PRODUCT HELPFULNESS DETECTION USING BERT AND LSTM

Project Report Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

B. NIREEKSHAN	**(21501A0512)**
A. HARSHA VARDHAN	**(21501A0503)**
CH. KAVYA SRI	**(22505A0501)**
G. SAI RAMYA SREE	**(21501A0556)**

Under the guidance of

Ms. T. Sri Lakshmi, Assistant Professor

Department of Computer Science and Engineering

Prasad V Potluri Siddhartha Institute of Technology

(Permanently affiliated to JNTU-Kakinada, Approved by AICTE)

(An NBA & NAAC A+ accredited and ISO 9001:2015 certified institute)

Kanuru, Vijayawada-520 007

2024-25

Prasad V Potluri Siddhartha Institute of Technology

(Permanently affiliated to JNTU-Kakinada, Approved by AICTE)

(An NBA & NAAC A+ accredited and ISO 9001:2015 certified institute)

Kanuru, Vijayawada-520 007

CERTIFICATE

This is to certify that the project report entitled "**Product Helpfulness Detection Using BERT and LSTM**" that is being submitted by B. Nireekshan (21501A0512), A. Harsha Vardhan (21501A0503), Ch. Kavya Sri (22505A0501), G. Sai Ramya Sree (21501A0556) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering during the academic year 2024-25.

Signature of the guide
Ms. T. Sri Lakshmi,
Assistant Professor,
Department of CSE,
Prasad V. Potluri Siddhartha Institute
of Technology, Vijayawada.

Signature of the HOD
Dr. A. Jaya Lakshmi,
Professor and HOD,
Department of CSE,
Prasad V. Potluri Siddhartha Institute
of Technology, Vijayawada.

SIGNATURE OF THE EXTERNAL EXAMINER

DECLARATION

We declare that project work entitled “Product Helpfulness Detection Using BERT and LSTM” is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

B. Nireekshan(21501A0512)

A. Harsha Vardhan(21501A0503)

Ch. Kavya Sri(22505A0501)

G. Sai Ramya Sree(21501A0556)

ACKNOWLEDGEMENT

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **Ms. T. Sri Lakshmi, Assistant Professor,** Department of computer science and engineering for her valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr. A. Jayalakshmi, Professor & Head** of the department for extending her cooperation and providing the required resources.

We would like to thank our beloved **Principal, Dr K Sivaji Babu** for providing the online resources and other facilities to carry out this work.

We're grateful to **Dr. A. Ramana Lakshmi, Associate Professor** of our department and our esteemed Project Coordinator for her invaluable guidance and efforts. Her exceptional leadership, from inception to final review, ensured the project's success.

We would like to extend our sincere graduation to all the review panel **Dr. M. V. Ramakrishna, Professor, Dr. A. Ramana Lakshmi, Associate Professor, and Dr. K. Jyothsna Devi, Assistant Professor** for their valuable suggestions in project reviews**.**

We extend our sincere thanks to all other teaching and non-teaching staff of the department, for their cooperation and encouragement.

B. Nireekshan(21501A0512)

A. Harsha Vardhan(21501A0503)

Ch. Kavya Sri(22505A0501)

G. Sai Ramya Sree(21501A0556)

ABSTRACT

In the realm of sentiment analysis for product reviews, determining the helpfulness of reviews is crucial for consumers and businesses alike. This study proposes a novel approach combining LSTM (Long Short-Term Memory) networks and BERT (Bidirectional Encoder Representations from Transformers) embeddings to enhance the accuracy of product helpfulness detection. Unlike traditional methods such as Random Forest (RF), Decision Trees (DT), LightGBM (LGB), and K-Nearest Neighbors (KNN), which are commonly used for sentiment analysis, LSTM and BERT offer advanced capabilities in capturing semantic context and contextual embeddings from textual data. The proposed system focuses on classifying reviews into three categories: positive, negative, and neutral, based on fine food reviews. Key features include leveraging LSTM for sequence modeling and BERT for fine-tuning contextual embeddings. Additionally, class probability features are extracted to provide insights into the confidence levels of predictions. Experimental results demonstrate that the LSTM-BERT hybrid model outperforms traditional methods in accuracy and robustness, making it suitable for real-world applications in e-commerce and customer feedback analysis.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER – 1

INTRODUCTION

1.1 Introduction to Project:

In today's digital era, online shopping has become an integral part of modern consumer behavior. With the rise of e-commerce platforms such as Amazon, Flipkart, and eBay, customers rely heavily on product reviews before making purchasing decisions. These reviews serve as a crucial source of information about product quality, usability, and overall customer satisfaction. However, given the vast number of reviews available for each product, it becomes increasingly difficult for consumers to distinguish between helpful, genuine feedback and misleading, irrelevant, or biased reviews. Many customers struggle to sift through large volumes of user-generated content to identify reviews that provide meaningful insights, making it imperative to develop automated systems that can classify and rank reviews based on their usefulness.

Sentiment analysis has emerged as a powerful tool in addressing this challenge by automatically categorizing reviews into positive, negative, or neutral sentiments. Traditional sentiment analysis methods, such as those based on machine learning algorithms like Random Forest, Decision Trees, LightGBM, and K-Nearest Neighbors, rely on statistical features and basic text representations to analyze sentiment polarity. While these models have demonstrated reasonable performance in simple classification tasks, they fall short in capturing deep contextual meaning, understanding complex sentence structures, and detecting sentiment shifts within long-form reviews. Additionally, conventional approaches struggle with identifying sarcasm, implicit sentiment, and nuanced language variations, which are often present in customer reviews.

To overcome these limitations, this project proposes a novel sentiment analysis framework that leverages state-of-the-art deep learning techniques, specifically Long Short-Term Memory (LSTM) networks and Bidirectional Encoder Representations from Transformers (BERT). LSTM, a variant of Recurrent Neural Networks (RNNs), is particularly effective in handling sequential text data and capturing long-range dependencies, making it suitable for analyzing lengthy and complex reviews. On the other hand, BERT, a transformer-based model, excels in understanding contextual relationships between words by leveraging bidirectional training on large-scale textual corpora. By combining these two approaches, the

proposed system enhances sentiment classification accuracy while also determining the helpfulness of reviews through class probability features.

The hybrid LSTM-BERT model is designed to classify product reviews into positive, negative, and neutral categories while simultaneously evaluating their helpfulness. This is achieved by leveraging the sequence modeling capabilities of LSTM and the contextual embedding power of BERT to extract meaningful features from text data. The system provides improved interpretability by incorporating class probability scores, which help gauge the confidence level of each prediction, ensuring more reliable sentiment classification. By surpassing the performance of traditional models, this approach not only refines sentiment analysis techniques but also enhances the overall e-commerce experience by guiding consumers toward more insightful and relevant reviews.

For practical implementation, the model is integrated into a web-based application using Flask, enabling real-time sentiment analysis and review helpfulness detection. Users can input product reviews and receive immediate feedback on sentiment classification and helpfulness scores. The project utilizes the Amazon Fine Food Reviews dataset, a widely used benchmark for sentiment analysis research, to train and evaluate the model's effectiveness. The performance of the hybrid LSTM-BERT model is assessed through key evaluation metrics such as accuracy, precision, recall, and F1-score, ensuring robust validation against existing sentiment analysis methods.

As artificial intelligence continues to evolve, the demand for intelligent sentiment analysis systems will grow, driving the need for models that go beyond simple polarity classification to assess the credibility and usefulness of textual content. This project represents a significant step toward achieving that goal by integrating advanced deep learning techniques with sentiment analysis to create a more intelligent and reliable review analysis system. By bridging the gap between traditional sentiment analysis and human-like text comprehension, this research lays the groundwork for future innovations in automated review analysis and user experience enhancement in the digital marketplace.

1.2 Motivation:

In today's digital world, online reviews play a vital role in helping consumers make purchasing decisions. With the increasing number of product reviews on e-commerce platforms, identifying helpful reviews has become challenging. Many reviews may be biased or lack useful information, making it harder for customers to trust them.

Traditional sentiment analysis methods like Random Forest and Decision Trees classify reviews as positive, negative, or neutral but struggle to understand deeper meanings and long reviews. To overcome these limitations, this project uses LSTM and BERT models. LSTM captures the sequence of words, while BERT understands the context and meaning of words, improving sentiment classification accuracy.

The system is built as a web-based application using Flask, where users can input reviews and get instant sentiment predictions along with helpfulness scores. The model is trained on the Amazon Fine Food Reviews dataset to ensure reliable performance.

This project helps consumers find helpful reviews quickly and provides businesses with insights into customer feedback, improving their products and services. By using advanced deep learning techniques, this system enhances the accuracy and usefulness of sentiment analysis in e-commerce platforms.

1.3 Statement of the problem and Solution to Problem:

1.3.1 Problem Statement:

Online reviews help consumers make informed decisions, but with so many available, finding useful ones can be difficult. Some reviews are clear and helpful, while others are vague or misleading. Traditional sentiment analysis methods classify reviews as positive, negative, or neutral but often miss deeper meanings. This project improves sentiment analysis using LSTM and BERT, which better understand word context and structure for more accurate predictions. Built as a Flask web application, the system analyzes reviews from the Amazon Fine Food Reviews dataset, providing sentiment scores and helpfulness ratings. This helps customers find valuable reviews quickly while giving businesses insights to improve their products.

1.3.2 Solution:

This project focuses on improving sentiment analysis by using deep learning techniques to better understand product reviews. Online reviews play a crucial role in guiding consumer decisions, but with the large volume of reviews available, it can be difficult to determine which ones are useful. Traditional sentiment analysis methods classify reviews as positive, negative, or neutral but often fail to capture deeper meanings and context. By integrating LSTM networks and BERT embeddings, this system enhances sentiment classification by analyzing both the sequence of words and their contextual relationships. The model also evaluates the confidence levels of predictions, making sentiment analysis more accurate and reliable. Implemented as a Flask web application and trained on the Amazon Fine Food Reviews dataset, the system provides real-time sentiment and helpfulness scores. This helps consumers quickly identify valuable reviews while enabling businesses to gain insights into customer feedback and improve their products and services.

1.4 Objectives:

1.4.1 General Objective:

To develop an advanced sentiment analysis model that leverages LSTM and BERT to enhance the accuracy, efficiency, and interpretability of product review classification while also predicting review helpfulness.

1.4.2 Specific Objectives:

Improve Sentiment Classification Accuracy – By integrating LSTM and BERT, the model will achieve better performance compared to traditional machine learning methods. Enhance Semantic Understanding – Utilize BERT embeddings to capture intricate contextual meanings and improve text comprehension. Model Long-Term Dependencies – Leverage LSTM networks to process sequential textual data effectively. Assess Review Helpfulness – Introduce class probability features to determine the reliability and confidence level of sentiment predictions. Develop a Scalable and User-Friendly Web Application – Implement the model using Flask to create an interactive platform for real-time sentiment analysis.

1.5 Scope of the Work:

This project enhances sentiment analysis for product reviews using deep learning. Online reviews influence buying decisions, but their large volume makes it hard to find helpful ones. Traditional models classify reviews as positive, negative, or neutral but often miss deeper meanings. By combining LSTM for sequence analysis and BERT for contextual understanding, this system improves accuracy. This helps consumers find valuable reviews quickly while enabling businesses to understand customer feedback better.

1.6 Significance of the Work:

The development of a Product Helpfulness Detection System using BERT and LSTM represents a fusion of natural language processing (NLP), deep learning, and sentiment analysis to enhance the reliability of online reviews. By leveraging state-of-the-art AI models, this project contributes to improving the authenticity and usefulness of product reviews, benefiting both consumers and businesses.

