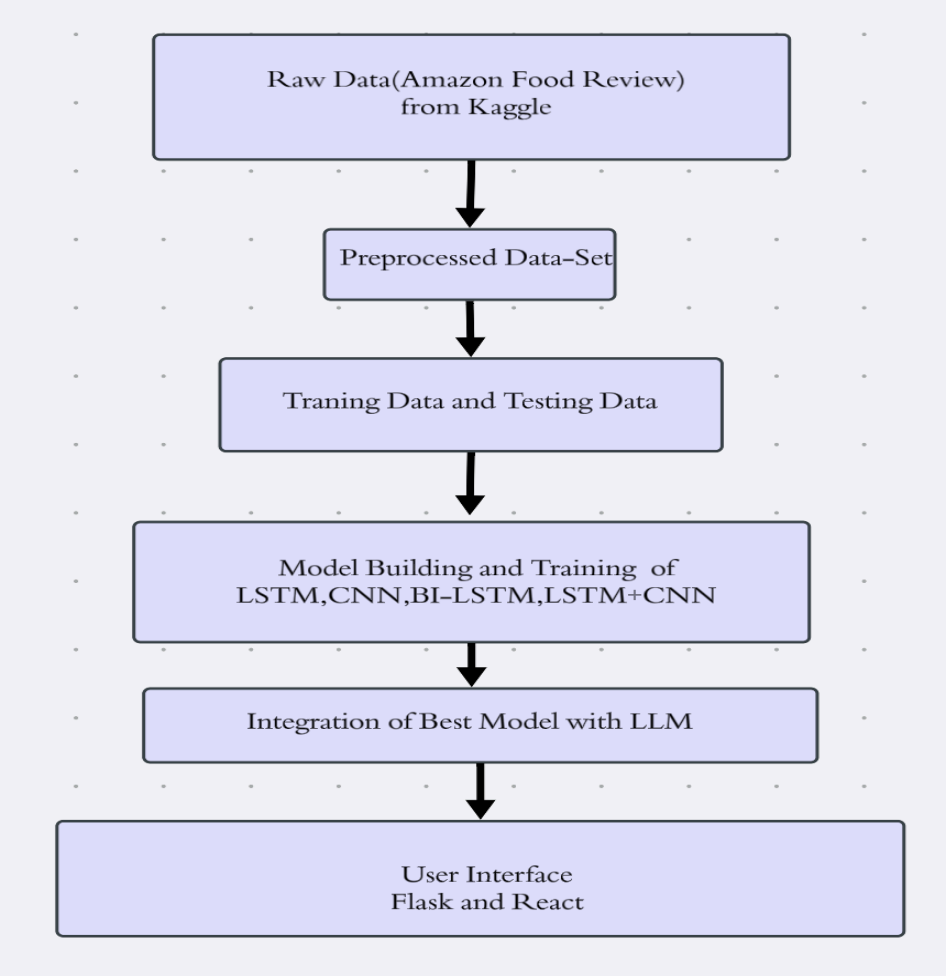


Figure 6 Methodology & Process Flow

7.1. DATASET EXPLORATION

For several reasons, the widely used and publicly accessible Amazon Food Reviews dataset is crucial for training LSTM, Bi-LSTM, CNN, and LSTM+CNN, Naive Bayes models for sentiment analysis. First off, the dataset is made up of a sizable number of customer reviews on different food goods that have been bought on Amazon. This variety and richness of textual data make it an excellent resource for sentiment analysis. Models may efficiently learn and generalise across multiple emotion polarity thanks to the vast range of sentiments represented in these evaluations, which vary from positive to severe negative.

Furthermore, the dataset provides a real-world environment that reflects authentic consumer views and encounters, which is essential for developing training models that effectively represent sentiment

in real-world situations. LSTM, the Bi-LSTM, CNN, for example, and LSTM+CNN models may be trained to identify idiomatic phrases, nuanced linguistic signals, and specific to the domain language patterns that are common in evaluations about food by using real data.

Furthermore, the Amazon Food Reviews dataset's labelled sentiment annotations make supervised learning easier and allow for the supervised development of sentiment analysis models. By learning to correlate textual cues with associated sentiment labels, models may reliably predict the polarity of sentiment on unseen data.

Moreover, the extensive utilisation of the Amazon Food Reviews dataset in scholarly investigations and commercial benchmarks promotes consistency and repeatability across diverse sentiment analysis methodologies. In order to promote advancement and innovation in sentiment analysis approaches, researchers and practitioners can compare their Developed Models against the body of current literature and standardised assessment metrics.

7.2 PREPROCESSING STEPS

Using Pandas to explore datasets to learn about and comprehend the features, substance, and structure of the dataset. Using procedures like read_csv() tand head(), tail(), and sample(), examined the contents of the dataset and imported it with ease using Pandas.

Due to constraint in computational resources and time 200000 samples are considered for analysis out of 500000 samples available. 200000 samples are picked randomly from the dataset.

The distribution and summary features of numerical columns in the dataset explored employing the describe() method in Pandas. This aided in locating possible problems, such as missing numbers or Statistics of Dataset, that could need preprocessing or more research.

During the data cleaning step, Pandas offered capabilities to manage missing values (dropna(), fillna()), remove duplicates (drop_duplicates()), and filter rows based on criteria (loc[], iloc[]). These programmes were used to clean and preprocess the dataset, ensuring data integrity and consistency before it was analysed.

TEXT CLEANING

In sentiment analysis, text cleaning is an essential preprocessing technique that eliminates noise and extraneous information from textual data. Python text cleaning activities are often performed with the Natural Language Toolkit (NLTK) and the BeautifulSoup (bs4) module.

Extracted text content from HTML pages and removed styling and HTML elements by using BeautifulSoup. By doing this, all HTML-related artefacts that might not be useful for sentiment analysis are removed.

After HTML elements are eliminated, NLTK(stopwords) is used to eliminate punctuation, special characters, and URLs, among other text cleaning tasks.

TOKENIZATION:

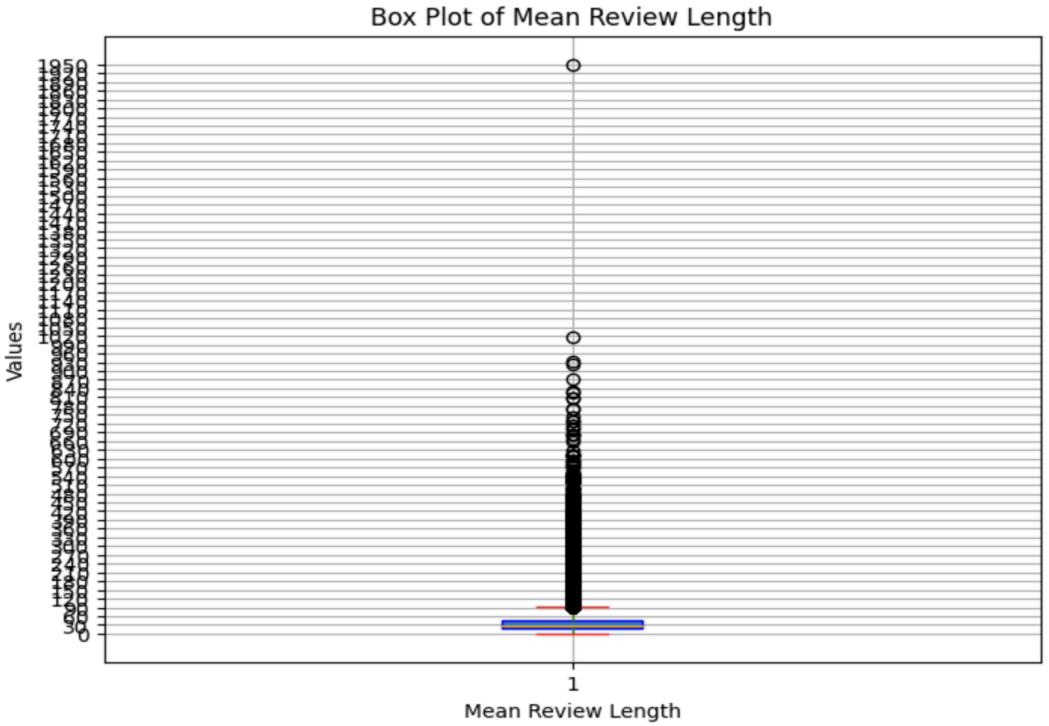
Tokenization is an essential preprocessing step in the natural language processing, especially in applications like sentiment analysis. The texts_to_sequences function is essential to tokenization, especially in frameworks like Keras' Tokenizer class. This is how it transpires:

Every text in the input list is converted into an integer sequence by using **texts_to_sequences**. This conversion is dictated by the language that was previously created. The text replaces each word or token with the relevant integer index from the lexicon. The outcome is a list of sequences, each of which is an integer representation of a text that has been transformed, and each integer indicates the index of a word or token in the vocabulary.

PADDING & TRUNCATION ON TEXT:

For the Amazon Food Reviews collection to be uniformly composed of text, input sequences must be padded or truncated. This is especially important when using the dataset to train sentiment analysis models.

As calculated the outliers most of the text length of the reviews is around 38.61 the standard length as input selected for the models choosed is 40 tokens.Standardising the sequence length to a specific amount 40 tokens, by padding or truncation assures consistency and makes training batch processing more effective, as the length of the text reviews in the dataset might vary greatly.




```
In [32]: df["Review_length"].mean()

Out[32]: 38.61781
```

Figure 7 Box Plot on review Text Length

Added zeros to the end of each sequence until it achieves the required length (here it is , 40) is known as padding shorter sequences with zeros. This preserved uniformity in the dataset by guaranteeing that every sequence, regardless of its initial length, has the same length.

However, in order to establish consistency, larger sequences can be truncated by removing surplus tokens that exceed the intended length of 40. This enabled efficient batch processing during training by limiting the length of each sequence to a predetermined size.Below is the Example outputafter padding

```
Actual Review:
 could possibly best tasting chip ever eaten nice smokey bacon flavor would make
sure item always stocked pantry price high amount get would sell amazing chip sta
tes would highly recomend someone paying price one time try

tweet sentence represented in terms of tokens and not padded :
  [38, 1908, 23, 184, 471, 102, 554, 65, 2648, 1108, 10, 8, 29, 112, 260, 79, 241
8, 1174, 24, 118, 196, 14, 8, 754, 330, 471, 1217, 8, 171, 3657, 499, 1118, 24,
4, 20, 33]

After padding :
 [   0    0    0    0   38 1908   23  184  471  102  554   65 2648 1108
   10    8   29  112  260   79 2418 1174   24  118  196   14    8  754
  330  471 1217    8  171 3657  499 1118   24    4   20   33]
```

Training and Test data sets:

20% of the samples are assigned to the test dataset and the remaining 80% to the training dataset when the entire dataset(200000) is split into training and test datasets using the scikit-learn train_test_split function with a ratio of 0.2 (or 20%) for the test dataset. This ensured that there is enough data available for training the model and that some is set aside for assessing its effectiveness.

Shape of the Processesed data:

Dataset Type	Sample Number	Shape
Sampled Dataset(Total)	200000	(200000,40)
Training dataset	160000	(160000,40)
Test dataset	40000	(40000,40)

Table 6 Shape of Input

7.3 MODEL BUILDING

Epoche Configuration:

- In order to evaluate the model's performance throughout a range of training times, it is trained using five and ten epochs.The model has more time to spot complex patterns in the data when there are 10 epochs instead of 5, which resulted in better performance.
- Higher epochs increase overall performance on validation or test sets by enabling stronger prediction generalisation to unknown data.

