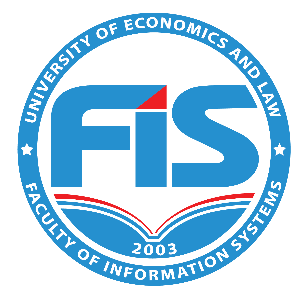
**UNIVERSITY OF ECONOMICS AND LAW**

**FACULTY OF INFORMATION SYSTEMS**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

****

**FINAL PROJECT REPORT**

**TEXT MINING COURSE**

**TOPIC:**

***LokeyTech: AI-driven text mining for E-commerce***

**Lecturer: Tran Duy Thanh, PhD.**

**Group Live Laugh Love:**

1. **K214162148 – Cao Nguyen Hai Nhu**
2. **K214162143 – Tran Hoang Anh**
3. **K214160989 – Tran Thi Minh Hien**

**Ho Chi Minh City, November 2024**

# Members

|  |  |  |  |
| --- | --- | --- | --- |
| *No.* | **Full name** | **Student ID** | **Signature** |
| *1* | Cao Nguyen Hai Nhu (Leader) | K214162148 | Shape  Description automatically generated |
| *2* | Tran Hoang Anh | K214162143 | A picture containing insect  Description automatically generated |
| *3* | Tran Thi Minh Hien | K214160989 | A picture containing insect, hanger  Description automatically generated |

# Acknowledgement

First and foremost, our group would like to extend our deepest gratitude to Mr Tran Duy Thanh for his unwavering support, guidance, and dedication. His expertise, knowledge, and experience have been invaluable to us, and his enthusiastic encouragement has been a constant source of motivation. We sincerely thank him for his commitment to our learning and research journey.

Lastly, while we have strived to execute this work with great care, we acknowledge that there may still be areas for improvement. We wholeheartedly welcome and greatly appreciate any feedback and suggestions from our teacher and peers, as they will help us continue to refine and perfect our research paper./.

Group Live Laugh Love

# Commitment

The team affirms that the results of the project “LokeyTech: AI-driven text mining for E-commerce” are the original work of the group. This project is built upon the knowledge acquired from the Text Mining course, guided and advised by Mr Tran Duy Thanh, and further supported by references from books, articles, and other reputable sources.

All references utilized in this project have been properly cited and presented by the research team./.

Ho Chi Minh City, November 2024

Group Live Laugh Love

# Table of Content

[Members I](#_Toc182315076)

[Acknowledgement II](#_Toc182315077)

[Commitment III](#_Toc182315078)

[Table of Content IV](#_Toc182315079)

[List of Figure VI](#_Toc182315080)

[List of Table VII](#_Toc182315081)

[Project Management Reporting 1](#_Toc182315082)

[Project Overview 6](#_Toc182315083)

[Reasons 6](#_Toc182315084)

[Objectives 7](#_Toc182315085)

[Objects and scopes 8](#_Toc182315086)

[**Objects** 8](#_Toc182315087)

[**Scopes** 8](#_Toc182315088)

[Structure of project 8](#_Toc182315089)

[Chapter 1. Introduction 9](#_Toc182315090)

[Chapter 2. Theoretical Basis 11](#_Toc182315091)

[***2.1.*** ***Natural Language Processing*** 11](#_Toc182315092)

[***2.2.*** ***Sentiment Analysis*** 13](#_Toc182315093)

[***2.3.*** ***MVC Framework*** 18](#_Toc182315094)

[***2.4.*** ***.NET Framework*** 20](#_Toc182315095)

[***2.5.*** ***Azure Machine Learning*** 21](#_Toc182315096)

[Chapter 3. System Architecture 23](#_Toc182315097)

[***3.1.*** ***Machine learning service*** 24](#_Toc182315098)

[*3.1.1.* *Model architecture* 24](#_Toc182315099)

[*3.1.2.* *Model deployment* 27](#_Toc182315100)

[***3.2.*** ***Web service*** 30](#_Toc182315101)

[*3.2.1.* *Database design* 30](#_Toc182315102)

[*3.2.2.* *Programming model* 35](#_Toc182315103)

[Chapter 4. Results and Discussion 41](#_Toc182315104)

[***4.1.*** ***Website appearance*** 41](#_Toc182315105)

[***4.2.*** ***Model evaluation*** 49](#_Toc182315106)

[Chapter 5. Conclusion and Future Works 52](#_Toc182315107)

[References 53](#_Toc182315108)

[Appendix A](#_Toc182315109)

# List of Figure

[Figure 2‑1. NLP process. Source: Authors' work 12](#_Toc182170193)

[Figure 2‑2. BERT pre-training/fine tuning flow chart (Li et al., 2022). 15](#_Toc182170194)

[Figure 2‑3. LSTM Cell Unit (Li et al., 2022) 16](#_Toc182170195)

[Figure 2‑4 General framework of BERT-BiLSTM (Xiong et al,. 2024) 17](#_Toc182170196)

[Figure 2‑5. MVC architecture. Source: geeksforgeeks.org 17](#_Toc182170197)

[Figure 3‑1. Overall architecture. Source: Authors' work 22](#_Toc182170198)

[Figure 3‑2. Model architecture. Source: Authors' work 23](#_Toc182170199)

[Figure 3‑3. Model deployment process. Source: Authors' work 25](#_Toc182170200)

[Figure 3‑4. Database design. Source: Authors' work 28](#_Toc182170201)

[Figure 3‑5. API microservice. Source: Authors’ work 34](#_Toc182170202)

[Figure 3‑6. Client microservice. Source: Authors' work 35](#_Toc182170203)

[Figure 4‑1 UI for sign-in. Source: Authors’ work 38](#_Toc182170204)

[Figure 4‑2 UI for sign-up. Source: Authors’ work 38](#_Toc182170205)

[Figure 4‑3 UI for admin - Overview page. Source: Authors’ work 39](#_Toc182170206)

[Figure 4‑4 UI for admin - Page for product list. Source: Authors’ work 39](#_Toc182170207)

[Figure 4‑5 Pop-up for product information editing. Source: Authors’ work 40](#_Toc182170208)

[Figure 4‑6 UI for admin - Transaction page. Source: Authors’ work 40](#_Toc182170209)

[Figure 4‑7 UI for admin - Report page. Source: Authors’ work 41](#_Toc182170210)

[Figure 4‑8 Homepage. Source: Authors’ work 42](#_Toc182170211)

[Figure 4‑9 Special offers section. Source: Authors’ work 43](#_Toc182170212)

[Figure 4‑10 Product detail page. Source: Authors’ work 44](#_Toc182170213)

[Figure 4‑11 Product category page. Source: Authors’ work 45](#_Toc182170214)

[Figure 4‑12. Shopping cart section. Source: Authors’ work 46](#_Toc182170215)

[Figure 4‑13. Confusion matrix. Source: Authors' work 47](#_Toc182170216)

# List of Table

[Table 3‑1. Database detail. Source: Authors' work 29](#_Toc182170217)

[Table 4‑1. Classification report. Source: Authors' work 46](#_Toc182170218)