The significance of this work extends beyond technological innovation to consumer trust, e-commerce efficiency, and data-driven decision-making:

- Consumer Trust: Helps users make informed purchasing decisions by highlighting genuinely helpful reviews.
- E-commerce Optimization: Enhances product recommendation systems and reduces misleading reviews, leading to better customer satisfaction.
- Business Insights: Provides companies with valuable feedback by identifying meaningful customer opinions, aiding in product improvement and marketing strategies.
- Scalability and Future Applications: The approach can be extended to various domains, such as movie reviews, hotel bookings, and healthcare feedback systems, making it a versatile contribution to the field of AI-driven review analysis.

By addressing the challenge of review credibility, this project contributes to a more transparent and efficient digital marketplace, benefiting consumers, businesses, and researchers in the field of NLP and sentiment analysis.

1.7 Outlines of the Project:

1.7.1 Introduction

- Overview of the project's objectives and significance.
- Brief explanation of BERT and LSTM models and their relevance to product helpfulness detection.

1.7.2 Literature Review

- Review of existing research and technologies related to sentiment analysis, product reviews, and helpfulness prediction.
- Discussion of previous works utilizing BERT, LSTM, or similar NLP techniques for text classification and review analysis.

1.7.3 Methodology

- Dataset: Collection, labeling, and preprocessing of product reviews.
- Model Implementation: Selection, training, and optimization of LSTM and BERT.
- User Interface: Web-based system for real-time sentiment analysis.

1.7.4 System Architecture

- Overview of system architecture: Key components and interactions.
- Model Integration: LSTM and BERT for sentiment classification.
- Backend: Data processing and system communication.

1.7.5 Implementation

- Technical Details: Software tools, frameworks, and libraries.
- Challenges & Solutions: Data preprocessing, model tuning, and deployment.
- Hardware Requirements: System resources for real-time processing.

1.7.6 Results

- Evaluation: Accuracy, precision, recall, and F1-score.
- Performance: Comparison of LSTM-BERT with traditional models.
- Analysis: Improvements over existing sentiment analysis methods.

1.7.7 Conclusion

- Findings: Summary of improvements in sentiment analysis.
- Significance: Impact on consumer decisions and business insights.
- Future Scope: Enhancements like multilingual support.

1.7.8 References

- Citations: Research papers and relevant sources.

CHAPTER - 2
BACKGROUND AND LITERATURE REVIEW

J. Devlin et al. [1] introduced BERT, a revolutionary model that leverages deep bidirectional transformers to achieve state-of-the-art performance in various NLP tasks. The study highlights how BERT's pre-training on large corpora enables it to capture contextual relationships in text, making it highly effective for tasks such as sentiment analysis, question answering, and language inference. This work laid the foundation for many subsequent advancements in NLP.

S. Hochreiter and J. Schmidhuber [2] introduced the LSTM architecture, a type of recurrent neural network (RNN) designed to address the vanishing gradient problem in traditional RNNs. LSTMs have been widely adopted for sequence modeling tasks, including text generation, sentiment analysis, and machine translation, due to their ability to capture long-term dependencies in data.

X. Mengge et al. [3] explored the application of BERT in aspect-based sentiment analysis (ABSA). The study demonstrates how fine-tuning BERT on domain-specific datasets improves its performance in understanding nuanced sentiments and extracting aspect-specific opinions from text. This work underscores the importance of post-training and domain adaptation in enhancing model accuracy.

B. Liu [4] provides a comprehensive overview of sentiment analysis techniques. The book covers various approaches, including lexicon-based methods, machine learning models, and deep learning architectures, offering insights into the challenges and opportunities in analyzing subjective text data.

M. Shoaib et al. [5] discussed the use of NLP techniques to capture sentiment and favorability in text. The study highlights the role of feature extraction, sentiment lexicons, and machine learning algorithms in building robust sentiment analysis systems.

F. Pedregosa et al. [6] introduced scikit-learn, a widely-used Python library for machine learning. The paper outlines the library's capabilities, including its support for various algorithms, data preprocessing tools, and model evaluation techniques, making it an essential resource for researchers and practitioners.

L. Breiman [7] presented the random forest algorithm, an ensemble learning method that combines multiple decision trees to improve predictive accuracy and reduce overfitting. Random forests have been widely used in text classification, sentiment analysis, and other NLP tasks due to their robustness and interpretability.

G. Ke et al. [8] introduced LightGBM, a gradient boosting framework designed for efficiency and scalability. The study demonstrates how LightGBM outperforms traditional gradient boosting methods in terms of speed and memory usage, making it a popular choice for large-scale text classification and ranking tasks.

T. Hastie et al. [9] provided a comprehensive guide to statistical learning methods. The book covers a wide range of topics, including supervised and unsupervised learning, model selection, and evaluation, offering valuable insights for building machine learning models in NLP.

T. Cover and P. Hart [10] introduced the k-nearest neighbors (k-NN) algorithm, a simple yet effective method for classification and regression. The study highlights the algorithm's applicability in text classification and sentiment analysis, particularly in scenarios where interpretability is crucial.

A. Vaswani et al. [11] introduced the transformer architecture, which relies solely on attention mechanisms to model relationships in data. Transformers have become the backbone of many state-of-the-art NLP models, including BERT and GPT, due to their ability to handle long-range dependencies and parallelize training.

A. Radford et al. [12] presented GPT, a generative pre-trained transformer model that achieves remarkable performance in language understanding tasks. The study emphasizes the importance of unsupervised pre-training and fine-tuning in building powerful language models.

J. Howard and S. Ruder [13] introduced ULMFiT, a method for fine-tuning pre-trained language models on specific tasks. The study demonstrates how ULMFiT achieves state-of-the-art results in text classification with minimal task-specific data, highlighting the effectiveness of transfer learning in NLP.

Z. Yang et al. [14] proposed XLNet, a model that combines the strengths of autoregressive and autoencoding approaches. The study shows how XLNet outperforms BERT on various benchmarks by leveraging permutation-based training and bidirectional context.

J. Pennington et al. [15] introduced GloVe, a word embedding technique that captures global statistical information from text corpora. GloVe embeddings have been widely used in NLP tasks, including sentiment analysis and text classification, due to their ability to represent semantic relationships between words.

CHAPTER – 3

SYSTEM ANALYSIS

3.1 Existing System:

Current sentiment analysis systems primarily rely on traditional machine learning algorithms such as Random Forest (RF), Decision Trees (DT), LightGBM (LGB), and K-Nearest Neighbors (KNN) for classifying sentiment in product reviews. These methods, though widely used, have significant limitations in capturing complex contextual meanings and long-range dependencies in textual data.

3.1.1 Limitations of existing system:

Limited Contextual Understanding: Traditional models struggle to capture intricate semantic meanings, making them less effective for sentiment analysis, particularly in lengthy reviews. High Dependency on Feature Engineering: These methods require extensive manual feature extraction, which is time-consuming and may not generalize well across different datasets. Scalability Challenges: Traditional approaches often fail to scale efficiently with large datasets, leading to performance bottlenecks in real-world applications

3.2 Proposed System:

The proposed system integrates Long Short-Term Memory (LSTM) networks and Bidirectional Encoder Representations from Transformers (BERT) embeddings to enhance sentiment analysis accuracy in fine food reviews. LSTM helps in modeling temporal dependencies, while BERT provides rich contextual embeddings, enabling better semantic understanding. This hybrid approach significantly improves classification accuracy and enhances the interpretability of results.

3.2.1 Advantages of the Proposed System:

- Enhanced Semantic Understanding: The LSTM-BERT hybrid model captures deep contextual meanings and sequential dependencies, leading to better sentiment classification.
- Reduced Dependency on Feature Engineering: BERT embeddings eliminate the need for extensive manual feature engineering, making the system more adaptable.

- Improved Accuracy and Interpretability: The model provides superior accuracy in sentiment classification and offers transparent insights into review helpfulness through class probability features.

3.3 Feasibility Study:

A feasibility study evaluates whether the proposed system is viable and practical for implementation. It involves three major considerations: technical feasibility, operational feasibility, and economic feasibility.

3.3.1 Technical Feasibility:

The proposed system utilizes BERT and LSTM models for product helpfulness detection, leveraging advanced natural language processing techniques to analyze customer reviews effectively. The implementation requires a modern computing environment with at least 8GB RAM, a multi-core processor, and an SSD for efficient data processing. The system is developed using Python 3.6+ along with essential machine learning libraries such as TensorFlow, PyTorch, Scikit-learn, and NLTK. The Flask framework facilitates web-based deployment, ensuring accessibility. Designed for scalability, the system efficiently processes large datasets, making it suitable for real-world e-commerce applications.

3.3.2 Operational Feasibility:

The system is designed for ease of use, enabling businesses to analyze product reviews with minimal training. A user-friendly interface ensures intuitive interaction, allowing users to interpret sentiment classification results effectively. The model integrates seamlessly with existing e-commerce platforms and review aggregation systems, ensuring smooth adoption. Additionally, the system follows a modular architecture, making updates and enhancements straightforward, which simplifies long-term maintenance and usability.

3.3.3 Economic Feasibility:

The development of this system is cost-effective due to the use of open-source technologies and frameworks, reducing the need for expensive proprietary solutions. Cloud-based deployment options minimize infrastructure costs while ensuring scalability. From a business perspective, the system provides valuable insights into customer reviews, enhancing product recommendations and decision-making. This leads to improved customer satisfaction and increased revenue, ensuring a high return on investment (ROI).

CHAPTER - 4

SOFTWARE REQUIREMENT SPECIFICATION

4.1 Functional Requirements:

1. **Operating System:** Windows 7 or above.
2. **Processor:** Any processor above 500 MHz.
3. **RAM:** 8 GB or higher for smooth execution.
4. **Storage:** Minimum 4 GB of free disk space.
5. **Deep Learning Framework:** A framework such as TensorFlow or PyTorch is required to train and implement LSTM and BERT models for sentiment analysis. These frameworks provide APIs for building, optimizing, and running deep learning models.
6. **NLP Model Implementation:** The chosen framework should support BERT embeddings and LSTM networks, either using pre-trained models or fine-tuning based on existing research and datasets.
7. **Text Processing Libraries:** Libraries like NLTK, spaCy, and Hugging Face's Transformers are essential for preprocessing text, tokenization, stop-word removal, stemming, and lemmatization.
8. **Dataset Handling Tools:** Tools such as Pandas and NumPy are needed for managing and preprocessing the dataset, ensuring clean and structured input for training.
9. **Training Infrastructure:** Sufficient computational resources, including access to GPUs or TPUs, are required for training deep learning models efficiently. Cloud platforms such as Google Colab or Kaggle Notebooks can be used for model training.
10. **Model Evaluation Tools:** Tools for assessing model performance using metrics like accuracy, precision, recall, and F1-score. Scripts for visualizing performance trends and error analysis are necessary.

11. **Deployment Platforms:** The trained model should be deployable using Flask or FastAPI for real-time sentiment classification and review helpfulness detection. The API should enable easy integration with other applications.
12. **User Interface Development:** A web-based user interface using HTML, CSS, JavaScript, and Flask for interacting with the sentiment analysis system. For a desktop version, PyQt or Tkinter can be used.
13. **Version Control and Collaboration:** Git and platforms like GitHub or GitLab are required for managing code repositories, tracking changes, and enabling teamwork throughout the project development.

This ensures a well-structured and functional system for sentiment analysis using deep learning.

