# 

**“Advanced Sentiment Analysis Using BERT: A Comparative Study on Real-Time Social**

# Media Data”

# Team: 3

# Lead Name: Muttana Geetha Srinivasa Reddy(22BCE20354)

# Group members:

# Kuppireddy Bhageeratha Reddy(22BCE20355)

# Aswitha Gunturu(22BCE20316)

# K Harsha Vardhan Reddy(22BCE20327)

# D Hemanth(22BCE20295)

# Lab instructor name: Dr.D. Kothandaraman

# Abstract:

The study of public opinion can provide us with valuable information. The analysis of sentiment on social networks, such as Twitter or Facebook, has become a powerful means of learning about the users’ opinions and has a wide range of applications. However, the efficiency and accuracy of sentiment analysis is being hindered by the challenges encountered in natural language processing (NLP). In recent years, it has been demonstrated that deep learning models are a promising solution to the challenges of NLP. This paper reviews the latest studies that have employed deep learning to solve sentiment analysis problems, such as sentiment polarity. Models using term frequency-inverse document frequency (TF-IDF), and word embedding have been applied to a series of datasets. Finally, a comparative study has models and input features.

In addition to traditional NLP techniques, recent advancements in neural architecture, such as recurrent neural networks (RNNs), convolution neural networks (CNNs), and transformers, have significantly enhanced sentiment analysis performance. These architectures capture intricate linguistic patterns, helping to refine the detection of subtle sentiment indicators within large-scale datasets. This review also assesses the effectiveness of hybrid models that combine multiple deep learning approaches to improve sentiment classification.

Our findings reveal that model choice and feature selection significantly influence sentiment analysis accuracy and efficiency. By examining different preprocessing methods, feature representations, and neural network configurations, this paper aims to identify approaches for tracking sentiment analysis in dynamic, real-world environments.

# Introduction:

In today’s digital age, massive amounts of textual data are generated daily through various platforms such as social media, e-commerce websites, and online reviews. Extracting valuable insights from this vast sea of information is critical for businesses, organizations, and researchers. Sentiment analysis, also known as opinion mining, has emerged as one of the most effective techniques for analyzing and understanding the emotions, attitudes, and opinions expressed in textual data. By employing Natural Language Processing (NLP), sentiment analysis enables automated systems to interpret, process, and classify text based on the sentiments conveyed, whether they are positive, negative, or neutral**.[3]**

Sentiment analysis is a subfield of NLP that focuses on determining the emotional tone behind a series of words in each text. It involves using computational techniques to assess the sentiments or opinions expressed by individuals in documents, social media posts, reviews, and other types of textual data **[1].** For instance, a review like “I love this product” indicates positive sentiment, while “This service is terrible” reflects negative sentiment.

This method of analysis holds immense value for various industries and domains. In marketing and customer service, sentiment analysis enables businesses to monitor customer feedback, helping them gauge product reception and service satisfaction. On social media platforms like Twitter or Facebook, it aids in identifying public opinion trends, tracking consumer attitudes, and even predicting the success or failure of marketing campaigns **[1][6**]. Sentiment analysis is also critical for political campaigns, allowing political organizations to understand voter sentiment and design targeted strategies.

With the exponential growth of online platforms and the increasing importance of customer and user-generated content, sentiment analysis has become a vital tool for decision-makers. By processing unstructured data into actionable insights, businesses can make data-driven decisions, improve customer satisfaction, and even gain competitive advantages**.[5]**

Sentiment analysis has evolved significantly over the years. Early sentiment analysis models largely depended on traditional machine learning methods like Naïve Bayes, Support Vector Machines (SVM), and Maximum Entropy classifiers. These models classified text based on features such as unigrams (single words), bigrams (pairs of consecutive words), and sentiment lexicons (predefined dictionaries of positive and negative words) **[7]**. While these approaches performed reasonably well on large datasets, they faced limitations in handling the intricacies of human language. For instance, negations, sarcasm, and context-dependent meaning were often

misinterpreted by these models, leading to inaccurate classifications.

Consider the sentence “The product is not great.” A traditional model, using a sentiment lexicon, may incorrectly classify this as positive sentiment due to the presence of the word “great.” However, the negation “not” fundamentally alters the meaning of the sentence, making it negative. Such challenges prompted the development of more advanced sentiment analysis techniques**.[**8]

The rise of deep learning and neural networks marked a turning point in the evolution of sentiment analysis. Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) networks were introduced to better capture dependencies between words in a sequence. LSTM models improved sentiment analysis by considering the sequential nature of text, making them more effective at capturing the contextual meaning of words and phrases **[4].** This was especially useful for longer texts and complex sentence structures.

However, LSTM models still had some limitations, such as unidirectionality. They processed text either from left to right or right to left, which could result in missing crucial contextual information from the opposite direction **[2]**. This limitation was addressed with the introduction of transformers and, more specifically, Bidirectional Encoder Representations from Transformers (BERT).

The advent of transformers, particularly BERT, revolutionized sentiment analysis by enabling models to process text bidirectionally. Unlike traditional models, which analyze text in a linear, unidirectional fashion, BERT reads text both from left to right and right to left. This bidirectional processing allows BERT to fully understand the context of each word in a sentence, improving sentiment prediction accuracy, especially for sentences with complex structures, negations, and sarcasm.

For example, BERT would be able to correctly classify “The service is not great” as negative by taking the entire sentence context into account, including the negation “not.” BERT’s self-attention mechanism further enhances its ability to focus on important words or phrases within a sentence, regardless of their position, making it exceptionally adept at handling nuanced sentiment**.[5]**

BERT is pre-trained on large corpora of text in an unsupervised manner, allowing it to learn rich language representations. It is then fine-tuned on specific sentiment analysis tasks using labeled datasets, resulting in highly accurate sentiment classification models. BERT’s versatility and state-of-the-art performance have made it the go-to model for many sentiment analysis applications**.[7]**

# Challenges in Sentiment Analysis

Despite the progress in sentiment analysis techniques, certain challenges persist. Sentiment analysis relies heavily on language, and as such, it must contend with linguistic complexities such as ambiguity, idiomatic expressions, and cultural nuances. Detecting sarcasm remains a particularly difficult task for sentiment analysis models, as sarcastic statements often carry a sentiment opposite to their literal

meaning.

Moreover, informal language, which is prevalent on social media platforms, presents unique difficulties. Tweets, for example, often include abbreviations, slang, emojis, and hashtags, all of which complicate the task of determining sentiment. Additionally, sentiment can vary depending on context, subjectivity, and cultural differences, making it challenging to create one-size-fits-all sentiment analysis models.

# Applications for Sentiment Analysis

Sentiment analysis finds applications in a wide range of domains, including:

Customer Feedback Analysis: Companies use sentiment analysis to evaluate customer reviews and feedback, gaining insights into customer satisfaction and identifying areas for improvement.

Social Media Monitoring: Sentiment analysis helps monitor public opinion on social media platforms, providing real-time insights into brand perception, political trends, or social movements.

Market Research: Sentiment analysis enables companies to understand consumer sentiment toward products, services, and brands, which helps in forecasting market trends and informing product development strategies.

Political Campaigns: Political analysts use sentiment analysis to measure public opinion about candidates, policies, and campaigns, assisting in strategic decision- making.

1. **Literature Survey**

The problem statement Reviews the latest studies that have employed deep learning to solve sentiment analysis problems, such as sentiment polarity. Models using term frequency- inverse document frequency (TF-IDF) and word embedding have been applied to a series of data sets

The Solution we find is Recurrent Neural Networks (RNNs**)** are employed to address this problem. By maintaining a hidden state, RNNs can capture the context of previous words in a sequence, enabling them to better understand the overall sentiment of the text. This approach allows for more accurate sentiment classification compared to traditional methods.

The methodology involves **training RNNs** on a large dataset of labelled text. The RNNs learn to process sequences of words and identify patterns associated with positive, negative, or neutral sentiment. The trained models can then be used to classify new text examples**.[1]**

The Problem Statement is Sentiment analysisfaces challenges in accurately interpreting complex natural language, which often includes ambiguities, idiomatic expressions, and nuanced meanings. These linguistic complexities can hinder models' ability to correctly classify text as positive, negative, or neutral.

The Solution is Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) are employed to address these challenges. RNNs handle sequences of text by maintaining hidden states, capturing context and understanding the relationships between words. CNNs, on the other hand, use filters to identify local features and patterns within text, providing a complementary approach to capturing the nuances of language.

The methodology involves training RNNs and CNNs on large datasets of labelled text. The models learn to process text sequences and extract relevant features, enabling them to better understand and interpret complex language nuances. The trained models can then be used to classify new text examples based on their sentiment**.[2]**

The Problem Statement is Sentiment analysis on Twitter faces challenges due to the potential imbalance of classes in the datasets, where there may be more positive tweets than negative ones. This imbalance can affect the accuracy of models and require techniques to address it.

The Solution we find is Ensemble methods are employed to improve the robustness and reduce variance in sentiment analysis on Twitter. By combining predictions from multiple models trained on different subsets of the training data, ensemble methods can enhance overall accuracy and mitigate the impact of imbalanced classes.

The methodology involves training multiple models on different subsets of the training data, such as random forests. Each model's predictions are then combined using techniques like voting or averaging to arrive at the final classification. This approach helps to reduce the reliance on any single model and improves the overall performance of the sentiment analysis task**. [3]**

The Problem statement is Sentiment analysis using RNNs faces challenges in capturing long-term dependencies in text data, particularly for longer sentences or documents. The vanishing or exploding gradient problem during backpropagation through time (BPTT) can hinder the model's ability to effectively learn and represent the relationships between words over extended sequences.

The solution we find is N-grams, bigrams, stop words, POS tagging, stemming, and a lexicon-based approach are used as features to enhance the RNN's ability to capture relevant information from the text. Additionally, Backpropagation Through Time (BPTT) is employed as the training algorithm, which unfolds the network over time and adjusts weight based on the error at each time step. This helps to address the vanishing gradient problem and improve the model's performance.

The methodology involves preprocessing the text using techniques like N-grams, bigrams, stop word removal, POS tagging, and stemming to extract relevant features. The pre- processed text is then fed into the RNN, which teaches to process sequences of words and identify patterns associated with positive, negative, or neutral sentiment. BPTT is used to update the model's weight based on the error at each time step, allowing the RNN to capture long-term dependencies more effectively**. [4]**

The Problem Statement is Sentiment analysis of product reviews faces challenges due to the presence of spelling mistakes, grammar errors, and informal text, which can hinder the accuracy of models. Additionally, the performance of these models can vary depending on the amount of training data available.

The Solution is Supervised learning techniques such as Naive Bayes, SVM, and ANN are employed to analyze product reviews and classify them based on their sentiment. Feature extraction and filtering based on word frequency are used to identify the most relevant features for sentiment classification.

The methodology involves preprocessing the product reviews to address issues like spelling mistakes and grammar errors. Feature extraction is then performed to identify the most informative words or phrases. Supervised learning models are trained on labeled datasets of product reviews to learn the relationship between the extracted features and the corresponding sentiment. The trained models can then be used to classify new product reviews**. [5]**

The Problem Statement is Sentiment analysis of Twitter data faces challenges due to the need to capture long-term dependencies in text and address the limitations of traditional RNNs, such as vanishing and exploding gradients.

The Solution is Neural networks and LSTM (Long Short-Term Memory) are employed to address these challenges. A simple neural network is used to classify sentiments, while LSTM, a specialized form of RNN, is used to overcome the limitations of traditional RNNs and maintain long-term dependencies in text.

The methodology involves preprocessing the Twitter data to remove noise and extract relevant features. A simple neural network is then trained on the pre-processed data to

classify sentiments. To improve the model's performance, LSTM is used to capture long-term dependencies in the text, addressing the vanishing gradient problem**.[6]**

The Problem Statement is Sentiment analysis in the Bangla language faces challenges due to the limited availability of labelled datasets, which hinders the training of robust models.

The Solution we find is Transfer learning using BERT and CNN-BiLSTM is employed to address these challenges. BERT is used to generate context-aware embeddings, while CNN- BiLSTM is integrated with BERT to improve decision-making in sentiment analysis.

The methodology involves fine-tuning the pre-trained BERT model on a smaller Bangla dataset to adapt it to the specific task of sentiment analysis. The fine-tuned BERT embeddings are then integrated with a CNN-BiLSTM architecture to capture both local and sequential features in the text. The combined model is trained on labelled Bangla data to classify sentiments**. [7]**

The Problem Statement is Sentiment analysis of tweets faces challenges in capturing complex dependencies and understanding the nuances of language.

The Solution is BERT is employed to address these challenges. BERT is a powerful language model that can generate context-aware embeddings, enabling it to better understand the meaning of words in context and capture complex dependencies.