# Project Management Reporting

**Total Number of Tasks:** 22

**Tasks Completed:** 21 (Done)

**Tasks Cancelled:** 1 (Cancelled)

**Start Date:** 10/10/2024

**End Date:** 10/11/2024

**Total Duration:** 30 days

Work assignment

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Task** | **Part** | **Status** | **PIC** | **Evaluation** | **Dependance** |
| MD-01 | Architecture Research | Architecture Research | Done | All | 100% | - |
| MD-02 | Architecture Research | Architecture Modification | Done | Nhu | 100% | MD-01 |
| MD-03 | Metrics Research | All | Done | Hien | 100% | MD-02 |
| MD-04 | Data Collection | All | Done | All | 100% | MD-03 |
| MD-05 | Project Setup | Project Skeleton | Done | Anh | 100% | MD-02, MD-04 |
| MD-06 | Project Setup | Project Architecture | Done | Anh | 100% | MD-02 |
| MD-07 | Coding | Train Model | Done | Nhu | 100% | MD-04, MD-05, MD-06 |
| MD-08 | Coding | Code and combine GUI with the model | Done | Anh | 100% | MD-07 |
| RP-01 | Project Overview | Project Overview | Done | Nhu | 100% | MD-03 |
| RP-02 | Introduction | Overview of the e-commerce situation and users' behaviors | Done | Nhu | 100% | RP-01 |
| RP-03 | Theoretical Basis | Theoretical Background | Done | Hien | 100% | RP-02 |
| RP-04 | Requirement Analysis | Requirement Analyzing | Done | All | 100% | RP-03 |
| RP-05 | Requirement Analysis | Use Case Definement | Done | Anh | 100% | RP-04 |
| RP-06 | System Architecture | Overall architecture | Done | Nhu | 100% | RP-05 |
| RP-07 | System Architecture | Machine learning service | Done | Nhu | 100% | RP-06 |
| RP-08 | System Architecture | Web service | Done | Nhu | 100% | RP-07 |
| RP-09 | Results and Discussion | Mockup | Done | Hien | 100% | RP-08 |
| RP-10 | Results and Discussion | Website appearance | Done | All | 100% | RP-09 |
| RP-11 | Results and Discussion | Website performance | Cancel | All | 0% | MD-08 |
| RP-12 | Results and Discussion | Model evaluation | Done | All | 100% | RP-10 |
| RP-13 | Conclusion and Future Works | Conclusion and Future Works | Done | All | 100% | RP-12 |
| RP-14 | Formatting | Document Formatting | Done | Nhu | 100% |  |

Team performance evaluation

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Member Id** | **Member Name** | **Evaluation** | **Note** |
| K214160989 | Tran Thi Minh Hien | 100% | Detail-oriented, responsible |
| K214162143 | Tran Hoang Anh | 100% | High performance, enthusiastic |
| K214162148 | Cao Nguyen Hai Nhu | 100% |  |

# Project Overview

## Reasons

In today’s digital economy, e-commerce has surged to become a central channel for retail and services, reflecting consumer preferences and behaviors in real time. A key driver of this growth is user-generated content (UGC), particularly customer reviews and feedback, which have emerged as critical indicators of customer satisfaction, product quality, and brand loyalty. This content provides valuable insights into consumer sentiment and can greatly influence purchasing decisions. As studies show, the majority of customers consult online reviews before making purchases, using them as a trusted source of information to assess product reliability and overall brand credibility.

However, while these reviews offer a wealth of information, they also present challenges due to their volume and the diversity of sentiments they express. Manually processing and interpreting thousands of reviews is both time-consuming and error-prone, leading businesses to seek automated solutions that can analyze sentiment efficiently. Advanced machine learning techniques have emerged as powerful tools in this domain, offering businesses the ability to automatically gauge the emotions, concerns, and satisfaction levels expressed by customers. This capability is especially valuable for e-commerce platforms, where understanding and responding to customer feedback can directly impact customer retention, engagement, and overall profitability.

This project responds to these broader trends by showcasing the potential of machine learning models to automate and enhance sentiment analysis in e-commerce, providing a template for how businesses can use customer feedback to inform product improvements, personalize marketing strategies, and foster stronger customer relationships.

## Objectives

The primary objective of this project is to design and develop an e-commerce website that leverages sentiment analysis to provide data-driven business insights. Key objectives include:

1. **Developing a realistic comment collection platform:** The website will simulate a genuine e-commerce environment with a comment section where customer feedback is collected. This section will serve as a robust data source for sentiment analysis, closely resembling real-world e-commerce scenarios.
2. **Implementing an integrated machine learning model for sentiment analysis:** An advanced hybrid BERT + BiLSTM model will be deployed within the website’s admin dashboard, enabling real-time sentiment extraction and prediction. This model combines BERT’s contextual understanding with BiLSTM’s capacity to process sequential data, allowing for a highly accurate analysis of customer sentiment.
3. **Leveraging cloud deployment for efficient processing and scalability:** To ensure high performance and resilience, the model will be trained locally and deployed on the cloud. This cloud setup will minimize processing loads on local systems and mitigate potential system instability, allowing the website to deliver reliable, real-time sentiment insights without disrupting core functionality.
4. **Providing actionable insights for business improvement:** The sentiment analysis tool will categorize customer feedback into positive, neutral, or negative sentiments. Additionally, it will support predictive insights, allowing businesses to gauge potential future customer responses. This functionality offers a powerful resource for companies aiming to improve customer satisfaction and foster loyalty through data-informed decisions.
5. **Showcasing the application of machine learning in e-commerce:** By utilizing machine learning to transform user-generated content into actionable insights, the project highlights how sentiment analysis can enhance customer relations and inform strategic decisions. This demonstration underscores the valuable role of text mining in transforming customer reviews into a critical asset for improving the customer experience.

By achieving these objectives, this project not only demonstrates the potential of cloud-based machine learning in e-commerce but also provides a scalable framework for businesses to harness sentiment analysis in real-world applications.

## Objects and scopes

### **Objects**

10,000 reviews about mobile phone products on Amazon.

### **Scopes**

**Time scope:** 10/2024 – 11/2024

**Space scope:** The research will be limited to the Amazon e-commerce platform, focusing specifically on the mobile phone product category.

## Structure of project

This project contains 54 pages, 24 figures, and 02 tables.. The report is divided into 05 parts, including:

Chapter 1: Introduction

Chapter 2: Theoretical Basis

Chapter 3: System Architecture

Chapter 4: Results and Discussion

Chapter 5: Conclusion and Future Works

# Introduction

In recent years, the explosive growth of e-commerce has significantly transformed consumer behavior and the retail landscape. More than ever, customers are not only purchasing online but also actively sharing their experiences through reviews, ratings, and comments on e-commerce websites. This user-generated content (UGC) has become a critical resource, providing businesses with authentic insights into consumer preferences, needs, and expectations. According to research, over 90% of consumers read online reviews before making a purchase (Dosa, 2022; Zhou, 2024), and more than 45% trust these reviews as much as personal recommendations (Clark, 2024). This trend underscores the influence of UGC in shaping brand perception, building customer trust, and driving purchasing decisions.

For businesses, understanding the content and sentiment of customer feedback is vital. Reviews and comments reveal not just what customers are buying but also their attitudes toward products, services, and the brand itself. Sentiment analysis—the process of identifying and categorizing opinions expressed in text—enables companies to gain insight into customer satisfaction levels and identify pain points. This ability to automatically interpret customer sentiment from large volumes of feedback offers a strategic advantage: it allows companies to respond proactively, whether by addressing common complaints, enhancing popular features, or tailoring marketing strategies.

In response to these trends, our project’s goal is to develop an e-commerce website with a comment section that mimics a real-world customer feedback environment. By using a dataset of simulated comments, we create a platform that resembles a genuine e-commerce scenario, providing a foundation for data mining. The site includes an admin dashboard where administrators can analyze customer sentiment via a machine learning model—specifically, a BERT + BiLSTM hybrid. This model combines BERT’s deep contextual analysis with BiLSTM’s sequential capabilities, allowing for accurate sentiment extraction and prediction.

To enhance performance and resilience, we plan to train the model locally and deploy it to the cloud. This setup addresses potential bottleneck issues, reducing the load on local systems and ensuring a smoother experience on the platform. Cloud deployment mitigates the risk of crashes, as any instability in the model will not affect the website’s primary functions. This approach provides a robust solution for real-time sentiment analysis, minimizing technical disruptions and enhancing the application’s reliability.

Through this approach, our project demonstrates the practical value of advanced text mining and ML models for e-commerce businesses. This setup not only provides a snapshot of customer attitudes but also supports predictive insights, enabling companies to identify trends and make informed, customer-focused decisions. By showcasing the website’s capacity to mimic real-world customer interactions, we highlight the potential for businesses to leverage such tools to enhance satisfaction and foster loyalty.