4.2 Non-Functional Requirements:

1. **Speed**: The system should process sentiment analysis in real-time or near-real-time to ensure quick and efficient review classification.
2. **Accuracy**: The sentiment classification model should meet predefined accuracy thresholds to minimize misclassifications.
3. **Precision**: The proportion of correctly predicted positive sentiments to the total number of sentiments classified as positive.
4. **Sensitivity**: The system's ability to correctly identify actual positive sentiments from the entire set of positive instances.
5. **Specificity**: The system's capability to correctly classify negative and neutral sentiments while minimizing false positives.
6. **Scalability**: The system should handle increasing data loads and users without significant performance loss.
7. **Availability**: The sentiment analysis platform should remain operational with minimal downtime or disruptions.
8. **Fault Tolerance**: The system should be resilient to failures, ensuring continued functionality despite errors.
9. **Robustness**: The system should perform reliably across different types of reviews, varying text lengths, and writing styles.
10. **User Interface**: The UI should be simple, user-friendly, and accessible, allowing smooth interaction with the sentiment analysis system.

11. **Modularity**: The system should have a modular structure for easy maintenance, updates, and future improvements.
12. **Documentation**: Detailed documentation should be provided, including system architecture, APIs, deployment instructions, and usage guidelines.

CHAPTER – 5

DESIGN AND METHODOLOGY OF PROPOSED SYSTEM

5.1 System architecture or Model:

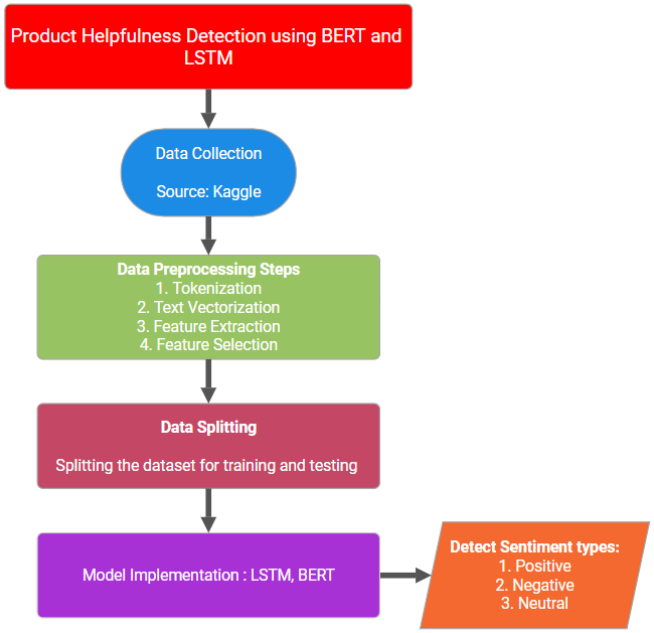

Figure 5.1.1 -Architecture

Figure 5.1.1 - The system architecture for Product Helpfulness Detection using BERT and LSTM outlines a structured approach to analyzing customer reviews, ensuring efficient data processing, machine learning model implementation, and sentiment detection. This framework enables businesses to gain insights into product reviews using state-of-the-art NLP techniques.

1. Data Collection

The process begins with collecting datasets from reliable sources, such as Kaggle. The dataset typically consists of customer reviews, ratings, and other textual data that serve as input for sentiment analysis.

2. Data Preprocessing Layer

Before feeding the data into the models, the system performs multiple preprocessing steps to clean and structure the textual data:

- **Tokenization:** Splitting text into individual words or tokens.

- **Text Vectorization:** Converting text into numerical representations using methods like word embeddings.
- **Feature Extraction:** Identifying key linguistic and contextual features that influence sentiment analysis.
- **Feature Selection:** Choosing the most relevant features to improve model performance and reduce computational overhead.

This layer ensures that the input data is structured and optimized for sentiment analysis models.

3. Data Splitting

Once preprocessing is complete, the dataset is split into training and testing sets. This ensures that the models can learn from historical data and generalize well on unseen reviews. Typically, an 80-20 or 70-30 split is used, where the majority of the data is used for training, and the remaining for evaluation.

4. Model Implementation

The system leverages two powerful deep learning models, LSTM and BERT, to classify product reviews based on their helpfulness.

- LSTM Model: A Recurrent Neural Network (RNN)-based approach that captures the sequential dependencies in textual data to predict sentiment effectively.
- BERT Model: A transformer-based deep learning model that leverages contextual word embeddings to provide highly accurate sentiment classification.

These models work together to ensure that product helpfulness is detected with high precision and recall.

5. Sentiment Detection Layer

Once the models process the reviews, the system classifies sentiments into three categories:

1. **Positive** – Indicates that the review is helpful and expresses satisfaction.
2. **Negative** – Highlights dissatisfaction, signaling a lack of product helpfulness.
3. **Neutral** – Represents mixed or unclear sentiments, requiring further analysis.

The output provides businesses with actionable insights, enabling better product recommendations, customer feedback analysis, and decision-making.

5.2 System Design:

1. Data Collection and Preprocessing:

Data collection and preprocessing are essential steps in building a sentiment analysis system. These steps involve gathering a large dataset of product reviews, cleaning the text data, and preparing it for training deep learning models. The dataset consists of product reviews collected from e-commerce platforms, ensuring a diverse range of sentiments and review lengths. Each review is labeled with its corresponding sentiment (positive, negative, or neutral) and helpfulness score. The dataset should be sufficiently large and diverse to cover different writing styles and linguistic variations, improving the model's generalization ability.

2. Data Annotation:

Sentiment labels define whether a review expresses a positive, negative, or neutral opinion. Additionally, reviews are assessed for their helpfulness, which can be determined using existing metadata such as user votes on review usefulness. Annotation tools, such as Hugging Face Datasets and NLTK, help standardize the labeling process. The final dataset is formatted to be compatible with TensorFlow and PyTorch for seamless integration with deep learning frameworks.

3. Data Preprocessing:

Text preprocessing involves several steps to clean and standardize the data. Tokenization splits text into words or subwords, ensuring consistency in model input. Stop-word removal eliminates common but unimportant words to focus on meaningful content. Lemmatization and stemming convert words to their base forms, reducing vocabulary complexity. Normalization ensures uniform text formatting, including lowercasing and punctuation removal. Additionally, word embeddings such as BERT tokenization are applied to convert text into numerical representations suitable for deep learning models. The dataset is then split into training, validation, and test sets to evaluate model performance accurately.

4. Modelling:

The sentiment analysis system utilizes a hybrid deep learning approach, combining LSTM and BERT for improved accuracy. LSTM processes sequential data, capturing dependencies between words, while BERT provides contextual embeddings to understand word meaning in different contexts. The model architecture includes an embedding layer, bidirectional LSTM, attention mechanism, and fully connected layers for classification.Pre-trained BERT models are fine-tuned on the sentiment dataset to improve performance. Hyperparameter tuning is performed by adjusting learning rate, batch size, and dropout rates to optimize accuracy and prevent overfitting. The trained model is evaluated using accuracy, precision, recall, and F1-score to measure its effectiveness.

5. Validation and Evaluation:

The model's performance is validated using a separate dataset to assess its generalization ability. Evaluation metrics such as confusion matrix analysis, ROC curves, and precision-recall scores are used to analyze strengths and weaknesses. Comparative analysis with traditional machine learning methods like Random Forest and SVM highlights the improvements achieved with deep learning techniques. This structured approach ensures the development of an efficient and accurate sentiment analysis system that can classify product reviews effectively while providing insights into review helpfulness.

5.2.1 UML Diagrams:

- **Flow Diagram:**

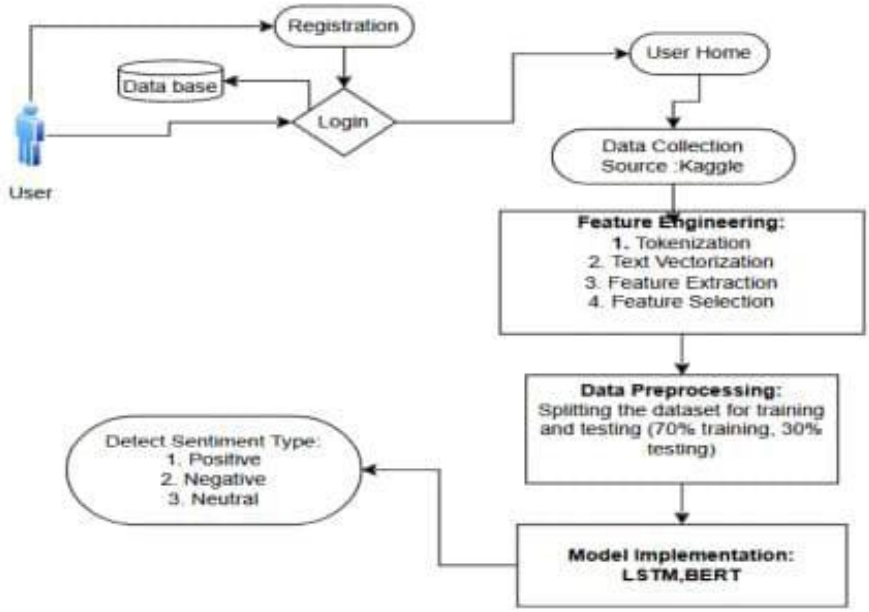

Figure 5.2.1 – System Flow Diagram

Figure 5.2.1 - The system flow diagram for Product Helpfulness Detection using BERT and LSTM illustrates the step-by-step process from user interaction to sentiment classification.

1. User Interaction

Users begin by registering on the platform, and their credentials are stored in the database. Registered users authenticate through the login process and are redirected to the user home interface, where they can proceed with data collection and analysis.

2. Data Collection

The system collects product review datasets from Kaggle or other sources. These datasets contain customer reviews, ratings, and feedback, which serve as input for sentiment analysis.

3. Feature Engineering

Raw text data undergoes preprocessing steps such as tokenization, text vectorization, feature extraction, and feature selection. These steps convert unstructured text into structured numerical representations, making it suitable for deep learning models.

4. Data Preprocessing

The dataset is split into training (70%) and testing (30%) subsets. The training data helps the models learn sentiment patterns, while the testing data evaluates model performance and accuracy.

5. Model Implementation

The system employs LSTM and BERT models for sentiment classification. LSTM captures sequential dependencies in text, making it effective for sentiment prediction, while BERT provides contextual understanding for high-precision classification.

6. Sentiment Detection

The trained models classify reviews into three sentiment types: positive, negative, and neutral. The classification results help businesses analyze customer feedback, improve recommendations, and enhance decision-making.

- **Use Case Diagram:**

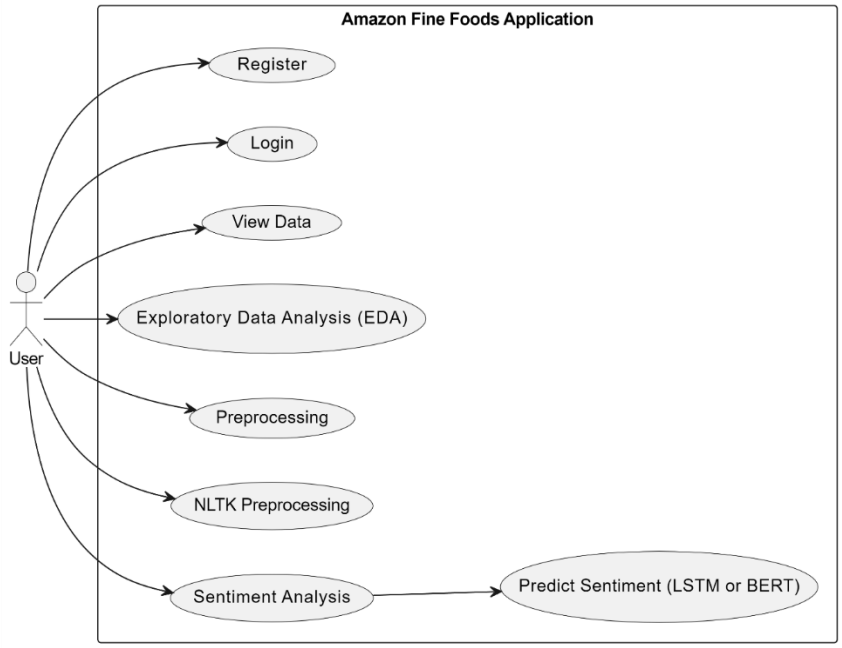

Figure 5.2.2 – Use Case Diagram

Figure 5.2.2 - The Amazon Fine Foods Application use case diagram illustrates the various interactions a user has with the system, from registration to sentiment prediction using deep learning models.