- For the indicators of overfitting as you increase the number of epochs to make certain the model doesn't retain its efficacy with unknown data and forgets what it learned from the training set.
- To minimise the danger of overfitting and maximise the model's capacity for generalisation, 10 epochs of the provided data are used.

LSTM Model:

When building an LSTM model for sentiment analysis, its architecture is influenced by a number of important elements. Below is a summary of the particular design decisions that were taken for the model:

Layer Configuration:

- One LSTM layer is used to reduce the overfitting hazards connected to deeper designs.
- In order to balance managing complexity with capturing temporal correlations, the model given 128 units per layer. The activation function of tanh is utilised.
- The inclusion of an embedding layer, using vocab_size and embed_size as inputs was done which improves the model's capacity to acquire word representations.
- After the LSTM layer, dropout regularisation is used to randomly deactivate units during training in order to minimise overfitting.To help in learning complicated patterns in the data, the model has two dense layers attached to it.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                 Output Shape
=================================================================
 embedding_2 (Embedding)      (None, None, 128)

 lstm_3 (LSTM)                (None, 128)

 dense_2 (Dense)              (None, 2)

=================================================================
```

Figure 8 Build LSTM Model

Compilation Settings:

- The compile approach ensures effective training by configuring necessary parameters for the learning process.
- The loss function that was used, sparse_categorical_crossentropy, computes the cross-entropy across true labels and predicted probability distributions, making it suitable for classifying text into many categories.
- The Adam optimizer is used in deep neural network optimisation for sentiment analysis because of its flexibility in managing sparse gradients and modifying learning rates per parameter.
- Evaluation measures, such accuracy, are defined in order to track the performance of the model throughout testing and training.

Training Model:

The model may acquire complex depictions of the training data by increasing the number of training epochs, which improves performance and generalisation on untrained data. To prevent overfitting and attain ideal performance, it is crucial to keep an eye on the model's output during epochs. so considering time computational resources epochs 5 and 10 are considered for this project

Epochs 5:

```
5000/5000 - 439s - loss: 0.3071 - accuracy: 0.8704 - val_loss: 0.2737 - val_accuracy: 0.8880 - 439s/epoch - 88ms/step
Epoch 2/5
5000/5000 - 436s - loss: 0.2223 - accuracy: 0.9088 - val_loss: 0.2718 - val_accuracy: 0.8902 - 436s/epoch - 87ms/step
Epoch 3/5
5000/5000 - 451s - loss: 0.1679 - accuracy: 0.9336 - val_loss: 0.2826 - val_accuracy: 0.8929 - 451s/epoch - 90ms/step
Epoch 4/5
5000/5000 - 460s - loss: 0.1202 - accuracy: 0.9538 - val_loss: 0.3241 - val_accuracy: 0.8813 - 460s/epoch - 92ms/step
Epoch 5/5
5000/5000 - 458s - loss: 0.0833 - accuracy: 0.9688 - val_loss: 0.3860 - val_accuracy: 0.8904 - 458s/epoch - 92ms/step
```

Figure 9 Training LSTM on 5 epochs

Epochs 10:

```
Epoch 1/10
5000/5000 - 770s - loss: 0.3060 - accuracy: 0.8706 - val_loss: 0.2662 - val_accuracy: 0.8912 -
770s/epoch - 154ms/step
Epoch 2/10
5000/5000 - 768s - loss: 0.2217 - accuracy: 0.9105 - val_loss: 0.2633 - val_accuracy: 0.8952 -
768s/epoch - 154ms/step
Epoch 3/10
5000/5000 - 770s - loss: 0.1669 - accuracy: 0.9344 - val_loss: 0.2816 - val_accuracy: 0.8909 -
770s/epoch - 154ms/step
Epoch 4/10
5000/5000 - 769s - loss: 0.1205 - accuracy: 0.9536 - val_loss: 0.3108 - val_accuracy: 0.8918 -
769s/epoch - 154ms/step
Epoch 5/10
5000/5000 - 771s - loss: 0.0836 - accuracy: 0.9689 - val_loss: 0.3735 - val_accuracy: 0.8938 -
771s/epoch - 154ms/step
Epoch 6/10
5000/5000 - 770s - loss: 0.0575 - accuracy: 0.9789 - val_loss: 0.4278 - val_accuracy: 0.8927 -
770s/epoch - 154ms/step
Epoch 7/10
5000/5000 - 770s - loss: 0.0405 - accuracy: 0.9851 - val_loss: 0.4790 - val_accuracy: 0.8893 -
770s/epoch - 154ms/step
Epoch 8/10
5000/5000 - 769s - loss: 0.0292 - accuracy: 0.9896 - val_loss: 0.5420 - val_accuracy: 0.8834 -
769s/epoch - 154ms/step
Epoch 9/10
5000/5000 - 770s - loss: 0.0221 - accuracy: 0.9921 - val_loss: 0.6486 - val_accuracy: 0.8842 -
770s/epoch - 154ms/step
Epoch 10/10
5000/5000 - 774s - loss: 0.0180 - accuracy: 0.9936 - val_loss: 0.6681 - val_accuracy: 0.8938 -
774s/epoch - 155ms/step
```

Figure 10 Training LSTM on 10 epochs

The model's prediction accuracy in relation to the actual desired values is gauged by the loss function. The loss value in this instance is 0.018, which shows that the model's predictions and the actual data agree rather well.As we can see is better compared to 5 epochs ie., 0.0833

BI- LSTM Model

Layer Configuration:

- In order to better grasp temporal linkages in the data, the model used bidirectional LSTM layers to gather information from both past and future contexts.

- To manage model complexity while capturing intricate patterns, two Bidirectional LSTM layers with 128 units each are used. To reduce overfitting, a 0.2 dropout rate is utilised.
- The use of an embedding layer enhances the model's comprehension of word representations by converting integer-encoded words into dense vectors.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape
=================================================================
 embedding_1 (Embedding)     (None, None, 128)

 bidirectional (Bidirectiona  (None, None, 256)
 l)

 bidirectional_1 (Bidirectio  (None, 256)
 nal)

 dense_1 (Dense)             (None, 2)
```

- ____

Figure 11 Build BI-LSTM Model

Compilation Settings:

- Through the computation of the cross-entropy between true labels and predicted probability distributions, the selected loss function, sparse_categorical_crossentropy, is appropriate for classifying text into categories.
- The Adam optimizer is chosen for optimising deep neural networks
- Evaluation measures, such accuracy, are defined to track the performance of the model during the training and testing stages.

Epochs 5:

```
5000/5000 - 639s - loss: 0.3051 - accuracy: 0.8727 - val_loss: 0.2718 - val_accuracy: 0.8882 - 639s/epoch - 128ms/step
Epoch 2/5
5000/5000 - 624s - loss: 0.2154 - accuracy: 0.9141 - val_loss: 0.2845 - val_accuracy: 0.8936 - 624s/epoch - 125ms/step
Epoch 3/5
5000/5000 - 623s - loss: 0.1534 - accuracy: 0.9409 - val_loss: 0.2914 - val_accuracy: 0.8916 - 623s/epoch - 125ms/step
Epoch 4/5
5000/5000 - 621s - loss: 0.1036 - accuracy: 0.9612 - val_loss: 0.3446 - val_accuracy: 0.8913 - 621s/epoch - 124ms/step
Epoch 5/5
5000/5000 - 624s - loss: 0.0673 - accuracy: 0.9753 - val_loss: 0.4085 - val_accuracy: 0.8867 - 624s/epoch - 125ms/step
```

Figure 12 Training BI-LSTM on 5 epochs

Epochs 10:

```
Epoch 1/10
5000/5000 - 1270s - loss: 0.3033 - accuracy: 0.8743 - val_loss: 0.2666 - val_accuracy: 0.8928 -
1270s/epoch - 254ms/step
Epoch 2/10
5000/5000 - 1256s - loss: 0.2157 - accuracy: 0.9138 - val_loss: 0.2573 - val_accuracy: 0.8967 -
1256s/epoch - 251ms/step
Epoch 3/10
5000/5000 - 1255s - loss: 0.1572 - accuracy: 0.9390 - val_loss: 0.2727 - val_accuracy: 0.8944 -
1255s/epoch - 251ms/step
Epoch 4/10
5000/5000 - 1279s - loss: 0.1080 - accuracy: 0.9594 - val_loss: 0.3216 - val_accuracy: 0.8963 -
1279s/epoch - 256ms/step
Epoch 5/10
5000/5000 - 1269s - loss: 0.0712 - accuracy: 0.9739 - val_loss: 0.3689 - val_accuracy: 0.8922 -
1269s/epoch - 254ms/step
Epoch 6/10
5000/5000 - 1245s - loss: 0.0489 - accuracy: 0.9824 - val_loss: 0.4655 - val_accuracy: 0.8877 -
1245s/epoch - 249ms/step
Epoch 7/10
5000/5000 - 1250s - loss: 0.0338 - accuracy: 0.9880 - val_loss: 0.5497 - val_accuracy: 0.8921 -
1250s/epoch - 250ms/step
Epoch 8/10
5000/5000 - 1252s - loss: 0.0263 - accuracy: 0.9908 - val_loss: 0.5848 - val_accuracy: 0.8884 -
1252s/epoch - 250ms/step
Epoch 9/10
5000/5000 - 1246s - loss: 0.0209 - accuracy: 0.9927 - val_loss: 0.6803 - val_accuracy: 0.8946 -
1246s/epoch - 249ms/step
Epoch 10/10
5000/5000 - 1252s - loss: 0.0159 - accuracy: 0.9944 - val_loss: 0.7141 - val_accuracy: 0.8927 -
1252s/epoch - 250ms/step
```

Figure 13 Training BI-LSTM on 10 epochs

In this case, the loss value is 0.0159, indicating a rather good agreement between the model's predictions and the actual data.As we can see, is superior than five epochs, or 0.0673.

CNN Model:

Specific design decisions impact the architecture of a CNN model, especially when building one for sentiment analysis. Below is a summary of the important elements considered:

Layers Configuration:

- To improve the model's perception of word representations, the embedding layer, which comes first in the model, transforms words using integer encoding into dense vectors. It is expected that the input length is max_len.
- In order to identify specific trends in the input text data, a 1D convolutional structure (Conv1D) with 32 filters and a kernel size of 3 is introduced. The use of the function that activates ReLU creates non-linearity.
- Using the MaxPooling1D layer, the input representation is down-sampled in order to extract significant features and reduce dimensionality.
- In order to prepare the result of the convolutional layer for input into the following dense layers, the Flatten layer transforms it into a one-dimensional array.
- The model is extended with two thick layers. With 1024 units in the first dense layer and a ReLU activation function, the model can recognise intricate patterns in the input. With two units and a softmax activation function, the second dense layer generates probabilities for every emotion class.