# Theoretical Basis

* 1. ***Natural Language Processing***

Natural language processing (NLP) is a branch of artificial intelligence focused on how computers and humans communicate using natural language. In recent years, NLP has seen significant growth due to the increasing availability of large datasets and the demand for more advanced, human-like interactions between computers and people. The primary aim of NLP is to create algorithms and models that can accurately understand, generate, and manipulate human language in a natural way. Its significance lies in enhancing the interaction between humans and computers, making communication more intuitive and relatable. NLP has various practical applications, including information retrieval, sentiment analysis, machine translation, and question answering. It holds the potential to transform numerous industries—such as healthcare, education, and customer service—by facilitating more effective communication and information management. Consequently, NLP has emerged as a vital area of research and development, attracting considerable investment for its growth.

Analyzing textual data often demands substantial human resources and investment. Significant time and financial resources are required for tasks like voice recognition, text mining and identification, topic categorization, and semantic reasoning. Developing a basic NLP model typically involves several key steps:

1. **Data Cleaning and Tokenization:** These are crucial processes in NLP that prepare textual data for analysis. Data cleaning includes removing stop words, converting text to lowercase, and normalizing words to a single form, which standardizes variations with the same meaning for easier grouping, such as in the “Bag-of-Words (BoW)” model. After cleaning, the text is tokenized into phrases or smaller units for more precise analysis.
2. **Vectorization (or Word Embedding):** Since computers cannot directly process strings, textual data must be converted into numerical representations. This mapping allows the computer to process the input effectively.
3. **Model Training:** This step is often conducted using deep learning techniques, particularly recurrent neural networks (RNN) and long short-term memory (LSTM) networks. RNNs are a type of deep learning artificial neural network that includes connections to previous hidden states, allowing them to capture sequential information. This capability is essential for understanding human language, as it helps model the dependencies between words in a text. Unlike traditional artificial neural networks, where information flows only forward, RNNs allow information to be passed back to previous nodes, enhancing their predictive capabilities (Yang & Huang, 2023).

A diagram of a diagram

Description automatically generated

Figure 2‑1. NLP process. Source: Authors' work

The advancement of new NLP models, now enhanced by artificial intelligence (AI), opens up a vast array of potential applications. Contrary to popular belief, NLP is already embedded in our daily lives, appearing in software, mobile devices, and increasingly in healthcare settings. Common applications of NLP include language translation, virtual assistants, and email spam detection.

Translation, one of the earliest uses of NLP, enables instant online conversion between languages. Many individuals utilize virtual assistants like Alexa, Siri, and Google Assistant, which are powered by NLP and AI. These assistants can comprehend both written and spoken language, allowing them to engage in meaningful conversations with users. Additionally, NLP plays a crucial role in email spam detection by analyzing the text to classify emails and determine whether they are spam. While spam detection primarily relies on Natural Language Understanding (NLU) for interpreting human language, virtual assistants utilize both NLU and Natural Language Generation (NLG) (Fanni et al., 2023).

* 1. ***Sentiment Analysis***

**Sentiment Analysis Techniques**

Sentiment analysis (SA) is a technique for mining textual data that enables companies to gauge public sentiment regarding their brand, product, or service. It identifies and extracts subjective information from various sources while monitoring online discussions (Zubair Nawaz et al., 2021).

Most companies classify sentiments into three main categories: positive, negative, and neutral. Monitoring these sentiments is crucial, as a surge in negative reviews can damage a brand’s reputation, impacting public relations and goodwill.

Recent advancements in deep learning have significantly enhanced the effectiveness of algorithms used for text analysis. Leveraging advanced artificial intelligence can provide valuable insights for in-depth research. It is essential to categorize customer conversations about a brand based on the following criteria:

1. Key aspects of the brand’s products and services that interest customers.
2. The underlying intent and reactions of users regarding these aspects.

By integrating these concepts, sentiment analysis becomes a vital tool for accurately assessing conversations about millions of brands on a human scale.

In natural language processing tasks, SA is crucial, and there are various ways to approach it, based on Machine Learning, Deep Learning, and a combination of them. For Machine Learning (ML) based techniques, Naive Bayes (NB), Decision Tree (DT), Logistic Regression (LR), and Support Vector Machines (SVM) are widely used methods in sentiment analysis. Among these, SVM and NB are particularly popular. Previous research has evaluated the effectiveness of various techniques, consistently finding that SVM and NB outperform others. NB is also useful for evaluating the quality of online shopping platforms and for spam detection, showcasing the adaptability of SA. Furthermore, some studies have compared SVM and NB with deep learning methods like RNN, revealing that RNN achieves better performance than other machine learning approaches. The following sections will explore deep learning techniques in more detail. (Huang et al., 2023)

In addition to machine learning techniques, deep learning is widely used in sentiment analysis. A well-designed deep learning model can determine the accuracy of its predictions based solely on its neural network, without the need for human intervention. In recent studies, bi-directional encoder representations from transformers (BERT), convolutional neural network (CNN), RNN, LSTM are among the most used techniques. As model development has progressed, more complex architectures like CNN-based bi-directional long short-term memory (CNN-BiLSTM) and attention-based bi-directional gated recurrent unit (BiGRU) models have emerged. These advancements aim to improve the accuracy and efficiency of deep learning models. The leading deep learning techniques in natural language processing (NLP) include RNN and its variations, such as LSTM and GRU. Although CNN was initially designed for computer vision, it has also been applied to text analysis. RNN algorithms come in various forms, including Bidirectional RNN and Bidirectional LSTM, which enhance their functionality. Moreover, RNN can be effectively combined with other deep learning methods or integrated with attention mechanisms to focus on specific elements of textual data. (Huang et al., 2023).

**BERT-BiLSTM Model**

BERT is a pre-trained neural network model that leverages transfer learning. Its Transformer-based bidirectional encoder processes words by considering both the preceding and succeeding context, enabling it to capture the full semantic meaning of each word in a sentence. The basic BERT model consists of 12 stacked encoding layers, with each layer featuring 12 self-attention heads and 768 hidden units in the feed-forward layer. The final output of the model, which serves as input for downstream tasks, produces high-quality word embeddings. As illustrated in ***Figure 2‑2***, BERT is initially pre-trained on a vast corpus, which allows its parameters to generalize across various NLP tasks. It is then fine-tuned for specific tasks to further optimize its performance (Li et al., 2022).

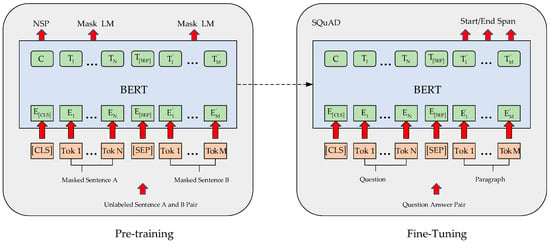


Figure 2‑2. BERT pre-training/fine tuning flow chart (Li et al., 2022).

A BiLSTM model is built from two LSTM networks: one that processes input sequences in the forward direction and another that processes them in the backward direction. The LSTM (Long Short-Term Memory) itself is a variant of the traditional recurrent neural network (RNN) designed to address the issue of vanishing gradients, which often hinders the learning of long-term dependencies in RNNs. To solve this, LSTM introduces gating mechanisms that regulate the flow of information, allowing it to better capture and retain long-range dependencies in sequential data.

Each LSTM cell is structured with four key components: the input gate ​, the output gate , the forget gate ​, and the cell state ​. The internal structure of a single cell of the LSTM module is shown in ***Figure 2‑3***. These gates work together to control the information that is stored, updated, or discarded at each time step. The input gate decides what new information to store in the cell state, the forget gate controls what information to erase, and the output gate determines what information to output based on the current cell state. This architecture enables the LSTM to maintain and update long-term dependencies, making it particularly effective for tasks that involve sequences with complex, long-range relationships.