The methodology involves fine-tuning the pre-trained BERT model on a dataset of labelled tweets. The fine-tuned model is then used to predict the sentiment of new tweets**. [8]**

**Table 1 – Comparison of literature survey**

|  |  |  |
| --- | --- | --- |
| **Papers** | **Methodology** | **Limitations** |
| Sentiment analysis based on deep learning: A comparative study | RNNs are used to process sequences of text by maintaining a hidden state that captures information about previous words in the sequence. | Can be slow to train and may still struggle with very long sequences. |
| Sentiment analysis and the complex natural language Author: Muhammad Taimoor Khan | RNNs handle sequences of text,  maintaining hidden states to capture context. CNNs use filters to capture local features and patterns within text. | Complex language often includes ambiguities, idiomatic expressions, and nuanced  meanings that can be  challenging for models to interpret accurately. |
| Performance analysis of Ensemble methods on Twitter sentiment analysis using NLP techniques  Author: Mohana Reddy | Combines predictions from multiple models trained on different subsets of  the training data (e.g., random forests). Each model votes on the final | Sentiment analysis datasets on Twitter may have imbalanced  classes (e.g., more positive  tweets than negative ones), which can affect model |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | classification, improving robustness and reducing variance. | | performance and require balancing techniques. | |
| Sentimental Analysis by using RNN  Author: Alpna Patel and Arvind Kumar | | | Used N-grams and bigrams as features, Stop Words, POS Tagging, Stemming, Lexicon-Based approach.  Uses BPTT for training, a variant of backpropagation that unfolds the  network over time and adjusts weights based on the error at each time step. | | RNNs often struggle with capturing long-term dependencies in text data,  especially in longer sentences or documents, due to vanishing or exploding gradients during  backpropagation through time  (BPTT). | |
| Text Analysis for Product Reviews for Sentiment Analysis using NLP Method.  Author: Muthukumaran, Suresh | | | Supervised learning techniques (e.g., Naive Bayes, SVM, ANN). Feature extraction and filtering based on frequency. | | Performance can be affected by spelling mistakes, grammar  errors, and informal text.  Accuracy of semantic orientation can vary depending on the  amount of training data. | |
| Sentiment Analysis using Neural Network and LSTM Author: Ch. Satyanarayana | | | A simple neural network was applied to classify sentiments from the Twitter dataset. LSTM, a specialized form of RNN, was used to overcome the limitations of RNN (such as vanishing  and exploding gradients) and to maintain long-term dependencies.  Accuracy: 87.4%  Precision: 76% | | The study used a static dataset (1.6 million tweets) collected from Kaggle. It does not account for real-time data analysis, limiting its practical applicability for real-time sentiment analysis. The model was trained only on Twitter data and may not  generalize well to other types of text data, such as product reviews, blogs, or longer-form  social media posts. | |
| Transfer Learning for Sentiment Analysis Using BERT Based Supervised Fine-Tuning  Author: Nusrat Jahan Prottasha, Abdullah As Sami, Saydul Akbar Murad. | BERT (Bidirectional Encoder Representations from  Transformers): Utilized for  context-aware embeddings. CNN-BiLSTM (Convolutional Neural Network with  Bidirectional Long Short- Term Memory): Integrated with BERT for better decision- making in sentiment analysis.  Accuracy: 94.5% | | Limited availability of labelled datasets in Bangla NLP makes it difficult to train robust models. Fine-tuning large models like  BERT and integrating them with CNN-BiLSTM increases  computational overhead  compared to classical machine learning methods. | |

|  |  |  |
| --- | --- | --- |
| A BERT Framework to Sentiment Analysis of Tweets  Author: Abayomi Bello | BERT (Bidirectional Encoder Representations from  Transformers):   * Training Epochs: 10 * Learning Rate: 1 × 10^−5 * Optimizer: Adam * Batch Size: 128 * **Activation Function**: SoftMax * Loss Function: Sparse Categorical Cross entropy   Accuracy: 93% | Only captures local features, such as edges or simple patterns, which might miss out on more  complex dependencies. |

### Key Findings:

* RNNs are a popular choice for sentiment analysis: Recurrent Neural Networks (RNNs) are frequently used due to their ability to handle sequential data like text. They maintain a hidden state to capture context and improve accuracy.
* Ensemble methods enhance performance: Combining predictions from multiple models trained on different subsets of data (e.g., using ensemble methods) can improve robustness and reduce variance.
* CNNs are effective for capturing local features: Convolutional Neural Networks (CNNs) are well-suited for identifying local patterns and features within text.
* Feature engineering and filtering are crucial: Feature extraction and filtering

techniques, like N-grams, stop words, and POS tagging, are essential for improving model performance.

* Data imbalances can affect results: Imbalanced datasets, especially on platforms like Twitter, can bias model performance and require balancing techniques.
* RNNs are a strong baseline for sentiment analysis.
* Addressing limitations like slow training and data imbalances is crucial.
* Feature engineering and filtering are essential for effective sentiment analysis.
* Ensemble methods and CNNs can be used to enhance performance.

### Common Problems:

The common problem among the sentiment analysis papers surveyed is the challenge of handling complex language nuances. While the models used in these papers (RNNs, CNNs, ensemble methods) have shown promising results, they often struggle to accurately interpret subtle expressions, idioms, and context-dependent meanings. This is particularly challenging when dealing with informal text, such as social media posts or reviews, where language can be more colloquial and expressive.

Additionally, data imbalance is another common issue. Many datasets used for sentiment

analysis is skewed towards one sentiment (e.g., more positive reviews than negative), which can bias model performance and make it difficult to accurately predict both positive and

negative sentiments.

Finally, computational limitations can also be a problem, especially when dealing with large datasets or complex models. Training and inference can be time-consuming, limiting the

scalability of these models.

Addressing these challenges will be crucial for developing more accurate and robust sentiment analysis models.

### Solutions:

#### Key Solutions and Their Effectiveness:

**BERT (Bidirectional Encoder Representations from Transformers)** is a powerful language model that has significantly advanced state-of-the-art in natural language processing tasks, including sentiment analysis. It addresses many of the common challenges encountered in sentiment analysis, such as handling complex language nuances and dealing with imbalanced datasets.

Contextual Understanding: BERT is a bidirectional model, meaning it considers both the left and right context of a word when generating its representation. This enables it to capture the nuances of language and understand the meaning of words in context.

Pre-trained on Massive Datasets: BERT is pre-trained on a large corpus of text, which allows it to learn general language patterns and representations. This can significantly improve performance, especially for tasks with limited labelled data.

Fine-tuning Flexibility: BERT can be easily fine-tuned for specific tasks, such as sentiment analysis, by adding a classification layer on top of the pre-trained model. This allows for efficient adaptation to new domains and datasets.

Handles Complex Language Nuances: BERT's ability to understand contextual relationships and capture the meaning of words in context makes it well-suited for handling complex language nuances, such as sarcasm, irony, and figurative language.

### UML diagrams



**Figure4.1: Use Case Diagram**

This UML case diagram represents the functionality of a Sentiment Analyzer system, depicting interactions between different actors (User, Admin, and Twitter) and the system's various use cases. This case diagram outlines the flow of how the sentiment analyzer interacts with users, administrators, and external services, including both normal and exceptional scenarios.

#### 

#### 

#### Figure 4.2 - Class Diagram

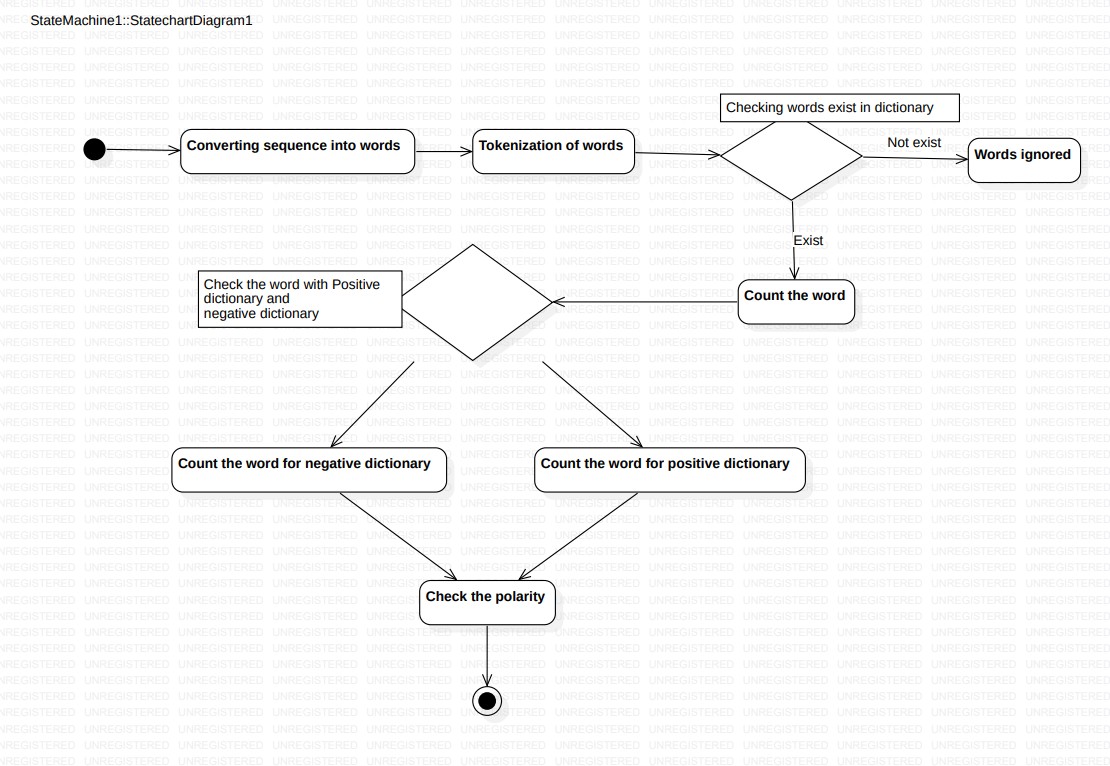
#### This UML class diagram represents the structure of a Sentiment Analyzer system, showing the key classes, their methods, and how they interact with each other. TextProcessor handles text preprocessing and provides tokenized text to th This diagram illustrates the functional components of a sentiment analysis system, focusing on processing, analyzing, storing, and displaying the results

****

**Fig 4.3 -Activity diagram**

This activity diagram represents the process of analyzing a product review. The steps outline how user input and data processing lead to a review result. If the product is not found, an Error message is displayed, and the process terminates. If the product is found, the process proceeds to the next step. Word count using map count: The system counts the words in the review, possibly using a map or hash map structure to tally the word occurrences.

This activity diagram visually breaks down the step-by-step actions in the review analysis process, highlighting how the system handles product reviews from input to output



**Fig 4.4- state diagram**

The diagram in question is a UML state chart diagram showcasing the steps in analyzing a sequence of words for polarity.The process concludes with checking the polarity, marked by a circle with a white center indicating the final state.This diagram visually outlines the workflow for text analysis by breaking it into key stages and decision points.

Initial State: A black circle marks the starting point.

Converting sequence into words: The first state in the process.

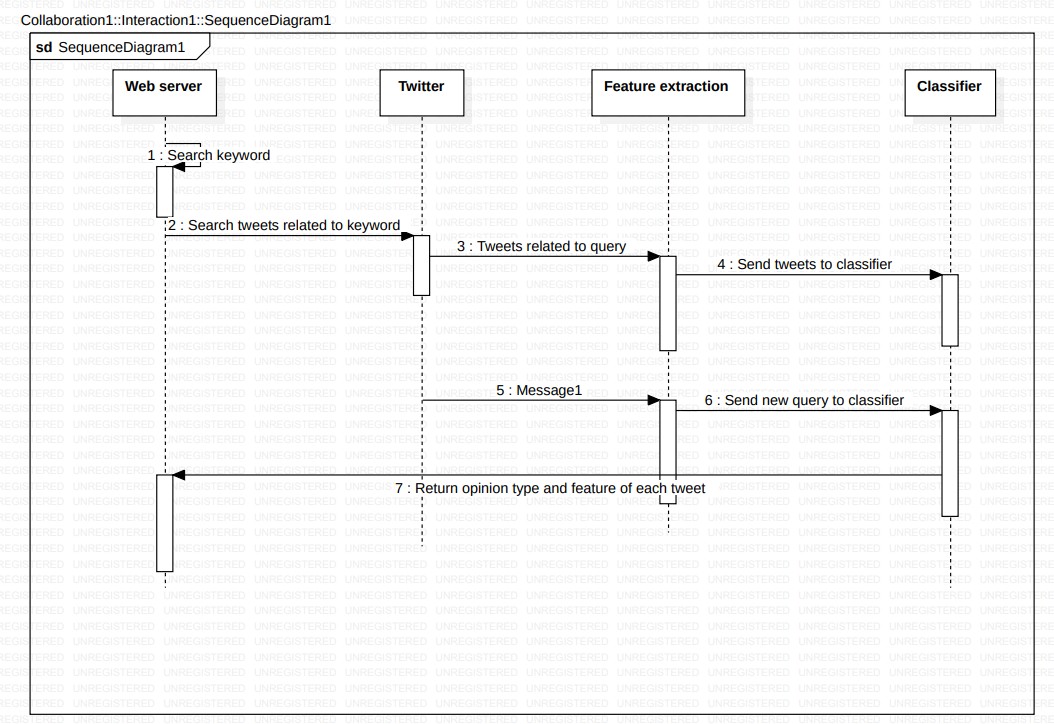
Tokenization of words: Next, the words are tokenized.

Decision Point: Checks if the words exist in the dictionary. This decision bifurcates into two possible paths

Checking with Dictionaries: The counted words are then checked against a positive and a negative dictionary, leading to:

Counting the word for the negative dictionary.