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape
=================================================================
 embedding_3 (Embedding)     (None, 40, 128)
 conv1d (Conv1D)             (None, 40, 32)
 max_pooling1d (MaxPooling1D  (None, 20, 32)
 )
 flatten (Flatten)           (None, 640)
 dense_3 (Dense)             (None, 1024)
 dense_4 (Dense)             (None, 2)
=================================================================
```

Figure 14 Build CNN Model

Compilation Settings:

- The compile technique establishes necessary parameters for the learning process, which guarantees effective training.
- Through the computation of the cross-entropy among true labels and predicted probability distributions, the selected loss function, sparse_categorical_crossentropy, is appropriate for classifying text into many categories.
- The Adam optimizer is chosen for optimising deep neural networks for use in sentiment analysis because of its proficiency in managing sparse gradients and fine-tuning learning rates per parameter.
- Evaluation measures, such accuracy, are defined to track the performance of the model during the training and testing stages.

Epochs 5:

```
Epoch 1/5
5000/5000 - 415s - loss: 0.2962 - accuracy: 0.8759 - val_loss: 0.2681 - val_accuracy: 0.8906 - 415s/epoch - 83ms/step
Epoch 2/5
5000/5000 - 400s - loss: 0.1695 - accuracy: 0.9342 - val_loss: 0.2882 - val_accuracy: 0.8864 - 400s/epoch - 80ms/step
Epoch 3/5
5000/5000 - 392s - loss: 0.0710 - accuracy: 0.9740 - val_loss: 0.3835 - val_accuracy: 0.8847 - 392s/epoch - 78ms/step
Epoch 4/5
5000/5000 - 392s - loss: 0.0345 - accuracy: 0.9877 - val_loss: 0.5139 - val_accuracy: 0.8858 - 392s/epoch - 78ms/step
Epoch 5/5
5000/5000 - 396s - loss: 0.0238 - accuracy: 0.9916 - val_loss: 0.5938 - val_accuracy: 0.8826 - 396s/epoch - 79ms/step
```

Figure 15 Training CNN on 5 epochs

Epochs 10:

```
Epoch 1/10
5000/5000 - 596s - loss: 0.2952 - accuracy: 0.8758 - val_loss: 0.2608 - val_accuracy: 0.8949 -
596s/epoch - 119ms/step
Epoch 2/10
5000/5000 - 587s - loss: 0.1698 - accuracy: 0.9336 - val_loss: 0.2695 - val_accuracy: 0.8977 -
587s/epoch - 117ms/step
Epoch 3/10
5000/5000 - 587s - loss: 0.0730 - accuracy: 0.9733 - val_loss: 0.4068 - val_accuracy: 0.8954 -
587s/epoch - 117ms/step
Epoch 4/10
5000/5000 - 584s - loss: 0.0372 - accuracy: 0.9867 - val_loss: 0.5148 - val_accuracy: 0.8814 -
584s/epoch - 117ms/step
Epoch 5/10
5000/5000 - 592s - loss: 0.0249 - accuracy: 0.9911 - val_loss: 0.6116 - val_accuracy: 0.8913 -
592s/epoch - 118ms/step
Epoch 6/10
5000/5000 - 593s - loss: 0.0181 - accuracy: 0.9937 - val_loss: 0.7030 - val_accuracy: 0.8925 -
593s/epoch - 119ms/step
Epoch 7/10
5000/5000 - 594s - loss: 0.0147 - accuracy: 0.9951 - val_loss: 0.7576 - val_accuracy: 0.8917 -
594s/epoch - 119ms/step
Epoch 8/10
5000/5000 - 593s - loss: 0.0131 - accuracy: 0.9956 - val_loss: 0.8929 - val_accuracy: 0.8887 -
593s/epoch - 119ms/step
Epoch 9/10
5000/5000 - 596s - loss: 0.0117 - accuracy: 0.9960 - val_loss: 0.8871 - val_accuracy: 0.8899 -
596s/epoch - 119ms/step
Epoch 10/10
5000/5000 - 590s - loss: 0.0110 - accuracy: 0.9964 - val_loss: 0.9177 - val_accuracy: 0.8882 -
590s/epoch - 118ms/step
```

Figure 16 Training CNN on 10 epochs

The loss value in this instance is 0.0110, which shows that the model's predictions and the actual data coincide rather well.As we can see, is better than 0..0249, or five epochs.

LSTM+CNN Model:

The architecture of an LSTM+CNN model for sentiment assessment Below is a summary of the important elements:

Layer Configuration:

- To improve the model's comprehension of word representations, it integrates an Embedding layer that transforms words encoded with integers into dense vectors.

- To extract local patterns and features from the text input, a Conv1D layer with 32 filters, a kernel size of 3, and a ReLU activation function is implemented.
- To capture the most prominent characteristics and lower the dimensionality of the feature maps, the MaxPooling1D layer is applied with a pool size of 2.
- Using 128 units and a 0.2 dropout rate, an LSTM layer is used to detect long-range relationships in the data and avoid overfitting.
- A Dense layer including a softmax activation mechanism is added to categorise the input text's emotion into either positive or negative categories.

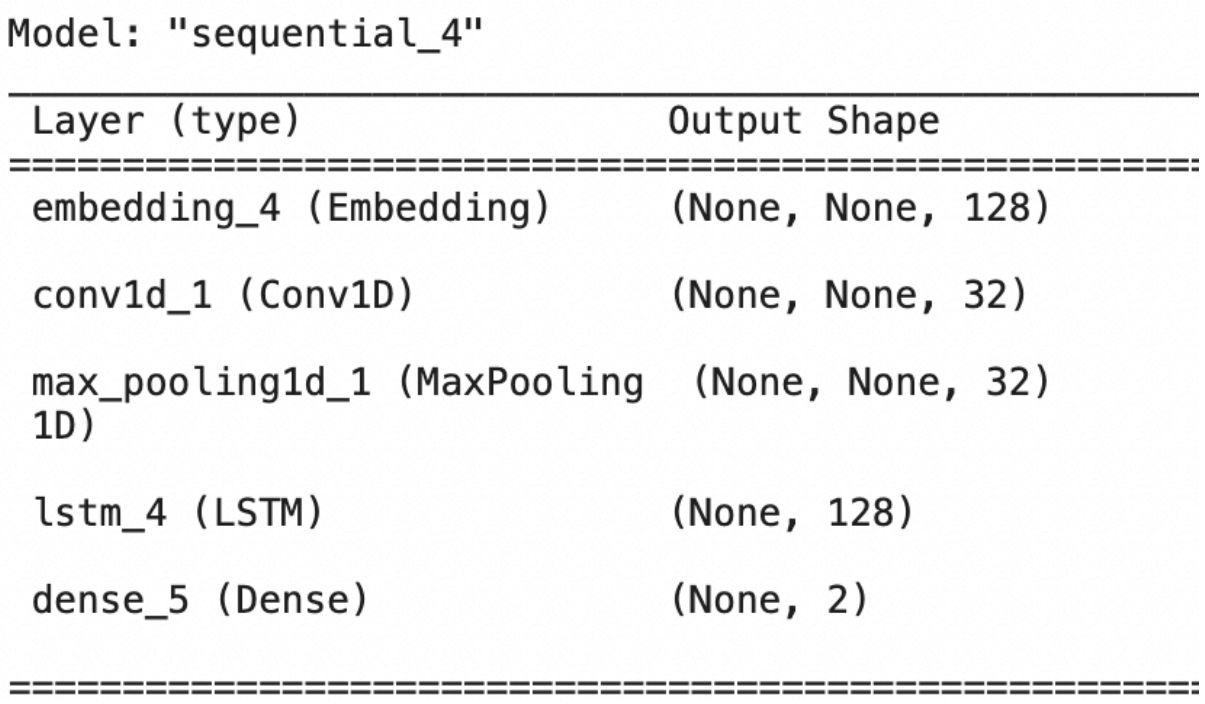


Figure 17 Build LSTM+CNN Model

Compilation Settings:

Through the computation of the cross-entropy among true labels and predicted probability distributions, the selected loss function is sparse_categorical_cross_entropy.

The Adam optimizer.

Evaluation measures, like accuracy is used.

Model Training:

Epochs 5:

```
5000/5000 - 386s - loss: 0.3103 - accuracy: 0.8694 - val_loss: 0.2789 - val_accuracy: 0.8859 - 386s/epoch - 77ms/step
Epoch 2/5
5000/5000 - 387s - loss: 0.2201 - accuracy: 0.9118 - val_loss: 0.2622 - val_accuracy: 0.8957 - 387s/epoch - 77ms/step
Epoch 3/5
5000/5000 - 389s - loss: 0.1546 - accuracy: 0.9407 - val_loss: 0.2794 - val_accuracy: 0.8971 - 389s/epoch - 78ms/step
Epoch 4/5
5000/5000 - 387s - loss: 0.1065 - accuracy: 0.9600 - val_loss: 0.3017 - val_accuracy: 0.8959 - 387s/epoch - 77ms/step
Epoch 5/5
5000/5000 - 386s - loss: 0.0758 - accuracy: 0.9718 - val_loss: 0.3911 - val_accuracy: 0.8943 - 386s/epoch - 77ms/step
```

Figure 18 Training LSTM+CNN on 5 epochs

Epochs 10:

```
Epoch 1/10
5000/5000 - 620s - loss: 0.3053 - accuracy: 0.8713 - val_loss: 0.2660 - val_accuracy: 0.8945 -
620s/epoch - 124ms/step
Epoch 2/10
5000/5000 - 626s - loss: 0.2180 - accuracy: 0.9128 - val_loss: 0.2612 - val_accuracy: 0.8979 -
626s/epoch - 125ms/step
Epoch 3/10
5000/5000 - 619s - loss: 0.1519 - accuracy: 0.9416 - val_loss: 0.2842 - val_accuracy: 0.8925 -
619s/epoch - 124ms/step
Epoch 4/10
5000/5000 - 618s - loss: 0.1055 - accuracy: 0.9612 - val_loss: 0.3255 - val_accuracy: 0.8992 -
618s/epoch - 124ms/step
Epoch 5/10
5000/5000 - 618s - loss: 0.0764 - accuracy: 0.9717 - val_loss: 0.3743 - val_accuracy: 0.8974 -
618s/epoch - 124ms/step
Epoch 6/10
5000/5000 - 621s - loss: 0.0569 - accuracy: 0.9793 - val_loss: 0.4080 - val_accuracy: 0.8949 -
621s/epoch - 124ms/step
Epoch 7/10
5000/5000 - 616s - loss: 0.0460 - accuracy: 0.9832 - val_loss: 0.4446 - val_accuracy: 0.8925 -
616s/epoch - 123ms/step
Epoch 8/10
5000/5000 - 610s - loss: 0.0367 - accuracy: 0.9868 - val_loss: 0.4543 - val_accuracy: 0.8922 -
610s/epoch - 122ms/step
Epoch 9/10
5000/5000 - 610s - loss: 0.0312 - accuracy: 0.9887 - val_loss: 0.5107 - val_accuracy: 0.8939 -
610s/epoch - 122ms/step
Epoch 10/10
5000/5000 - 608s - loss: 0.0270 - accuracy: 0.9904 - val_loss: 0.5401 - val_accuracy: 0.8883 -
608s/epoch - 122ms/step
```

Figure 19 Training LSTM+CNN on 10 epochs

The loss value in this instance is 0.0270, which shows that the model's predictions and the actual data coincide rather well.As we can see, is better than 0.0758, or five epochs.
The model's prediction accuracy in relation to the true target values is gauged by the loss function. Below are the loss values after respective Epoches for above Models build

Model	Loss Values for 5 Epochs	Loss Value for 10 Epochs
LSTM	0.0833	0.018
BI-LSTM	0.0673	0.0159
CNN	0.0249	0.0110
LSTM+CNN	0.0758	0.0270

Table 7 Loss values for 5 and 10 epochs

Naive Bayes Model:

Built models by importing from sklearn library and Evaluated the accuracy of each model, Accuracy is calculated for below three models and discussed in results section of thesis BernoulliNB, MultinatinominalNB, and ComplementNB for same Data Set preprocessed as below:

Complement NB:

```
from sklearn.naive_bayes import ComplementNB
from sklearn.metrics import classification_report, confusion_matrix
CNB = ComplementNB()
CNB.fit(X_train, y_train)

from sklearn import metrics
predicted = CNB.predict(X_test)
accuracy_score = metrics.accuracy_score(predicted, y_test)
```

Figure 20 Model Building and fit code for Complement NB

Multinomial NB:

```
from sklearn.naive_bayes import MultinomialNB

MNB = MultinomialNB()
MNB.fit(X_train, y_train)

predicted = MNB.predict(X_test)
accuracy_score = metrics.accuracy_score(predicted, y_test)
```

Figure 21 Model Building and fit code for Multinomial NB

Bernoulli NB:

```
from sklearn.naive_bayes import BernoulliNB

BNB = BernoulliNB()
BNB.fit(X_train, y_train)

predicted = BNB.predict(X_test)
accuracy_score_bnb = metrics.accuracy_score(predicted,y_test)
```

Figure 22 Model Building and fit code for Bernoulli NB

7.4. MODELS EVALUATION

Results for all the built models are presented for further evaluation & Accuracy followed by other metrics are calculated and tabled below and best model for given data sample is verified later in document .

Experimental Results for 5 Epochs on Deep Learning NN:

LSTM:

	precision	recall	f1-score	support
0	0.9792	0.9536	0.9662	35317
1	0.9870	0.9943	0.9906	124683
accuracy			0.9853	160000
macro avg	0.9831	0.9739	0.9784	160000
weighted avg	0.9852	0.9853	0.9852	160000

Testing Results

	precision	recall	f1-score	support
0	0.7684	0.7148	0.7406	8754
1	0.9216	0.9396	0.9305	31246
accuracy			0.8904	40000
macro avg	0.8450	0.8272	0.8356	40000
weighted avg	0.8881	0.8904	0.8890	40000

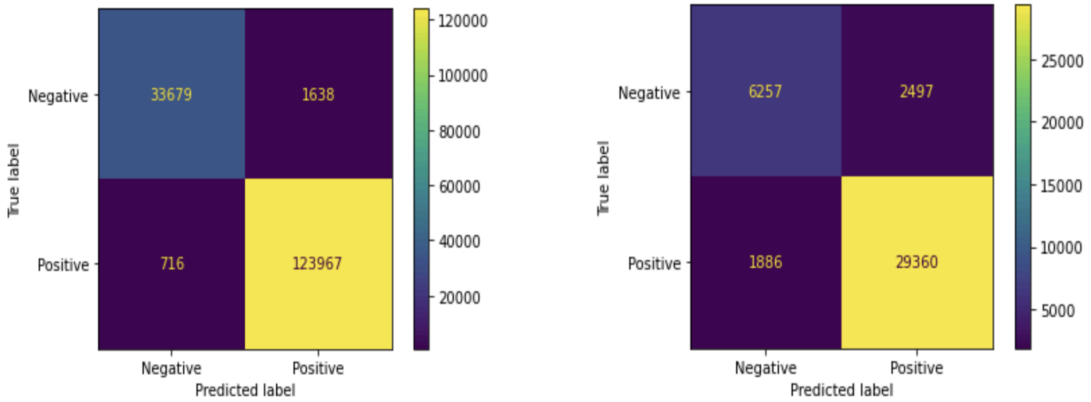


Figure 23 Results for LSTM 5 epochs

BI-LSTM:

	precision	recall	f1-score	support
0	0.9833	0.9732	0.9782	35317
1	0.9924	0.9953	0.9939	124683
accuracy			0.9904	160000
macro avg	0.9879	0.9843	0.9861	160000
weighted avg	0.9904	0.9904	0.9904	160000

Testing Results

	precision	recall	f1-score	support
0	0.7548	0.7143	0.7340	8754
1	0.9211	0.9350	0.9280	31246
accuracy			0.8867	40000
macro avg	0.8380	0.8247	0.8310	40000
weighted avg	0.8847	0.8867	0.8856	40000

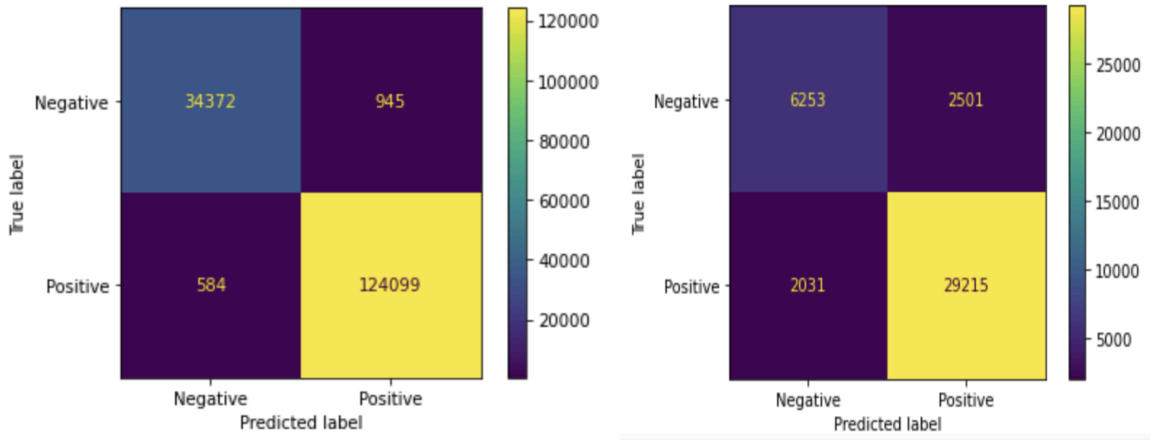


Figure 24 Results for BI LSTM 5 epochs

CNN:

Training Results

	precision	recall	f1-score	support
0	0.9924	0.9924	0.9924	35317
1	0.9979	0.9979	0.9979	124683
accuracy			0.9967	160000
macro avg	0.9951	0.9951	0.9951	160000
weighted avg	0.9967	0.9967	0.9967	160000

Testing Results

	precision	recall	f1-score	support
0	0.7419	0.7111	0.7262	8754
1	0.9200	0.9307	0.9253	31246
accuracy			0.8826	40000
macro avg	0.8309	0.8209	0.8257	40000
weighted avg	0.8810	0.8826	0.8817	40000

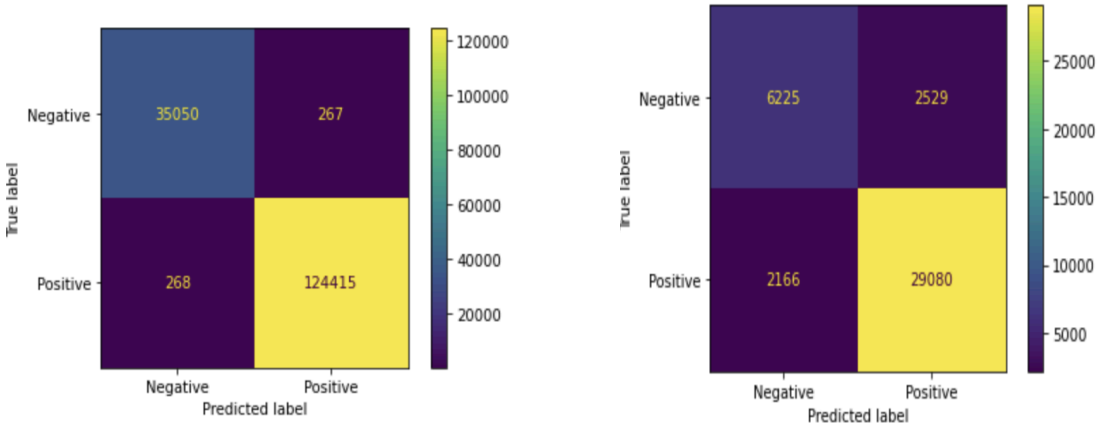


Figure 25 Results for CNN 5 epochs

LSTM+CNN:

Training Results

	precision	recall	f1-score	support
0	0.9910	0.9495	0.9698	35317
1	0.9859	0.9976	0.9917	124683