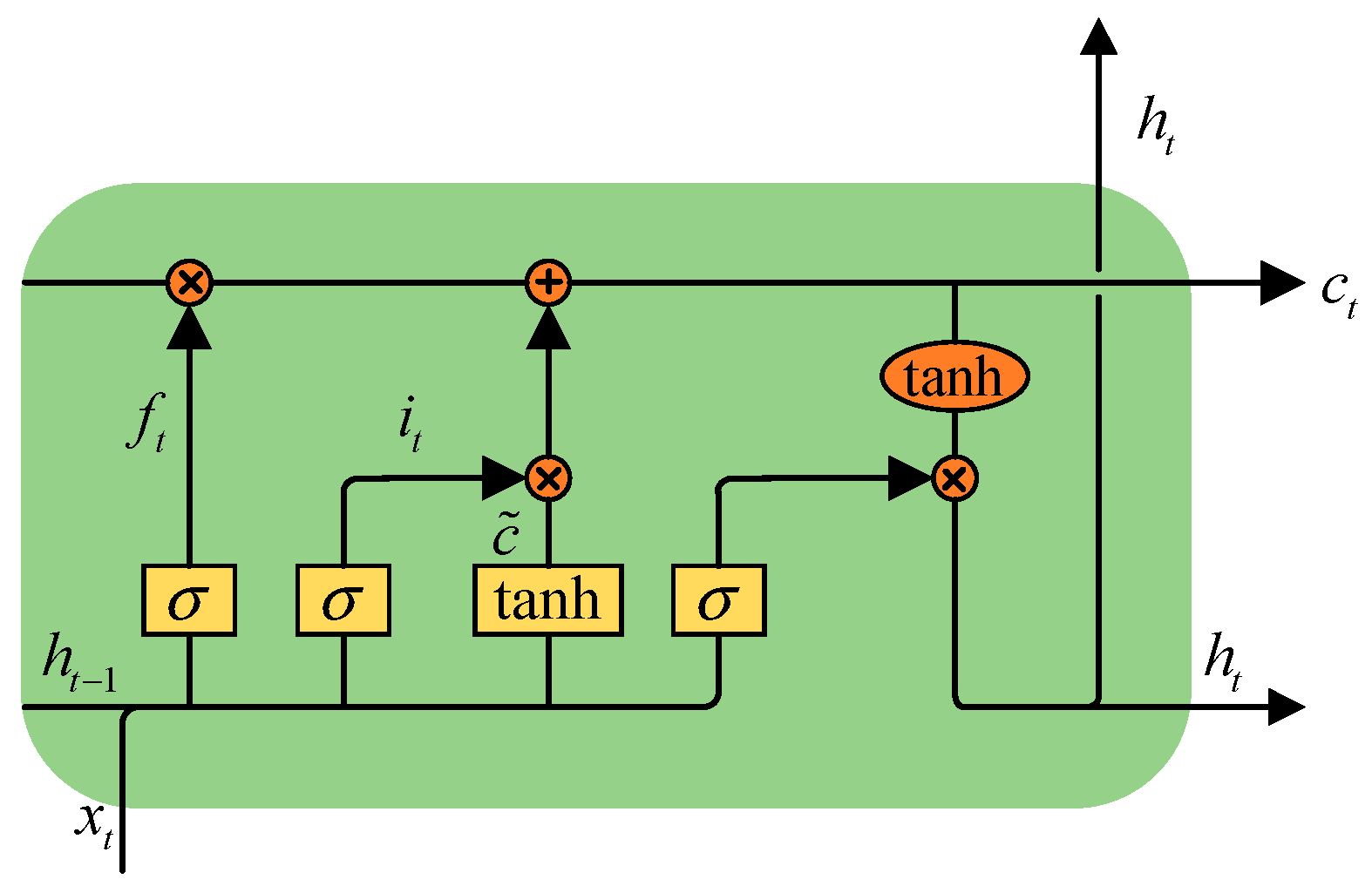


Figure 2‑3. LSTM Cell Unit (Li et al., 2022)

Based on BERT and BiLSTM, a method combining the above two methods, BERT-BiLSTM, is applied to construct the model and predict the sentiment orientation.

Li et al. (2022) also used this combination to conduct SA of Online Chinese Buzzwords. Their approach overcomes the problem of static word vectors by enabling more flexible and context-aware representations. The generated word vectors are then passed to the BiLSTM model for feature extraction, allowing the model to capture both local and global semantic features of the text, with a particular focus on identifying sentiment polarity for sentiment classification. Likewise, BERT-BiLSTM was also used in exploring consumption intent in live e-commerce barrage and in SA about investors and consumers in energy market (Cai et al., 2020; Xiong et al,. 2024). The general model can be illustrated as in ***Figure 2‑4***.

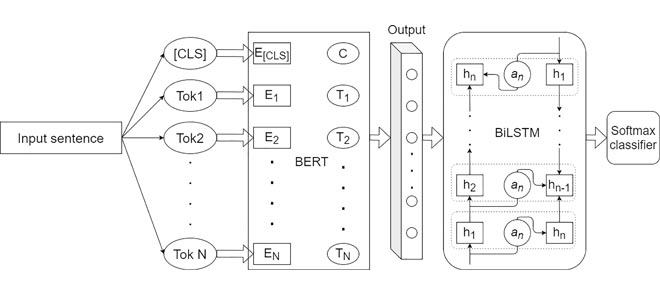


Figure 2‑4 General framework of BERT-BiLSTM (Xiong et al,. 2024)

* 1. ***MVC Framework***

A diagram of a computer

Description automatically generated

Figure 2‑5. MVC architecture. Source: geeksforgeeks.org

The Model-View-Controller (MVC) pattern is a widely adopted architectural framework in software development, particularly in the design of user interfaces and web applications. This pattern separates an application into three interconnected components, each with distinct responsibilities: the Model, the View, and the Controller. This separation facilitates modularization, making the application easier to manage, scale, and test (Sadik Khan et al., 2023).

The Model represents the core data and business logic of the application. It is responsible for managing the data, including retrieval, storage, and manipulation. The Model also defines the rules for how the data can be changed. By isolating the data management from the user interface, the Model allows developers to focus on the business logic without being concerned about how the data is presented to the user.

The View is the component that presents the data to the user. It is responsible for rendering the user interface and displaying the information contained in the Model. The View listens for updates from the Model and reflects any changes to the user interface. This separation of concerns allows developers and designers to work independently on the user interface (UI) without interfering with the underlying data management. By keeping the View separate from the Model, developers can modify the presentation layer without altering the core functionality of the application.

The Controller acts as an intermediary between the Model and the View. It processes user inputs, translates them into requests for the Model, and updates the View accordingly. Essentially, the Controller interprets user actions, such as clicks and key presses, and triggers the necessary updates in the Model or the View. This decoupling of components enables developers to modify one part of the application without necessitating changes in others, thus enhancing maintainability.

One of the significant advantages of the MVC architecture is its ability to promote a clean separation of concerns, which simplifies the development process. By organizing code around distinct components, developers can work on different aspects of the application simultaneously. This modular approach also aids in testing and debugging, as each component can be tested independently. Furthermore, MVC frameworks often come with built-in support for routing, session management, and other common functionalities, allowing developers to focus more on the unique aspects of their applications.

* 1. ***.NET Framework***

The .NET Framework is a software development platform created by Microsoft, designed to support the building and running of applications on Windows. It provides a comprehensive set of libraries, tools, and runtime environments, allowing developers to create desktop, web, and mobile applications. The .NET Framework is based on the Common Language Runtime (CLR), which manages the execution of code and provides features such as garbage collection, type safety, and exception handling. Additionally, the framework includes the .NET Class Library, a vast collection of pre-built classes that handle common programming tasks, from file manipulation to database connectivity.

One of the core advantages of the .NET Framework is its language interoperability. The CLR supports multiple programming languages, including C#, VB.NET, and F#, allowing developers to use the best tool for the job while maintaining seamless integration between different codebases. This flexibility enhances productivity, as developers can reuse code written in different languages within the same application. The framework also offers features such as Integrated Security, which helps developers create secure applications with built-in authentication and encryption mechanisms.

