

عبدالحافظ العصار

Sentiment Analysis for Mobile App Reviews A Comparative Study.pdf

 Midocean University

Document Details

Submission ID

trn:oid::2945:337599096

Submission Date

Dec 16, 2025, 2:20 AM GMT+2

Download Date

Dec 16, 2025, 10:45 AM GMT+2

File Name

Sentiment Analysis for Mobile App Reviews A Comparative Study.pdf

File Size

2.7 MB

82 Pages

19,843 Words

118,640 Characters

23% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

-  **36 AI-generated only 23%**
Likely AI-generated text from a large-language model.
-  **0 AI-generated text that was AI-paraphrased 0%**
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?

The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?

Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.





Sentiment Analysis for Mobile App Reviews: A Comparative Study

Prepared By

Name: Sultan Khalaf Alotaibi

ID No: 222000226

Name: Abdel Hafez Adeeb Alassar

ID No: 251002398

Submitted in fulfillment of the requirements for the degree Master of
Science in Artificial Intelligence

At

College of Informatics

Midocean University

UAE

Supervisor Name: Dr. Wesam Ahmed

College of Informatics, Midocean University, Comoros

Email: wesamahmed@midocean.edu.km

TABLE OF CONTENTS

TABLE OF CONTENTS.....	1
Introduction	4
1.1 Introduction and Background	5
1.2 Problem Statement.....	5
1.3 Research Objectives	6
1.4 Research Contributions	6
1.5 Thesis Organization	7
Overview: Background	8
2.1 Introduction to sentiment Analysis.....	9
2.2 Levels of Sentiment	9
2.2.1 Document-Level Sentiment Analysis	10
2.2.2 Sentence-Level Sentiment Analysis.....	10
2.2.3 Phrase-Level Sentiment Analysis	10
2.2.4 Aspect-Level Sentiment Analysis	11
2.3 Feature Extraction	11
2.4 Term Frequency (TF) and TF-IDF.....	11
2.5 Bag of Words (BoW).....	12
2.6 Word Embeddings	12
2.6.1 Word2Vec.....	12
2.6.2 GloVe (Global Vectors for Word Representation)	13
2.6.3 FastText.....	13
2.6.4 ELMo (Embeddings from Language Models)	13
2.7 Sentiment Analysis Approaches.....	14
2.7.1 Machine Learning Approach in Sentiment Analysis	14
2.7.1.1 Supervised Learning.....	15
2.7.2 Machine learning Models.....	15
2.7.2.1 Naive Bayes.....	15
2.7.2.2 Decision Tree	16
2.7.2.3 Support Vector Machines (SVM)	18
2.7.2.4 Random Forest	19
2.7.3 Deep Learning Approach in Sentiment Analysis.....	20
2.7.4 Deep Learning Models Approaches to Sentiment Analysis.....	21
2.7.4.1 Convolutional Neural Networks (CNNs)	21
2.7.4.2 Recurrent Neural Networks (RNNs)	21
2.7.4.3 Long Short-Term Memory (LSTM).....	22

2.7.4.4 Gated Recurrent Unit (GRU)	23
2.7.5 Large Language Models (LLMs Approaches to Sentiment Analysis	23
2.7.5.1 Introduction to Large Language Models (LLMs).....	23
2.7.5.2 BERT (Bidirectional Encoder Representations from Transformers).....	24
2.7.5.3 GPT (Generative Pre-trained Transformer).....	25
2.7.5.4 RoBERTa (Robustly Optimized BERT Approach	25
2.7.5.5 DistilBERT.....	26
2.7.5.6 XLNet.....	26
2.8 Evaluation Matrices	28
2.9 Summary	29
Literature Review	30
3.1 Machine Learning Models.....	31
3.2 Deep Learning Models	35
3.3 Large Language Models	39
Proposed Approach	48
4.1 Introduction	49
4.2 Dataset Information	50
4.3 Preprocessing.....	50
4.4 Machine Learning Models.....	52
4.4.1 Support Vector Machine (LinearSVC).....	52
4.4.2 Logistic Regression	53
4.4.3 Random Forest Classifier.....	53
4.4.4 LightGBM Classifier	54
4.4.5 Naïve Bayes Classifier.....	55
4.5 Large Language Model.....	55
4.5.1 DistilBERT-Base-Uncased	55
4.5.2 DeBERTa-v3-Base	57
4.5.3 RoBERTa-Base	58
4.5.4 BERT-Base-Uncased.....	59
4.5.5 ELECTRA-Large-Discriminator	60
4.6 Deep Learning Models	61
4.6.1 Bi-LSTM.....	61
4.6.2 CNN + BiGRU Hybrid Model.....	62
4.7 Summary.....	64
Experimental Results and Discussion	65
5.1. Introduction	66

5.2. Results of Individual Models.....	66
5.2.1. Traditional Machine Learning Models.....	66
5.2.2. Deep Learning Models.....	68
5.2.3. Results of Transformer-Based Models.....	69
5.3. Discussion of Findings.....	74
5.4. Limitations of the Proposed Work	75
5.5. Summary of the Chapter	75
References	76

Chapter 1

Introduction

1.1 Introduction and Background

With the rapid expansion of mobile applications in the digital marketplace, user comments have emerged as a valuable source of data both for stakeholders and developers [1]. Out of all the different kinds of user-generated data, app reviews are crucial for influencing new downloads, forming user perception, and directing product enhancements for the business. Therefore, it is becoming increasingly important to analyze these reviews systematically [2]. The detection of sentiments from textual content is an area of natural language processing (NLP) that allows for the automated identification and classification of opinions expressed in text data [3]. Sentiment analysis helps extract useful information from massive amounts of user reviews by classifying them into positive, negative, and neutral sentiment [4]. This can help developers enhance overall user satisfaction, address long-term issues, and gain insight into how people perceive a product.

For the analysis, processing, and interpretation of unstructured text data, NLP methods offer robust solutions. NLP models can transform unprocessed text input into meaningful structures suitable for machine learning and deep learning tasks, using methods such as tokenization, lemmatization, and vectorization. These processes enable scalable and uniform analysis of feedback, well beyond human review capabilities [5].

This project involves applying sentiment analysis to user reviews of the mobile game Mobile Legends: Bang Bang by comparing different sentiment classification methods ranging from classical machine learning to deep learning architectures and transformer-based models. We aim to identify the most effective methods for understanding user sentiment in reviews of mobile apps. This study's findings can provide valuable insights to app developers, marketers, and UX designers seeking to enhance app performance through user feedback.

1.2 Problem Statement

Mobile applications have become an integral part of daily life, offering services that range from communication and entertainment to education and finance. Despite their widespread adoption, apps often face challenges such as performance issues, frequent crashes, poor user interface design, and unresponsive updates. Negative user experiences, such as these, can lead to low ratings, poor retention, and ultimately, reduced market success. For developers and stakeholders, understanding user concerns quickly and effectively is critical to maintaining app quality and user satisfaction.

User reviews in app stores provide rich, unstructured feedback that reflects real user experiences and sentiments. However, manually analyzing thousands of reviews is impractical. Sentiment analysis offers a scalable solution by automatically classifying user opinions as positive, negative, or neutral, enabling developers to identify and prioritize areas that require improvement.

While various approaches to sentiment analysis exist, challenges remain. There is a lack of studies that systematically compare traditional machine learning, deep learning, and transformer-based models in the specific context of mobile app reviews.

This study aims to fill this gap by applying and comparing multiple sentiment analysis techniques to user reviews of the Mobile Legends app. By identifying the most effective models and preprocessing strategies, this research seeks to support developers in leveraging user feedback for product improvement.