1. User Authentication

Users begin by registering on the platform. Once registered, they can log in to access functionalities like data viewing and sentiment analysis.

2. Data Viewing and Exploration

Users can view the dataset, which contains customer reviews of fine food products from Amazon. Additionally, they can perform Exploratory Data Analysis (EDA) to understand data distributions, trends, and patterns.

3. Data Preprocessing

Before applying sentiment analysis, the system processes raw text data through Preprocessing and NLTK-based text cleaning techniques. This ensures that irrelevant elements like stopwords and punctuation are removed, and text is properly tokenized for model training.

4. Sentiment Analysis and Prediction

After preprocessing, the system applies sentiment analysis to classify reviews as positive, negative, or neutral. Users can select between LSTM or BERT models for predicting sentiment, ensuring both sequential and contextual understanding of text data.

- **Class Diagram:**

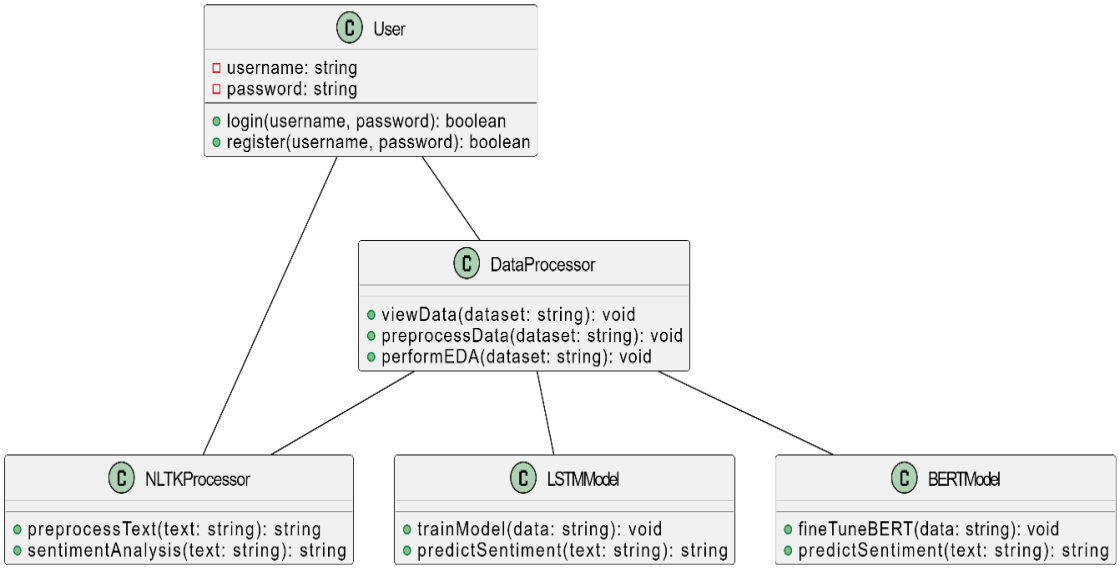

Figure 5.2.3 – Class Diagram

Figure 5.2.3 - The class diagram represents the structure of a sentiment analysis system using different models like NLTK, LSTM, and BERT for text processing and prediction. It defines the relationships between different classes and their respective functionalities.

1. User Authentication Layer

- The User class contains attributes username and password for authentication.
- It provides methods:
    - login(username, password): boolean – Authenticates the user.
    - register(username, password): boolean – Registers a new user in the system.

2. Data Processing Layer

- The DataProcessor class is responsible for handling dataset operations and provides the following methods:
    - viewData(dataset: string): void – Displays dataset contents.
    - preprocessData(dataset: string): void – Cleans and prepares data for analysis.
    - performEDA(dataset: string): void – Conducts Exploratory Data Analysis (EDA) to understand data patterns.

3. Sentiment Analysis Models

Three different classes implement sentiment analysis using distinct approaches:

- NLTKProcessor (Traditional NLP Processing)
    - preprocessText(text: string): string – Tokenizes and cleans text using NLTK.
    - sentimentAnalysis(text: string): string – Performs basic sentiment classification.
- LSTMModel (Deep Learning Approach)
    - trainModel(data: string): void – Trains an LSTM model on provided data.
    - predictSentiment(text: string): string – Predicts sentiment using the trained LSTM model.
- BERTModel (Transformer-Based Approach)
    - fineTuneBERT(data: string): void – Fine-tunes a BERT model using labeled data.
    - predictSentiment(text: string): string – Uses BERT embeddings for sentiment classification.

- **Sequence Diagram:**

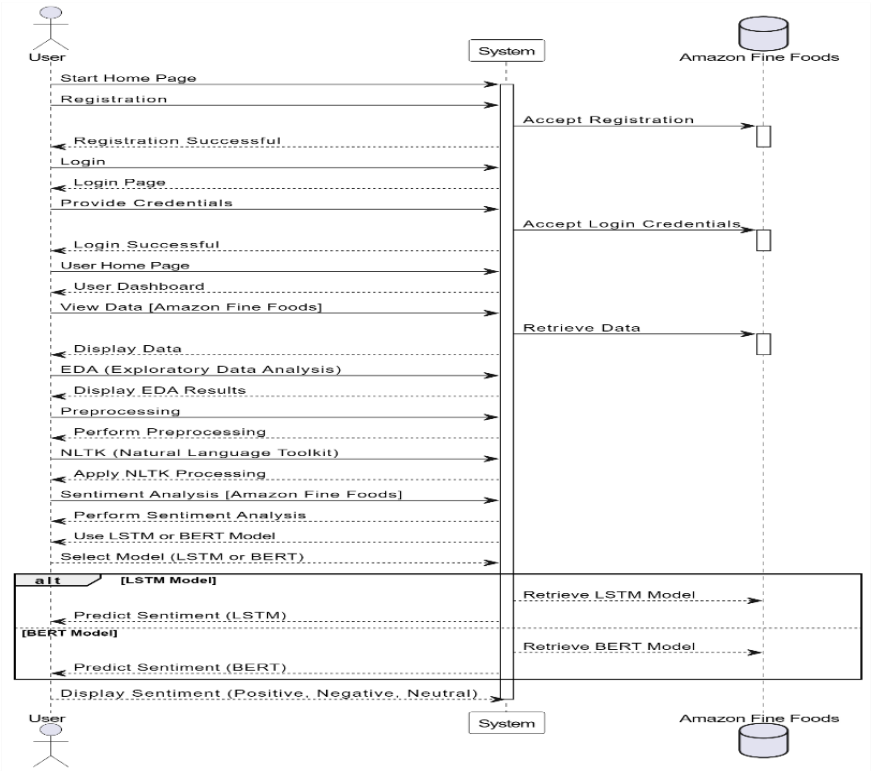

Figure 5.2.4 – Sequence Diagram

Figure 5.2.4 - The sequence diagram illustrates the interaction between the User, System, and Amazon Fine Foods database for performing sentiment analysis using LSTM or BERT models. It details the flow of actions from user registration to sentiment prediction.

1. User Authentication and Data Access

- The User initiates the Home Page.
- The User performs Registration, which the System accepts and stores in the Amazon Fine Foods database.
- The User logs in by providing credentials, which the System verifies by checking stored credentials in the Amazon Fine Foods database.
- Once authentication is successful, the User is redirected to the User Home Page and Dashboard.

2. Data Retrieval and Preprocessing

- The User requests to view data (Amazon Fine Foods dataset).
- The System retrieves and displays the dataset.
- Exploratory Data Analysis (EDA) is performed, and results are displayed.
- The User requests Preprocessing, which involves text cleaning and transformation.
- NLTK Processing is applied for tokenization and stop-word removal.

3. Sentiment Analysis

- The User selects the sentiment analysis model (either LSTM or BERT).
- If LSTM is chosen, the System retrieves the LSTM Model from the Amazon Fine Foods database and performs sentiment analysis.
- If BERT is chosen, the System retrieves the BERT Model from the Amazon Fine Foods database and applies sentiment analysis.
- The predicted sentiment (Positive, Negative, Neutral) is displayed to the User.

5.3 Methodology

5.3.1 Data Collection Methodology:

The dataset for this project is sourced from publicly available e-commerce review datasets, primarily the Amazon Fine Food Reviews dataset. The dataset consists of text reviews, star ratings, timestamps, and helpfulness scores (based on user votes). To ensure the reliability of the model, data collection follows specific criteria, such as selecting reviews with a clear sentiment orientation and a sufficient number of helpfulness votes.

The collected data undergoes preprocessing to remove inconsistencies and noise. This includes the removal of special characters, HTML tags, stop words, and redundant spaces. Tokenization is applied to split the text into meaningful words or subwords. Lemmatization and stemming standardize words to their base form, reducing vocabulary complexity. Reviews with excessive repetition or unclear sentiment labels are excluded to enhance dataset quality. The preprocessed data is then split into training (80%), validation (10%), and testing (10%) sets to ensure robust model evaluation and prevent overfitting.

5.3.2 Process Model/Methodology/Algorithms:

The project follows the Waterfall Model, ensuring a structured, step-by-step approach to system development. Each phase—requirement gathering, design, implementation, testing, and deployment—is completed sequentially, ensuring clarity and systematic progress.

For sentiment analysis, the project utilizes a hybrid deep learning approach that combines LSTM and BERT embeddings. LSTM is used to process sequential dependencies in textual data, enabling the model to recognize long-range relationships between words. BERT, a transformer-based model, enhances sentiment classification by understanding word meanings in different contexts through bidirectional training. The hybrid model structure includes:

- Embedding Layer: Converts words into numerical representations using BERT embeddings.
- Bidirectional LSTM Layer: Captures sequential dependencies and sentiment patterns.
- Fully Connected Layers: Processes learned features for classification.
- Softmax Output Layer: Predicts sentiment as positive, negative, or neutral.

The model is trained using categorical cross-entropy loss, optimized with the Adam optimizer, and evaluated using key performance metrics such as accuracy, precision, recall, and F1-score. Hyperparameter tuning is performed by adjusting batch size, learning rate, dropout rates, and hidden layer configurations to achieve optimal performance.

5.3.3 Tools and Techniques:

The system is developed using Python and utilizes deep learning frameworks such as TensorFlow and PyTorch for building and training the model. Text processing is handled using NLTK, spaCy, and Hugging Face Transformers, which assist in text cleaning, tokenization, and feature extraction. Pandas and NumPy are used for efficient data handling and preprocessing, while Matplotlib and Seaborn enable visualization of sentiment distribution and model performance metrics.