The .NET Framework has evolved over the years, particularly with the introduction of the .NET Core, which is a cross-platform, open-source variant of the original framework. .NET Core enables the development of applications that run not only on Windows but also on macOS and Linux. This shift reflects Microsoft's growing focus on cross-platform support and cloud computing, particularly with the integration of Azure and cloud services. .NET Core, now part of .NET 5 and later versions, provides developers with greater flexibility and performance improvements, such as a smaller runtime footprint and faster application startup times.

Moreover, the .NET Framework has a rich ecosystem of tools and frameworks that simplify development. For example, ASP.NET is a popular framework for building dynamic web applications and services, while Windows Forms and WPF (Windows Presentation Foundation) enable the creation of desktop applications with rich graphical user interfaces (GUI). With the addition of Xamarin, the framework also facilitates mobile application development, allowing developers to write cross-platform mobile apps for iOS and Android using a shared C# codebase (Fowler, 2021).

In conclusion, the .NET Framework remains a powerful, versatile platform for building applications across a wide range of domains. Its continued evolution with .NET Core and its modern variants offers developers enhanced capabilities, better performance, and expanded platform support. Despite the rise of other development platforms, .NET continues to be a cornerstone of enterprise-level application development, thanks to its robust ecosystem, strong community support, and alignment with the latest technological advancements.

* 1. ***Azure Machine Learning***

Azure Machine Learning (Azure ML) is a comprehensive cloud-based platform developed by Microsoft for building, training, and deploying machine learning models. It is designed to support the full machine learning lifecycle, providing tools for data preparation, model development, model training, and deployment, all within a unified environment. Azure ML integrates a wide array of open-source frameworks, such as TensorFlow, PyTorch, Scikit-learn, and others, allowing researchers and practitioners to leverage their preferred libraries and algorithms while benefiting from the scalability and computational power of the cloud. Additionally, it provides access to a rich set of pre-built algorithms and cognitive services, facilitating the development of AI applications in domains such as computer vision, natural language processing, and predictive analytics.

A notable feature of Azure ML is its support for automated machine learning (AutoML), which simplifies the process of model selection and hyperparameter tuning. AutoML allows users with limited expertise in machine learning to automatically experiment with various algorithms, preprocessing techniques, and hyperparameter configurations, thereby accelerating the development cycle. The platform’s AutoML functionality is particularly valuable in addressing the challenge of model selection in complex tasks, where manually tuning models can be time-consuming and computationally expensive. This democratization of machine learning enables more efficient experimentation and model iteration, ultimately reducing the time to insights for organizations and researchers.

Azure Machine Learning also emphasizes collaboration, reproducibility, and model management. It features integrated experiment tracking, version control, and model registry tools, which are essential for ensuring the transparency and reproducibility of machine learning projects. These features allow teams to document the various stages of model development, from data collection and preprocessing to algorithm selection and performance evaluation, ensuring that experiments can be easily revisited or reproduced. Furthermore, Azure ML facilitates seamless deployment of models to various environments, including cloud-based services (e.g., Azure Kubernetes Service) and on-premises systems, which ensures that machine learning models can be scaled and integrated into production systems effectively.

# System Architecture

**A blue oval with black text

Description automatically generated**

Figure 3‑1. Overall architecture. Source: Authors' work

The system architecture of this website is designed to be modular and scalable, leveraging microservices to ensure flexibility and maintainability. The architecture consists of 03 key components, dividing into 02 parts (***Figure 3-1***), that work together seamlessly to deliver a responsive and robust application:

**Web service**

***Product Management (API) Microservice***

This microservice handles all aspects related to products within the system. It is responsible for managing product data, including product details, categories, brands, pricing, images, and reviews. The product management microservice communicates with the database to store and retrieve product-related information, ensuring consistency and availability of data across the system.

***Client Microservice***

The client microservice serves as the interface through which users interact with the system. It provides user-facing features such as browsing products, submitting reviews, managing user profiles, and authenticating users. This microservice is optimized for handling user requests and delivers a smooth and intuitive experience, while also interacting with other microservices to retrieve or update data as needed.

**Machine learning service**

***Machine Learning Microservice***

The machine learning microservice is designed to handle all the machine learning-related tasks. It is responsible for running model inferences, such as sentiment analysis and other predictive tasks, based on user input. By isolating the machine learning functionality into its own microservice, the system ensures that computationally intensive tasks do not interfere with the performance of other services. This service is tightly integrated with Azure Machine Learning, using APIs to send input data and return model predictions.

This architecture allows each component to scale independently, making the system more resilient and capable of handling increasing demands without compromising performance. Additionally, the separation of concerns promotes modular development and facilitates easier updates to individual microservices without affecting the overall system.

* 1. ***Machine learning service***
     1. *Model architecture*



Figure 3‑2. Model architecture. Source: Authors' work

In this project, we use the BERT + BiLSTM model as its architecture is designed to combine the advantages of pre-trained language models and recurrent networks to achieve effective sentiment analysis on text data.

As is shown in ***Figure 3-2***, this model begins with a BERT layer, which has been pre-trained on a vast corpus to understand the context of words based on their surrounding words. When an input text is fed into the model, it is first tokenized into input\_ids and attention\_mask by the BERT tokenizer. The tokenizer converts raw text into token IDs and marks tokens that should be attended to in each input, ensuring that padding tokens are ignored in subsequent computations. This preprocessing step provides a uniform format for the BERT model to process each text sample.

After tokenization, the data passes through the BERT model itself, where it is transformed into contextual embeddings for each token. BERT, as a transformer-based model, captures both local and global dependencies within the text, providing a deep, context-aware representation of each word. The output from BERT consists of hidden states for each token in the input sequence, which hold rich contextual information, capturing relationships among tokens across the entire sequence. This output serves as a high-quality representation of the input text and acts as an input to the next layer in the model architecture.

Following the BERT layer, the embeddings are passed into a BiLSTM layer. This layer is a recurrent network that processes the sequence in both directions—forward and backward—allowing the model to capture dependencies between words regardless of their position in the text. By processing the input from both directions, the BiLSTM layer further enhances the contextual understanding, building a sequential representation that can capture long-range dependencies more effectively. The BiLSTM outputs a hidden state for each token, but only the final hidden state from the last token in the sequence is used for the next stage of processing.

Next, the output from the BiLSTM is passed into a Fully Connected (FC) layer. This layer acts as a classifier by reducing the hidden states from the BiLSTM to a fixed-size vector that corresponds to the sentiment classes: negative, neutral, and positive. The FC layer applies a linear transformation to map the sequential embeddings from the BiLSTM to these sentiment labels. Finally, a softmax function is applied to the output logits to obtain a probability distribution over the sentiment classes, resulting in the model’s sentiment prediction.

The model training is optimized using Cross-Entropy Loss to measure the error between predicted probabilities and true sentiment labels, and Adam optimizer is used to adjust the model’s parameters for better performance. The BERT + BiLSTM model thus benefits from the strengths of both BERT and BiLSTM, achieving a deep understanding of the input text’s contextual and sequential dependencies. This combination enhances the model’s ability to accurately classify sentiment, especially in cases where both word relationships and sequence structure are crucial.

* + 1. *Model deployment*

A diagram of a process flow

Description automatically generated

Figure 3‑3. Model deployment process. Source: Authors' work

The above figure describes the pipeline implemented in the pipeline\_entry\_script.py file, which handles the inference process for deploying the model on Azure Machine Learning. This script orchestrates the steps from receiving user input to returning model predictions:

1. ***Input Data***

The pipeline begins when input data is sent to the model’s API endpoint. This data is typically sent as a JSON payload, containing the text that needs analysis, such as for sentiment detection. This input initiates the model inference process by triggering an HTTP request to the Azure Machine Learning endpoint.

1. ***API Endpoint***

Azure Machine Learning receives the request at the model’s deployed API endpoint. This service layer handles incoming requests, manages security, and ensures the data is passed to the model’s code securely and efficiently. This infrastructure abstracts away the complexities of deployment, allowing users to interact directly with the model through simple API requests.