1.3 Research Objectives

The primary goal of this study is to conduct a comprehensive sentiment analysis of user reviews for the Mobile Legends app and compare the performance of various classification models.

1.4 Research Contributions

This study has the following contributions based on the research objectives:

1. Data Preprocessing Framework: Developed a robust and efficient preprocessing pipeline to clean up and prepare the app review data, with challenges being on noise, slang, emojis, inconsistent formatting on user-generated data.
2. Model implementation and evaluation: Multiple sentiment analysis methods, such as those based on conventional machine learning methods e.g., SVM, Random Forest, deep learning e.g., LSTM, CNN and transformer e.g., BERT methods, have been implemented and evaluated rigorously with regard to mobile app review classification.
3. Comparison with current models: performed a thorough analysis of the performance of alternative sentiment analysis models and their strengths, weaknesses, and metrics as defined by accuracy, F1-score, and computational efficiency.
4. Recommendation of Best Approach: Established most efficient and feasible model to making review analysis of mobile apps, actionable suggestions to mobile app

developers and researchers to gain insight into user sentiment to enhance the experience of their mobile apps.

5. **Practical Insights:** Equipped an organized approach, along with the results that could be used in further research and implementation of sentiment analysis in the mobile application world, specifically in games such as the Mobile Legend.

1.5 Thesis Organization

Chapter 2: provides an overview of sentiment analysis methods, including machine learning, deep learning, and transformer-based approaches, as well as feature extraction techniques.

Chapter 3: reviews the existing literature on sentiment analysis, identifies its strengths and weaknesses, and highlights the research gaps addressed in this study.

Chapter 4: outlines the research framework, including data preparation, model development, and evaluation strategies for sentiment classification.

Chapter 5: presents the performance results of all models using evaluation metrics and discusses their practical implications.

Chapter 2

Overview: Background

2.1 Introduction to sentiment Analysis

Sentiment analysis (SA) is a process used in natural language processing (NLP) that helps computers understand and interpret human emotions or opinions expressed in text. It is primarily used to determine whether a piece of text such as a review, comment, or post conveys a positive, negative, or neutral sentiment. This process works by analyzing the words, phrases, and context in which they are used to determine the writer's emotional state or attitude. The reason sentiment analysis is critical is that it provides businesses, researchers, and developers with a way to make sense of large amounts of feedback without having to read every single sentence. For example, companies can use it to measure customer satisfaction with their products or services by scanning customer reviews. In mobile app development, sentiment analysis helps developers understand what users like or dislike about an app by analyzing app store reviews. This can help improve the app, resolve common issues, and enhance the overall user experience. It is also used in various other fields, such as politics, where it can analyze public opinion on a policy, or in finance, where it helps track market sentiment by examining news or social media. As more people write online, sentiment analysis becomes increasingly useful for quickly transforming text data into valuable insights.

2.2 Levels of Sentiment

Sentiment analysis can be performed at various levels of detail depending on the nature and complexity of the text. The primary levels include document level, sentence level, phrase level, and aspect level, each offering a different depth of insight. Figure 2.1 illustrates these levels and how they are used to capture sentiment from general to more specific parts of a text.

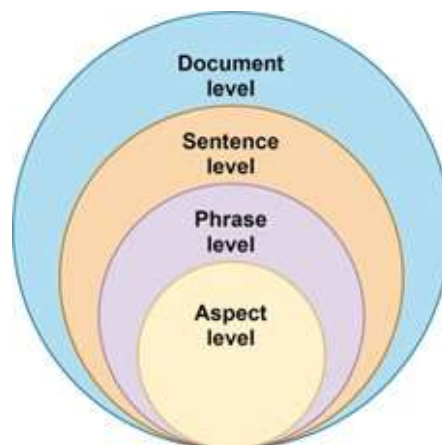


Figure 0.1 Levels of Sentiment Analysis

2.2.1 Document-Level Sentiment Analysis

Document-level sentiment analysis is the most basic form of sentiment analysis, where the entire document such as a review, article, or feedback post is treated as one unit. The goal is to classify the overall sentiment of the document as positive, negative, or neutral. This approach works best when a single opinion or feeling is consistently expressed throughout the text, as commonly seen in product reviews or app feedback, where the writer focuses on a single topic. For example, if a user writes a review like “This app is amazing! I love how smooth it runs,” the entire document reflects a positive sentiment. However, it may not perform well when the document contains mixed opinions about different parts of the app, as it averages everything into a single label [6].

2.2.2 Sentence-Level Sentiment Analysis

In sentence-level sentiment analysis, the text is broken down into individual sentences, and each sentence is analyzed separately to determine its sentiment. This is particularly useful when a review or document contains multiple sentences that express different views. For instance, in a review like “The app design is great. But it crashes often,” the first sentence is positive, and the second is negative. Sentence-level analysis allows us to understand such variations in sentiment within a single document. It offers more detailed insights than document-level analysis, but may still struggle with complex sentences or when multiple sentiments are conveyed in a single sentence [7].

2.2.3 Phrase-Level Sentiment Analysis

Phrase-level sentiment analysis, also known as fine-grained sentiment analysis, focuses on smaller parts of a sentence, such as phrases or expressions that carry emotional weight. These could include adjectives, verbs, or short noun phrases that reflect opinion, such as “very useful,” “too slow,” or “extremely satisfied.” This level of analysis is more detailed and can detect sentiment within phrases, even if the sentence as a whole is neutral. For example, in the sentence “I expected more from the update, although the new features are quite useful,” the phrase “expected more” carries a negative sentiment. In contrast, “quite useful” reflects a positive one. Phrase-level analysis is more challenging but valuable when trying to capture nuanced opinions [8].

2.2.4 Aspect-Level Sentiment Analysis

Aspect-level sentiment analysis is the most detailed and advanced level of analysis. It involves identifying specific features or components called “aspects” of a product or service that users are commenting on and then determining the sentiment associated with each aspect. This is especially helpful in reviews where multiple features are discussed. For instance, in the sentence “The UI is clean, but the loading time is too long,” the model identifies two aspects: “UI” (positive sentiment) and “loading time” (negative sentiment). This level of analysis is fundamental in mobile app reviews, where users frequently discuss usability, speed, bugs, or battery consumption. Aspect-based sentiment analysis provides developers with precise feedback to improve specific aspects of their app, rather than interpreting reviews in a general way [9] [10].

2.3 Feature Extraction

In natural language processing (NLP), feature extraction is the process of transforming raw text into a structured format that machines can understand and analyze. Since algorithms cannot interpret human language in its natural form, it is necessary to extract features that represent the essential characteristics of the text. These features can include word frequency, word importance, grammatical patterns, and sentence structures. Effective feature extraction plays a crucial role in enhancing the performance of sentiment analysis models by enabling them to identify emotional tones, patterns, and relevant information within textual data [11].

2.4 Term Frequency (TF) and TF-IDF

Term Frequency (TF) refers to the number of times a word appears in a document. For example, in the sentence “The app is fast, smooth, and fast-loading”, the term “fast” appears twice, so it has a higher term frequency. This method assumes that more frequent words are more important. However, common words like “is,” “the,” or “app” may not carry much sentiment. Therefore, TF-IDF (Term Frequency–Inverse Document Frequency) is used to weigh words by their uniqueness across a collection of documents. TF-IDF increases the importance of rare but meaningful words and decreases the importance of standard terms that appear in many documents. For instance, the term “crashes” in app reviews might have a high TF-IDF score because it's strongly linked to negative sentiment and doesn't appear in every review [12].