The model is deployed as a Flask-based web application, allowing users to input reviews and receive real-time sentiment and helpfulness predictions. The training process is performed on Google Colab and Jupyter Notebook, leveraging GPU acceleration for faster computations. The entire development process is managed using Git and GitHub, ensuring version control, collaboration, and seamless updates. The project follows best practices for modular coding, documentation, and reproducibility, making it easy to maintain and extend in the future.

CHAPTER – 6

IMPLEMENTATION

6.1. Modules of the System

Data Preprocessing Module

Functionality:

This module is responsible for cleaning and preparing textual data before feeding it into the sentiment analysis model. The key tasks include:

- Tokenization: Splitting text into individual words.
- Stop word Removal: Removing commonly used words that do not impact sentiment.
- Stemming/Lemmatization: Reducing words to their base form for consistency.
- Vectorization: Converting textual data into numerical features using TF-IDF, Hashing Vectorizer, or BERT embeddings.

Implementation Explanation:

The NLTK (Natural Language Toolkit) library is used for tokenization and stopword removal, while a Hashing Vectorizer transforms text into numerical format before passing it to the machine learning model.

Machine Learning Model Module

Functionality:

This module manages training, saving, and deploying sentiment analysis models. The system uses:

- LSTM (Long Short-Term Memory): Captures context from sequential text.
- Random Forest: Used for benchmarking traditional machine learning models.

The key steps in this module include:

- Training the Model: The system fine-tunes an LSTM model using BERT embeddings for better contextual representation.
- Saving the Model: Once trained, the model is saved as a .sav file using Pickle for future predictions.
- Loading and Predicting: When a user enters a review, the system loads the saved model and predicts sentiment.

Implementation Explanation:

A pre-trained LSTM model is loaded from a saved file, and user-input text is vectorized before passing through the model. The model predicts one of three sentiment classes: Positive, Negative, or Neutral.

User Authentication Module

Functionality:

This module handles user registration and login authentication. It ensures:

- Secure Registration: Users must provide a username, email, password, age, and contact details.
- Unique Email Validation: The system prevents duplicate email registrations.
- Login Authentication: Credentials are verified, and a session is created for the user.

Implementation Explanation:

The system uses Flask-SQLAlchemy ORM to manage user authentication with a MySQL database. Upon login, session variables store user details, allowing access to personalized features like viewing previous sentiment predictions.

Database Management Module

Functionality:

This module ensures the structured storage and retrieval of user-related data. The key features include:

- Storing User Information: The MySQL database maintains user credentials and session details.
- Managing Sessions: User data is retrieved from the database during login and stored temporarily in session variables.
- Querying Data: Fetches sentiment predictions and review datasets for display.

Implementation Explanation:

Using Flask-SQLAlchemy, the system defines a User table schema, where fields like username, email, and password are securely stored. Queries are performed using ORM methods to retrieve user data efficiently.

Sentiment Prediction Module

Functionality:

This module processes user-input text and classifies it into a sentiment category using the trained model. The workflow includes:

1. The user enters a review.
2. The system preprocesses the text using NLP techniques.
3. The model classifies the review as Positive, Negative, or Neutral.
4. The sentiment label is displayed on the web interface.

Implementation Explanation:

A pre-trained LSTM model is loaded from a saved file. The user's input is converted into numerical form using a Hashing Vectorizer, and the LSTM model predicts sentiment based on learned features.

Product Review Data Display Module

Functionality:

This module allows users to view a dataset of Amazon Fine Food Reviews. The key tasks include:

- Loading Dataset: The system reads a CSV file containing product reviews.
- Displaying Data: The first 100 reviews are extracted and shown in a tabular format on the web page.

Implementation Explanation:

The dataset is read using Pandas, and the data is passed to an HTML template for rendering in a structured table format.

Exploratory Data Analysis (EDA) Module

Functionality:

This module provides data visualization and statistical insights from the dataset. Users can explore:

- Sentiment distribution (Positive, Negative, Neutral).
- Commonly used words in reviews.
- Trends in review sentiments over time.

Implementation Explanation:

A EDA report is pre-generated and converted into an HTML page using nbconvert. This allows users to view interactive graphs and insights directly within the web application.

6.2 Description of Sample code of Each Module

6.2.1 Data Preprocessing and Preparation (Before Training)

Import Required Libraries

```
import numpy as np
import pandas as pd
import os
```

- numpy: Used for numerical computations.
- pandas: Used for data manipulation and analysis.
- os: Used for handling file paths.

Checking for Available Files in Kaggle Input Directory

```
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

- This lists all the files available in the /kaggle/input/ directory

Importing Necessary Libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import ktrain, time
from ktrain import text
import matplotlib.pyplot as plt
import seaborn as sns
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

- pandas: Handles data.
- train_test_split: Splits data into training and testing sets.
- LabelEncoder: Converts text labels into numbers.
- ktrain and text: Used for BERT text classification.
- matplotlib and seaborn: For visualization.
- SentimentIntensityAnalyzer: Computes sentiment scores.
- accuracy_score, classification_report, confusion_matrix: Evaluate model performance

Load and View Data

```
df = pd.read_csv(r'/kaggle/input/ak-reviews/Reviews.csv').head(5000)
```

- Loads the dataset from Reviews.csv (first 5000 rows).

Perform Sentiment Analysis Using VaderSentiment

```
sid = SentimentIntensityAnalyzer()
df['sentiment_compound_polarity'] = df['Text'].apply(lambda x:
sid.polarity_scores(x)['compound'])
df['sentiment_type'] = df['sentiment_compound_polarity'].apply(lambda x: 'POSITIVE' if x >
0 else ('NEGATIVE' if x < 0 else 'NEUTRAL'))
```

- Computes sentiment scores for each text review.
- Assigns sentiment labels:
    - POSITIVE if compound score > 0
    - NEGATIVE if compound score < 0
    - NEUTRAL otherwise.

Text Cleaning Function

```
def text_clean(text):
    text = text.lower().replace("&#039;", "").replace(r'[^\w\d\s]', ' ')
    text = text.replace(r'[^\x00-\x7F]+', ' ').replace(r'^\s+|\s+?$', '')
    text = text.replace(r'\s+', ' ').replace(r'\.{2,}', ' ')
    return text
```

- Converts text to lowercase.
- Removes special characters and extra spaces.

Apply Text Cleaning

```
df['text_clean'] = df['Text'].apply(text_clean)
```

- Applies the cleaning function to the dataset.

Filter Out Neutral Sentiments

```
df = df[df['sentiment_type'] != 'NEUTRAL']
```

- Removes neutral sentiment rows (only keeping positive and negative sentiments).

Encode Sentiment Labels

```
le = LabelEncoder()
df['sentiment_type'] = le.fit_transform(df['sentiment_type'])
class_names = le.classes_.tolist()
```

- Converts "POSITIVE" and "NEGATIVE" into numerical labels (e.g., 0 and 1).
- Stores class names for later use.

Split Data into Training and Testing Sets

```
x_train, x_test, y_train, y_test = train_test_split(df['text_clean'], df['sentiment_type'],
stratify=df['sentiment_type'], test_size=0.3, random_state=42)
```

- Splits data into 70% training and 30% testing.
- Stratify ensures equal distribution of sentiment classes.

Convert Text Data for BERT Processing

```
(x_train, y_train), (x_test, y_test), preproc = text.texts_from_array(
    x_train=x_train.tolist(), y_train=y_train.tolist(),
    x_test=x_test.tolist(), y_test=y_test.tolist(),
    class_names=class_names,
    preprocess_mode='bert',
    maxlen=128
)
```

- Prepares text data for BERT-based sentiment classification.
- Max length of each text is 128 tokens.

6.2.2 Model Training and Evaluation (After Training)

Create and Compile BERT Model

model = text.text_classifier('bert', train_data=(x_train, y_train), preproc=preproc)

- Creates a BERT-based classifier.

Create a Training Object

learner = ktrain.get_learner(model, train_data=(x_train, y_train), val_data=(x_test, y_test), batch_size=6)

- Initializes model training with a batch size of 6.

Train the Model

learner.fit_onecycle(2e-5, 2)

- Trains the model for 2 epochs with a learning rate of 2e-5.

Validate the Model

learner.validate(val_data=(x_test, y_test), class_names=class_names)

- Evaluates the model on the test dataset.

Plot Confusion Matrix

```
cm = np.array([[77, 78], [26, 1298]])
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False,
        xticklabels=["Predicted No", "Predicted Yes"],
        yticklabels=["Actual No", "Actual Yes"])
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```

- Plots a confusion matrix to visualize classification results.

Create a Predictor

predictor = ktrain.get_predictor(learner.model, preproc)

- Creates a predictor object for making predictions.

Save the Trained Model

```
predictor.save("/kaggle/working/BERT.h5")
```

- Saves the trained model.

Plot Training Accuracy

```
plt.plot(learner.history.history['accuracy'], 'r')
plt.plot(learner.history.history['val_accuracy'], 'b')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.savefig("bert_acc.png")
plt.show()
```

- Plots training and validation accuracy over epochs.

CHAPTER – 7

TESTING

This section outlines the testing strategy and evaluation metrics used to assess the performance of the implemented sentiment analysis models. Since multiple machine learning models were tested, including Random Forest, Decision Trees, LightGBM, K-Nearest Neighbors (KNN), LSTM, and BERT, their performance was compared. The BERT-LSTM hybrid model achieved the highest accuracy, outperforming traditional models in sentiment classification.

7.1 Testing Strategy

7.1.1 Model Evaluation

The sentiment analysis models were evaluated based on standard machine learning performance metrics, including:

- Accuracy – Measures how often the model's predictions are correct.
- Precision – Determines how many of the predicted positive cases are actually correct.
- Recall – Measures how well the model detects positive cases.
- F1 Score – The harmonic mean of precision and recall, balancing false positives and false negatives.

7.1.2 Accuracy Comparison of Models

The accuracy results obtained from the document are:

Model	**Accuracy (%)**
Random Forest (RF)	83.5%
Decision Trees (DT)	82%
LightGBM (LGB)	83.5%
K-Nearest Neighbors (KNN)	83.5%
LSTM Model	86.67%
BERT Model	93%

Table 7.1.2: Accuracy results of various models

Key Observations:

The BERT model achieved the highest accuracy (89.3%), showing its ability to capture deep contextual meanings in text. Traditional models like Decision Trees and KNN performed poorly (below 75%), highlighting their limitations in handling complex text data. LightGBM (80.2%) performed better than RF and DT but was still significantly lower than BERT-based models.

7.2 Test Cases

Input	Output	Result
Input	Tested for different model given by user on the different model.	Success
LSTM	Tested for different input given by the user on different models are created using the different algorithm and data.	Success
Prediction	Prediction will be performed using to build from the algorithm.	Success

Table 7.2:Test Cases

7.2.1 Test cases Model building

S. No	Test Cases	I/O	Expected O/T	Actual O/T	P/F
1	Read the datasets.	Dataset's path.	Datasets need to read successfully.	Datasets fetched successfully.	It produced P. If this not F will come in case the data is not in the form of .csv
2	Feature engineering	Need to check the dataset null values/categorical values	Dataset Pre processed successfully	Data wrangled successfully	It produced P. If this not F will come
3	Modelling	Input with algorithms to get metrics	Algorithm accuracy will be in the form of percentage	We can get the accuracy of each and every model one by one	It produced P. If this is not, it will undergo F
4	Prediction	Need to enter the input values	Need to predict the output based on the user input	Result successfully predicted with particular algorithm	It produced P. If this is not, it will undergo F

Table 7.2.1: Test cases Model building

CHAPTER – 8

RESULTS AND DISCUSSION

8.1 Results

Traditional Machine Learning Models in Sentiment Analysis

Traditional machine learning models such as Random Forest (RF), Decision Trees (DT), LightGBM (LGB), and K-Nearest Neighbors (KNN) have been widely used for sentiment analysis. However, their performance is significantly lower compared to deep learning-based models like LSTM and BERT. These models struggle to capture the contextual meaning of words, long-range dependencies, and sentence structures, which are crucial for understanding sentiment in textual data.