accuracy			0.9870	160000
macro avg	0.9884	0.9736	0.9808	160000
weighted avg	0.9870	0.9870	0.9869	160000

Testing Results

	precision	recall	f1-score	support
0	0.8132	0.6715	0.7356	8754
1	0.9122	0.9568	0.9340	31246
accuracy			0.8943	40000
macro avg	0.8627	0.8141	0.8348	40000
weighted avg	0.8906	0.8943	0.8906	40000

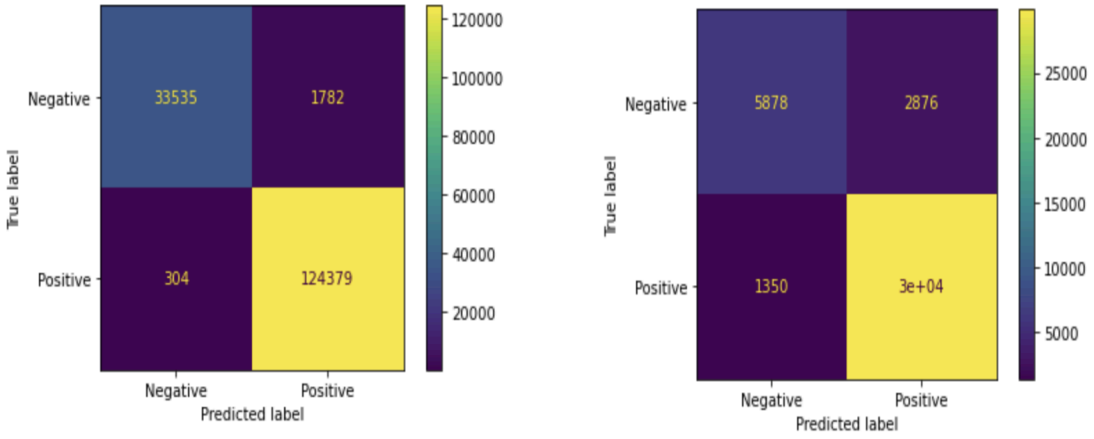


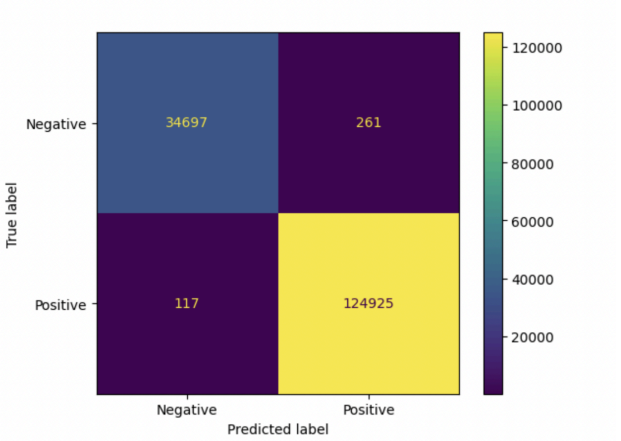
Figure 26 Results for LSTM+CNN 5 epochs

Experimental Results for 10 Epochs on Deep Learning NN:

LSTM:

Training Results

	precision	recall	f1-score	support
0	0.9966	0.9925	0.9946	34958
1	0.9979	0.9991	0.9985	125042
accuracy			0.9976	160000
macro avg	0.9973	0.9958	0.9965	160000
weighted avg	0.9976	0.9976	0.9976	160000



Testing Results

	precision	recall	f1-score	support
0	0.7715	0.7349	0.7528	8802
1	0.9262	0.9386	0.9324	31198
accuracy			0.8938	40000
macro avg	0.8489	0.8368	0.8426	40000
weighted avg	0.8922	0.8938	0.8928	40000

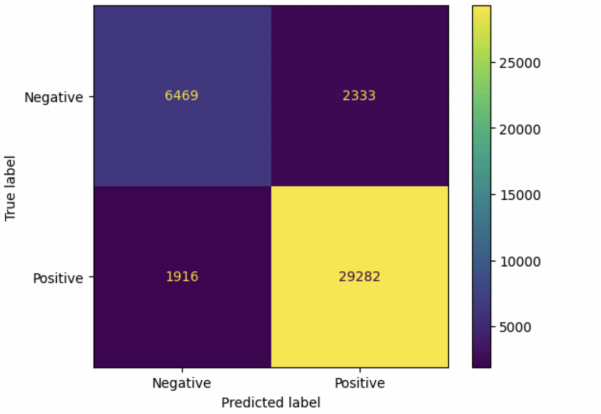
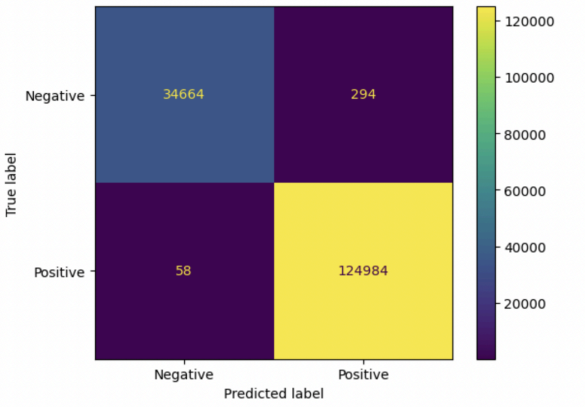


Figure 27 Results for LSTM 10 epochs

BI-LSTM:

Training Results

	precision	recall	f1-score	support
0	0.9983	0.9916	0.9949	34958
1	0.9977	0.9995	0.9986	125042
accuracy			0.9978	160000
macro avg	0.9980	0.9956	0.9968	160000
weighted avg	0.9978	0.9978	0.9978	160000



Testing Results

	precision	recall	f1-score	support
0	0.7797	0.7142	0.7455	8802
1	0.9212	0.9431	0.9320	31198
accuracy			0.8927	40000
macro avg	0.8505	0.8286	0.8388	40000
weighted avg	0.8901	0.8927	0.8910	40000

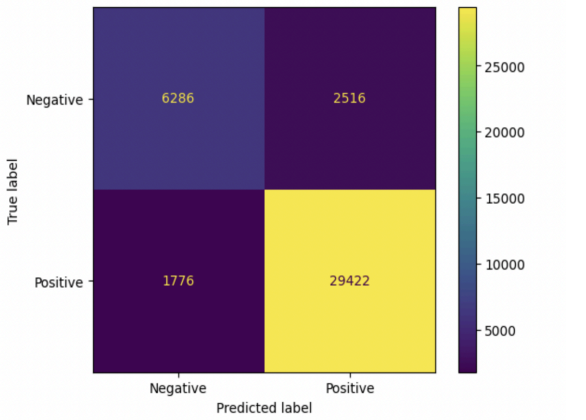
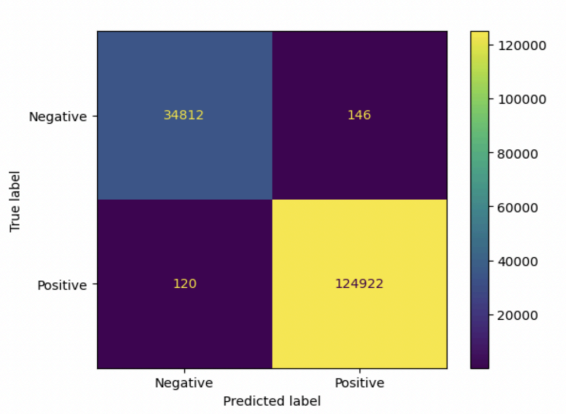


Figure 28 Results for BI LSTM 10 epochs

CNN:

Training Results

	precision	recall	f1-score	support
0	0.9966	0.9958	0.9962	34958
1	0.9988	0.9990	0.9989	125042
accuracy			0.9983	160000
macro avg	0.9977	0.9974	0.9976	160000
weighted avg	0.9983	0.9983	0.9983	160000



Testing Results

	precision	recall	f1-score	support
0	0.7581	0.7227	0.7400	8802
1	0.9228	0.9349	0.9288	31198
accuracy			0.8882	40000
macro avg	0.8404	0.8288	0.8344	40000
weighted avg	0.8865	0.8882	0.8873	40000

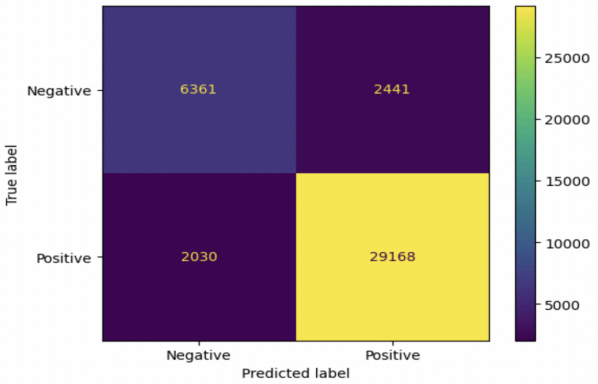
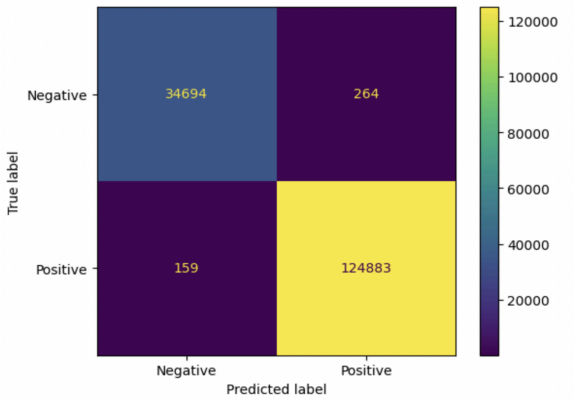


Figure 29 Results for CNN 10 epochs

LSTM+CNN:

Training Results

	precision	recall	f1-score	support
0	0.9954	0.9924	0.9939	34958
1	0.9979	0.9987	0.9983	125042
accuracy			0.9974	160000
macro avg	0.9967	0.9956	0.9961	160000
weighted avg	0.9974	0.9974	0.9974	160000



Testing Results

	precision	recall	f1-score	support
0	0.7526	0.7334	0.7429	8802
1	0.9253	0.9320	0.9286	31198
accuracy			0.8883	40000
macro avg	0.8390	0.8327	0.8357	40000
weighted avg	0.8873	0.8883	0.8878	40000

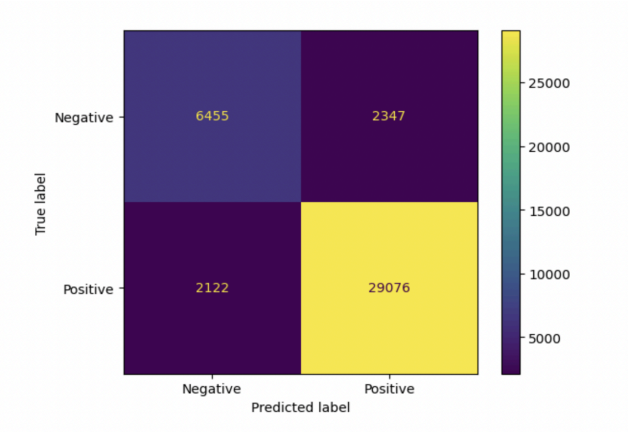


Figure 30 Results forLSTM+ CNN 10 epochs

Results for Naive Bayes variants for same dataset:

ComplementNB Model Test Result