2.5 Bag of Words (BoW)

The Bag of Words (BoW) model is a technique where a document is represented as a collection (or "bag") of its words, disregarding grammar and word order. It only considers whether known words occur in the text and how many times they appear. For example, if we have two reviews:

- *Review 1: "The app is good and fast."*
- *Review 2: "Fast performance but not good support."*

BoW will create a vocabulary of all words (e.g., "app," "good," "fast," "performance," "support") and count the occurrences of each word in both reviews. Although it ignores sentence structure, BoW is useful for capturing general sentiment patterns, especially when paired with algorithms like Naive Bayes or SVM [13].

2.6 Word Embeddings

In natural language processing (NLP), word embeddings are techniques used to represent words as numerical vectors, enabling machines to understand and work with them. Unlike simple models like Bag of Words or TF-IDF that only consider word frequency, word embeddings capture the semantic meaning and context of words [14][15]. These vector representations enable similar words (such as "happy" and "joyful") to occupy similar positions in a high-dimensional space. The idea is that words with similar meanings will be close together mathematically, even if they are not identical in meaning. Word embeddings have greatly improved the performance of models in sentiment analysis, machine translation, and question answering [16].

2.6.1 Word2Vec

Word2Vec is one of the earliest and most influential word embedding models. It was developed by Google in 2013 and learns the meaning of words based on their context. Word2Vec has two training methods: CBOW (Continuous Bag of Words), which predicts a word given its surrounding words, and Skip-Gram, which does the opposite—predicts surrounding words given a word. For example, in the sentence "The app is easy to use and very fast", the model learns that words like "easy," "use," and "fast" often appear together and thus likely share related meanings. The result is a vector representation for each word that reflects its usage in real-world language. Word2Vec vectors can capture relationships, such as king - man + woman = queen [17].

2.6.2 GloVe (Global Vectors for Word Representation)

GloVe is another word embedding model developed by Stanford University. Unlike Word2Vec, which learns word meaning from local context (neighboring words), GloVe is trained using global statistics from the entire text corpus. It builds a word co-occurrence matrix, meaning it looks at how often words appear together across all documents. Then it uses mathematical techniques to reduce that matrix into useful word vectors. For example, in millions of reviews, if the word “battery” frequently appears near “long” and “life,” GloVe will place them close in vector space. This global context helps the model understand more nuanced relationships. GloVe is especially effective for large-scale corpora like Wikipedia or Twitter data [18].

2.6.3 FastText

FastText was introduced by Facebook AI Research and is an extension of Word2Vec. It improves upon Word2Vec by considering not just whole words, but also subword information (e.g., character n-grams). This means FastText can understand the meanings of morphologically similar words or even words it has never seen before. For example, if the model has seen the words “running” and “runner,” it can use their shared parts (“run,” “ing,” “er”) to guess the meaning of “runs” or “runnable.” This is extremely useful for processing noisy data like social media text or reviews that include typos or slang. FastText supports out-of-vocabulary words, which many other embedding models cannot handle [19].

2.6.4 ELMo (Embeddings from Language Models)

ELMo, developed by the Allen Institute for AI, is a contextualised word embedding model. This means it generates different embeddings for the same word, depending on the sentence in which it's used. Unlike Word2Vec or GloVe, which assign a single fixed vector to each word, ELMo utilises a deep bidirectional LSTM (a type of neural network) to generate word vectors that comprehend the surrounding context. For example, the word “bank” in “river bank” and “savings bank” will have different vector representations in ELMo. This makes ELMo more powerful and accurate, especially for tasks that require understanding word meaning in different contexts, such as sentiment analysis or question answering [20].

2.7 Sentiment Analysis Approaches

Sentiment Classification approaches are broadly categorized into Machine Learning-based Approaches and Lexicon-based Approaches, presented in Figure 2.2. The Machine Learning-based branch encompasses methods such as Supervised Learning, Unsupervised Learning, Semi-Supervised Learning, and Deep Learning, each utilizing labelled or unlabeled data to train sentiment models. On the other hand, the Lexicon-based approach is divided into Dictionary-based and Corpus-based Approaches, which rely on predefined sentiment lexicons to determine the polarity of text. These two primary paths reflect the evolution and diversity in sentiment analysis techniques, from rule-based systems to more data-driven, automated models.

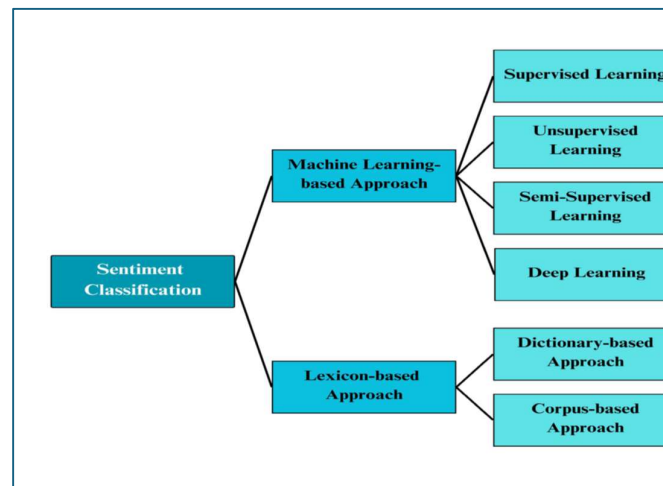


Figure 0.2: Sentiment Analysis Approaches

2.7.1 Machine Learning Approach in Sentiment Analysis

The Machine Learning (ML) approach to sentiment analysis involves training a computer model to understand emotions in text by learning from data. It utilizes algorithms that identify patterns in labelled or unlabeled text and categories them as positive, negative, or neutral. Compared to rule-based or lexicon-based approaches, ML methods can learn from context, handle complex sentences, and improve with additional data. Machine learning for sentiment analysis is generally categorized into four approaches: supervised, unsupervised, semi-supervised, and deep learning [21].

2.7.1.1 Supervised Learning

In supervised learning, the model is trained on a dataset that contains text data and corresponding sentiment labels [22].

The algorithm learns to associate certain words or patterns with sentiment classes. Common supervised algorithms used in sentiment analysis include:

- **Naive Bayes:** A probabilistic classifier that assumes features (words) are independent. It's fast and performs well with small data.
- **Support Vector Machines (SVM):** It finds a boundary (hyperplane) between sentiment classes in a high-dimensional space.
- **Logistic Regression:** A statistical model that predicts probabilities of classes (positive, negative, neutral).
- **Decision Trees / Random Forests:** Rule-based classifiers that split text features to reach a decision path.

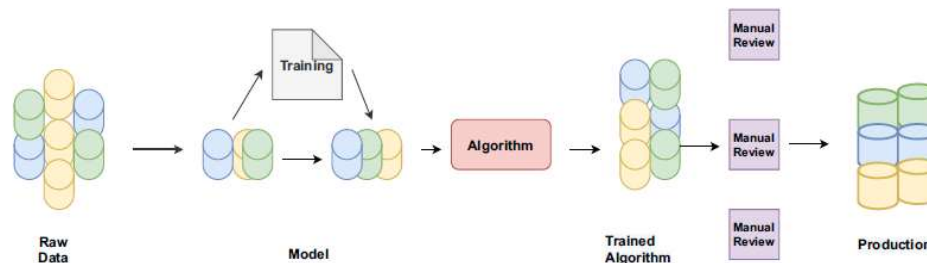


Figure 0.3: Process of Supervised ML Approaches

2.7.2 Machine learning Models

2.7.2.1 Naive Bayes

Naive Bayes is one of the simplest and most widely used algorithms for sentiment analysis [23]. It is a probabilistic model, meaning it predicts sentiment based on the likelihood of how often words appear in each sentiment category (such as positive, negative, or neutral). The core idea is to calculate the chance that a given piece of text belongs to a particular sentiment class by looking at how often the words in that text have shown up in texts of that class in the past.