1. ***Data Parsing***

Upon receiving the request, the code parses the incoming JSON payload to extract relevant fields, such as input\_text. This parsing step ensures that the data is in a suitable format for subsequent processing. If the input\_text field is missing, the system will return an error message, prompting the user to provide valid input data.

1. ***Data Preprocessing***

Once the input text is extracted, the preprocessing step tokenizes the text. Tokenization is essential for transforming the raw text into a structured format that the model can interpret. The tokenizer converts the input into numerical token IDs, applies padding or truncation to standardize the length, and creates an attention mask to indicate which parts of the text should be attended to by the model.

1. ***Prepare Tensors***

With the tokenized data ready, the pipeline converts these tokens and the attention mask into PyTorch tensors. The tensors are then transferred to the appropriate device—either CPU or GPU—for efficient processing. This step prepares the data for rapid inference by ensuring it is optimized for the underlying hardware.

1. ***Model Inference***

The model is now ready to process the input tensors. During this inference step, the model takes in the tensors and generates output in the form of logits. These logits represent the model’s confidence scores for each possible class, providing raw predictions that will be further interpreted in the next step.

1. ***Post-processing***

The logits output by the model are analyzed to determine the predicted class with the highest probability. This predicted class, represented as a numeric label, is mapped to a more understandable sentiment label, such as “positive,” “neutral,” or “negative.” This step ensures that the model’s output is meaningful and easily interpretable by the user.

1. ***Return Response***

Finally, the pipeline returns the sentiment prediction as a JSON-formatted response. If an error occurred during any of the previous steps, an error message is returned instead, indicating the issue. This response completes the inference process, providing the user with an actionable result based on the input text.

* 1. ***Web service***
     1. *Database design*

A screenshot of a computer

Description automatically generated

Figure 3‑4. Database design. Source: Authors' work

The database design (see ***Figure 3-4***) consists of 14 tables that represent data related to both the application and products. These tables are structured to support various functions within the system, from managing user accounts and authentication to organizing and tracking product information. The design ensures that both application-related data (such as user management and access permissions) and product-related data (such as product details, pricing, and reviews) are efficiently stored and can be easily queried. Below is an overview of the tables:

Table 3‑1. Database detail. Source: Authors' work

|  |  |  |
| --- | --- | --- |
| **Name** | **Attribute** | **Description** |
| APP\_Account | Email (PK)  Password  IsActive | Stores user account information, including email, password, and whether the account is active. |
| APP\_Account\_Scope | ScopeId (PK, FK)  AccountId (PK, FK) | Acts as a junction table linking accounts with their permissions (scopes). |
| APP\_Authentication | Id (PK)  Email (FK)  Code  Expired | Stores authentication codes for validating email addresses. This table tracks code expiry and user email. |
| APP\_RefreshToken | RefreshTokenId (PK)  AccountId (FK)  ExpiredTime | Stores refresh tokens, which allow users to renew their session without re-entering login credentials. Each refresh token is linked to a specific account. |
| APP\_Scope | Id (PK)  Value  Description | Defines the different user permissions that control access to specific system resources. |
| Brand | Id (PK)  BrandName  IsArchived | Stores product brand information. The IsArchived attribute marks whether the brand is currently in use or has been discontinued. |
| Category | Id (PK)  CategoryName  CategoryDescription  IsArchived | Contains categories for organizing products. Includes a description for each category and an archive status. |
| Product | ASIN (PK)  SubCategoryId (FK)  BrandId (FK)  OwnedBy (FK)  ProductShortDescription  ProductDetailDescription  ProductName  IsArchived | Stores detailed product information. Links to subcategory, brand, and owner. Includes short and long descriptions of the product and an archive status. |
| ProductColor | Id (PK)  ASIN (FK)  ColorHex  IsArchived | Tracks the color information for each product. The ColorHex value represents the color in hexadecimal format. |
| ProductImage | Id (PK)  ASIN (FK)  ImageUrl  IsArchived | Stores image URLs associated with products. Links to products via ASIN and includes an archive status. |
| ProductPriceHistory | Id (PK)  ASIN (FK)  Price  Discount  DateUpdated | Tracks historical price changes for products. Includes price, any discounts applied, and the date of the update. |
| Reviews | Id (PK)  ASIN (FK)  OwnedBy (FK)  Title  ImageUrl  StarRating  ReviewContent  ReviewDate  Sentiment | Stores customer reviews for products. Includes star rating, review title, content, date, sentiment, and optional image URL. |
| SubCategory | Id (PK)  CategoryId (FK)  SubCategoryName  SubCategoryDescription  IsArchived | Stores subcategory information. Links to a parent category and includes a description of the subcategory. |
| User | Id (PK)  Email (FK)  FirstName  LastName  DateOfBirth  IsArchived | Stores user profile information. Links to user accounts via email and includes personal information and archive status. |

This design supports the management of users and products, while ensuring that the system can easily handle product-related operations, such as managing price histories, handling reviews, and organizing products within categories and subcategories.

* + 1. *Programming model*

In terms of programming, the system follows a clean and structured approach for both the Product Management and Client microservices. Below is an overview of the programming model used:

**Product Management (API)**

The Product Management (API) microservice consists of 05 main components:

***Controllers***

Controllers manage the flow of requests between the client and the services. They handle incoming HTTP requests and route them to the appropriate service methods for processing.

***Entities***

Entities represent the database models and are created from the database schema. These entities define the structure of the data used by the application, corresponding to tables in the database, such as Product, User, Category, etc.

***Profiles***

Profiles serve as a layer that helps in mapping data between entities and the external data structures used by the service, such as DTOs (Data Transfer Objects). They assist in converting complex entity data into formats suitable for response and vice versa.

***Repositories***

Repositories act as intermediaries between services and entities. They manage the data access logic, ensuring that the application can interact with the database to perform CRUD (Create, Read, Update, Delete) operations. The services will call the repository to handle data-related tasks, and the repositories will interact with the database entities.

***Services***

Services define the business logic of the application. They perform the necessary actions by using repositories and other components, such as entities and profiles. For example, a product service might handle operations like adding a new product or updating product details.

The interaction flow of this microservice can be envisioned as in the following figure:

A diagram of services and profiles

Description automatically generated

Figure 3‑5. API microservice. Source: Authors’ work

***Controllers → Services***

The controller receives incoming HTTP requests, which it passes to the corresponding service. The service is responsible for handling the business logic.

***Services → Repositories***

Within the service, business logic is applied. For tasks like adding a product, the service will call the repository to interact with the database.

***Repositories → Entities***

The repository uses the entity models to perform CRUD operations. The repository translates requests into actions on the database tables using the entity definitions.

***Services → Profiles***

In some cases, the service may also call the Profile layer to map entities to DTOs (or vice versa). This is especially useful when transforming data before sending a response.

***Profiles → Entities (optional)***

If the profile is transforming data from entities into a suitable format (e.g., DTOs), it may call the entity layer.

***Controllers → Responses (External Client)***

Finally, after business logic is processed, the controller sends a response to the client. This response is typically in a transformed format using the profile (DTOs).

**Client (Frontend)**

A screenshot of a computer

Description automatically generated

Figure 3‑6. Client microservice. Source: Authors' work

As can be seen in ***Figure 3-6***, the client-side of the application is developed using Angular, a popular framework for building single-page applications. Below is a detailed description of the Client microservice structure, based on the provided project structure:

***Folder and File Structure***

*Root directory:*

* node\_modules/: This directory contains all the external libraries and dependencies required for the project. It is managed using NPM (Node Package Manager) and includes libraries like Angular and other third-party packages.
* src/: The source folder holds all the application-specific code. This is where the core of the application logic resides, and it is divided into various subdirectories and files.