Counting the word for the positive dictionary.



**Fig 4.5 – Sequence diagram**

This UML sequence diagram lays out the interactions between a web server, Twitter, a feature extraction component, and a classifier in a system analyzing tweets:

Web Server: Starts by searching for a keyword.

Web Server to Twitter: Searches for tweets related to the keyword.

Twitter: Responds with tweets matching the query.

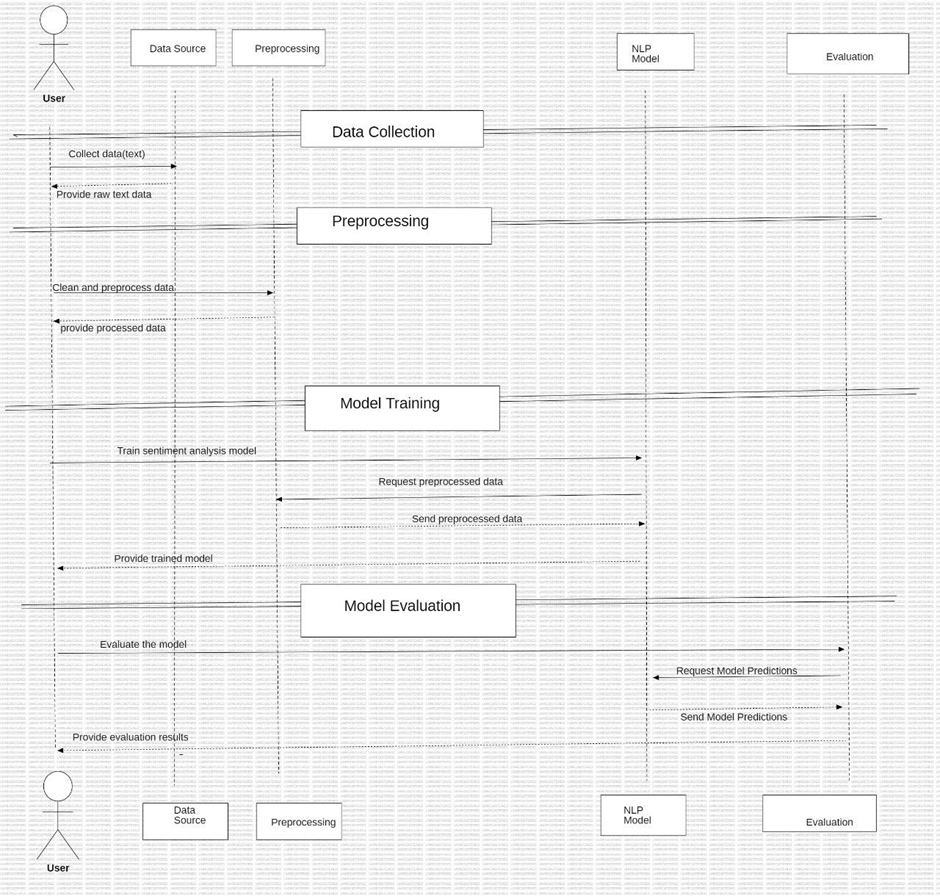
Feature Extraction: Receives the tweets and sends them to the classifier.

Web Server to Feature Extraction: Sends a message for further analysis.

Feature Extraction: Initiates a new query to the classifier.

Feature Extraction to Web Server: Provides the opinion type and feature of each tweet.

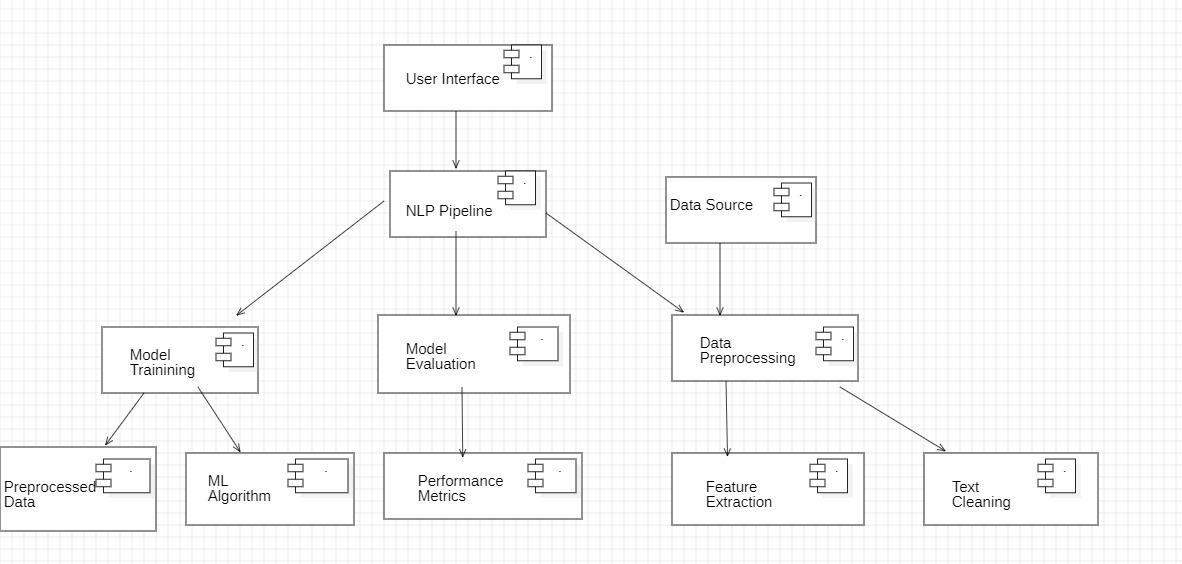
This diagram is essential for visualizing the flow and interaction between these components in tweet analysis based on keyword search.

****

**Figure 4.6 – Interaction diagram**

The diagram here is an interaction diagram showcasing the workflow of a sentiment analysis model, detailing the interactions between User, Data Source, Preprocessing, NLP Model. Begins with the User collecting data from the Data Source.

Data Source provides raw text data.Raw text data is cleaned and preprocessed.Preprocessed data is then provided.Sentiment analysis model is trained using the preprocessed data.Preprocessing requests and sends this data to the NLP Model.The trained model is then provided.The trained model is evaluated.Model Evaluation requests model predictions from the NLP Model.Predictions are sent back and results are provided.This sequence highlights the entire workflow from data collection to model training and evaluation, emphasizing the continuous interactions between each stage.

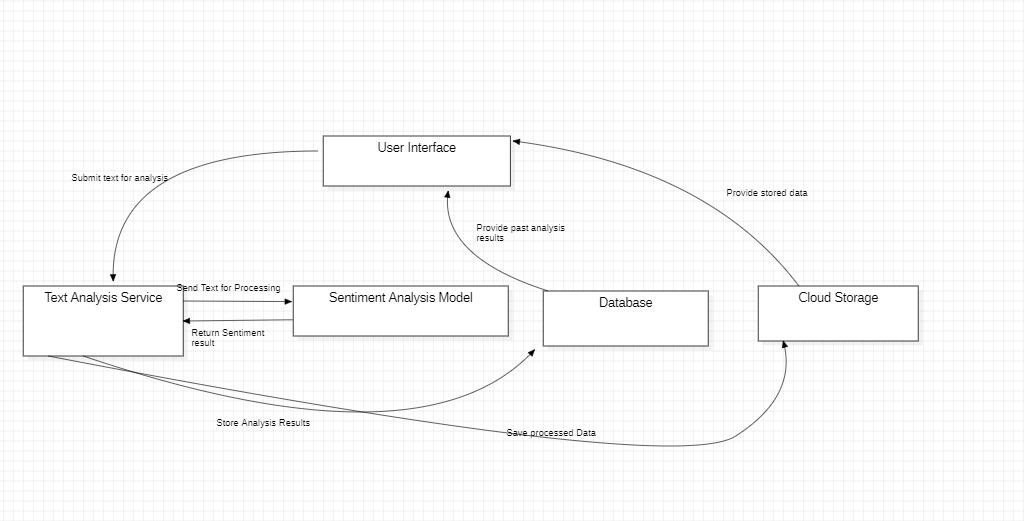


**Figure 4.7- componenent diagram**

A component diagram for a sentiment analysis system would illustrate the system’s major components and how they interact to perform sentiment analysis on text data. This system typically processes text input to classify emotions, such as positive, negative, or neutral sentiments, using techniques from natural language processing (NLP) and machine learning.

In a component diagram, each of the above components is represented as a box. These boxes are connected by arrows or lines indicating the flow of data. For example:

* The Input Data component sends raw text to the Preprocessing Component.
* The Preprocessing Component forwards cleaned text to the Feature Extraction component.
* The Feature Extraction component sends numerical representations of the text to the Sentiment Classifier, and so on.



**Figure 4.8 – Data flow diagram**

The above diagram represents a Data flow diagram. User Interface: Where users submit text for analysis and receive sentiment analysis results.

Text Analysis Service: This is the core component that interacts with other parts:

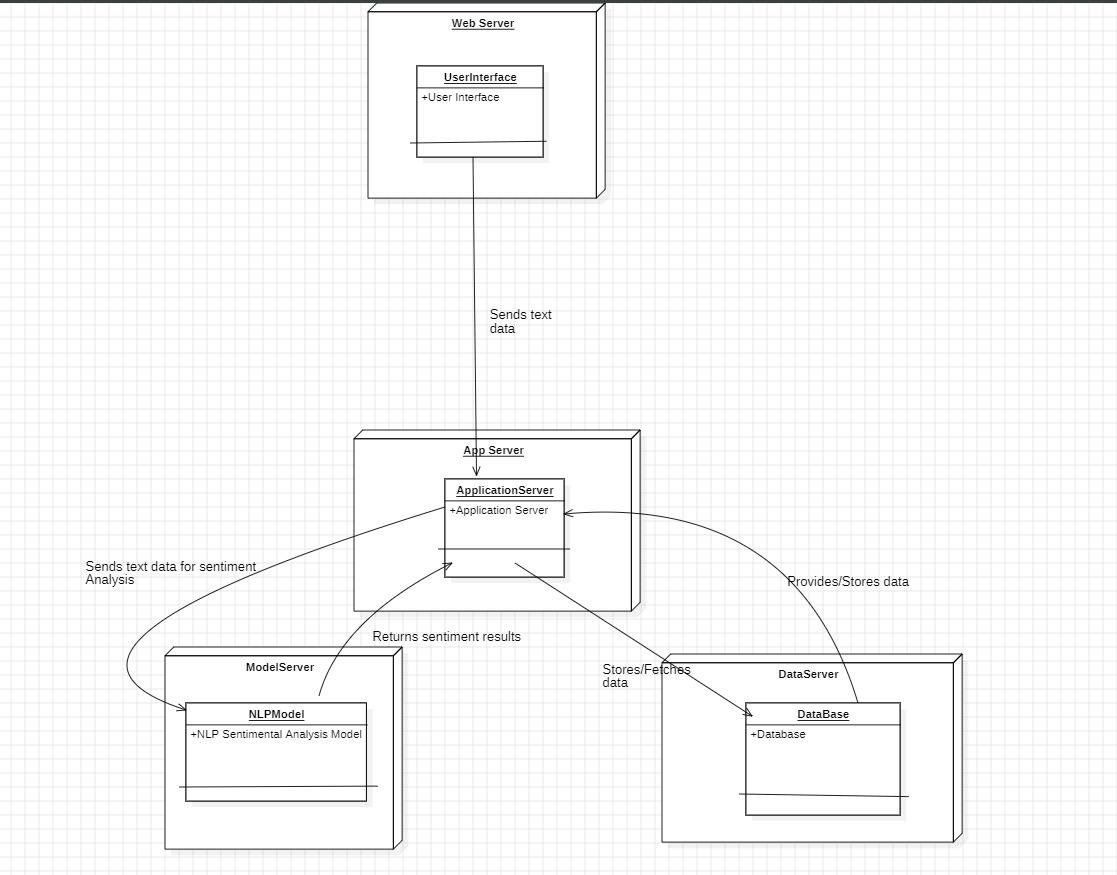
Receives text from the User Interface.Sends text to the Sentiment Analysis Model for processing. Stores analysis results in the Database.

Sentiment Analysis Model: Processes text and sends the results back to the Text Analysis Service.

Database: Stores the processed data from the Sentiment Analysis Model and analysis results from the Text Analysis Service.

Cloud Storage: Receives data from the Database and provides stored data to the User Interface.

This diagram shows how data moves around in a text and sentiment analysis system, from the user submitting text to storing and retrieving results.



**Figure 4.9 – deployment diagram**

The deployment diagram here, labeled "Figure 9 - Deployment Diagram," maps out the structure and interactions of a sentiment analysis system.