```
Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.77      0.65      5888
           1       0.36      0.24      0.29      3087
           2       0.92      0.89      0.91     31025

    accuracy                           0.82     40000
   macro avg       0.61      0.63      0.61     40000
weighted avg       0.83      0.82      0.82     40000
```

Figure 31 Results for Complement NB

MultinomialNB Model Test Results

```
Classification Report:
              precision    recall  f1-score   support

           0       0.66      0.61      0.64      5888
           1       0.35      0.24      0.29      3087
           2       0.89      0.93      0.91     31025

    accuracy                           0.83     40000
   macro avg       0.63      0.60      0.61     40000
weighted avg       0.82      0.83      0.82     40000
```

Figure 32 Results for Multinomial NB

BernoulliNB Model Test Results

```
Classification Report:
              precision    recall  f1-score   support

           0       0.66      0.48      0.55      5888
           1       0.23      0.23      0.23      3087
           2       0.86      0.91      0.88     31025

    accuracy                           0.79     40000
   macro avg       0.58      0.54      0.55     40000
weighted avg       0.78      0.79      0.78     40000
```

Figure 33 Results for Beroulli NB

7.4.1 SUMMARY OF RESULTS AND OBSERVATIONS

S.No	Model Name	Model Accuracy(Training Dataset - 160000)		Model Accuracy(Test Dataset - 40000)	
		Epochs -5	Epochs -10	Epochs- 5	Epochs- 10
1	LSTM	98.53	99.76	89.04	89.38
2	BI-LSTM	99.04	99.78	88.67	89.27
3	CNN	99.67	99.83	88.26	88.82
4	LSTM+CNN	98.70	99.74	89.43	88.83
5	ComplementNB	82.17			
6	MultinominalNB	83.10			
7	BernoulliNB	79.09			

Table 9.1.1 Table of Results Summary

It is clear from above results for Amazon Food Review Dataset with the 200,000 sample dataset that LSTM performs better than alternative models when it comes of analysing sentiment accuracy. In particular, it outperformed previously created models with an accuracy of about 89.38% after developing the model using LSTM for 10 epochs. We have thus chosen to move on with LSTM as the

primary framework for integrating with the LLM model and ensuing sentiment prediction tasks in light of these results.

7.5. LLM INTEGRATION

The careful examination of reviews using the Amazon Fine Food Reviews dataset, which combines sentiment prediction with an LSTM model and reason categorization with an LLM. Based on the review's content, the LLM uses the given prompt to provide replies that include sentiment and reasons descriptors.

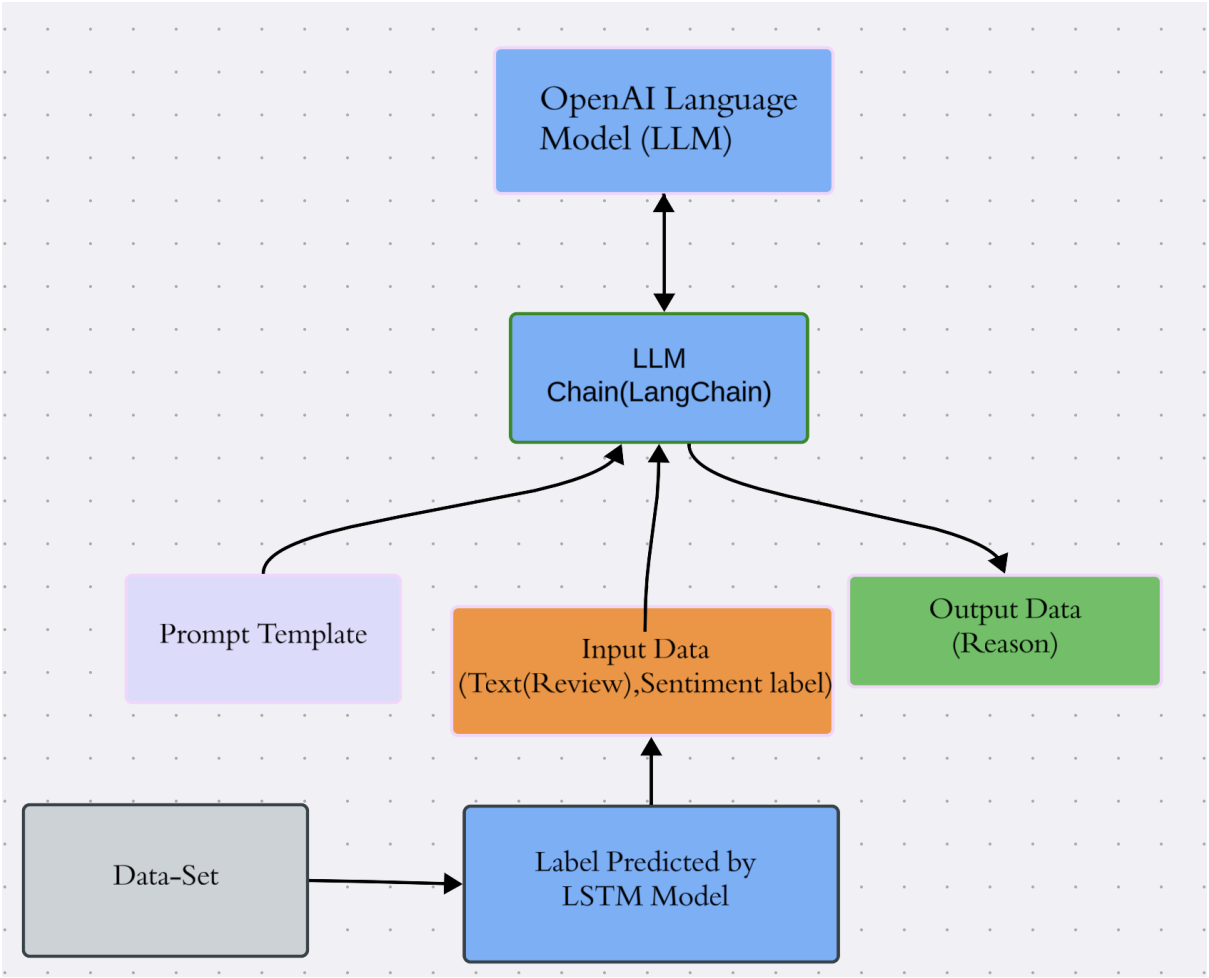


Figure 34 Process flow in Integration of LSTM to LLM

Template Setup:

- A specified template called template1 is developed, which acts as the LLM prompt. It gives background information on the assignment and guidelines for doing sentiment and rationale analysis using a review's content.Both the anticipated sentiment label ({Sentiment}) generated from the developed LSTM model and the review content ({Review}) have placeholders in the template.

Prompt Template:

- The template that was previously specified (template1) and the parameters that were provided (Review and Sentiment) are used to instantiate the PromptTemplate class.

LLM Initiation and Execution:

- A Language Modelling with Prompt Engineering (LLM) instance is constructed with certain parameters, such as the OpenAI API key. The "gpt-3.5-turbo-instruct" model was selected here because it is best suited for activities requiring instructions.
- The LLM instance (llm1) and the prompt template (prompt_template1) are used to initialise an LLMChain. This chain makes it easier to communicate with the LLM by enabling the passing of inputs and the generation of answers.
- The review text and the anticipated sentiment label are used as input variables when the LLMChain's execute method is called. The emotion and rationale analysis of the review is finished when the LLM responds to the given prompt.The LLM output, which comprises the sentiment and reason labels anticipated based on the review text, is printed together with the sentiment anticipated by the LSTM model (d[o[0]]).

Output:

Figure 35 Console output of Integration of LSTM to LLM

7.6 WEB APP BUILDING

Structure of web Application:

Backend (Flask):

- The backend server, which interfaces through the LLM for reasoning categorization, processes requests received from the front end, analyses sentiment using a model based on machine learning, and is built using Flask.
- Flask routes are intended to manage a variety of requests, including reviews where they need to forecast the sentiment and reason labels.
- Flask receives a request, parses the input data, applies a machine learning algorithm to predict sentiment, then uses the LLM to provide reason tags based on the content of the reviews.
- The anticipated sentiment and rationale labels are returned to the frontend by the Flask server as JSON replies.

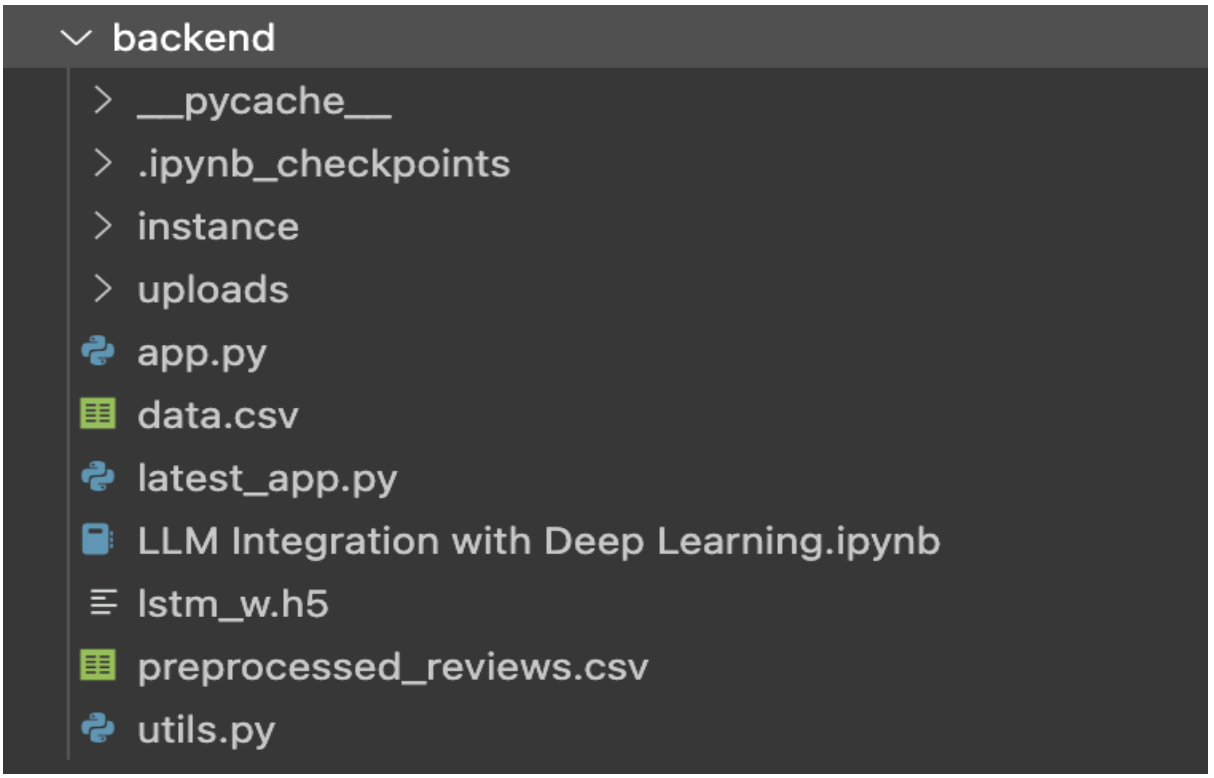


Figure 36 Flask Backend Folder Structure

- To run Backend simply we need to run the file app.py the command python app.py by opening the backend folder in the terminal

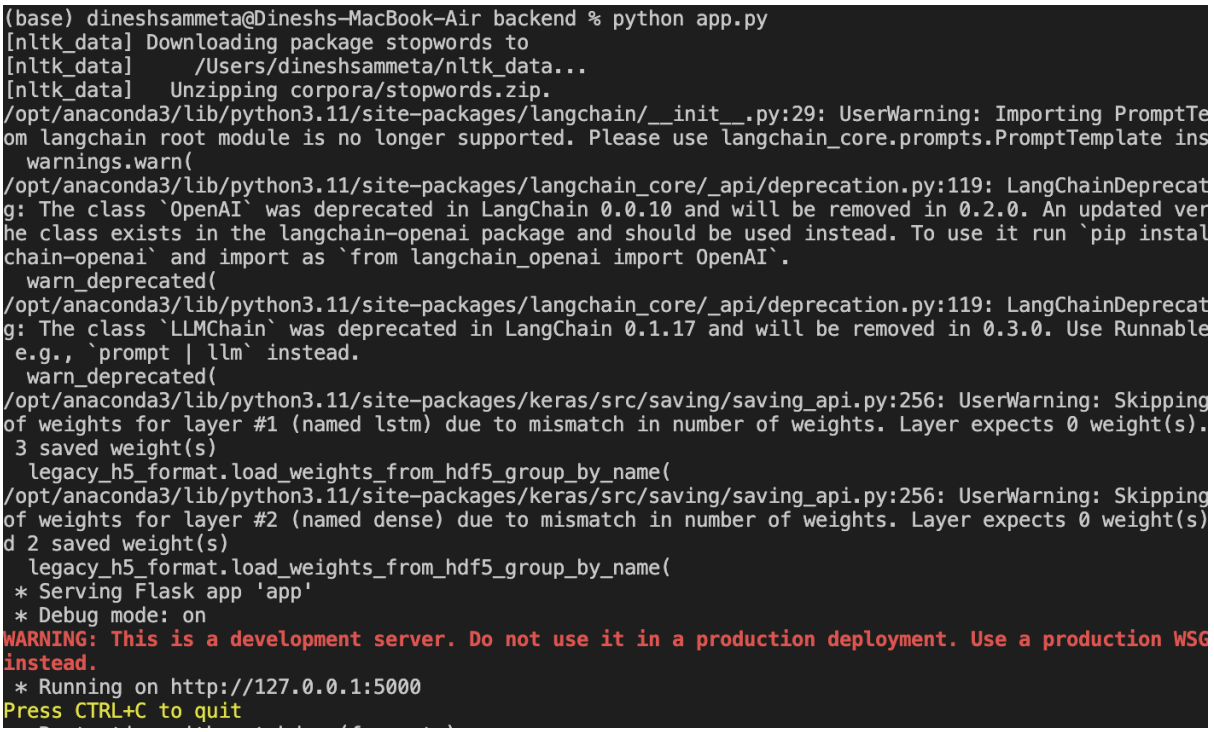


Figure 37 Console Output of Flask Server

Frontend(React JS)

- The web application's user interface (UI) is developed using React, which offers a responsive and user-friendly interface.