*Configuration files:*

* .editorconfig: This configuration file is used to define and maintain consistent coding styles across different editors and IDEs (e.g., Visual Studio). It ensures a uniform code style across all developers working on the project.
* .gitignore: This file specifies which files and directories should be ignored by Git when committing changes. Typically, files like node\_modules/, build artifacts, and local environment configurations are excluded.
* angular.json: This file contains the configuration settings for Angular CLI, allowing customization of build and serve commands. It controls aspects like build optimization, file replacements, and environment settings.
* karma.conf.js: This file configures Karma, a testing tool used to run unit tests and integration tests for the Angular application. It specifies the test framework, browser options, and other testing configurations.
* package.json: This file lists all the required dependencies for the project. It includes both project-specific and third-party libraries. If any new libraries are added, they are declared in this file.

*node\_modules/:* This directory is created by running npm install and contains all the installed packages and dependencies. These packages are managed through NPM and can be updated or removed as needed.

*src/:* The src/ directory holds all the source code for the application, including various components, services, models, and routing logic.

***Key Subdirectories and Files within src/***

*src/app/:* This directory contains the main application logic. Angular CLI typically generates this structure as a starter template for building Angular applications. It is here that the main components and services are organized.

* app.component.html: This file contains the HTML markup for the main component, forming the view layer of the application that users interact with. It defines the structure and layout of the user interface.
* app.component.ts (Component Class): This file contains the business logic for the main component. It is similar to a controller in other frameworks. Here, developers handle data manipulation, event handling, and interaction with services or APIs.
* app.component.css: This file defines the CSS styles for the main component. It allows developers to apply styles specific to this component, making it visually distinct.
* app.module.ts: This is the root module of the Angular application. It configures and imports all necessary modules, components, and services, initializing the application. It declares the components and tells Angular how to bootstrap the application.
* app-routing.module.ts: This module is responsible for routing. It defines the paths for navigating between different views or pages in the application. By using Angular’s router, developers can create dynamic and single-page applications that update the UI without a full page reload.
* app.component.spec.ts: This file contains the unit tests for the app.component.ts class. It allows developers to test the logic, functionality, and behavior of the component in isolation.

By following this model, both the backend (Product Management microservice) and the frontend (Client microservice) are organized in a modular way, facilitating clear separation of concerns, and making it easier to maintain and scale each component independently.

# Results and Discussion

* 1. ***Website appearance***

The website user interface was designed in 2 flows for customers and for admin, whose mockups can be accessed via the link: [Lokeytech Figma](https://www.figma.com/proto/Zd1b6rQs3EK8W6mKdO3Nyt/Wireframe-%2B-Mockup?node-id=0-1&t=nvjKcKOBTREzeVLt-1).

A screen shot of a login form

Description automatically generated

Figure 4‑1 UI for sign-in. Source: Authors’ work

A screenshot of a login screen

Description automatically generated

Figure 4‑2 UI for sign-up. Source: Authors’ work

A screenshot of a computer

Description automatically generated

Figure 4‑3 UI for admin - Overview page. Source: Authors’ work

A screenshot of a computer

Description automatically generated

Figure 4‑4 UI for admin - Page for product list. Source: Authors’ work

When admin is in need of editing the product information, the pop up appears to fill in necessary field as in ***Figure 4‑5.***

A pink rectangular box with black text

Description automatically generated

Figure 4‑5 Pop-up for product information editing. Source: Authors’ work

A screenshot of a computer

Description automatically generated

Figure 4‑6 UI for admin - Transaction page. Source: Authors’ work

A screenshot of a computer

Description automatically generated

Figure 4‑7 UI for admin - Report page. Source: Authors’ work

A screenshot of a cell phone advertisement

Description automatically generated

Figure 4‑8 Homepage. Source: Authors’ work

A screenshot of a phone

Description automatically generated

Figure 4‑9 Special offers section. Source: Authors’ work

A screenshot of a phone

Description automatically generated

Figure 4‑10 Product detail page. Source: Authors’ work

A screenshot of a phone

Description automatically generated

Figure 4‑11 Product category page. Source: Authors’ work

A screenshot of a computer

Description automatically generated

Figure 4‑12. Shopping cart section. Source: Authors’ work

* 1. ***Model evaluation***

Table 4‑1. Classification report. Source: Authors' work

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ***precision*** | ***Recall*** | ***F1-score*** | ***Support*** |
| ***Negative*** | 0.53 | 0.65 | 0.58 | 529 |
| ***Neutral*** | 1.00 | 0.0 | 0.01 | 246 |
| ***Positive*** | 0.56 | 0.69 | 0.02 | 545 |
| ***Accuracy*** |  |  | 0.55 | 1320 |
| ***Macro.avg*** | 0.7 | 0.45 | 0.4 | 1320 |
| ***Weighed.avg*** | 0.63 | 0.55 | 0.49 | 1320 |

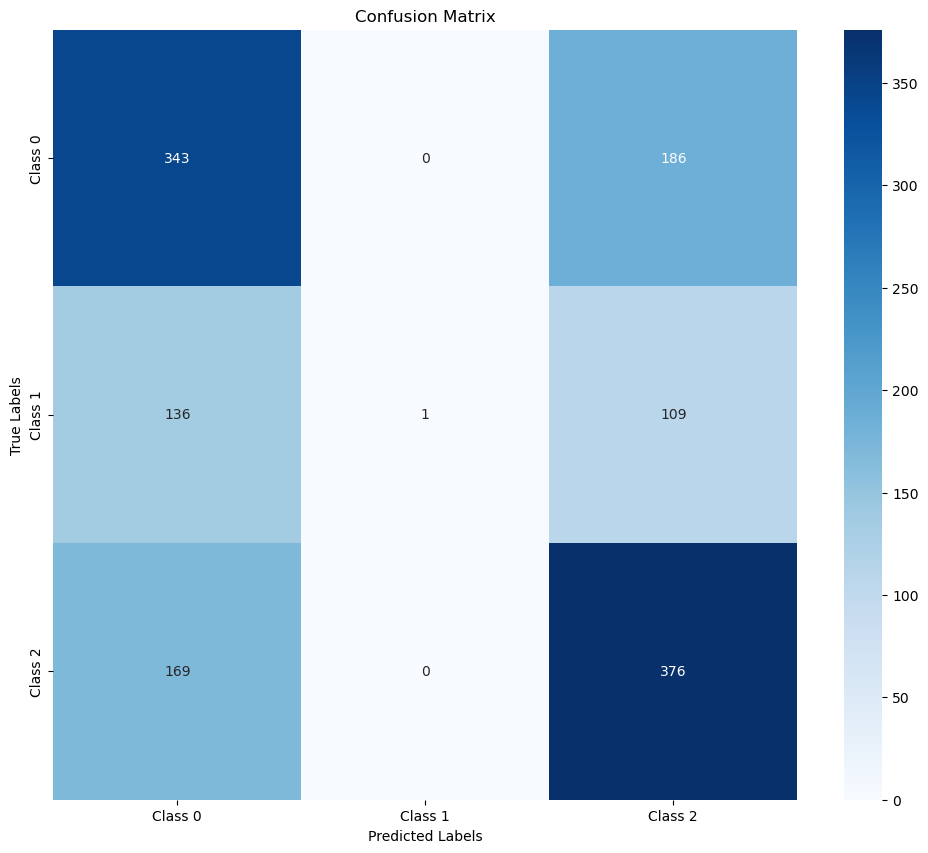


Figure 4‑13. Confusion matrix. Source: Authors' work

Class 0 (Negative) shows decent performance, with 346 true positives out of 529 total instances. However, the model misclassifies 147 negative instances as positive and 36 as neutral, indicating some overlap with the other sentiment classes. This class has a precision of 0.61, suggesting that when the model predicts a negative sentiment, it is correct 61% of the time. The recall of 0.65 implies that the model identifies 65% of actual negative instances. The F1-score of 0.63 reflects a balance between precision and recall, but there is room for improvement to reduce the misclassification rate.