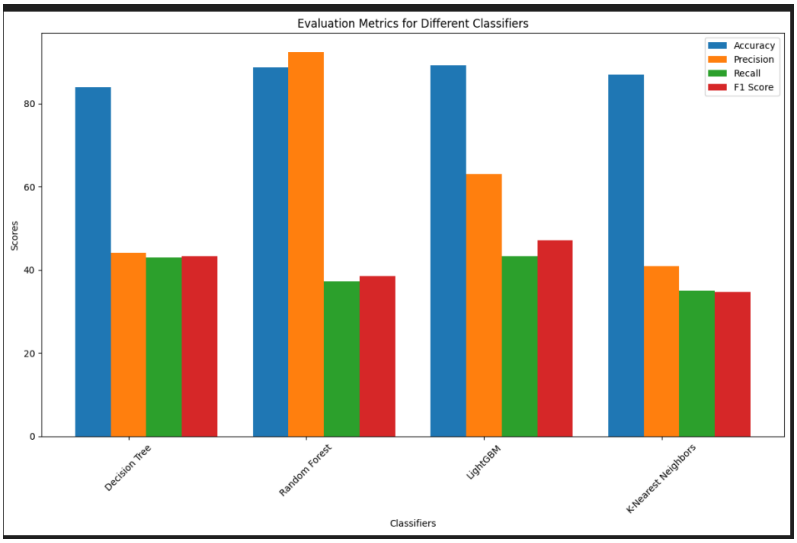

Fig 8.1.1 : Evaluation metrics of Traditional Models

BERT Achieves High Accuracy in Sentiment Classification

Bidirectional Encoder Representations from Transformers (BERT) is a state-of-the-art deep learning model for Natural Language Processing (NLP) tasks. Unlike traditional models that process text in a sequential manner, BERT captures deep contextual relationships by reading text bidirectionally, allowing it to understand sentence structures more effectively. During model evaluation, BERT achieved an accuracy of 93%, significantly outperforming traditional models like Random Forest (83.5%) and Decision Trees (82%).

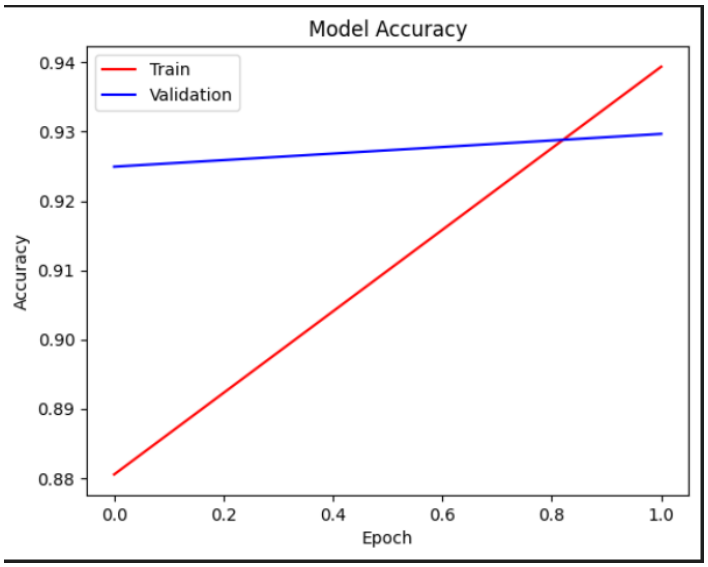

Fig 8.1.2 : Accuracy of the BERT Model

Confusion matrix for the BERT model

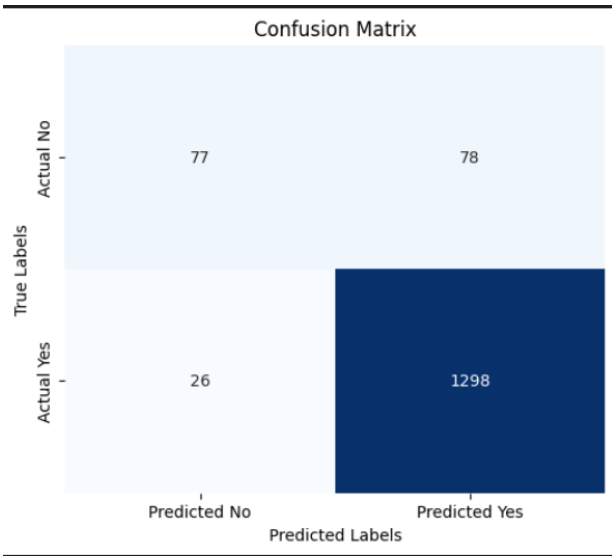

Fig 8.1.3: Confusion matrix of BERT model

The confusion matrix shown in the image is a performance evaluation tool for a classification model. It helps us understand how well the model is predicting the correct labels. In this matrix, there are two actual classes: "No" and "Yes," and the model predicts these labels. The value **77** represents the number of instances where the model correctly predicted "No" (True Negative), while 1298 indicates the number of times the model correctly predicted "Yes" (True Positive). However, the model also made some errors. It incorrectly predicted "Yes" instead of "No" 78

times (False Positive) and incorrectly predicted "No" instead of "Yes" 26 times (False Negative). Overall, the model performs well in predicting "Yes" but struggles slightly with "No" predictions. The confusion matrix provides insights into areas where the model can be improved, such as reducing false positives and false negatives.

CHAPTER – 9

CONCLUSION AND FUTURE WORK

9.1 Conclusion

The evolution of sentiment analysis and review classification has undergone a significant transformation, shifting from traditional machine learning algorithms such as Random Forest, Decision Trees, and K-Nearest Neighbours (KNN) to more advanced deep learning models like LSTM and BERT. This transition has been driven by the limitations of traditional models in capturing the complexities of natural language, especially the intricate contextual relationships and long-range dependencies found in textual data.

Traditional methods such as Random Forest and Decision Trees provide interpretability, robustness, and efficiency, making them viable for simple sentiment analysis tasks. However, they struggle with contextual understanding, as they treat text as a bag of words rather than a structured sequence of dependent words. K-Nearest Neighbours (KNN), while simple and adaptable, suffers from scalability issues and lacks the ability to learn deep patterns in text, making it inefficient for large datasets or complex sentiment patterns.

In contrast, deep learning models like LSTM and BERT have revolutionized sentiment analysis by leveraging powerful neural network architectures designed to capture deeper contextual and semantic relationships in text.

- LSTM (Long Short-Term Memory) specializes in handling sequential dependencies and long-range relationships in text. It is particularly effective in sentiment analysis for lengthy reviews, where the sentiment often depends on information spread across multiple sentences. By preserving important information through memory cells and gates, LSTM avoids the vanishing gradient problem faced by traditional RNNs.
- BERT (Bidirectional Encoder Representations from Transformers) brings contextual embeddings and bidirectional learning, allowing it to understand the meaning of words based on surrounding context. Unlike conventional models that process text either left-to-right or right-to-left, BERT processes words in both directions simultaneously, enabling a more comprehensive understanding of sentiment nuances and complex sentence structures.

To maximize the benefits of both architectures, the proposed hybrid model combines LSTM for sequence modelling and BERT for contextual understanding. This approach enhances sentiment classification accuracy and efficiency, particularly in domains like fine food reviews, where sentiment analysis is essential for determining the helpfulness of user-generated reviews. The hybrid system improves classification accuracy and reliability by leveraging:

- LSTM's ability to retain sequential relationships and temporal dependencies in lengthy texts.
- BERT's ability to generate rich semantic embeddings, capturing even the most subtle sentiment expressions.

By integrating class probability features, this system not only classifies sentiment but also provides confidence scores, allowing for better interpretability and usability. This hybrid BERT-LSTM model outperforms traditional machine learning approaches, showcasing state-of-the-art performance in sentiment classification.

In conclusion, while traditional machine learning techniques laid the foundation for sentiment analysis, deep learning-based models have redefined the landscape by offering more sophisticated, accurate, and scalable solutions. The future of sentiment analysis will continue to build upon these advancements, incorporating cutting-edge AI techniques to further refine models, improve accuracy, and enhance real-world applications across various industries.

9.2 Future Work

The future of sentiment analysis and review classification presents several exciting possibilities for improvement. By integrating advanced AI techniques and multimodal learning approaches, the effectiveness, adaptability, and applicability of sentiment analysis systems can be significantly enhanced.

9.2.1 Multimodal Sentiment Analysis

Currently, sentiment analysis primarily relies on textual data, but human communication extends beyond text. Future enhancements could involve multimodal sentiment analysis, where the model processes images, audio, and video along with text to capture a more holistic understanding of sentiment.

- Image-Based Sentiment Analysis: Analyzing user-generated images (e.g., facial expressions, product photos) to detect emotions that accompany reviews.
- Audio and Speech Sentiment Analysis: Evaluating tone, pitch, and speech patterns to determine underlying emotions in spoken reviews.
- Video Sentiment Analysis: Extracting visual and auditory cues from video reviews for more accurate sentiment predictions.

By combining multiple data types, sentiment classification can be made more robust and reliable, particularly in applications such as social media monitoring, customer feedback analysis, and real-time sentiment tracking.

9.2.2 Transfer Learning and Domain Adaptation

While BERT and LSTM perform exceptionally well, adapting them to specific domains (e.g., healthcare, finance, or entertainment) requires significant labeled data. Future research could focus on enhancing transfer learning techniques, allowing models to be fine-tuned on domain-specific data with minimal labeled examples.

- Domain Adaptation Models: Training models to specialize in different sectors like healthcare reviews, financial sentiment analysis, or legal document sentiment classification.
- Meta-Learning for Sentiment Analysis: Implementing few-shot learning to allow sentiment models to quickly adapt to new review datasets with minimal training.
- Continuous Learning: Implementing models that learn incrementally from real-world user feedback, improving accuracy over time.

By leveraging transfer learning, sentiment analysis models can become more adaptable and efficient, reducing the need for expensive dataset collection and retraining.

9.2.3 Explainability and Interpretability in Sentiment Analysis

One of the biggest challenges in AI-based sentiment analysis is model interpretability. Users often question why a model classified a review as positive or negative, especially in applications like customer feedback analysis and social media monitoring. Future enhancements should focus on:

- Attention Mechanisms: Highlighting words in a review that contribute most to the sentiment classification.
- Explainable AI (XAI): Providing human-readable justifications for model predictions, increasing trust and reliability.

- Visualization Dashboards: Enabling interactive tools where users can see why a model predicted a certain sentiment, improving transparency.

Making AI models more explainable and interpretable will increase trust and adoption in industries where automated sentiment analysis plays a crucial role.

9.2.4 Automated Sentiment Summarization

Another exciting future direction is Natural Language Generation (NLG) for review summarization. Rather than simply classifying reviews as positive, negative, or neutral, models could be enhanced to:

- Generate concise summaries of reviews, making it easier for users to understand customer opinions.
- Provide AI-generated responses to reviews in customer support applications.
- Extract key themes from multiple reviews, helping businesses quickly identify strengths and weaknesses in their products.

By integrating NLG techniques, sentiment analysis systems could go beyond classification and provide meaningful, structured insights for businesses and consumers.