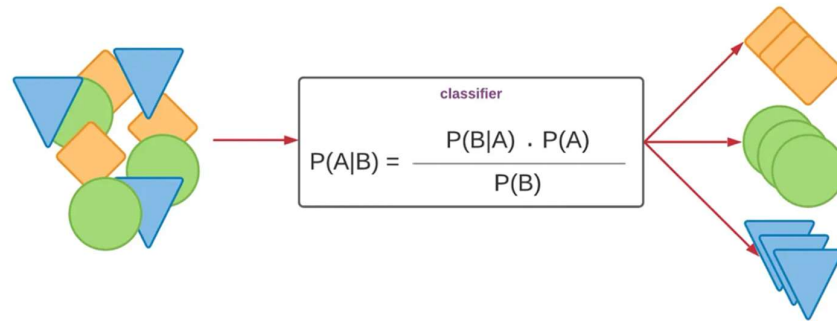


Figure 0.4: The architecture of Naïve bayes classifier [24]

The method is called “naive” because it assumes that each word in the sentence contributes to the sentiment independently of all the other words. For example, in the sentence “The app is fast and reliable,” Naive Bayes would treat “fast” and “reliable” as separate, unrelated indicators of sentiment, even though they are related in meaning. While this assumption is not always accurate in real language, the algorithm still performs surprisingly well, especially when trained on a large amount of labelled data.

During training, the algorithm learns how often words occur in each sentiment category. Then, when a new sentence needs to be classified, Naive Bayes uses the word frequencies it has learned to estimate which sentiment class the sentence is most likely to belong to. The class with the highest probability is selected as the final output. Naive Bayes is fast, efficient, and requires very little training data, making it ideal for large-scale applications such as classifying thousands of mobile app reviews. However, it may struggle when words have contextual meanings or when word order matters, which are not captured in this model.

2.7.2.2 Decision Tree

A Decision Tree is a machine learning model that works like a flowchart to make decisions based on the input data [25]. It splits the data into smaller parts by asking a series of yes/no or condition-based questions. Each question checks for a single feature (such as the presence of a word or phrase), and depending on the answer, the data is directed down a different branch of the tree. This process continues until the model reaches a final decision or label, like “positive” or “negative” sentiment. In sentiment analysis, a decision tree might ask questions such as: Does the sentence contain the word “excellent”? or Is the word “bad” present? Based on the answers, it keeps narrowing down the possibilities until it classifies the review into a sentiment category. The nodes in the tree represent the questions (or conditions), the branches represent the possible answers, and the leaf nodes represent the final sentiment label.

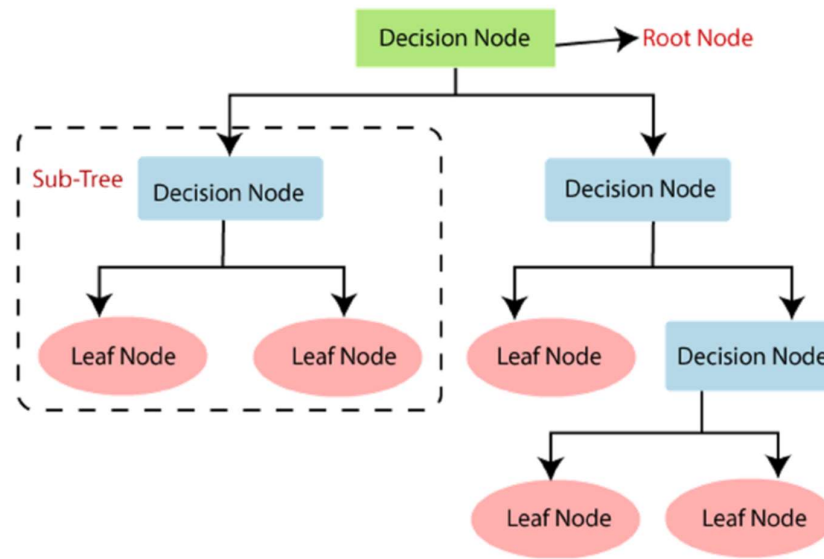


Figure 0.5: The architecture of Decision tree classifier [26]

Decision trees are easy to understand and interpret, making them very useful when you want to explain how a model makes its predictions. They can capture non-linear relationships in data, which makes them more flexible than simple linear models. However, they can sometimes become too complex and overfit the data especially when they grow intense and try to memorize the training data too precisely. To avoid this, techniques such as pruning (cutting back the tree) or using ensembles like Random Forest are often employed. Overall, decision trees are a powerful yet intuitive method for classifying sentiment, especially when working with structured or pre-processed features extracted from text.

2.7.2.3 Support Vector Machines (SVM)

Support Vector Machines (SVMs) are powerful machine learning models used for classification tasks, including sentiment analysis [27]. The main idea behind SVM is to find the optimal boundary (also known as a hyperplane) that separates different classes of data, in this case, different sentiments, such as positive or negative. This boundary is chosen in such a way that it creates the most significant possible gap, or margin, between the two classes. The wider the margin, the better the model is expected to generalize to unseen data.

SVMs are especially effective in high-dimensional spaces, which is typical in text data, where each word can be considered a feature. The model focuses on only the most critical data points called support vectors that lie closest to the decision boundary. These points are crucial because they define the position and direction of the separating hyperplane.

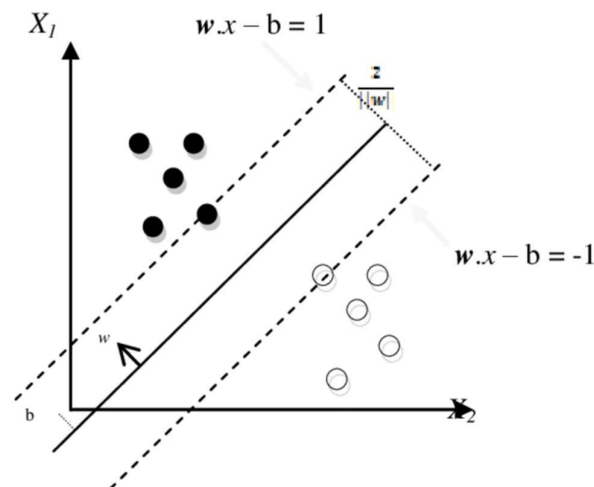


Figure 0.6: The architecture of Support vector machine [28]

One of the strengths of SVM is its ability to handle both linear and non-linear classification problems. When the data is not linearly separable (i.e., it cannot be divided by a straight line), SVM uses mathematical functions known as kernels to transform the data into a higher-dimensional space where a clear boundary can be found. This makes it highly flexible and suitable for complex sentiment classification tasks.

SVMs tend to perform very well in practice, especially when the number of features is much larger than the number of samples, which is often the case in natural language processing. However, they can be computationally intensive and require careful parameter tuning to achieve optimal performance.

2.7.2.4 Random Forest

Random Forest is an advanced machine learning model that works by combining the predictions of multiple decision trees to make a more accurate and stable final decision [29]. It belongs to a group of models called ensemble methods, which means it builds many models and then merges their results to improve overall performance. The idea is that while a single decision tree might make errors or overfit the training data, combining the decisions of multiple trees helps reduce these mistakes and increases accuracy.