- React is used to develop components that let users enter review text and see the anticipated sentiment and explanation labels. Examples of these components are input forms, buttons, and result displays.
- Sending HTTP requests (such as a POST request with review data) and handling the server's answers are how React components communicate with the Flask backend.
- The frontend dynamically updates in response to the backend's replies, presenting the user with an ordered and aesthetically pleasing presentation of the anticipated emotion and explanation labels.

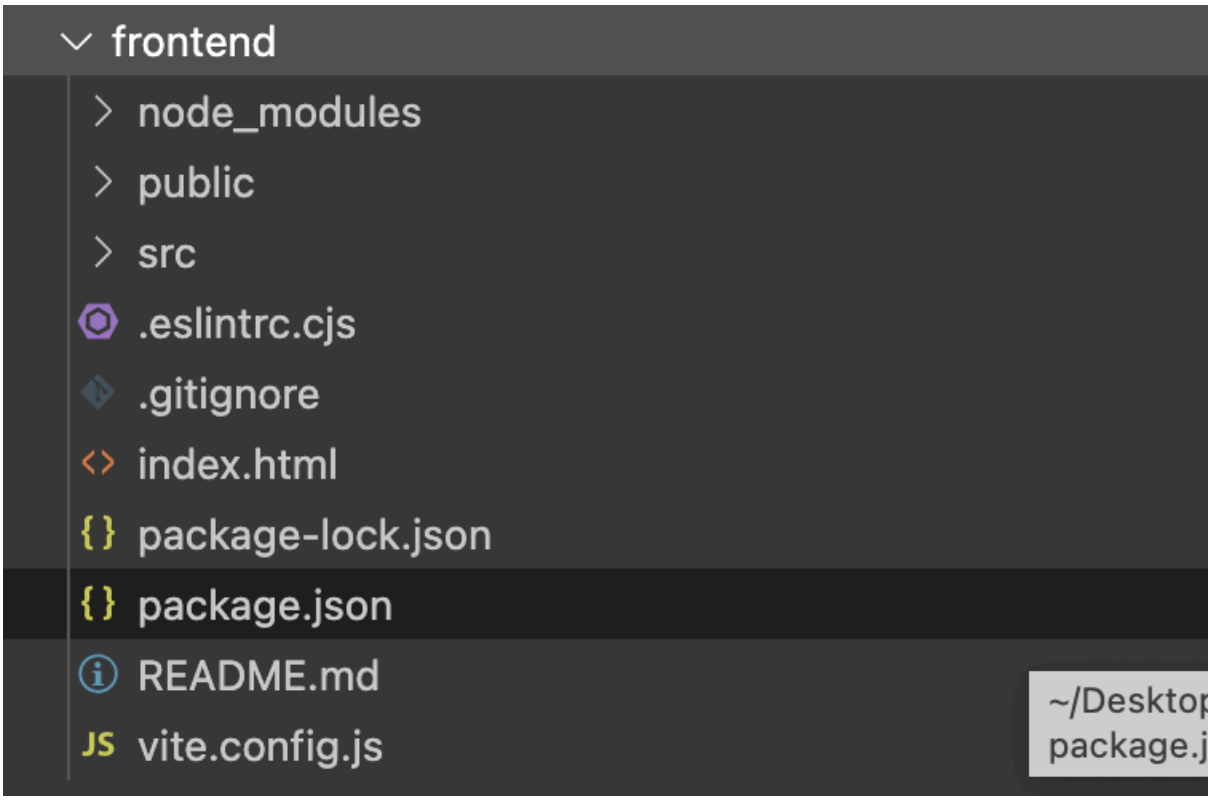


Figure 38 Folder Structure of React FrontEnd

- To run frontend simply we need to run with the command npm run dev by opening the frontend folder in the terminal

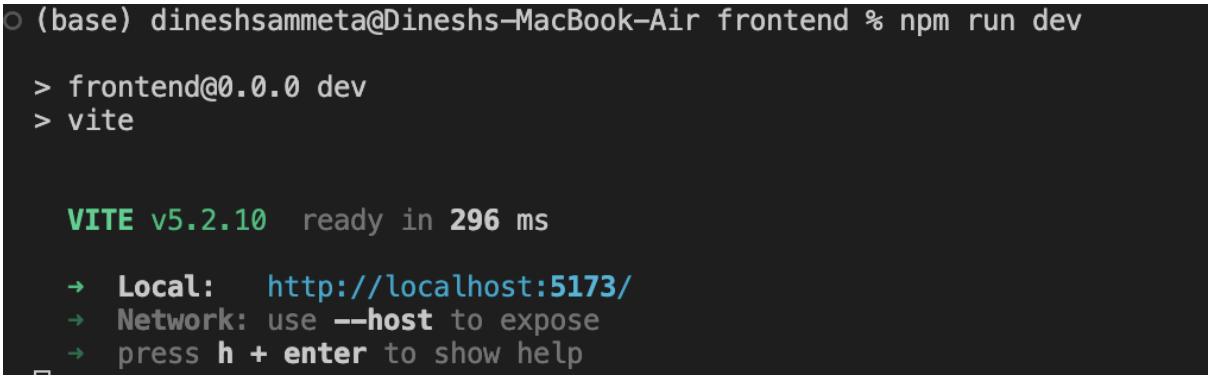


Figure 39 Console Output of React FrontEnd

Main features in WebApp

Using Flask as the server side and React as the frontend, the web application designed for sentiment assessment and reason categorization combines machine learning models with Language Models using Prompt Modelling (LLM). These are the web application's salient features:

User Authentication:

- In order to provide safe access to the programme's functions, the web application has user authentication mechanism.
- To access customised features and ensure session security, users can register for an account, log in, and log out of the programme.

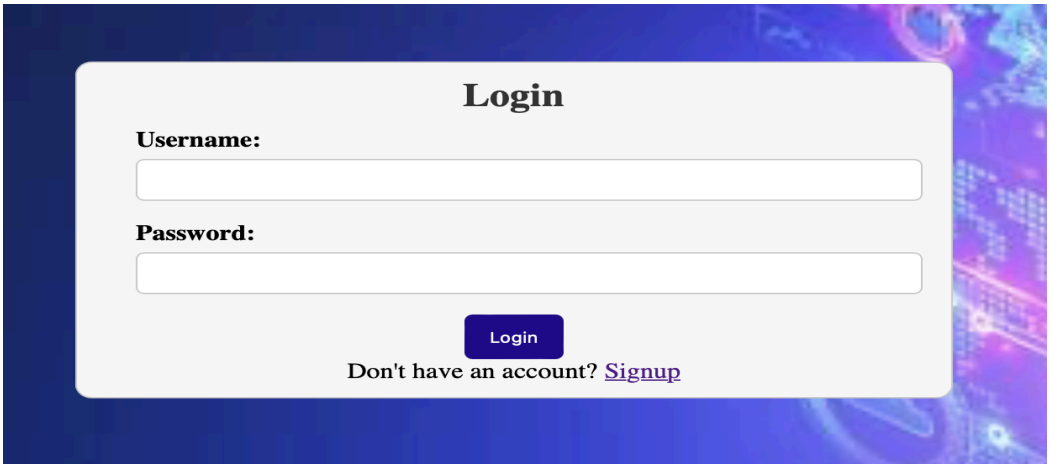


Figure 40 Login Web Page UI

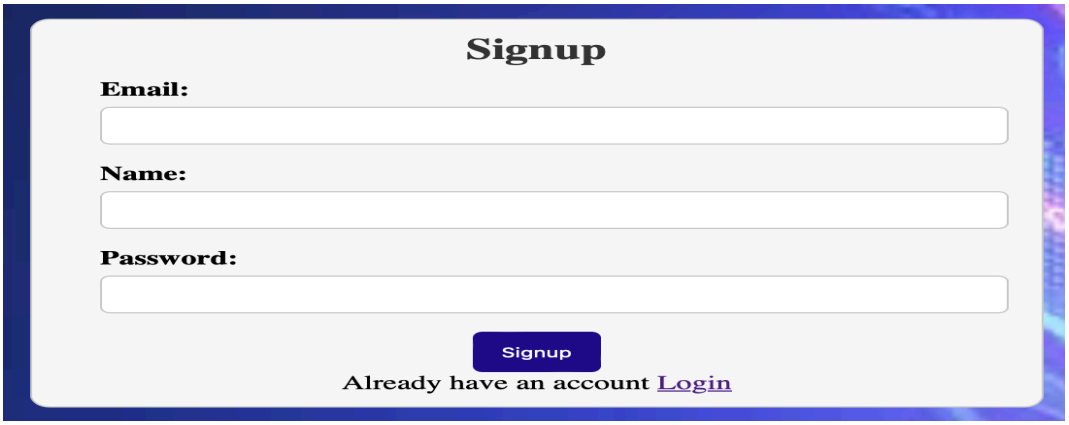


Figure 41 Signup Web Page UI

Portfolio Management:

- A portfolio management function is available to authenticated users, giving them the ability to view their saved data, see their analysis history, and preserve their reviewed ratings.
- Every user's queries are recorded in the programme, which keeps track of how they utilise the CSV file analysis and text analysis functions.
- To examine earlier analysis, monitor their usage trends, and go back and review earlier outcomes, users may access their query history.

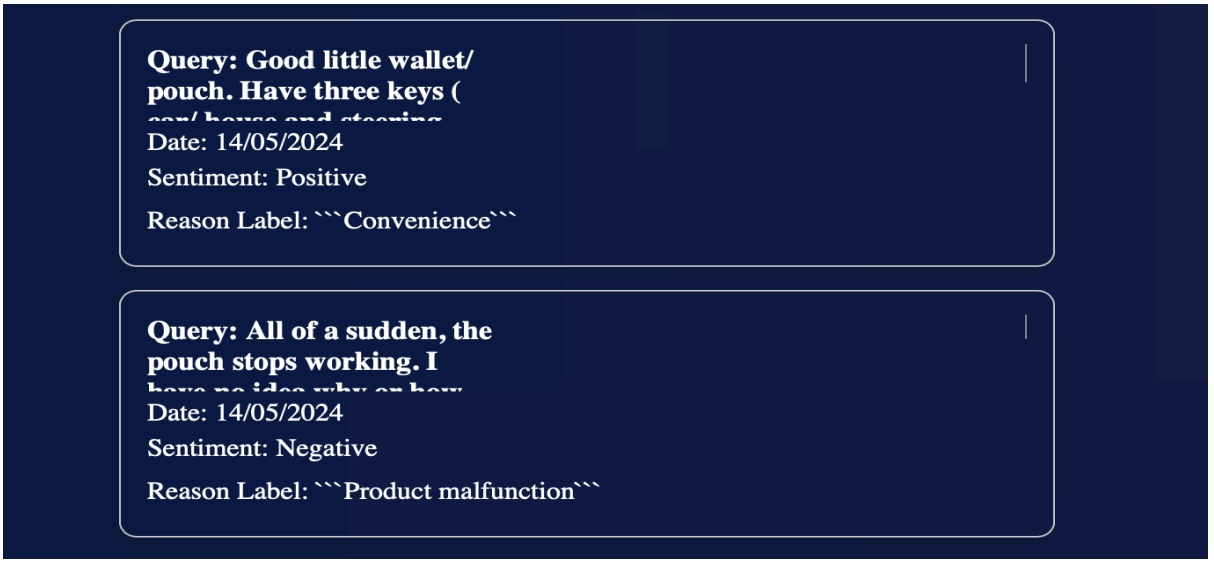


Figure 42 History Web Page UI

Single Text Analysis:

- The UI Developed allows users to provide text reviews for sentiment analysis and reason categorization.
- In order to forecast sentiment and reason labels based on the content of the review, the backend evaluates the input text using the LLM integration and machine learning model.
- The user is presented with the findings, which include the anticipated sentiment label and the rationale for the evaluation.

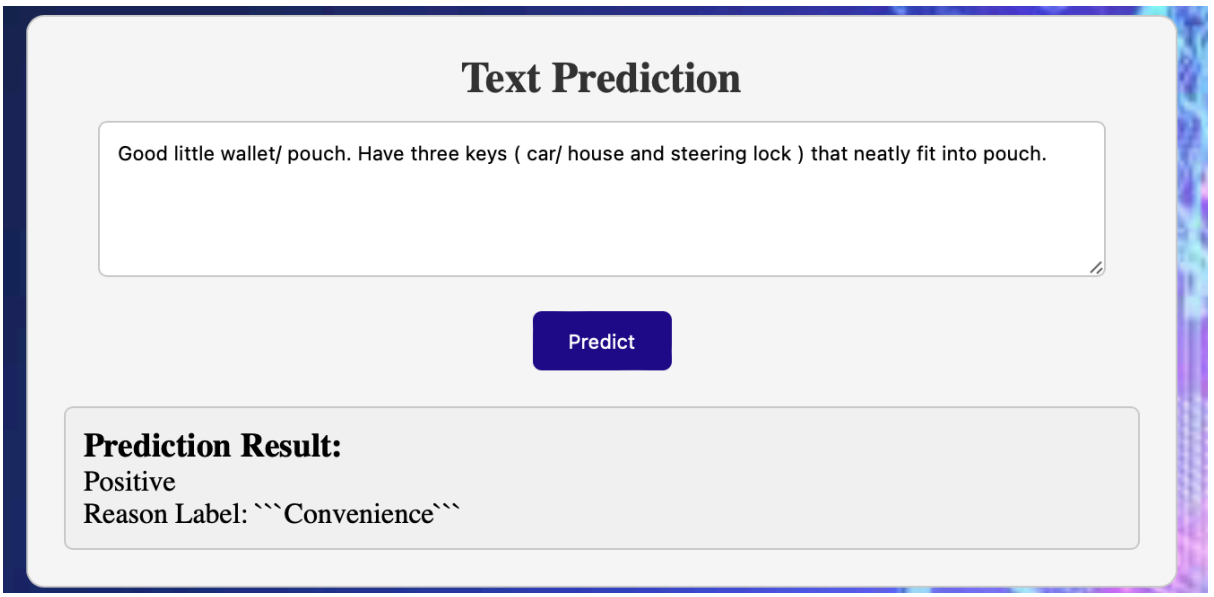


Figure 42 Single Text Input Web Page

CSV File Analysis:

- For batch analysis, users can choose to submit CSV files with numerous reviews.
- Using the machine learning model and LLM integration, the backend processes the CSV file, goes through each review iteratively, and uses sentiment analysis and reason categorization.
- The anticipated emotion and cause labels for each review in the supplied CSV file are displayed to the user in a tabular style.

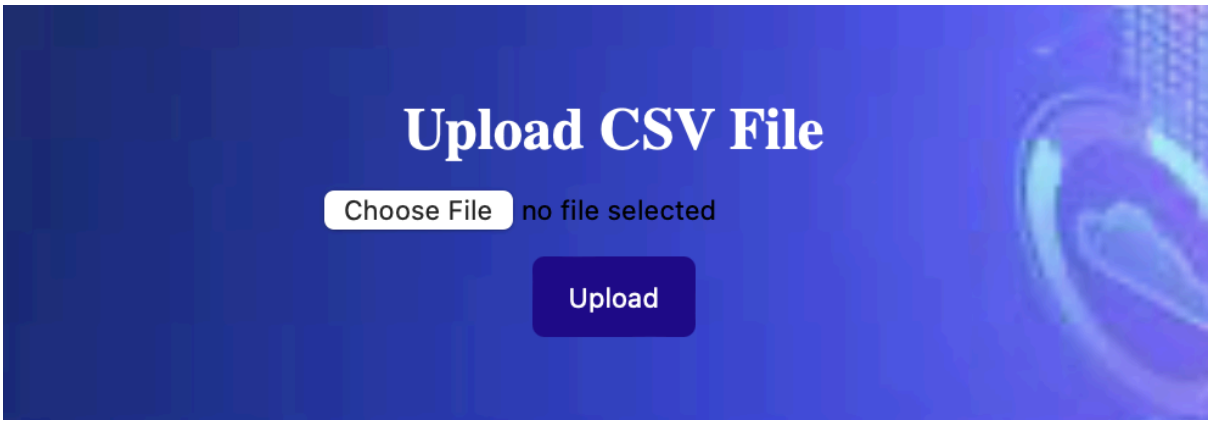


Figure 43 CSV File Input Web Page

8. RECOMMENDATIONS FOR FUTURE RESEARCH

Within the framework of the project on sentiment analysis as well as logic categorization that has been detailed, the following suggestions for further study might be investigated and expanded around in a dissertation:

Optimising and Improving Models:

- Examine more sophisticated deep learning architectures to improve sentiment analysis and reason classification performance, such as attention mechanisms or transformer-based models (BERT, GPT, etc.).
- Try varying the hyperparameter tuning and optimisation procedures to enhance the model's performance, accelerate its convergence, and resolve any possible overfitting problems.

Multi-Model Analysis:

- Include extra modalities in the study, such as photographs, audio, or video, in addition to textual data. These may offer more contextual data for sentiment analysis and rational categorization.
- Create multi-modal fusion strategies to efficiently merge data from several modalities and enhance the interpretability and prediction accuracy of the final product.

Domain Specific Adoption:

- To enhance model performance in specialised areas (like healthcare or finance), where the language and emotion expressions may differ dramatically from general-purpose datasets, investigate methods for domain adaptation or domain-specific fine-tuning.
- To guarantee model robustness and generalisation, look at methods for resolving domain shift and information distribution discrepancies across training and deployment contexts.

9. CONCLUSION

From the bottom up, this study has used the Amazon Food Reviews dataset to conduct a thorough investigation of emotion evaluation and rational categorization. By means of the application and assessment of several deep learning models, such as LSTM, CNN, Bi-LSTM, and LSTM+CNN, Naive Bayes significant knowledge has been obtained concerning the effectiveness of these models in interpreting sentiment and pinpointing the motivations behind reviews.For the aforementioned Amazon Food Reviews dataset, out of all of the above simulations, the LSTM architecture proved to be the most beneficial, performing better in regards to accuracy and generalisation. The LSTM model demonstrated strong sentiment analysis skills by utilising its capacity to record sequential dependencies, hence offering significant sentiment labelling for the reviews.Additionally, the