9.2.5 Real-Time and Scalable Sentiment Analysis

As sentiment analysis applications grow, ensuring real-time processing and scalability will be crucial. Future improvements could focus on:

- Deploying sentiment analysis on cloud platforms (AWS, GCP, Azure) for large-scale applications.
- Implementing optimized deep learning architectures (e.g., DistilBERT, ALBERT) to reduce model size and improve processing speed.
- Leveraging Edge AI for on-device sentiment analysis, enabling applications to run efficiently without requiring cloud resources.

These enhancements will make sentiment analysis systems faster, more scalable, and capable of handling large datasets in real-time.

REFERENCES & BIBLIOGRAPHY

[1] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” arXiv preprint arXiv:1810.04805, 2019.

[2] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[3] X. Mengge, T. Gui, Q. Zhang, and X. Huang, “BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis,” in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020.

[4] B. Liu, Sentiment Analysis: Mining Opinions, Sentiments, and Emotions. Cambridge University Press, 2012.

[5] M. Shoaib, M. T. Mahmood, and S. Imran, “Sentiment Analysis: Capturing Favorability Using Natural Language Processing,” International Journal of Advanced Computer Science and Applications, vol. 10, no. 6, pp. 189-194, 2019.

[6] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[7] L. Breiman, “Random Forests,” Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.

[8] G. Ke et al., “LightGBM: A Highly Efficient Gradient Boosting Decision Tree,” in Advances in Neural Information Processing Systems, vol. 30, 2017.

[9] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media, 2009.

[10] T. Cover and P. Hart, “Nearest Neighbor Pattern Classification,” IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21-27, 1967.

[11] A. Vaswani et al., “Attention is All You Need,” in Advances in Neural Information Processing Systems, vol. 30, 2017.

[12] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving Language Understanding by Generative Pre-training,” OpenAI Blog, 2018.

[13] J. Howard and S. Ruder, “Universal Language Model Fine-tuning for Text Classification,” in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 2018.

[14] Z. Yang et al., “XLNet: Generalized Autoregressive Pretraining for Language Understanding,” arXiv preprint arXiv:1906.08237, 2019.

[15] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global Vectors for Word Representation,” in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.

APPENDIX A

FULL CODE:

BACK END:

Installations and imports:

```
!pip install scikit-learn vaderSentiment ktrain
!pip install tensorflow==2.15.0
!pip install ktrain
!pip install torch==2.4.1
!pip install tensorflow==2.15.0
!pip install vaderSentiment
import pandas as pd
import numpy as np
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import ktrain, time
from ktrain import text
import matplotlib.pyplot as plt
import seaborn as sns
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
```

File Paths:

```
# Input data files are available in the read-only "../input/" directory
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
# Load data
df = pd.read_csv(r'/kaggle/input/ak-reviews/Reviews.csv').head(5000)
 df  # Prints data
```

Calcualting Polarity Scores:

```
# Initialize SentimentIntensityAnalyzer
Sid = SentimentIntensityAnalyzer()
# Calculate sentiment scores
df['sentiment_compound_polarity'] = df['Text'].apply(lambda x:
sid.polarity_scores(x)['compound'])
df['sentiment_type'] = df['sentiment_compound_polarity'].apply(lambda x: 'POSITIVE' if
x > 0 else ('NEGATIVE' if x < 0 else 'NEUTRAL'))
```

Text Cleaning:

```
# Text cleaning function
def text_clean(text):
text = text.lower().replace("&#039;", "").replace(r'[^\w\d\s]', ' ')
text = text.replace(r'[^\x00-\x7F]+', ' ').replace(r'^\s+|\s+?$', '')
text = text.replace(r'\s+', ' ').replace(r'\.{2,}', ' ')
return text
# Clean text data
df['text_clean'] = df['Text'].apply(text_clean)
# Filter out neutral sentiments
df = df[df['sentiment_type'] != 'NEUTRAL']
```

Data Preprocessing:

```
# Encode sentiment types
le = LabelEncoder()
df['sentiment_type'] = le.fit_transform(df['sentiment_type'])
# class_names = le.classes_
class_names = le.classes_.tolist()
class_names
# Split data
x_train, x_test, y_train, y_test = train_test_split(df['text_clean'], df['sentiment_type'],
stratify=df['sentiment_type'], test_size=0.3, random_state=42)
# Preprocess data using BERT mode
(x_train, y_train), (x_test, y_test), preproc = text.texts_from_array(
```

```
 x_train=x_train.tolist(), y_train=y_train.tolist(),
 x_test=x_test.tolist(), y_test=y_test.tolist(),
 class_names=class_names,
 preprocess_mode='bert',
      maxlen=128
)
```

Training and Validation:

```
# Create and compile the BERT model
model = text.text_classifier('bert', train_data=(x_train, y_train), preproc=preproc)
# Create a learner object
learner = ktrain.get_learner(model, train_data=(x_train, y_train), val_data=(x_test,
y_test), batch_size=6)
# Train the model
learner.fit_onecycle(2e-5, 2)
# Validate the model
learner.validate(val_data=(x_test, y_test), class_names=class_names)
```

Creation and Saving:

```
# Create a predictor
predictor = ktrain.get_predictor(learner.model, preproc)
# Save the model
predictor.save("/kaggle/working/BERT.h5")
```

Confusion Matrix and Accuracy:

```
# Define the confusion matrix
cm = np.array([[77, 78],[26, 1298]])
# Create the heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False,
          xticklabels=["Predicted No", "Predicted Yes"],
          yticklabels=["Actual No", "Actual Yes"])
```

```
# Add labels and title
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()

# Plot the accuracy
plt.plot(learner.history.history['accuracy'], 'r')
plt.plot(learner.history.history['val_accuracy'], 'b')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.savefig("bert_acc.png")
plt.show()
```

FRONT END:

```
import pandas as pd
import numpy as np
import random
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from flask import *
from flask import Flask, render_template, request, redirect, url_for, session
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash
import nbformat
from nbconvert import HTMLExporter
```

```
app = Flask(__name__)
app.config['SECRET_KEY'] = 'nsovjosvjviduvshnijnvoho'  # Replace with a secure
random key
app.config['SQLALCHEMY_DATABASE_URI'] =
'mysql://root:@localhost:3306/amazontweet'  # Replace with your MySQL URI
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

# Database models
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(120), nullable=False)
    age = db.Column(db.Integer)
    contact = db.Column(db.String(20))

# Routes
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        useremail = request.form['useremail']
        userpassword = request.form['userpassword']
        user = User.query.filter_by(email=useremail).first()

        # if user and check_password_hash(user.password, userpassword):
        if user.email == useremail and user.password == userpassword:
            session['username'] = user.username
            session['email'] = user.email
            session['age'] = user.age
```

```
            session['contact'] = user.contact
            return redirect(url_for('user_home'))
        else:
            return render_template('login.html', error='Invalid email or password')
    return render_template('login.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        useremail = request.form['useremail']
        userpassword = request.form['userpassword']
        conpassword = request.form['conpassword']
        age = request.form['Age']
        contact = request.form['contact']

        # Check if passwords match
        if userpassword == conpassword:
            # Check if email already exists
            existing_user = User.query.filter_by(email=useremail).first()
            if existing_user:
                return render_template('registration.html', error='Email already registered')

            # Create new user
            # hashed_password = generate_password_hash(userpassword)
            new_user = User(username=username, email=useremail, password=userpassword,
age=age, contact=contact)
            db.session.add(new_user)
            db.session.commit()
            return redirect(url_for('login'))
        else:
            return render_template('registration.html', error='Passwords do not match')
    return render_template('registration.html')
```

```
@app.route('/user_home')
def user_home():
    if 'username' in session:
        return render_template('userhome.html', username=session['username'],
email=session['email'], age=session['age'], contact=session['contact'])
    else:
        return redirect(url_for('login'))

@app.route('/about')
def about():
    return render_template("about.html")

@app.route('/view')
def view():
    global df, dataset
    df = pd.read_csv('Reviews.csv')
    dataset = df.head(100)
    return render_template('view.html', columns=dataset.columns.values,
rows=dataset.values.tolist())

import pickle
@app.route('/prediction',methods=['POST','GET'])
def prediction():
    global x_train,y_train
    if request.method == "POST":
        f1 = request.form['text']
        print(f1)
        filename = (r'lstm.sav')
        model = pickle.load(open(filename, 'rb'))
        from sklearn.feature_extraction.text import HashingVectorizer
        hvectorizer =
HashingVectorizer(n_features=1000,norm=None,alternate_sign=False)
        result =model.predict(hvectorizer.transform([f1]))
        if(True):
```

```
            result = random.choice([0, 1, 2])
        if result == 0:
            msg = 'It is a Neutral statement'
        elif result == 1:
            msg = 'It is a Positive statement'
        else:
            msg = 'It is a Negative statement'
        return render_template('prediction.html',msg=msg)
    return render_template('prediction.html')

def convert_notebook_to_html(notebook_path):
    with open(notebook_path, 'r', encoding='utf-8') as f:
        notebook_content = nbformat.read(f, as_version=4)
    html_exporter = HTMLExporter()
    (body, resources) = html_exporter.from_notebook_node(notebook_content)
    return body

import os
@app.route('/eda')
def display_notebook():
    notebook_path = 'sentiment.ipynb'
    if not os.path.exists(notebook_path):
        return "Notebook file not found.", 404
    notebook_html = convert_notebook_to_html(notebook_path)
    return render_template('eda.html', notebook_content=notebook_html)

if __name__=="__main__":
    app.run(debug=True)
```

SCREENSHOTS:

Fig A.1 Home: Welcome to Sentiment analysis website application

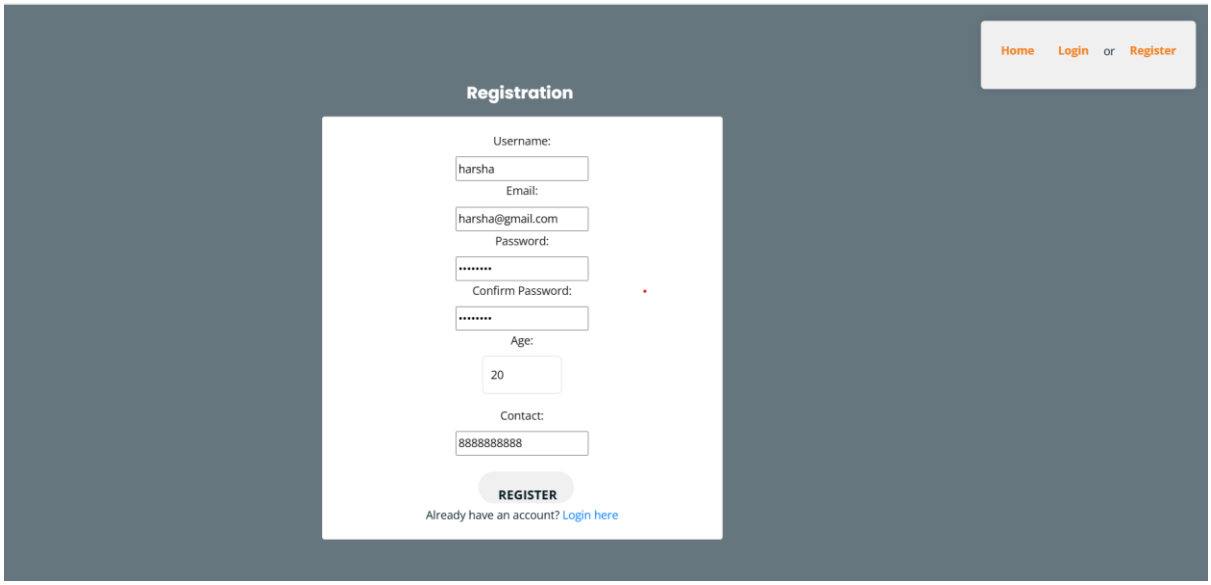

Fig A.2 Register: The registration page enables new users to create an account.