To create a random forest, the algorithm first builds several individual decision trees. However, instead of using the complete dataset for each tree, it creates a different sample of the data for each one by randomly selecting data points with replacement a technique known as bagging. Additionally, when each tree is being built, only a random subset of the features is considered at each split. This randomness helps ensure that the trees are distinct from one another and do not all make the same errors.

Once all the trees are trained, the Random Forest model makes predictions by aggregating the outputs of all the trees. For classification tasks such as sentiment analysis, this typically involves taking a majority vote from all trees. For example, if most of the trees say a review is “positive,” then the final output will also be “positive.”

One of the most significant advantages of Random Forest is its ability to handle large feature spaces and noisy data, which is common in text analysis. It is also very good at capturing complex patterns and interactions between words without requiring heavy feature engineering. However, since it involves training many trees, it can be computationally more expensive and more complicated to interpret than simpler models, such as single decision trees or logistic regression. In sentiment analysis, Random Forests are often employed when accuracy and reliability are more important than model transparency, particularly for classifying large volumes of user reviews, social media posts, or customer feedback.

2.7.2.5 Logistic Regression

Logistic Regression is a widely used statistical model for classification tasks, particularly useful when the goal is to predict whether something belongs to one of two categories like positive or negative sentiment [30]. Despite the name, logistic Regression is used for classification, not Regression. It works by estimating the probability that a given input belongs to a particular class and then assigning the corresponding class label.

In sentiment analysis, the model learns from labelled training data, where the sentiment of each text (such as a product review or tweet) is already known. Each word or feature in the text is

assigned a numerical value, and logistic Regression uses these values to calculate the probability of the text being positive, negative, or neutral. If the probability for “positive” is higher than 0.5, the model classifies the text as positive; otherwise, it is classified as negative. What makes logistic Regression effective is its ability to model the relationship between input features and the likelihood of an outcome in a very efficient way. It is particularly suitable for datasets with a large number of features, such as word frequencies in documents, which is common in natural language processing.

One of the advantages of logistic Regression is that it is easy to implement and interpret. The model’s output reveals the contribution of each word or feature to the final prediction, making it valuable for researchers who seek not only high accuracy but also insight into why a text was classified in a certain way. However, it assumes a linear relationship between the features and the log-probability of the outcome, which may limit its performance in highly complex or non-linear problems.

Overall, logistic Regression is often used as a baseline model in sentiment analysis tasks because of its simplicity, speed, and interpretability, making it a solid choice for both minor- and large-scale text classification problems.

2.7.3 Deep Learning Approach in Sentiment Analysis

Deep learning is a subfield of machine learning that uses neural networks with multiple layers to automatically learn complex patterns from text data. It is especially powerful in handling long sentences, contextual meaning, and sarcasm.

Common deep learning models include:

- **Convolutional Neural Networks (CNNs):** Originally used for images, CNNs can also identify local features or n-gram patterns in text (e.g., “very bad service”) that influence sentiment.
- **Recurrent Neural Networks (RNNs):** Especially Long Short-Term Memory (LSTM) models, are used to understand sequence and context of words. For example, they can learn that “not good” is negative, even if “good” is usually positive.
- **Transformer-based Models (e.g., BERT):** BERT understands context by looking at the entire sentence at once and is one of the most accurate models for sentiment tasks today [31].

2.7.4 Deep Learning Models Approaches to Sentiment Analysis

2.7.4.1 Convolutional Neural Networks (CNNs)

Convolutional neural networks began with image work, yet they also read sentiment in text. Each filter scans a few neighboring words, much like a sliding window [32]. It can flag phrases such as “very good” or “not happy.” The network keeps the strongest matches, stacks them, and passes the stack to a layer that picks the final label.

This approach runs fast and suits short posts, tweets, and chat messages. It has limits. The filters focus on nearby words, so the model may miss clues spread across long sentences or several lines. Recurrent nets or transformers track wider links but need more time and power.

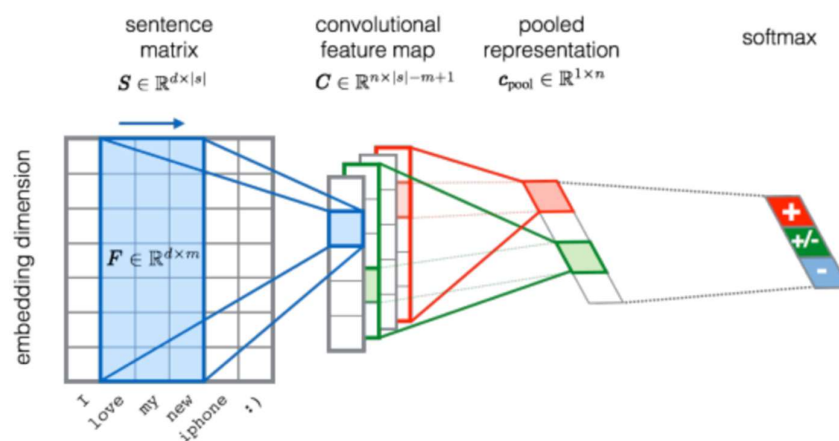


Figure 0.7: The architecture of Convolutional Neural Network for sentiment analysis [33]

2.7.4.2 Recurrent Neural Networks (RNNs)

Recurrent neural networks, or RNNs, read text one word at a time. Each word updates an internal state that stores what the model has already seen [34]. This running memory lets the network track context. In sentiment work, that memory links early words to later ones, so the system can spot shifts in tone inside a sentence.

Basic RNNs still face a hurdle. Gradients can fade during training. Long sentences then overwhelm the memory and the network forgets early cues. A long review may start with praise and end with a complaint. A plain RNN may miss that turn and label the text wrongly.

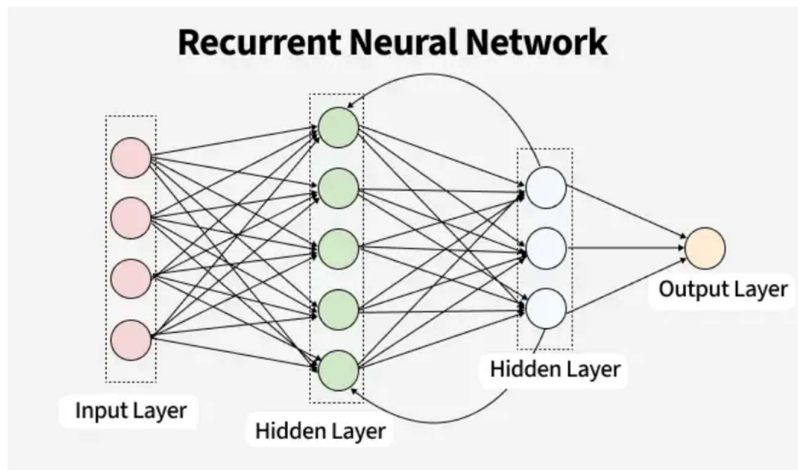


Figure 0.8: The architecture of Recurrent Neural Network [35]

2.7.4.3 Long Short-Term Memory (LSTM)

LSTM is a special type of RNN designed to solve the limitations of standard RNNs [36]. An LSTM is a variant of a recurrent neural network. Standard RNNs lose track of information after many steps. LSTM fixes this issue with small units called gates that store, discard, or pass data forward. The gates let the model notice events spread across long text, like a positive start and a negative finish. In sentiment analysis, LSTM can effectively track emotional tone across entire paragraphs or conversations, and it works well with both short and long texts. It is one of the most widely used deep learning models for SA tasks, thanks to its ability to handle contextual shifts and negation.