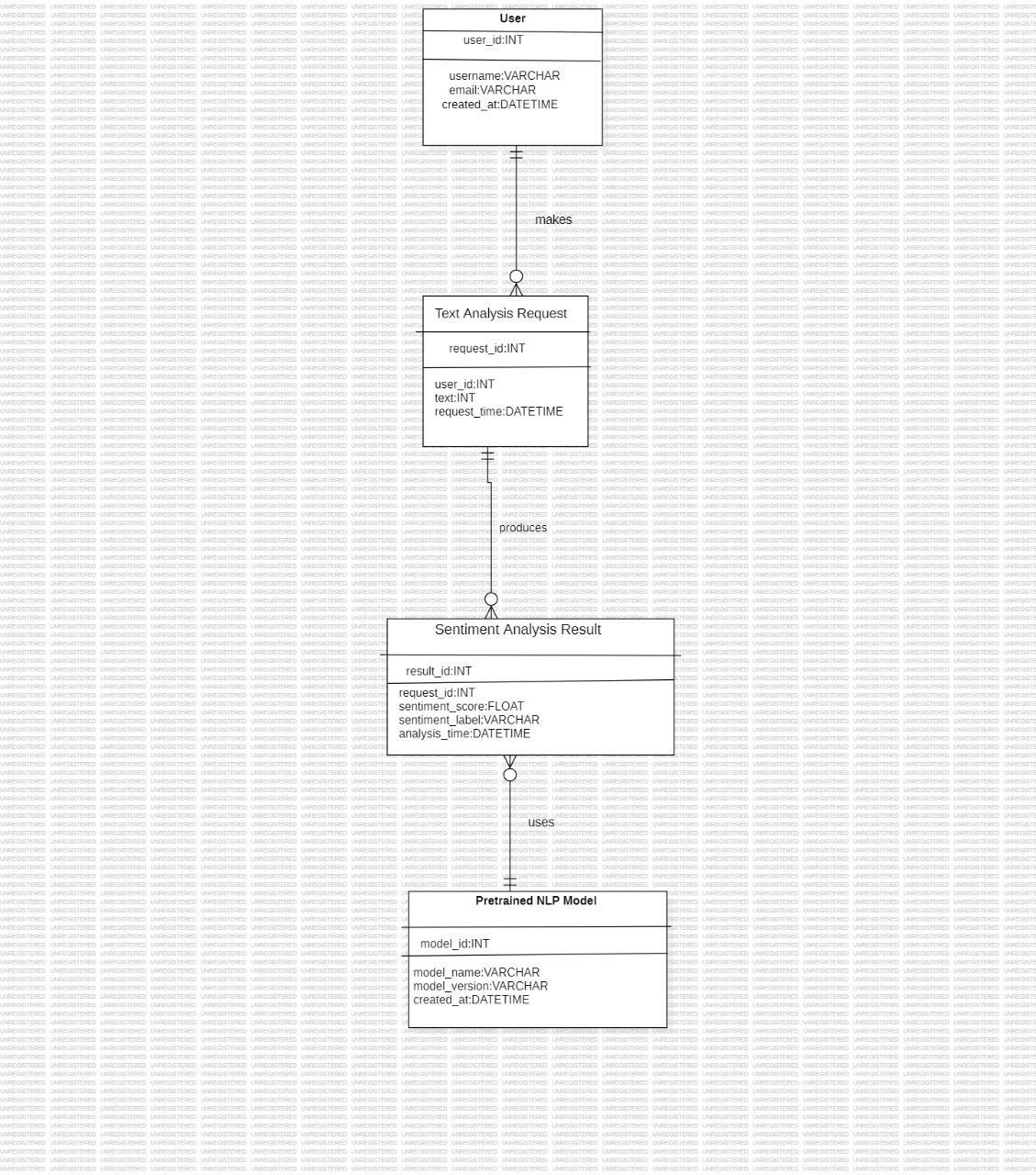
Web Server: Hosts the User Interface component. Responsible for collecting text data from users and sending it to the App Server.

App Server: Contains the Application Server component.

Model Server: Houses the NLP Sentiment Analysis Model component. Handles sentiment analysis, processing text data received from the App Server, and returning the results.

Data Server: Features the Database component. Stores and retrieves data for the App Server.

This deployment diagram visually represents how different servers and components interact to enable sentiment analysis, outlining their roles and connections within the system.



**Figure 4.10- ER diagram**

A Sentiment Analysis Result utilizes one Pretrained NLP Model. It mainly contains attributes, relationships and users. The connection between user and attributes is established.

This diagram captures the relationships and key attributes within a sentiment analysis system, helping visualize the flow and interaction of data.

**5.PROPOSED SYSTEM**

This project implements sentiment analysis on movie reviews using various machine learning models, including Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Random Forest, and Bidirectional Encoder Representations from Transformers (BERT). The goal is to classify reviews as either positive or negative.

**1. Setup and Data Acquisition**

Key libraries like TensorFlow, transformers, pandas, BeautifulSoup, and scikit-learn are imported. The IMDb dataset, containing 50,000 labeled reviews, is downloaded and split into training and testing datasets, each with equal numbers of positive and negative reviews.

**2.Data Loading and Preparation**

A custom function loads the movie reviews, mapping them to their respective sentiments (1 for positive, 0 for negative). Reviews undergo text cleaning where HTML tags and special characters are removed using BeautifulSoup and regular expressions.

**3. Tokenization and Padding**

The reviews are tokenized using the Keras Tokenizer, converting them into integer sequences. These sequences are then padded to a uniform length to ensure compatibility with neural network models.

**4. Data Visualization**

Sentiment distribution is visualized using a bar chart, providing insights into the balance of positive and negative reviews. Additionally, word clouds for both positive and negative reviews are generated to highlight the most frequent terms.

**5. Model Construction**

**Four models are built:**

**CNN:** Includes embedding, convolution, max pooling, flattening, and dense layers. It is compiled with Adam optimizer and trained on padded text sequences.

**RNN (LSTM):** Features embedding and LSTM layers for capturing temporal dependencies in the reviews.

**Random Forest**: Uses Bag-of-Words for feature extraction and trains a random forest classifier.

**BERT:** BERT tokenizer encodes reviews, and the TFBertModel classifies the sentiment using dense layers.

**6. Model Evaluation**

Each model is evaluated based on test accuracy and loss. Random Forest uses a classification report and confusion matrix, while CNN, RNN, and BERT are evaluated on metrics such as test accuracy and loss.

**7. Sentiment Prediction and Feature Importance**

A custom function predicts the sentiment for new reviews using all four models. Feature importances from Random Forest are visualized to identify the key words influencing the predictions.

**8. Conclusion**

The project highlights the versatility of traditional and deep learning models in handling sentiment analysis. Model performance is visualized through metrics like training and validation accuracy, while feature importance reveals the most influential words in the Random Forest model.

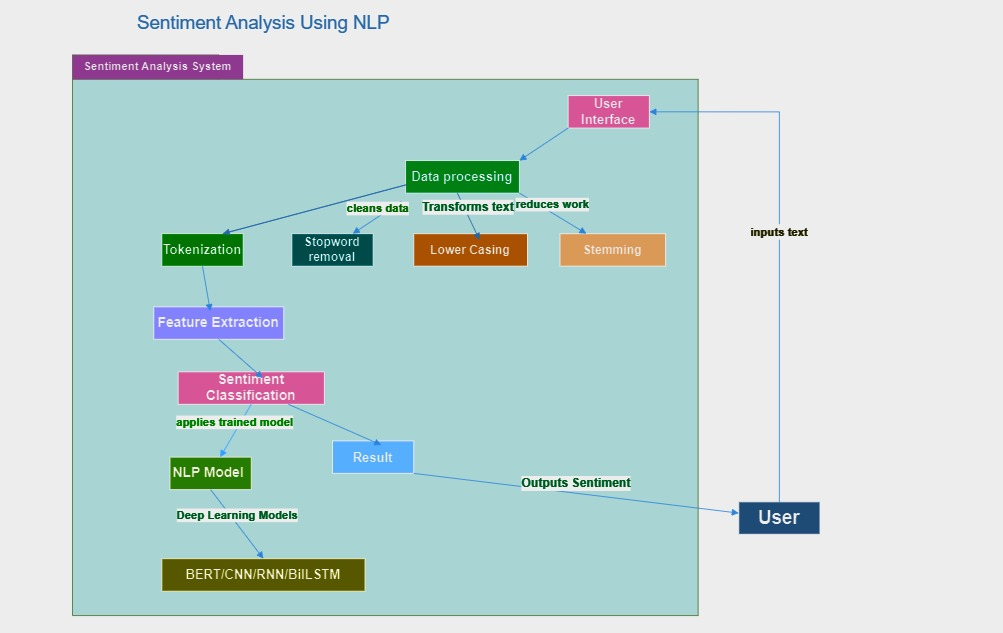
**6.1. Dataset**

The IMDb Large Movie Review Dataset, used for this project, is a comprehensive dataset specifically designed for binary sentiment classification tasks in natural language processing (NLP). It contains 50,000 highly polarized movie reviews from IMDb, with an equal split between positive and negative reviews. The reviews are divided into two equal-sized sets: a training set and a testing set, each containing 25,000 reviews. Each set has 12,500 positive and 12,500 negative reviews, ensuring balance for training and testing. This dataset focuses on extreme sentiments by excluding neutral reviews, making it particularly useful for binary sentiment analysis tasks.

The reviews themselves vary in length, style, and complexity, offering a real-world challenge in analyzing natural language. Positive reviews are defined as those with a rating of 7 or higher (out of 10), while negative reviews have a rating of 4 or lower. Ratings of 5 and 6 are excluded, removing any potential ambiguity in sentiment. The dataset also provides additional unlabeled reviews, which can be used for unsupervised learning or semi-supervised learning tasks. The reviews in both positive and negative categories come from a wide range of movies, capturing diverse genres and writing styles, which makes the dataset an excellent resource for experimenting with sentiment analysis techniques.

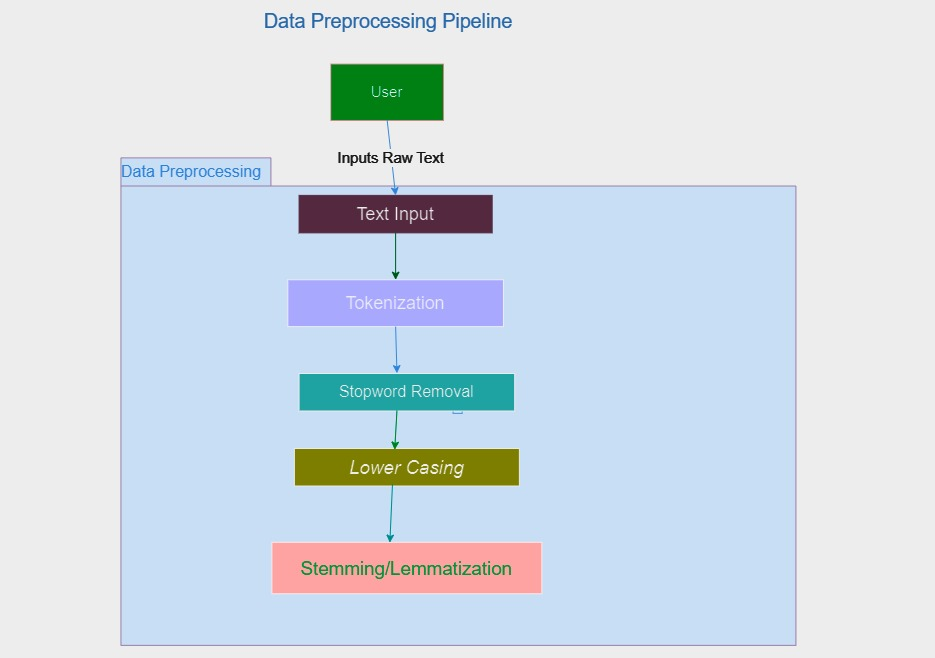
The structure of the dataset includes various files for labeled and unlabeled data, Bag-of-Words (BoW) features, and metadata such as IMDb URLs for each review. The dataset is widely used in research to benchmark machine learning models, especially in text classification tasks. Preprocessing is necessary to clean the reviews, remove HTML tags, and tokenize the text, enabling models like CNN, RNN, and BERT to learn from the data efficiently. This makes the IMDb dataset an ideal candidate for exploring various machine learning techniques in the context of natural language understanding and sentiment classification.

**6.2. Block Diagrams**



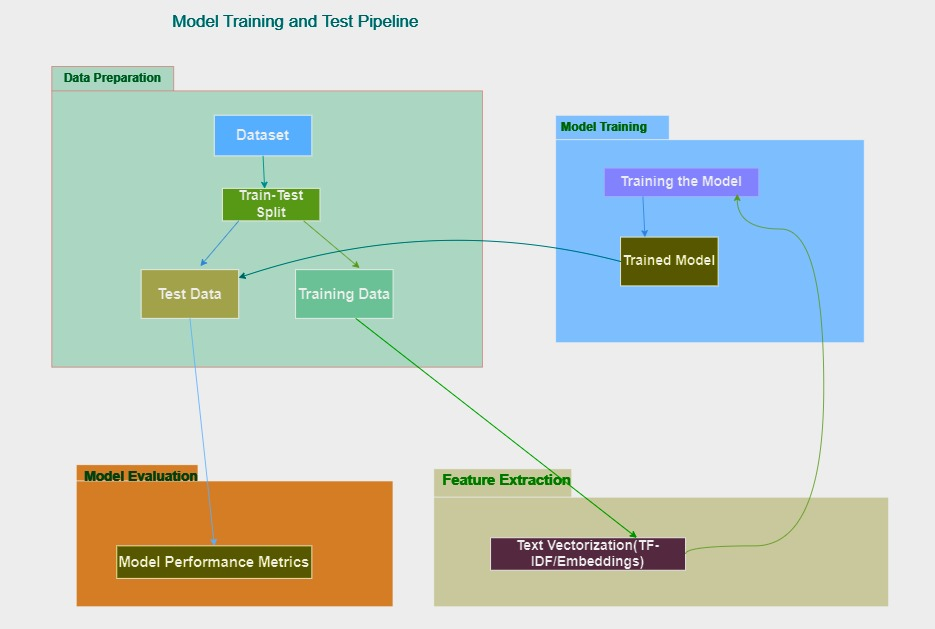
**Fig 6.2.1-Sentiment Analysis System using NLP**

The flowchart Fig11 represents a Sentiment Analysis System using NLP. It starts with user input text, which undergoes data processing (tokenization, stopword removal, lowercasing, stemming). Features are then extracted and fed into an NLP model (e.g., BERT, CNN, RNN, BiLSTM) for sentiment classification, resulting in the final sentiment output.