analysis has been enhanced by the inclusion of Language Model using Prompt Modelling (LLM), which generates intelligent reason labels depending on the content of the reviews. Deeper comprehension of the underlying feelings represented in the reviews is made possible by the combination of artificial intelligence models with language models, which improves the comprehension and understanding of the sentiment analysis findings.The creation of an intuitive web interface with Flask and React has made sentiment analysis and logical categorization more accessible to a wider audience and enabled smooth user interaction with the models. The interface encourages user participation and makes it easier to make well-informed decisions based on sentiment insights by providing user-friendly input forms & visually appealing result displays.This work has established a solid basis for the advancement of language models, cutting-edge models, sentiment analysis, and reason categorization, as well as the creation of user-friendly interfaces. Future initiative like Model Refinement, Multi-Model Analysis, Interpretibility and Domain Specific Adoption in this field have the potential to uncover new capabilities and applications with a big influence on society if they embrace continuous research and innovation.

10. REFERENCES

Alzahrani, M., Aldhyani, T., Alsubari, S., Althobaiti, M., & Fahad, A. (2022). Developing an intelligent system with deep learning algorithms for sentiment analysis of e-commerce product reviews. Computational Intelligence and Neuroscience, 2022, 1-10.

Dang, C., García, M., & Prieta, F. (2021). Hybrid deep learning models for sentiment analysis. Complexity, 2021, 1-16.

Hardjita, P. and Hidayat, R. (2022). Sentiment analysis of tweets on prakerja card using convolutional neural network and naive bayes. Ijid (International Journal on Informatics for Development), 10(2), 82-91.

Lin, C. and He, Y. (2009). Joint sentiment/topic model for sentiment analysis..

Mrhar, K., Benhiba, L., Bourekkache, S., & Abik, M. (2021). A bayesian cnn-lstm model for sentiment analysis in massive open online courses moocs. International Journal of Emerging Technologies in Learning (Ijet), 16(23), 216-232.

Nawaz, Z., Zhao, C., Nawaz, F., Safeer, A., & Irshad, W. (2021). Role of artificial neural networks techniques in development of market intelligence: a study of sentiment analysis of ewom of a women's clothing company. Journal of Theoretical and Applied Electronic Commerce Research, 16(5), 1862-1876.

Omara, E., Mosa, M., & Ismail, N. (2022). Applying recurrent networks for arabic sentiment analysis. Menoufia Journal of Electronic Engineering Research, 31(1), 21-28.

Salur, M. and Aydin, I. (2020). A novel hybrid deep learning model for sentiment classification. Ieee Access, 8, 58080-58093.

Zhang, X. and Qin, X. (2022). Research on sentiment analysis algorithm for comments on online ideological and political courses. International Journal of Advanced Computer Science and Applications, 13(11).

Truhn, D. (2023). Extracting structured information from unstructured histopathology reports using generative pre‐trained transformer 4 (gpt‐4). The Journal of Pathology, 262(3), 310-319.

Zhou, J., Lu, Y., Dai, H., Wang, H., & Hong, X. (2019). Sentiment analysis of chinese microblog based on stacked bidirectional lstm. Ieee Access, 7, 38856-38866.

Arunabala, C., Jwalitha, P., & Nuthalapati, S. (2019). Text sentiment analysis based on cnns and svm. International Journal of Research -Granthaalayah, 7(6), 77-83.

Nawaz, Z., Zhao, C., Nawaz, F., Safeer, A., & Irshad, W. (2021). Role of artificial neural networks techniques in development of market intelligence: a study of sentiment analysis of ewom of a women's clothing company. Journal of Theoretical and Applied Electronic Commerce Research, 16(5), 1862-1876.

Nurrohmat, M. and Sn, A. (2019). Sentiment analysis of novel review using long short-term memory method. Ijccs (Indonesian Journal of Computing and Cybernetics Systems), 13(3), 209.

Salur, M. and Aydin, I. (2020). A novel hybrid deep learning model for sentiment classification. Ieee Access, 8, 58080-58093.

Alzahrani, M., Aldhyani, T., Alsubari, S., Althobaiti, M., & Fahad, A. (2022). Developing an intelligent system with deep learning algorithms for sentiment analysis of e-commerce product reviews. Computational Intelligence and Neuroscience, 2022, 1-10.

Chen, H., Zheng, G., Awadallah, A., & Ji, Y. (2022). Pathologies of pre-trained language models in few-shot fine-tuning.

Creswell, A., Shanahan, M., & Higgins, I. (2022). Selection-inference: exploiting large language models for interpretable logical reasoning.. https://doi.org/10.48550/arxiv.2205.09712

Gondhi, N., Sharma, E., Alharbi, A., Verma, R., & Shah, M. (2022). Efficient long short-term memory-based sentiment analysis of e-commerce reviews. Computational Intelligence and Neuroscience, 2022, 1-9.

Rajasekharan, A., Zeng, Y., Padalkar, P., & Gupta, G. (2023). Reliable natural language understanding with large language models and answer set programming.. https://doi.org/10.48550/arxiv.2302.03780

Saparov, A. and He, H. (2022). Language models are greedy reasoners: a systematic formal analysis of chain-of-thought.

The python tutorial," Python documentation, https://docs.python.org/3/tutorial/index.html (accessed june. 10, 2023).

Welcome to flask," Welcome to Flask - Flask Documentation (2.3.x), https://flask.palletsprojects.com/ (accessed Aug. 18, 2023).