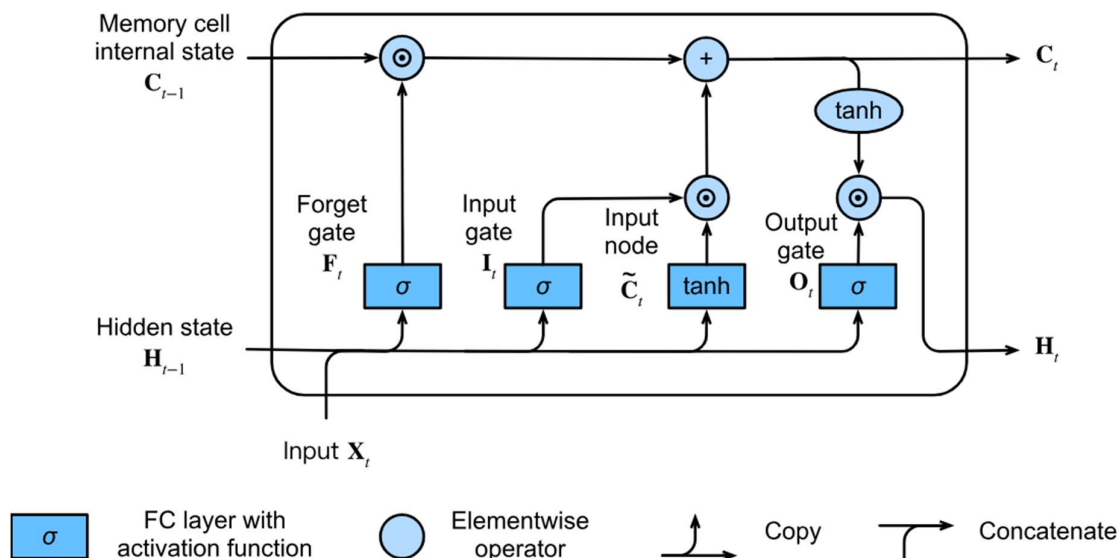


Figure 0.9: The architecture of LSTM [37]

2.7.4.4 Gated Recurrent Unit (GRU)

GRU is a simpler and faster alternative to LSTM that also solves the problem of vanishing gradients in RNNs [38]. It combines the memory and forget gates into fewer components, making it less complex while still being effective at capturing sequential patterns. GRUs are often used when computational efficiency is important or when the dataset is not large enough to train more complex models like LSTM or transformers. In sentiment analysis, GRU models perform similarly to LSTM in many cases and are especially useful for real-time or mobile applications where memory and speed are critical.

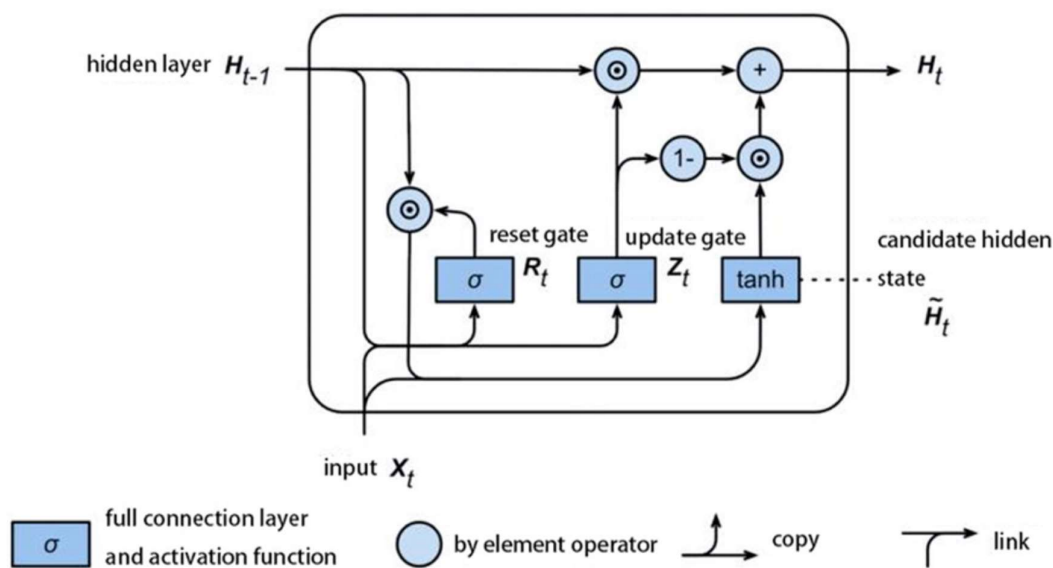


Figure 0.10: The architecture of GRU [39]

2.7.5 Large Language Models (LLMs Approaches to Sentiment Analysis

2.7.5.1 Introduction to Large Language Models (LLMs)

Large Language Models (LLMs) learn language from huge text collections [40]. They include GPT from OpenAI, BERT from Google, and LLaMA from Meta. Each network holds billions of parameters. These weights store patterns about grammar, facts, and tone. In sentiment analysis, LLMs excel. They study full sentences and paragraphs rather than fixed word lists. This depth lets them catch irony and shifting context.

Transformers give LLMs this skill. Self-attention lets the model weigh every word against every other word in one step. The process captures links between distant terms. The model then flags sentiment shifts, double meanings, and subtle cues. Engineers can fine-tune an LLM for a new dataset with only a brief training run, or use it in zero-shot form.

Vaswani et al. introduced the transformer in 2017. The design replaced recurrent nets such as RNNs and LSTMs. It processes the full input at once, so training runs on many GPUs in parallel and finishes sooner. The lack of recurrence also helps the model learn long-range patterns.

During training the network views the entire sentence, paragraph, or document as a single sequence. This parallel view cuts training time and raises accuracy on large corpora. Modern LLMs benefit from that speed and size. New variants keep improving sentiment analysis, question answering, and text generation.

2.7.5.2 BERT (Bidirectional Encoder Representations from Transformers)

BERT is a transformer-based model developed by Google that reads text in both directions (left and right), unlike earlier models that only read in one direction [41]. This bidirectional training helps BERT understand the full context of a word by considering the words that come before and after it. BERT is pre-trained on large text corpora using two techniques: masked language modeling (predicting missing words) and next sentence prediction (understanding sentence relationships).

In sentiment analysis, BERT is powerful because it captures subtle meanings, sarcasm, and the influence of context on word interpretation. For example, the word "great" might mean something very different when preceded by "not" or "barely."

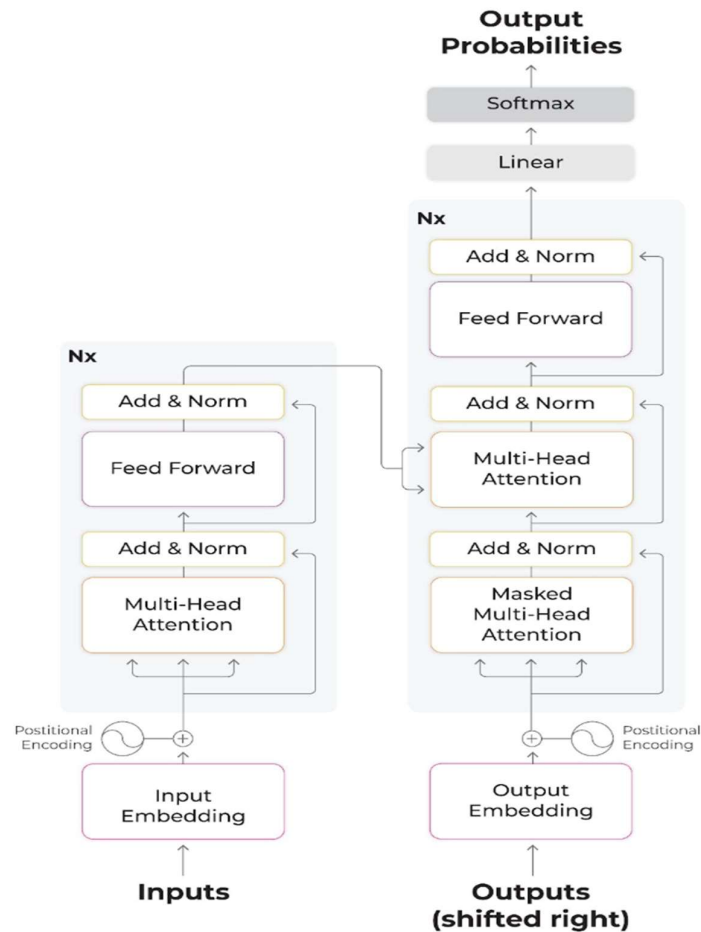


Figure 0.11: The architecture of BERT [42]