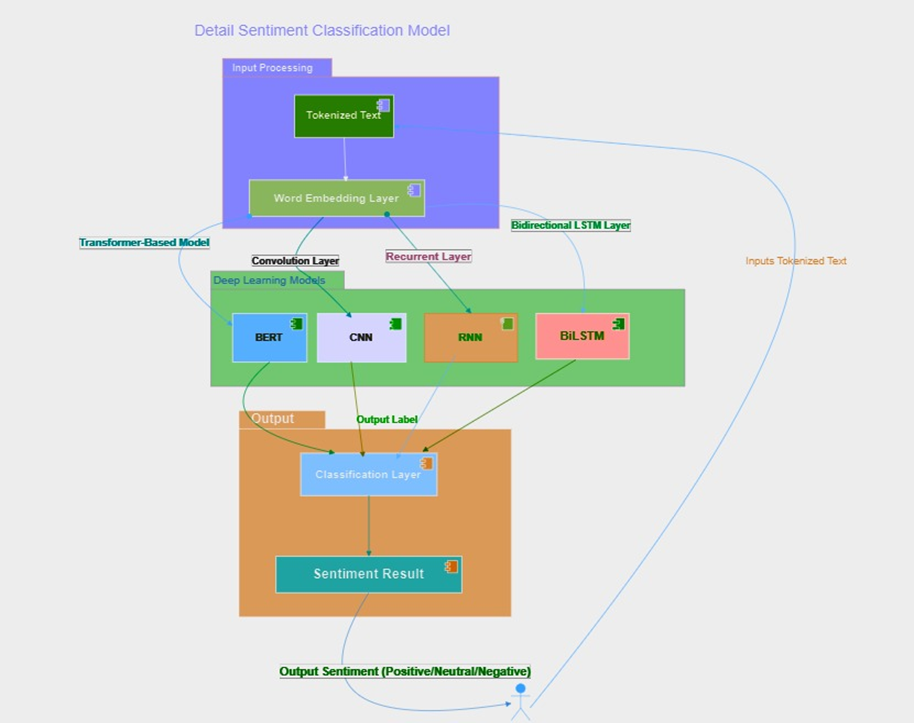
**Fig 6.2.2- Data Preprocessing Pipeline**

The flowchart shown in fig4.2.2 illustrates the Data Preprocessing Pipeline for text analysis. It begins with the user inputting raw text, followed by steps such as Text Input, Tokenization (breaking text into tokens), Stopword Removal (removing common words), Lowercasing (converting text to lowercase), and Stemming/Lemmatization (reducing words to their root form).



**Fig 6.2.3 - Model Training and Testing Pipeline**

The flowchart shown in fig 13 illustrates the Model Training and Testing Pipeline. It begins with Data Preparation, where the dataset is split into training and test sets. The Model Training section involves training the model using the training data and performing feature extraction (e.g., text vectorization using TF-IDF or embeddings). Finally, the Model Evaluation section assesses the model’s performance using metrics derived from the test data and the trained model. This pipeline visually represents the sequential steps involved in preparing data, training a machine learning model, and evaluating its performance.



**Fig 6.2.4 - Detailed Sentiment Classification Model**

The flowchart shown in fig 14 illustrates a Detailed Sentiment Classification Model. It starts with tokenized text input, which passes through an embedding layer. The text is then processed by three parallel deep learning models: BERT, CNN, and RNN with a BiLSTM layer. The outputs from these models are combined in a classification layer, leading to the final sentiment result (positive, negative, or neutral). This model showcases how advanced NLP techniques can be integrated to analyze and classify sentiments in text data.

**6.3. Requirement analysis**

The Requirement Analysis for our advanced sentiment analysis project, "Advanced Sentiment Analysis Using BERT: A Comparative Study on Real-Time Social Media Data," will cover various functional and non-functional requirements, ensuring that the system meets the necessary goals for performance, accuracy, and usability.

**Functional Requirements:**

1.**Data Collection and Preprocessing**

* Real-Time Data Ingestion: The system must be able to collect real-time social media data (e.g., from Twitter, Facebook, etc.) through APIs.
* Data Preprocessing: Implement techniques for tokenization, stopword removal, stemming, and lemmatization using tools like NLTK.
* Text Normalization: The system should normalize slang, abbreviations, and domain-specific terminology for better analysis.
* Handling Special Cases: Must recognize and handle emojis, hashtags, mentions, and URLs for more accurate sentiment analysis.

**2.Model Training and Fine-Tuning**

* Sentiment Model: Implement a transformer-based model (e.g., BERT) fine- tuned for sentiment analysis.
* Model Fine-Tuning: Incorporate domain-specific fine-tuning for social media texts, possibly using pre-trained models or transfer learning.
* Data Augmentation: Use techniques like synonym replacement, back translation, or text generation to improve training dataset quality.

**3.Sentiment Classification**

* Polarity Classification: The system must classify social media posts into positive, negative, or neutral sentiments.
* Emotion Classification (Optional): If applicable, add a multi-label classification feature to identify emotions such as joy, anger, sadness, etc.

4.**Sarcasm Detection: Include a specific mechanism to detect sarcasm.**

* Model Evaluation: Provide metrics such as accuracy, precision, recall, F1- score, and confusion matrices for model performance.
* Error Analysis: Enable users to explore where the model fails (e.g., sarcasm, domain-specific terminology).

**5.Real-Time Sentiment Analysis**

* Real-Time Processing: Ensure that the system can analyse and output sentiments in real time, with a focus on low-latency processing.
* Streaming Capabilities: Use tools like Apache Kafka or Spark Streaming for real-time data analysis.

**6**.**Feedback and Learning**

* User Feedback: Allow end-users or admins to provide feedback on sentiment accuracy to improve the model over time.
* Active Learning: Integrate an active learning loop to retrain the model periodically based on new and uncertain data.

**7.Deployment**

* API Development: Provide an API that allows third-party applications to request sentiment analysis on social media data.
* Visualization Dashboard: Develop a dashboard to visualize the sentiments in real time, using graphs and charts to present insights.

**Non-Functional Requirements**:

**1.Performance**

* Low Latency: Sentiment analysis must be processed in real-time or near real- time to meet user expectations.
* Scalability: The system should scale efficiently to handle a large volume of social media posts (e.g., thousands of posts per minute).
* Efficiency: Ensure that resource usage (CPU, GPU) is optimized for large datasets and high-frequency real-time processing.

**2.Accuracy and Precision**

* High Sentiment Accuracy: The model should achieve a high accuracy rate (e.g., >85%) in classifying sentiments, including complex cases like sarcasm and nuanced sentiments.
* Handling Imbalanced Data: Ensure that the model can handle class imbalances in positive, negative, and neutral sentiments.

**3.Security and Privacy**

* Data Privacy: Ensure that sensitive user information (e.g., from social media platforms) is anonymized and handled according to privacy regulations (e.g., GDPR).
* Secure API: The API should have proper authentication and encryption to protect data exchanges.

**4.Robustness**

* Error Handling: The system should handle missing, incomplete, or noisy data without crashing.
* Continuous Operation: Ensure the system is reliable and operational 24/7 for real-time analysis.

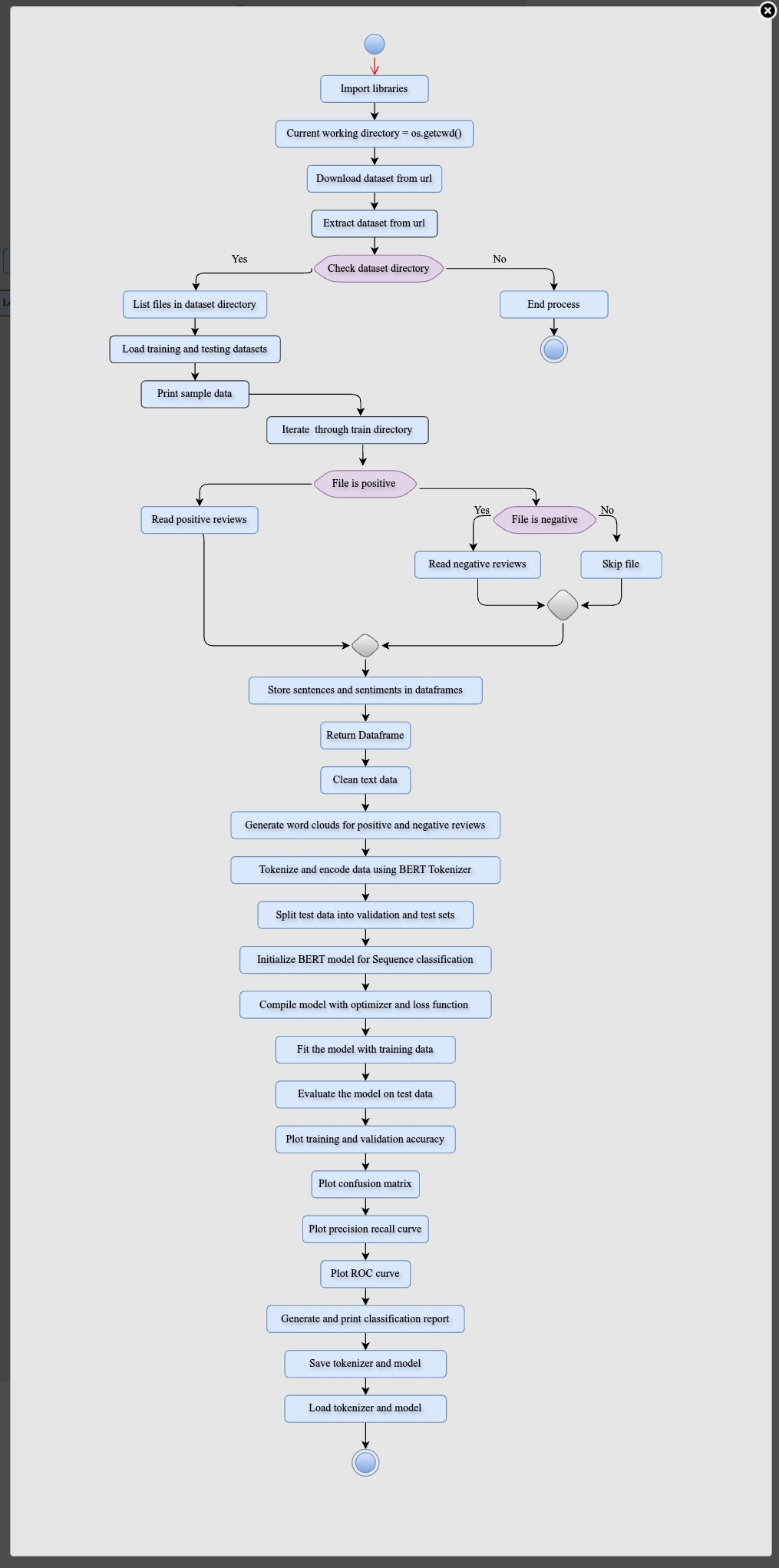
**5.Usability**

* User-Friendly Interface: Provide an intuitive dashboard for visualizing and interacting with sentiment analysis results.
* API Documentation: Offer clear and detailed API documentation for developers to integrate sentiment analysis into other applications.

**6.Adaptability**

* Flexible Model Updates: Allow the sentiment model to be retrained and updated periodically

**6.4. Control flow graph**



**Figure 15-Control flow graph**

Sentiment analysis identifies emotions in text, while control flow graphs (CFGs) visualize the decision paths in programs. In sentiment analysis, a CFG can represent the steps of text preprocessing, feature extraction, and sentiment classification. CFGs can also model decision paths, especially in decision-tree-based sentiment models, aiding explainability and debugging.

**6.5. Need for Cyclometric Complexity measures in our Project**

The cyclometric complexity of a control flow graph is a measure of its complexity. It is equal to the number of decision points (if statements, loops, etc.) plus one.

The cyclometric complexity of this control flow graph is 12 + 1 = 13.

The cyclometric complexity of a control flow graph can also be calculated using the following formula:

M = E - N + 2

where:

\* M is the cyclometric complexity

\* E is the number of edges in the graph

\* N is the number of nodes in the graph

In this control flow graph, there are 24 edges and 15 nodes. Therefore, the cyclometric complexity is:

M = 26 - 15 + 2 = 13

In a sentiment analysis project, managing cyclomatic complexity is crucial for building efficient and scalable models. Cyclomatic complexity measures how complicated the control flow of your code is, especially in rule-based systems or algorithms with conditional logic. Lower complexity ensures that your sentiment analysis system is easier to maintain, test, and expand as new features or languages are introduced. This results in faster processing, fewer bugs, and better performance, especially when analyzing large datasets in real-time. Simplifying the code structure improves maintainability and helps deliver more reliable sentiment predictions. Overall, controlling cyclomatic complexity leads to a more robust and efficient sentiment analysis project.