Class 1 (Neutral) is where the model struggles most, as shown by only 98 correctly classified neutral instances out of 246. The confusion matrix highlights that 89 neutral instances are mistakenly classified as negative, and 59 as positive. The precision of 0.55 indicates that only 55% of neutral predictions are accurate, and the low recall of 0.40 means that only 40% of actual neutral instances are captured correctly by the model. An F1-score of 0.46 underscores that the model's effectiveness in identifying neutral sentiments is limited. This poor performance suggests that the model may not be distinguishing neutral sentiments well from the other two classes, potentially due to insufficient or overlapping features that make neutral cases harder to identify.

Class 2 (Positive) has the strongest performance among the three classes. With 370 true positives out of 545 instances, the model performs fairly well in identifying positive sentiments. The confusion matrix indicates that 132 positive cases are misclassified as negative, and 43 as neutral, which is better compared to the misclassification rates for the other classes. The precision of 0.64 and recall of 0.68 highlight that the model is reasonably reliable in predicting positive sentiment. The F1-score of 0.66 shows a well-balanced precision and recall, reinforcing that the model can more accurately identify positive instances compared to negative or neutral.

Overall Metrics show that the model's accuracy stands at 0.62, indicating that it correctly classifies 62% of all sentiment instances. While this is a moderate result, it leaves considerable room for enhancement, particularly with neutral sentiments. The macro average precision, recall, and F1-score hover around 0.58-0.60, pointing out that performance is inconsistent across the three sentiment categories, heavily impacted by the poor classification of neutral instances. The weighted average precision, recall, and F1-score align with the overall accuracy, showing that the imbalanced support for different classes influences these averages, particularly with the neutral class dragging down the overall scores.

# Conclusion and Future Works

This project underscores the growing role of UGC, particularly customer reviews, in shaping the landscape of e-commerce. As online retail continues to thrive, businesses are increasingly turning to customer feedback to better understand consumer preferences, improve products, and refine marketing strategies. The sentiment analysis model developed in this project, utilizing an advanced hybrid BERT - BiLSTM approach, effectively processes and categorizes large volumes of customer reviews, providing businesses with valuable insights into customer sentiment.

By creating a realistic e-commerce platform with a comment collection system and integrating an advanced sentiment analysis model, this project demonstrates how businesses can leverage machine learning to automatically interpret customer feedback. The ability to quickly categorize feedback into positive, neutral, or negative sentiments allows businesses to respond more effectively to customer concerns, make data-driven improvements, and enhance customer satisfaction.

The insights gained from sentiment analysis enable companies to make informed decisions regarding product development, customer service, and marketing strategies. This project highlights the power of machine learning in transforming raw customer reviews into actionable data, allowing businesses to stay ahead in a highly competitive market by better aligning their offerings with customer needs and expectations.

In the future, this approach could be expanded to other product categories or enhanced with additional features, such as the analysis of multimedia content or the integration of real-time sentiment tracking. As e-commerce continues to evolve, the ability to harness the power of sentiment analysis will be a key driver in improving customer experiences, fostering brand loyalty, and ensuring long-term business success.

# References

Cai, R., Qin, B., Chen, Y., Zhang, L., Yang, R., Chen, S., & Wang, W. (2020). Sentiment analysis about investors and consumers in energy market based on BERT-BiLSTM. *IEEE access*, *8*, 171408-171415.

Clark, J. (2024). 18 Online review statistics Every marketer should know. Search Engine Journal. [https://www.searchenginejournal.com/online-review-statistics/329701/](https://www.google.com/url?q=https://www.searchenginejournal.com/online-review-statistics/329701/&sa=D&source=docs&ust=1731247734843544&usg=AOvVaw13NTstl5Pe2V_TVMGj_cBK)

Dosa, H. (2022). Brand Rated: “Nine out of ten customers read reviews before buying a product.” GlobeNewswire News Room. [https://www.globenewswire.com/news-release/2022/01/13/2366090/0/en/Brand-Rated-Nine-out-of-ten-customers-read-reviews-before-buying-a-product.html](https://www.google.com/url?q=https://www.globenewswire.com/news-release/2022/01/13/2366090/0/en/Brand-Rated-Nine-out-of-ten-customers-read-reviews-before-buying-a-product.html&sa=D&source=docs&ust=1731247734842872&usg=AOvVaw2x51bYRTqvrLXw3iaLSDUx)

Fanni, S.C., Febi, M., Aghakhanyan, G., Neri, E. (2023). Natural Language Processing. In: Klontzas, M.E., Fanni, S.C., Neri, E. (eds) Introduction to Artificial Intelligence. Imaging Informatics for Healthcare Professionals. Springer, Cham. [https://doi.org/10.1007/978-3-031-25928-9\_5](https://www.google.com/url?q=https://doi.org/10.1007/978-3-031-25928-9_5&sa=D&source=docs&ust=1731247734837348&usg=AOvVaw1YSqpal5kjyLjQCMO-DUaU)

Fowler, M. (2012). Patterns of enterprise application architecture. Addison-Wesley.

H. Huang, A. A. Zavareh and M. B. Mustafa, "Sentiment Analysis in E-Commerce Platforms: A Review of Current Techniques and Future Directions," in IEEE Access, vol. 11, pp. 90367-90382, 2023, doi: 10.1109/ACCESS.2023.3307308

Khan, S., & Khanam, A. T. (2023). Study on MVC Framework for Web Development in PHP. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 414-419.

Li X, Lei Y, Ji S. BERT- and BiLSTM-Based Sentiment Analysis of Online Chinese Buzzwords. *Future Internet*. 2022; 14(11):332. <https://doi.org/10.3390/fi14110332>

Xiong, Y., Wei, N., Qiao, K., Li, Z., & Li, Z. (2024). Exploring Consumption Intent in Live E-Commerce Barrage: A Text Feature-Based Approach Using Bert-BiLSTM Model. *IEEE Access*.

Yang C, Huang C. Natural Language Processing (NLP) in Aviation Safety: Systematic Review of Research and Outlook into the Future. Aerospace. 2023; 10(7):600. [https://doi.org/10.3390/aerospace10070600](https://www.google.com/url?q=https://doi.org/10.3390/aerospace10070600&sa=D&source=docs&ust=1731247734838126&usg=AOvVaw3Pdp1qvN1YJIwUo8WaZsRE)

Zhou, L. (2024). Online Review Statistics: The Definitive list (2024 data). Luisa Zhou. [https://luisazhou.com/blog/online-review-statistics/](https://www.google.com/url?q=https://luisazhou.com/blog/online-review-statistics/&sa=D&source=docs&ust=1731247734844067&usg=AOvVaw0Mm8x1ljqQX4L1NqpKkl55)

Zubair Nawaz et al., "Role of Artificial Neural Networks Techniques in Development of Market Intelligence: A Study of Sentiment Analysis of eWOM of a Women's Clothing Company", Journal of Theoretical and Applied Electronic Commerce Research, vol. 16, no. 5, pp. 1862-1873, 2021.

# Appendix

**Materials**

All materials can be accessed via this link: [Google Drive](https://drive.google.com/drive/folders/124OLicJ3sOWBTeD94OTCZKb-t-l1MLpe?usp=drive_link)

* Word: [Link](https://docs.google.com/document/d/1onVRPxTZAFkPv3i5ctNLYCutj-V9QzjL/edit?usp=drive_link&ouid=103292648504092090526&rtpof=true&sd=true) and/or [Link](https://drive.google.com/file/d/1ekm3KLY0nEzjKoBf2daQPzMjNLTWHYj9/view?usp=drive_link)
* Excel: [Link](https://docs.google.com/spreadsheets/d/1c8wvDqni_0SaAp_8ZYmEQKVWtKC05wV3CbSofxYxEz4/edit?usp=drive_link)
* Sourcecode: [Link](https://drive.google.com/drive/folders/1LfxQgNVVG8gMrtOxbWCaqdTl2A0s3Ztk?usp=drive_link)
* Video: [Link](https://drive.google.com/file/d/1Jk-F0yKsgbB-X6ktcd5nFiqtLSC0DOFl/view?usp=drive_link)
* Slides: [Link](https://drive.google.com/file/d/13iYwvRYIVvqopAMSSJRZ5h5TR73Q7GoH/view?usp=drive_link)