2.7.5.3 GPT (Generative Pre-trained Transformer)

GPT, developed by OpenAI, is a unidirectional transformer that generates text from left to right [43]. It is trained to predict the next word in a sequence, which makes it excellent for language generation tasks. Unlike BERT, GPT focuses on generative tasks, but it also performs well on classification tasks like sentiment analysis when fine-tuned on labeled datasets.

The model's large size and training on massive datasets allow it to understand grammar, world knowledge, and even conversational tone. In SA tasks, GPT can identify complex expressions of sentiment, irony, and opinion across multiple sentences or even entire documents.

2.7.5.4 RoBERTa (Robustly Optimized BERT Approach)

RoBERTa is an improved version of BERT created by Facebook AI. It keeps BERT's two-way transformer but drops the "next sentence prediction" stage [44]. Facebook AI fed it ten times more text: 160 GB from Common Crawl plus BookCorpus and news. They also raised

the batch size and the learning rate, then trained for 500 k steps. These choices let the model learn richer word patterns and longer contexts.

Benchmarks confirm the gains. On GLUE, RoBERTa-large scores 90.2, while BERT-large scores 82.2. On the SST-2 sentiment task, RoBERTa reaches 96.4 % accuracy, two points above BERT. Similar jumps appear on emotion and intent datasets like GoEmotions and SNIPS. For a production pipeline, fine-tune RoBERTa on labeled reviews or tweets for three epochs. Use a learning rate near 2×10^{-5} and a batch size of 32. Freeze nothing; the full model adapts fast. With these settings, you can expect one to three accuracy points over an equally tuned BERT model.

2.7.5.5 DistilBERT

DistilBERT is a smaller and faster version of BERT designed to be efficient while keeping most of BERT's performance [45]. It is created using a technique called knowledge distillation, where a smaller model is trained to imitate a larger one. DistilBERT is particularly useful in real-time applications or on devices with limited resources like mobile phones. Even with fewer parameters, DistilBERT still achieves strong performance on tasks such as text classification and sentiment analysis, making it popular in industry settings.

2.7.5.6 XLNet

XLNet is another transformer model that improves on BERT by using a technique called permutation-based training, which considers all possible orders of words during training [46]. This helps XLNet learn better context and sequence understanding without relying on masking or losing word order information. It often outperforms BERT on a range of NLP tasks and is particularly good at sentiment classification when context and word order are critical.

Table 0.1 Comparison Matrix of Sentiment Analysis Methods

Category	Model/Method	Key Characteristics	Advantages	Limitations	Common Use Cases
Machine Learning	Naive Bayes	Probabilistic model assuming word independence	Fast and works well on small datasets	Assumes features are independent (not realistic for language)	Spam filtering, basic SA

	Decision Tree	Tree-like structure using rules to classify sentiment	Interpretable and easy to visualize	Prone to overfitting	Structured text classification
	Random Forest	Ensemble of decision trees	Higher accuracy and robustness	Slower, less interpretable	Large text datasets
	SVM	Maximizes margin between classes	Performs well on high-dimensional data	Requires careful tuning; slower on large data	Review classification
	Logistic Regression	Predicts probability of sentiment class	Interpretable and efficient	Limited expressiveness for complex data	Binary sentiment tasks
Deep Learning	CNN	Uses filters to extract local features from text	Good at detecting n-grams and phrases	Weak at capturing long-term dependencies	Short text sentiment
	RNN	Processes sequences word by word with memory	Maintains word order and context	Struggles with long sequences	Sequential sentiment tasks
	LSTM	Gated RNN variant that remembers long-term dependencies	Handles long and complex sequences	Computationally heavy	Paragraph/document-level SA
	GRU	Simplified LSTM with fewer gates	Faster than LSTM; good for small data	Slightly less expressive than LSTM	Real-time sentiment tasks
LLMs / Transformers	BERT	Bidirectional transformer trained on masked word prediction	Strong contextual understanding	Requires fine-tuning; resource intensive	Complex sentence-level SA
	GPT	Left-to-right transformer trained to predict next word	Excellent at text generation and coherent output	Unidirectional (less context-aware); large model size	Generative SA tasks

	RoBERTa	Optimized version of BERT with better pretraining	State-of-the-art performance	Requires extensive computing power	Sentiment benchmarks
	DistilBERT	Compressed BERT with fewer parameters	Fast and lightweight	Slight performance drop from full BERT	On-device SA
	XLNet	Permutation-based transformer with full context	Handles sequence order and context well	Complex training; slower than BERT	Long text and context-heavy SA

2.8 Evaluation Matrices

To assess the performance of various machine learning models for sentiment analysis, standard evaluation metrics such as Accuracy, Precision, Recall, and F1-Score are used.

1. Accuracy

Accuracy measures the overall correctness of the model. It shows how many total predictions were correct.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

TP = True Positive

TN = True Negative

FP = False Positive

FN = False Negative

2. Precision

Precision tells how many of the predicted positive results are actually correct. It reflects the model's ability to avoid false positives.

$$\text{Precision} = TP / (TP + FP)$$

3. Recall

Recall indicates how many of the actual positive cases were correctly predicted. It reflects the model's ability to avoid false negatives.

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

4. F1 Score

F1 Score is the harmonic mean of Precision and Recall. It balances the two when there is an uneven class distribution.

$$\text{F1 Score} = 2 \times [(\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})]$$

2.9 Summary

The current chapter has given a detailed knowledge of sentiment analysis, its significance, its levels of granularity (document to aspect-level), its use in areas of mobile application analysis, social media analytics and customer satisfaction. Several feature selection methods such as Bag of Words feature extraction, TF-IDF feature extraction, advanced word embeddings (Word2Vec, GloVe, FastText and ELMo) were described to transform unstructured data into data that would be machine-readable. The chapter also discussed various methods of sentiment classification, including various traditional machine learning models (Naive Bayes, Decision Trees, SVM, Logistic Regression) and deep learning models (CNN, RNN, LSTM, GRU), as well as the latest transformer-based large language models (BERT, GPT, RoBERTa, XLNet). Lastly, the chapter discussed practical tasks and difficulties like sarcasm, idiom, lexical ambiguity and context dependency. Knowledge created in this chapter forms the basis that the rest of the chapters build on. It prepares the ground on which sentiment analysis methods are implemented and processed.

Chapter 3

Literature Review

Over the past decade, sentiment analysis has experienced considerable growth because of the very high rise of user-generated content on most social media platforms, review sites, and other online communication platforms. With the development of new strategies and approaches to identify valuable knowledge in this huge mass of textual data, the necessity of developing proper sentiment classifications models is more relevant than ever before. The literature review in this research will present an analysis of modern development in the sentiment analysis field, particularly in the spheres of machine learning (ML), deep learning (DL), and large language models (LLMs). The literature review will compare models based on their accuracy, interpretability, computational cost, and generalization on various tasks e.g. product reviews, social media posts, and multilingual data.