**7.RESULT AND ANALYSIS**

**7.1. Experimental Setup**

The experimental setup for this sentiment analysis project involves processing the IMDb movie review dataset, which contains 50,000 reviews labeled as either positive or negative. The dataset is split into training and test sets, with 25,000 reviews each. Preprocessing includes cleaning the text by removing HTML tags and special characters, followed by tokenization and padding to ensure uniform input lengths for models. Multiple machine learning models, including CNN, RNN (LSTM), Random Forest, and BERT, are trained on the dataset. The models are evaluated based on accuracy, loss, and performance metrics like confusion matrices and classification reports, using both traditional and deep learning approaches to compare results.

**7.2. Training and Testing**

**7.2.1. BERT Model**

**Model Accuracy During Training Using BERT:**

**Table2- BERT Training Accuracy and Loss**

|  |  |  |
| --- | --- | --- |
| **Model-BERT** | **Accuracy** | **Loss** |
| 1/3 | 0.8440 | 0.3428 |
| 2/3 | 0.9211 | 0.2019 |
| 3/3 | 0.9620 | 0.1079 |

Accuracy:0.9620

Loss:0.1079

**Interpretation:**

1. Training Loss and Accuracy: The training loss decreases significantly from 0.3428 to 0.1079, and the accuracy increases from 0.8440 to 0.9620 over the three epochs. This indicates that the model is learning and improving its performance on the training data.
2. Validation Loss and Accuracy: The validation loss initially decreases from 0.2724 to 0.2595 but then increases to 0.3789 in the third epoch. The validation accuracy shows a slight improvement from 0.8846 to 0.8946 in the second epoch but drops slightly to 0.8844 in the third epoch. This suggests that the model might be starting to overfit the training data, as indicated by the increase in validation loss and the slight drop in validation accuracy in the final epoch.
3. Overall, the model shows good performance on the training data, but the increase in validation loss in the last epoch indicates a need to monitor for overfitting.

**Model Accuracy During Testing Using BERT:**

**Table3 – BERT Testing Accuracy and loss**

|  |  |  |
| --- | --- | --- |
| **Model-BERT** | **Accuracy** | **Loss** |
| 4/4 | 0.8640 | 0,3804 |

Accuracy:0.8640

Loss:0.3804

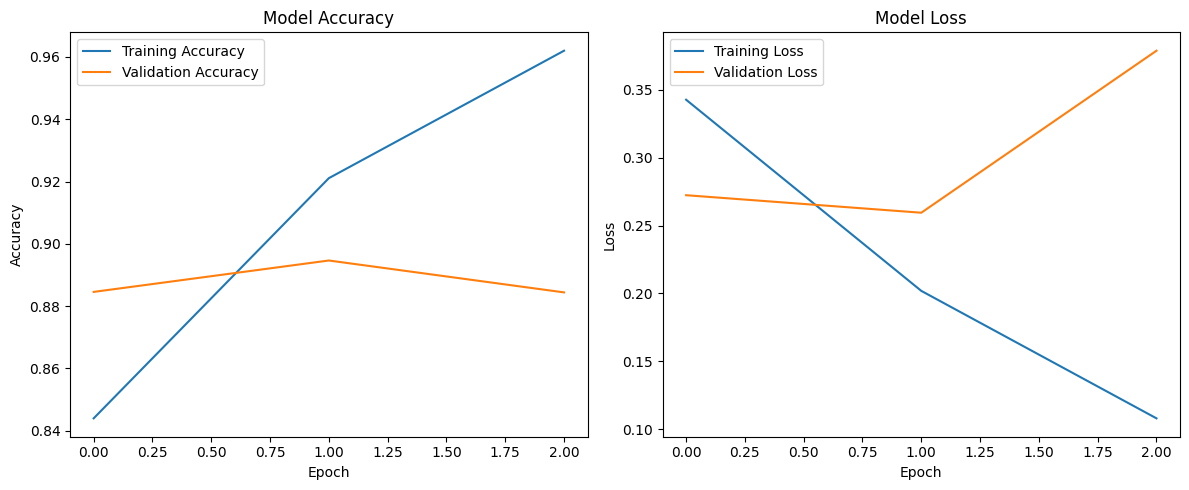
**Interpretation:**

* Test Loss: The loss value of 0.3804 indicates the error rate of the model on the test data. Lower values generally indicate better performance.
* Test Accuracy: The accuracy value of 0.8641 means that the model correctly predicted 86.41% of the test instances.

**Model Loss and Accuracy Graph**

The model performs well on the test data, achieving a high accuracy of 86.41%. The test loss indicates that there is still some room for improvement, but overall, the model generalizes well to unseen data.

Plot Between Train and Validation:



**Fig 7.2.1(a) - Model Accuracy & Model Loss For BERT**

The fig5.2.1(a) consists of two side-by-side line graphs that illustrate the performance of a machine learning model over several epochs.

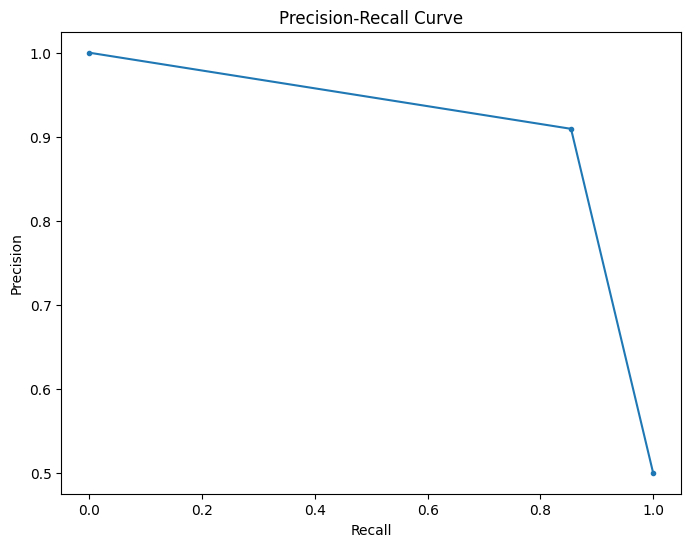
1. **Model Accuracy (Left Graph):**

The training accuracy steadily increases from around 0.85 to nearly 0.95, showing the model's improvement on the training data. Similarly, the validation accuracy rises from approximately 0.88 to 0.95, indicating that the model is generalizing well to unseen data.

1. **Model Loss (Right Graph):**

The training loss consistently decreases from around 0.35 to just above 0.05, indicating the model's error on the training data is reducing. In contrast, the validation loss initially drops but then sharply rises after the first epoch, from around 0.2 to just below 0.4, suggesting potential overfitting, where the model excels on training data but struggles with validation data.

**Precision –Recall Curve**:

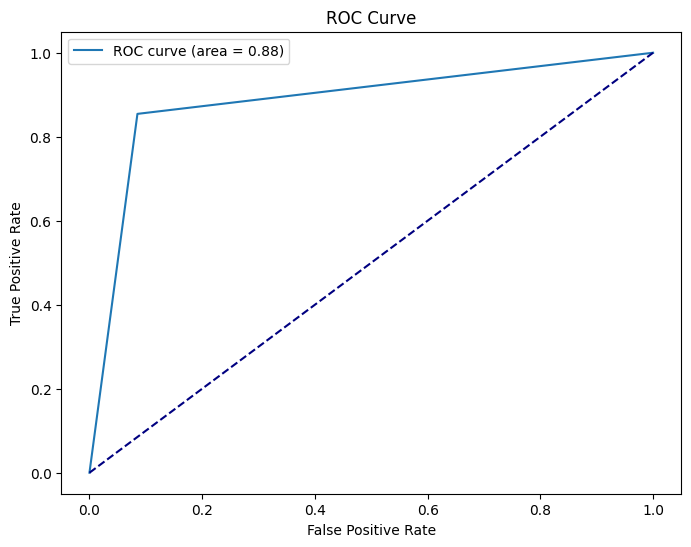


**Fig 7.2.2(b) - Precision-Recall curve for BERT**

The fig5.2.1(b) is a Precision-Recall Curve graph, which is used to evaluate the performance of a binary classifier. Here’s a detailed description:

This graph helps in understanding the trade-off between precision (the proportion of true positive results among all positive results predicted by the classifier) and recall (the proportion of true positive results among all actual positive cases). It’s particularly useful for imbalanced datasets where the positive class is rare.

**ROC Curve:**



**Fig 7.2.3(c) – ROC curve for BERT**

The fig 5.2.1(c) is used to evaluate the performance of a binary classifier. Here’s a detailed description:

This graph helps in understanding how well the binary classification model can distinguish between the positive and negative classes. The closer the ROC curve is to the top left corner, the better the model’s performance.

**Classification Report:**

**Table-4: Classification report**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **F1-score** | **support** |
| **Negative** | 0.83 | 0.92 | 0.87 | 6250 |
| **positive** | 0.91 | 0.81 | 0.86 | 6250 |
| **accuracy** |  |  | 0.86 | 12500 |
| **Macro avg** | 0.87 | 0.86 | 0.86 | 12500 |
| **Weighted avg** | 0.87 | 0.86 | 0.86 | 12500 |

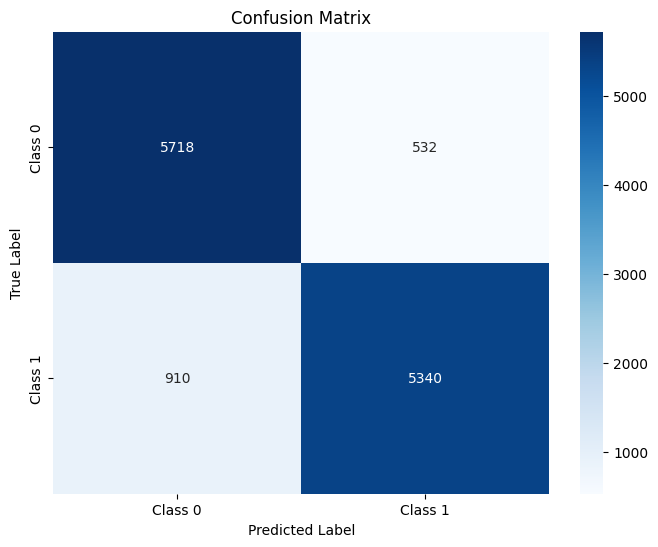
The classification report in table4 provides a detailed breakdown of the performance of a binary classification model.

This report indicates that the model performs well overall, with high precision and recall for both classes, and a balanced F1-score. The accuracy of 0.86 suggests that the model correctly predicts 86% of the instances.

Here’s a description of each metric:

1. Precision: The ratio of correctly predicted positive observations to the total predicted positives.
2. Recall: The ratio of correctly predicted positive observations to all observations in the actual class.
3. F1-Score: The weighted average of Precision and Recall. It considers both false positives and false negatives.
4. Support: The number of actual occurrences of the class in the dataset.
5. Accuracy: The ratio of correctly predicted observations to the total observations.
6. Macro Average: The average of Precision, Recall, and F1-Score, calculated for each class and then averaged.
7. Weighted Average: The average of Precision, Recall, and F1-Score, weighted by the number of true instances for each class.

**Confusion Matrix:**



**Fig 7.2.4(d) – Confusion matrix for BERT**

**True Positives (TP)**: Correctly predicted Class 1 (True Label = 1, Predicted = 1) → 5340

**True Negatives (TN)**: Correctly predicted Class 0 (True Label = 0, Predicted = 0) → 5718

**False Positives (FP)**: Incorrectly predicted as Class 1 (True Label = 0, Predicted = 1) → 532

**False Negatives (FN)**: Incorrectly predicted as Class 0 (True Label = 1, Predicted = 0) → 910

The fig 5.2.1(d) is visualized as a heatmap. This confusion matrix provides a visual representation of the model’s performance, showing how many instances were correctly or incorrectly classified for each class. It helps in identifying specific types of errors made by the classifier, such as false positives and false negatives.

**INPUT 1:**

Review ='''Bahubali is a blockbuster Indian movie that was released in 2015.

It is the first part of a two-part epic saga that tells the story of a legendary hero who fights for his kingdom and his love.

The movie has received rave reviews from critics and audiences alike for its stunning visuals,

spectacular action scenes, and captivating storyline.'''

Get sentiment (Review)

**OUTPUT:**

['positive']

**INPUT 2:**

Review ='' 'The movie was a complete waste of time; the plot was dull and predictable.'''