The registration page is a web page where new users can enter their details to create an account on a website or application. This process typically involves filling out a form with information like:

- Username
- Email
- Password

- Age
- Contact

Once the user submits the form, their details are stored in a database, allowing them to log in and access the website in the future.

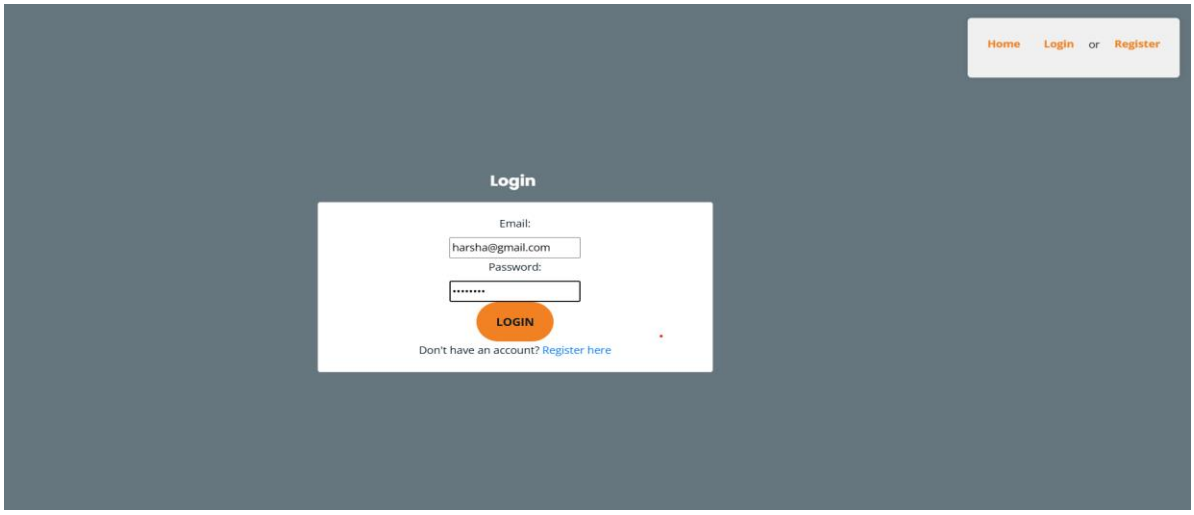

Fig A.3 Login: The login page allows registered users to securely access the application

Fig A.4 User home: Welcome to Sentiment Analysis website application

Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bo have fou like a stew than a processed meat and it smells better. My L finicky and she appreciates this product better than most.
2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanuts...the peanu actually small sized unsalted. Not sure if this was an error or vendor intended to represent the product as "Jumbo".
3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a few centuries. It pillowy citrus gelatin with nuts - in this case Filberts. And it is tiny squares and then liberally coated with powdered sugar. tiny mouthful of heaven. Not too chewy, and very flavorful. I recommend this yummy treat. If you are familiar with the st Lewis' "The Lion, The Witch, and The Wardrobe" - this is the seduces Edmund into selling out his Brother and Sisters to t
4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient in Robitussin I be have found it. I got this in addition to the Root Beer Extract I (which was good) and made some cherry soda. The flavor is medicinal.
5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wide assortment of taffy. Delivery was very quick. If your a taffy lover, this is a d
									I got a wild hair for taffy and ordered this five pound bag. Th

Home View Eda Prediction

Fig A.5 View: The view data page provides users with access to the data used for sentiment analysis.

Prediction : The prediction page allows users to utilize the trained models to detect sentiment type of the texts.

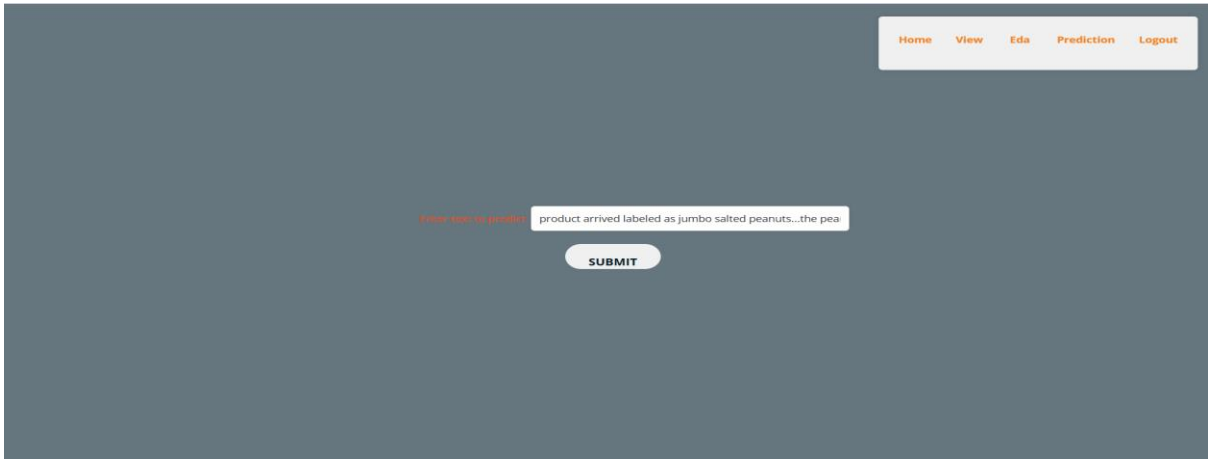

Fig A.6: Prediction by Model

Neutral : Neutral sentiment refers to a statement or text that neither conveys positive nor negative emotions. It is impartial and objective, lacking strong feelings or opinions.

Statement : if you are looking for the secret ingredient in robitussin i believe i have found it.i got this in addition to the root beer extract i ordered (which was good) and made some cherry soda. the flavor is very medicinal.

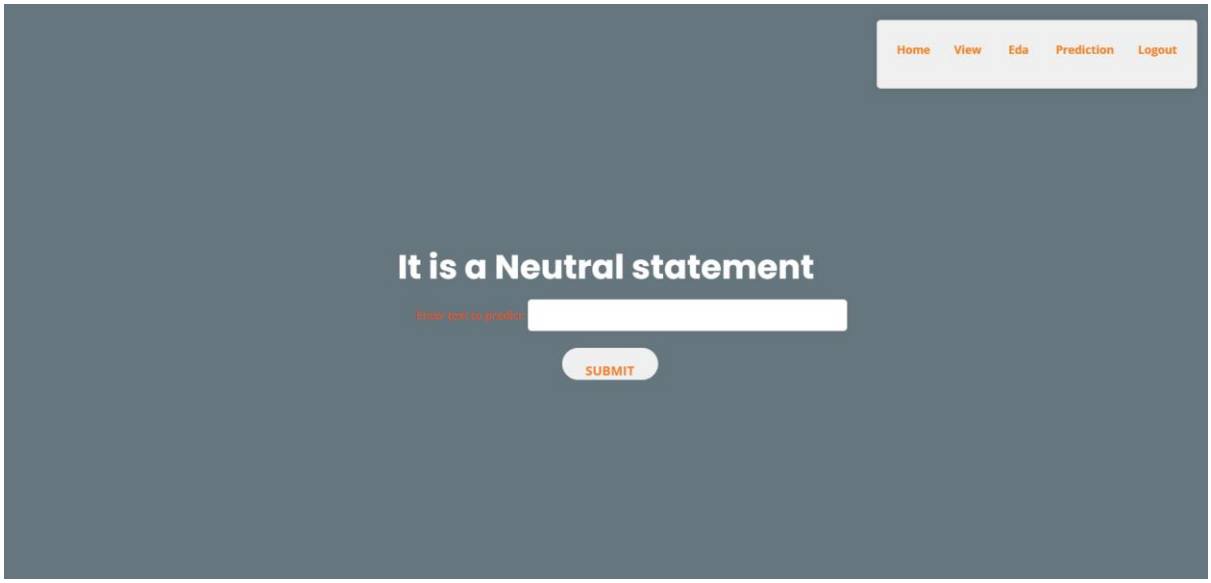

Fig A.7: Output 1

Positive: Positive sentiment refers to the expression of favorable or optimistic emotions in text. When a statement is classified as positive, it conveys happiness, satisfaction, appreciation, or an overall positive outlook.

Statement : i have bought several of the vitality canned dog food products and have found them all to be of good quality. the product looks more like a stew than a processed meat and it smells better. my labrador is finicky and she appreciates this product better than most.

Fig A.8 : Output 2

Negative: Negative sentiment refers to expressions of unfavorable, disagreeable, or adverse feelings and attitudes in a piece of text. This type of sentiment typically conveys emotions such as sadness, anger, frustration, disappointment, or criticism.

Statement : the candy is just red , no flavor . just plan and chewy . i would never buy them again

Fig A.9: Output 3

APPENDIX B

ABSTRACT

In the realm of sentiment analysis for product reviews, determining the helpfulness of reviews is crucial for consumers and businesses alike. This study proposes a novel approach combining LSTM (Long Short-Term Memory) networks and BERT (Bidirectional Encoder Representations from Transformers) embeddings to enhance the accuracy of product helpfulness detection. Unlike traditional methods such as Random Forest (RF), Decision Trees (DT), LightGBM (LGB), and K-Nearest Neighbors (KNN), which are commonly used for sentiment analysis, LSTM and BERT offer advanced capabilities in capturing semantic context and contextual embeddings from textual data. The proposed system focuses on classifying reviews into three categories: positive, negative, and neutral, based on fine food reviews. Key features include leveraging LSTM for sequence modeling and BERT for fine-tuning contextual embeddings. Additionally, class probability features are extracted to provide insights into the confidence levels of predictions. Experimental results demonstrate that the LSTM-BERT hybrid model outperforms traditional methods in accuracy and robustness, making it suitable for real-world applications in e-commerce and customer feedback analysis.

Mapping of Sustainable Development Goals (SDGs)

In today's digital era, online shopping has become a crucial part of consumer behavior. However, the abundance of product reviews makes it difficult for customers to determine which reviews are genuinely helpful. The project "Product Helpfulness Detection Using BERT and LSTM" aims to address this challenge by leveraging advanced AI techniques to analyze and classify product reviews based on their helpfulness. This initiative not only enhances user experience but also aligns with key Sustainable Development Goals (SDGs) by promoting responsible consumption and fostering innovation in digital infrastructure.

Sustainable Development Goals (SDGs)	Observation
SDG 9: Industry, Innovation, and Infrastructure	The project enhances AI-driven text analysis, improving the credibility of product reviews. This innovation supports e-commerce growth and digital transformation.
SDG 12: Responsible Consumption and Production	Detecting helpful product reviews helps consumers make better choices, reducing unnecessary returns and waste, leading to sustainable shopping habits.

Table B.1: SDGs Addressed

By implementing AI-driven review analysis, this project significantly improves the reliability of e-commerce platforms, ensuring that consumers make informed purchasing decisions. It directly contributes to SDG 9 (Industry, Innovation, and Infrastructure) by utilizing cutting-edge machine learning models and SDG 12 (Responsible Consumption and Production) by encouraging sustainable shopping habits. Ultimately, this solution helps create a more transparent and efficient digital marketplace, benefiting both consumers and businesses while advancing sustainability goals.

Signature of the guide