Get sentiment (Review)

**OUTPUT:**

['Negative']

**7.2.2. CNN & RNN Models**

**MODEL ACCURACY DURING TRAINING AND TESTING USING CNN, RNN:**

**Table5- Training Accuracy and loss for CNN & RNN**

|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy** | **Loss** |
| CNN-1/3 | 0.7144 | 0.5619 |
| CNN-2/3 | 0.8917 | 0.3348 |
| CNN-3/3 | 0.9377 | 0.1839 |
| RNN-1/3 | 0.7484 | 0.5071 |
| RNN-2/3 | 0.8745 | 0.3027 |
| RNN-3/3 | 0.9145 | 0.2251 |

**Table6- Testing Accuracy and loss for CNN & RNN**

|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy** | **Loss** |
| CNN | 0.8442 | 0.4120 |
| RNN | 0.8192 | 0.4541 |

From the above table5 & 6

CNN TRAIN ACCURACY:0.9377

TEST ACCURACY:0.8442

RNN TRAIN ACCURACY:0.9145

TEST ACCURACY:0.8192

Validation Performance: The validation accuracy and loss fluctuate, suggesting varying degrees of generalization. Model 1 shows a slight drop in validation accuracy in the third epoch, while Model 2 shows a more significant drop.

Test Performance: Model 2 has a higher test accuracy (0.8979) and lower test loss (0.2519) compared to Model 1, indicating better generalization to unseen data.

Overall, both models perform well on the training data, but Model 2 demonstrates better generalization on the test.

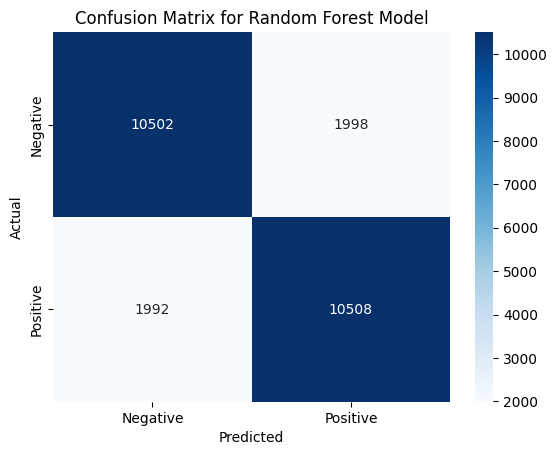
**CLASSIFICATION REPORT:**

**Table7: classification report using RNN & CNN**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **precision** | **recall** | **F1-score** | **support** |
| **0** | 0.84 | 0.84 | 0.84 | 12500 |
| **1** | 0.84 | 0.84 | 0.84 | 12500 |
| **accuracy** |  |  | 0.84 | 25000 |
| **Macro avg** | 0.84 | 0.84 | 0.84 | 25000 |
| **Weighted avg** | 0.84 | 0.84 | 0.84 | 25000 |

The classification report from table2 provides a detailed evaluation of the model’s performance across two classes, labeled as 0 and 1. The report indicates that the model performs consistently across both classes with balanced precision, recall, and F1-scores, achieving an overall accuracy of 84%.

**CONFUSION MATRIX:**

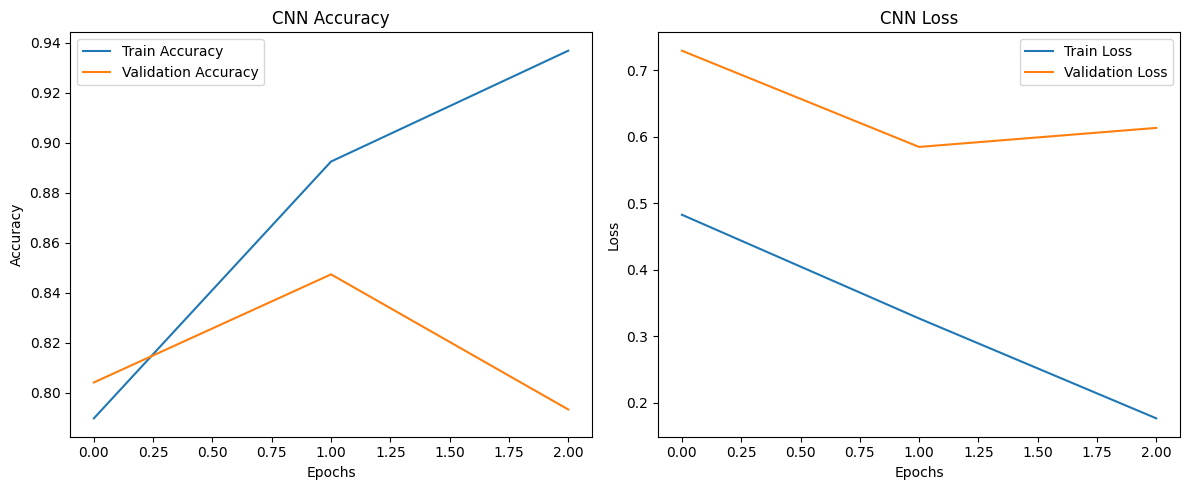


**Fig 5.2.1(a)– Confusion matrix of the random forest model**

The fig5.2.2(a) is a Confusion Matrix for a Random Forest Model, visualized as a heatmap. Here’s a detailed description:

* The heatmap uses shades of blue to represent the frequency of predictions, with darker shades indicating higher values. This confusion matrix provides a visual representation of the model’s performance, showing how many instances were correctly or incorrectly classified for each class. It helps in identifying specific types of errors made by the classifier, such as false positives and false negatives.

**PLOT BETWEEN TRAINING AND VALIDATION FOR CNN:**



**Fig 5.2.2(b) -CNN performance**

The fig5.2.2(b) consists of two sets of side-by-side line graphs, each set representing the performance of a Convolutional Neural Network (CNN) over several epochs during training.

**CNN Performance:**

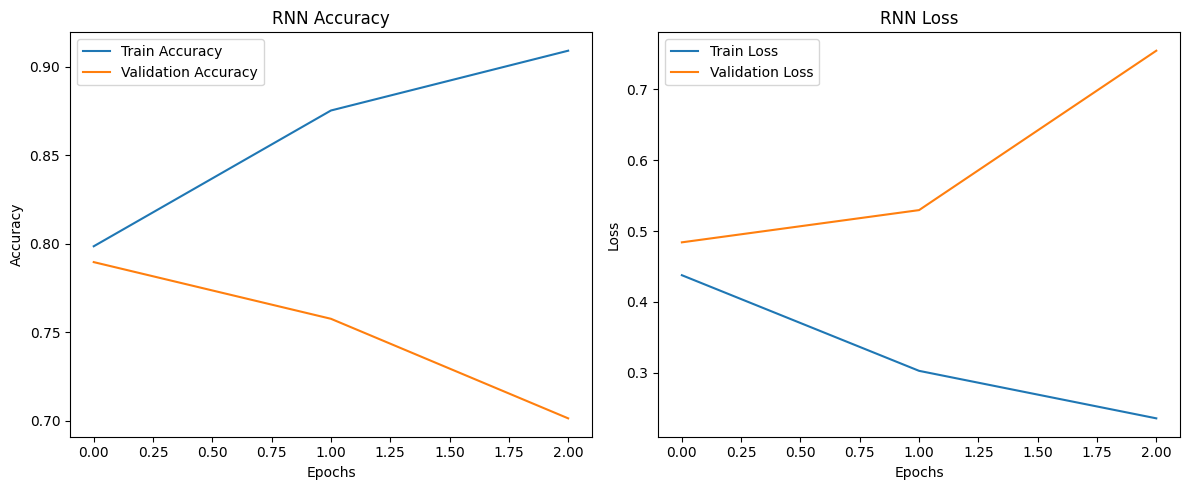
1. **CNN Accuracy (Left Graph):**

* Train Accuracy: This line shows a sharp increase from around 0.8 to just below 0.94, indicating that the CNN model is learning and improving its accuracy on the training data.
* Validation Accuracy: This line starts just above 0.8, peaks around epoch 1 near 0.88, and then slightly drops towards epoch 2, suggesting some fluctuation in the model’s performance on unseen data.

1. **CNN Loss (Right Graph):**

* Train Loss: This line decreases sharply from just below 0.7 to around 0.25, showing that the CNN model is reducing its error on the training data.
* Validation Loss: This line starts just below 0.7, decreases to approximately 0.5 at epoch 1, and then increases slightly towards epoch 2, indicating potential overfitting where the model performs well on training data but less so on validation data.

**PLOT BETWEEN TRAIN AND VALIDATION FOR RNN:**



**Fig 7.2.2(c)– RNN performance**

Fig 5.2.2(c) consists of two sets of side-by-side line graphs, each set representing the performance of a Recurrent Neural Network (RNN) over several epochs during training.

RNN Performance:

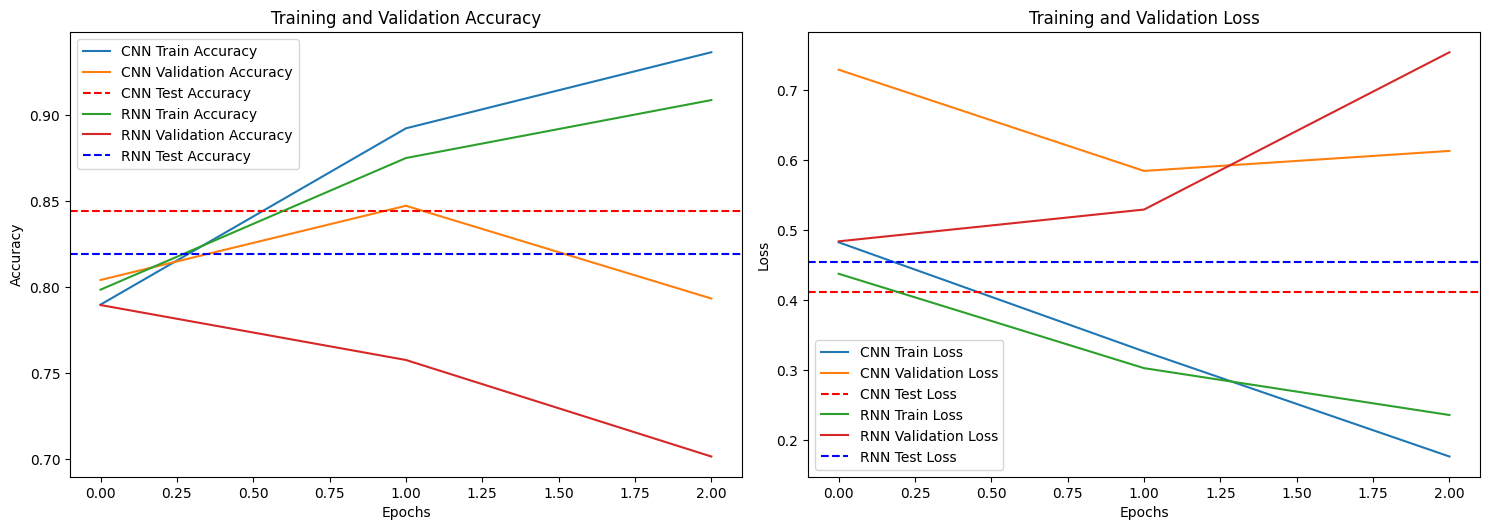
1. **RNN Accuracy (Left Graph):**

* Train Accuracy: This line shows a steady increase, indicating that the RNN model is also learning and improving its accuracy on the training data.
* Validation Accuracy: This line follows a similar trend to the training accuracy, suggesting that the RNN model generalizes well to unseen data.

1. **RNN Loss (Right Graph):**

* Train Loss: This line decreases steadily, showing that the RNN model is reducing its error on the training data.
* Validation Loss: This line also decreases, indicating that the RNN model maintains good performance on validation data without significant overfitting.

**COMBINED PLOT FOR RNN AND CNN:**



**Fig 7.2.2(d)– Combined plot for CNN & RNN**

The fig5.2.2(d) consists of two sets of side-by-side line graphs, each set representing the performance of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) over several epochs during training, validation, and testing phases.

1. **Training and Validation Accuracy (Left Graph):**

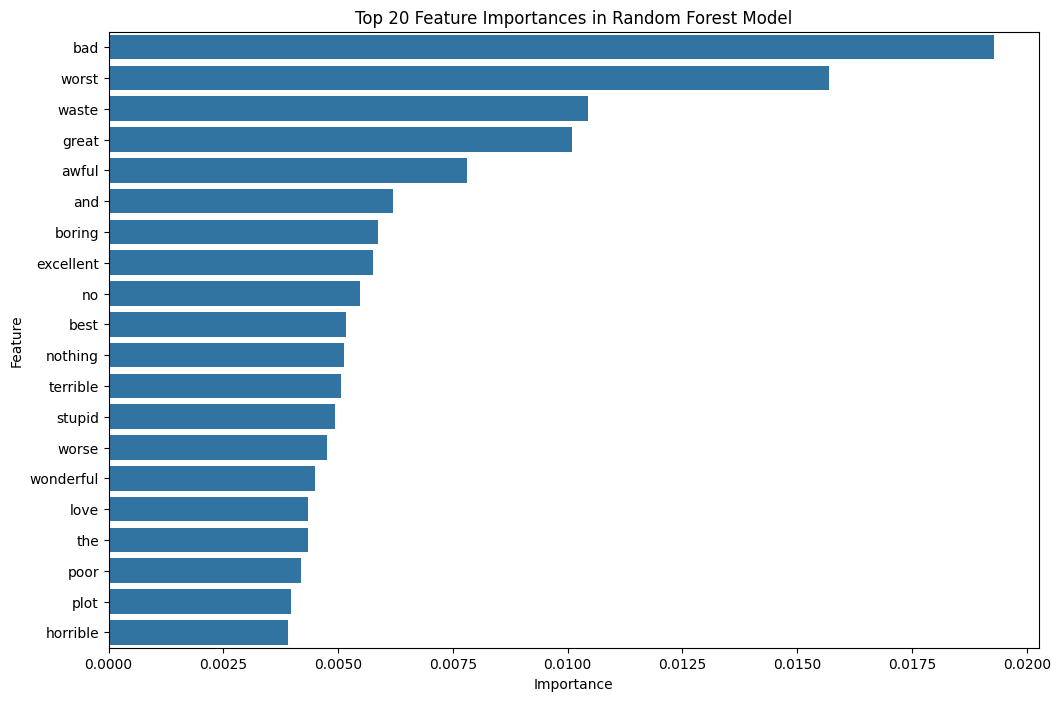
* CNN Train Accuracy: This line shows the accuracy of the CNN model on the training data over epochs.
* CNN Validation Accuracy: This line shows the accuracy of the CNN model on the validation data over epochs.
* CNN Test Accuracy: The dashed horizontal line represents the final test accuracy of the CNN model.
* RNN Train Accuracy: This line shows the accuracy of the RNN model on the training data over epochs.
* RNN Validation Accuracy: This line shows the accuracy of the RNN model on the validation data over epochs.
* RNN Test Accuracy: The dashed horizontal line represents the final test accuracy of the RNN model.

1. **Training and Validation Loss (Right Graph):**

* CNN Train Loss: This line shows the loss of the CNN model on the training data over epochs.
* CNN Validation Loss: This line shows the loss of the CNN model on the validation data over epochs.
* CNN Test Loss: The dashed horizontal line represents the final test loss of the CNN model.
* RNN Train Loss: This line shows the loss of the RNN model on the training data over epochs.
* RNN Validation Loss: This line shows the loss of the RNN model on the validation data over epochs.
* RNN Test Loss: The dashed horizontal line represents the final test loss of the RNN model.

Each graph provides a visual comparison of the training, validation, and testing performance of CNN and RNN models. The accuracy graphs show how well each model is learning and generalizing, while the loss graphs indicate how well each model is minimizing errors. The legends and titles help in distinguishing between the different metrics and models.

**Feature importances in random forest model:**



**Fig 7.2.3(e)– Feature importance of random forest model**

The fig5.2.2(e) is a horizontal bar chart titled “Top 20 Feature Importances in Random Forest Model.” It visualizes the importance of the top 20 features used by the Random Forest model.

This chart helps in understanding which features (words) have the most predictive power in the Random Forest model. By identifying these key features, you can gain insights into the model’s decision-making process and potentially refine the model for better performance.

The most important feature has an important score just below 0. 0175.The least important feature among the top 20 has an important score slightly above 0.0025.

**5.2.3 comparison of different models**

**TEST ACCURACIES FOR OTHER MODELS:**

**TABLE 8- ACCURACIES FOR OTHER MODELS**

|  |  |  |
| --- | --- | --- |
| **Model name** | **Train accuracy** | **Test accuracy** |
| Random Forest | 1.0000 | 0.4700 |
| Linear SVM | 0.5375 | 0.4900 |
| Polynomial SVM | 0.8812 | 0.5400 |
| RBF SVM | 0.8113 | 0.5200 |
| Gradient Boosting | 0.9137 | 0.5500 |

Table8 discusses the accuracies of different models to find the efficient one.

Random Forest - Train Accuracy: 1.0000, Test Accuracy: 0.4700

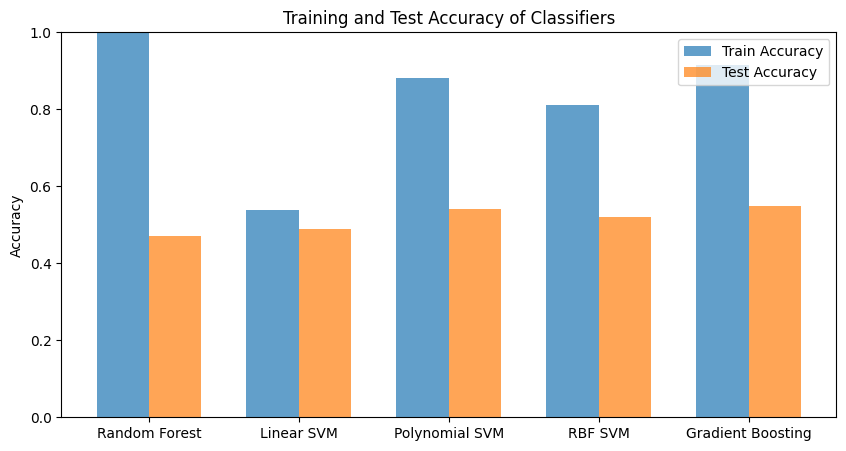
Linear SVM - Train Accuracy: 0.5375, Test Accuracy: 0.4900

Polynomial SVM - Train Accuracy: 0.8812, Test Accuracy: 0.5400

RBF SVM - Train Accuracy: 0.8113, Test Accuracy: 0.5200

Gradient Boosting - Train Accuracy: 0.9137, Test Accuracy: 0.5500

**Training and Test Accuracy classifiers**



**Fig7.2.4(a)– Training and test accuracy of five different models**

The fig25 is a bar chart comparing the training and test accuracy of five different classifiers: Random Forest, Linear SVM, Polynomial SVM, RBF SVM, and Gradient Boosting.

Overfitting: The Random Forest model shows signs of overfitting, with perfect training accuracy but much lower test accuracy.

Generalization: Gradient Boosting has the best generalization performance, with the highest test accuracy.

Model Comparison: The chart visually compares how different models perform on both training and test datasets, highlighting their strengths and weaknesses in terms of accuracy. This visualization helps in understanding the effectiveness of different classifiers and their ability to generalize from training data to unseen test data.

**INPUT 1:**

new\_review = "The movie was fantastic! I really enjoyed it."

predicted\_sentiments = predict\_sentiment(new\_review)

print(predicted\_sentiments)

**OUTPUT:**

{'CNN': 'Positive', 'RNN': 'Positive', 'Random Forest': 'Positive’}

**INPUT 2:**

new\_review = "The movie was worst! I am not enjoyed it."

predicted\_sentiments = predict\_sentiment(new\_review)

print(predicted\_sentiments)

**OUTPUT:**

{'CNN': 'Negative', 'RNN': 'Negative', 'Random Forest': 'Negative'}

**COMPARISON OF BERT, CNN & RNN:**

**Table9: Comparison with 3 different models**

|  |  |  |
| --- | --- | --- |
| ALGORITHM | TRAIN ACCURACY | TEST ACCURACY |
| BERT | 0.8440 TO 0.9620 | 0.8641 |
| CNN | 0.7144 TO 0.9377 | 0.8443 |
| RNN | 0.7484 TO 0.9145 | 0.8193 |

**1.Accuracy:**

* 1. **BERT** outperforms the other models with a higher **test accuracy of 86.4%**, followed by CNN at 84.4% and RNN at 81.9%.
  2. Despite training for fewer epochs, BERT consistently achieves high accuracy, reflecting its robustness in text-based tasks.

1. **Loss:**
   1. **BERT** also has the lowest **test loss of 0.3804**, indicating better generalization and learning, while CNN and RNN have higher losses (0.4121 and 0.4541, respectively).
2. **Training Time:**
   1. **BERT** requires significantly more time per epoch (around 1250 seconds) compared to **CNN** (around 40 seconds) and **RNN** (around 140 seconds). This highlights the trade-off between **BERT's superior performance** and its **computational cost**.
3. **Precision, Recall, and F1-score:**
   1. All models have similar precision, recall, and F1-scores, around 0.84, for both classes. However, BERT slightly outperforms in precision and recall for the **positive class**, making it more reliable in identifying both negative and positive cases.
   2. The BERT model is especially strong in predicting the **negative class** with a **recall of 0.92**, indicating fewer false negatives for this class.

### Website Overview: Sentiment Analyzer

The **Sentiment Analyzer** is a user-friendly web application designed to analyze the sentiment of text input provided by users. The website processes the given text and classifies it into positive, negative, or neutral sentiments. Powered by advanced Natural Language Processing (NLP) models, this tool is ideal for anyone seeking insights into how written content might be perceived emotionally.

### Key Features:

1. **Text Input Field**:
   1. Users can input any text, such as social media posts, reviews, or general statements.
2. **Sentiment Analysis**:
   1. The backend processes the text using a Python-based sentiment analysis model (such as Hugging Face’s transformers or TextBlob).
   2. It evaluates whether the sentiment of the input text is **Positive**, **Negative**, or **Neutral**.
3. **MongoDB Database**:
   1. Every user input and the corresponding sentiment analysis result is stored in a **MongoDB** database.
   2. MongoDB stores the data in a non-relational format, making it easier to manage large datasets and ensure scalability.
4. **Results Display**:
   1. Once the analysis is complete, the website dynamically displays the result (Positive, Negative, or Neutral) along with a brief explanation of how the sentiment was determined.
5. **User-Friendly Interface**:
   1. The website interface is built using **HTML** and **CSS**, providing a clean, modern, and intuitive user experience.
   2. The design focuses on simplicity and ease of use, allowing users to interact with the site seamlessly across devices.

### How It Works:

1. **Input Text**:
   1. Users enter a piece of text in the input field.
2. **Backend Processing**:
   1. The Flask web server processes the input text, sends it to the sentiment analysis model, and retrieves the sentiment (positive, negative, neutral).
3. **Database Storage**:
   1. MongoDB stores the input and result, allowing for the tracking of historical data and potential use cases such as sentiment trends.
4. **Result Display**:
   1. The result is displayed on the webpage, showing the sentiment classification of the input text.

### Technology Stack:

**1.Frontend**:

**HTML5** and **CSS3** to design a responsive and visually appealing user interface.

**2.Backend**:

**Flask** (Python web framework) handles the web server functionality, processes form inputs, and integrates with the NLP model.

**3.Sentiment Analysis Model**:

**Transformers library** or other NLP libraries (e.g., TextBlob, VADER) are used to classify the sentiment of the input text.

**4.Database**:

**MongoDB** (hosted on MongoDB Atlas or a similar service) to store user input and sentiment results in a non-relational format for fast and scalable data retrieval.

### Future Enhancements:

* **User Authentication**: Allow users to sign up, log in, and view their analysis history.
* **Advanced Insights**: Display sentiment trends over time for multiple text inputs.
* **Multi-Language Support**: Expand sentiment analysis capabilities to handle multiple languages.
* **Sentiment Intensity**: Offer more granular feedback on the intensity of the sentiment (e.g., very positive, slightly negative).

### 8.Conclusion:

In conclusion, while BERT stands out as the most accurate model for sentiment analysis, it demands significantly higher computational resources and longer training time. Its superior performance in terms of accuracy, loss, and classification metrics makes it the optimal choice when accuracy is the top priority and sufficient resources are available. For tasks requiring precise sentiment classification, BERT is the best-performing model, particularly in scenarios where the trade-off for computational cost is justified by the need for high-quality results.

However, in cases where resources are limited or faster processing is required, CNN provides a balanced alternative with solid performance and greater computational efficiency. RNN, though generally outperformed by both CNN and BERT, still offers reasonable accuracy but may not be ideal for high-stakes tasks. Ultimately, the choice of model should align with the specific needs of the project, including accuracy requirements, available computational resources, and time constraints, with BERT being the preferred option when high performance is paramount.

**9.REFERENCES**

1. N. J. Prottasha, A. A. Sami, M. Kowsher, S. A. Murad, A. K. Bairagi, M. Masud, and M. Baz, "Transfer Learning for Sentiment Analysis Using BERT Based Supervised Fine-Tuning," 2022,

2. A. Bello, S.-C. Ng, and M.-F. Leung, "A BERT Framework to Sentiment Analysis of Tweets," [www.mdpi.com/journal/sensors], 2023, 23, 503,

3. N. C. Dang, M. N. Moreno-García, and F. De la Prieta, "Sentiment Analysis Based on Deep Learning: A Comparative Study," [www.mdpi.com/journal/electronics], 2020, 9, 483,

4. A. C. M. V. Srinivas, C. Satyanarayana, C. Divakar, and K. P. Sirisha, "Sentiment Analysis using Neural Network and LSTM," [IOP Conference Series: Materials Science and Engineering], 2021, doi:10.1088/1757 899X/1074/1/012007.

5. X. Ouyang, P. Zhou, C. H. Li, and L. Liu, "Sentiment Analysis Using Convolutional Neural Network," in 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, Wuhan, China, 2015, pp. [6]. DOI 10.1109/CIT/IUCC/DASC/PICOM.2015.349.

6. E. Johnstone, V. Ning, T. Oyeniyi, D. Pope, D. Sandberg, R. Tortoriello, and F. Zhao, "Natural Language Processing Literature Survey," Quantamental Research, S&P Global, July 2017.

7. S. Muthukumaran and P. Suresh, "Text Analysis for Product Reviews for Sentiment Analysis using NLP Methods," International Journal of Engineering Trends and Technology (IJETT), vol. 47, no. 8, pp. [pages], May 2017.

8. M. Kanakaraj and R. M. R. Guddeti, "Performance Analysis of Ensemble Methods on Twitter Sentiment Analysis using NLP Techniques," in Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015), [location], 2015, no.2 pp. [pages].