These insights will be used to shape the design and future of this thesis, especially in determining the most promising and viable methods of developing an effective sentiment analysis system. Following sections are structured in three key segments that include literature on traditional machine learning, deep learning, and transformer-based LLMs; and concluding with identification of research gaps and reasons to work on the proposed project.

3.1 Machine Learning Models

Jannah et al. [47] conducted a research study aims to analyze the Naive Bayes and Support Vector Machine (SVM) performance in sentiment classification of e-commerce product reviews focusing on beauty products like Shopee. The authors used 10-fold cross-validation to assess its robust performance evaluation. Preprocessing pipeline included the step of lemmatization and TF-IDF vectorization that transforms raw text into numerical features that can be used with machine learning. The study utilized a synthetic dataset that mimicked real-world Shopee-like reviews and was labelled based on the review as positive, negative, or neutral manually. Comparing the outcomes, SVM model exceeded Naive Bayes, with the average accuracy of 86.14%, whereas Naive Bayes delivered the accuracy of 85.53% in some folds. Although the study has achieved promising results, it has some limitations. It lacks error analysis and feature importance assessment, which are essential to insight into the model behavior and misclassifications. Moreover, the fact that the research was conducted on a synthetic dataset and not tested on real Shopee data implies the lack of ecological validity and generalizability of the findings.

Juliane et. al. [48] used traditional machine learning pipeline in their study to apply Naive Bayes classifier to Twitter data sentiment analysis. The approach was to retrieve the tweets through an API of Twitter, followed by the preprocessing that includes lowercasing, stemming,

and the removal of stop-words. Term frequency (TF) was then used to convert the preprocessed text into features and finally the classification process was achieved through the Weka software environment. The data was made up of Twitter comments indicating views of people on different social matters. It was split into training set and a small validation set with final 30 tweets as validation. The model got a really high accuracy of 93.33 on the training set but validation accuracy was a bit less at 86.66 which implies a fairly good level of generalization. The findings of the study are however limited based on the small size dataset obtained, especially the small size of the validation set thus limiting size of performance estimates. Also, the authors did not compare Naive Bayes to other models, such as the SVM or the Logistic Regression which would have reinforced the analysis in terms of the bench mark data.

The study by Azeraf et al. [49] presents a Neural Naive Bayes (NNB) classifier, which is a new improvement to the classic Naive Bayes idea since it uses shallow neural representations that are more context-dependent in defining semantics. Unlike traditional models, which use Bag of Words or TF-IDF, this model uses pre-trained FastText embeddings, so it is able to take word-level context into consideration at the time of the classification. Authors have tested the model on the IMDb movie reviews database, where sentiment is predicted in a binary mode (as positive or negative). Their findings indicate that the NNB classifier was very useful in reducing classification error 4.5 times as compared to standard Naive Bayes and results were impressive especially addressing long or context-dependent reviews. The study has its limitations despite good performance. It lacks a comparison to other popular models like Logistic Regression or SVM, when utilizing the same embeddings. Further, only one dataset was used in the experiment, and thus, limits the validity generalization characteristic shown by the model.

In a recent study [50], Support Vector Machines (SVM) and SMOTE Tomek link oversampling were used in classification of sentiments in fashion e-commerce reviews. The SVM model after preprocessing e.g., cleaning, TF-IDF vectorization has reached 92% accuracy, 0.89 precision, 0.89 recall, 0.89 F1-score whereas the non-augmented baseline has displayed an insignificant degree of accuracy 0.68, F1-score 0.71. The research is restricted to binary classification (positive/negative), no investigation of neutral sentiment, or comparison with other classifiers under an identical oversampling plan has been conducted.

The research work presented by Wirawan et. al. [51] tackles the problem of detecting cyberbullying in the comments of Instagram using easy data augmentation (EDA) in connection with SVM. The enriched data enhanced resistance, with the accuracy of 92.5%, with an additional advantage of 2.5 percent over the usual SVM model. It targets one of the

domain-specific datasets but its numeric improvement is not compared to specifying logistic regression or decision tree baselines.

The sentiment analysis of the reviews on the e-commerce products used a Logistic Regression, Decision Tree, Random Forest, and SVM in the Turkish language were made by C. G. Özmen and S. Gündüz [52]. SVM performed persistently better than the others, yet Logistic Regression managed to display decent performance in a range of categories, proving once again to be useful in large text classification, as well. Although the several models were compared, hyperparameter tuning, model interpretability analysis of the classifier behavior were not done. In their study, Danyal et al. [53] compared and evaluated the multiple machine learning algorithms to conduct sentiment analysis in a comprehensive way with references to the Linear Support Vector Machines (SVM), Multinomial Naive Bayes (MNB), and Logistic Regression (LR) apart from other baseline classifiers. The objective of the study was to determine which traditional classifiers are most effective with regard to popular sentiment datasets. Two gold standards of datasets were used: IMDb 50K containing 50,000 positive or negative movie reviews, and Sentiment Polarity Dataset, a long-proven benchmark with positive and negative sentence-based reviews.

The researchers implemented conventional preprocessing steps, such as tokenization and TF-IDF vectorization, prior to training each of the models. Linear SVM was one of the models that performed best with an accuracy of 89.48 %. Nevertheless, Logistic Regression, particularly when combined with hyperparameter tuning, just slightly surpassed SVM, achieving an accuracy of 90.27%. Passive Aggressive Classifier also performed competitively when fine-tuned suggesting that simple, linear models are still very useful in sentiment classification especially on clean and well-labeled datasets like IMDb.

Regardless of the promising results, the study has certain limitations. One of the shortcomings is that the analysis paid much attention to the accuracy as the only performance measure, without much attention given to precision, recall, or F1-score, which is acute in the case of imbalanced or real-life data. Second, the research did not address the model performance on multilingual data or on domain-specific reviews like a product review, app store reviews, or social media text, which are linguistically diverse. Finally, the models have not used more stamp-contextualized embeddings instead of the more common TF-IDF base features, which would theoretically help the classification performance on more convoluted texts.

Khan et al. [54] have compared sentiments analysis of Random Forest and Support Vector Machine (SVM) classifiers on news and social media posts. The objective of the study was to examine the performance of classical machine learning methods in classifying opinionated text

using various kinds of real-world data. The authors used both models on sentiment dataset with pre-labelled headlines of news, samples of blog articles, and social media messages that were labelled either positive or negative. For feature extraction, the steps of standard natural language processing, including tokenization, stop-word removal, and TF-IDF vectorization, were applied. Overall, SVM model performed marginally better than Random Forest, with an accuracy of about 80.39% and 78.56%, respectively. Nevertheless, the authors observed that Random Forest produced a more consistent output and could process data faster, especially on a noisy and informal text, as is always the case in social media. The ensemble nature of Random Forest enabled it to deal with variability and outliers more effectively compared to SVM that is prone to feature noise.

Although all of the above are strengths of the study, there are some limitations. In particular, the authors have not explored the polarity of phrases at the segment level, e.g., sentiment of sentences or polarity of aspects, which restricts the level of information provided by the text. All the judgments were performed at a document level that can hide the divided feelings in a single piece of material. Moreover, the paper has not used recent representations of features, yet only TF-IDF as a way to vectorize features. This limits the effectiveness with which the models can derive semantic relationships among words, idioms and contextual meaning. Moreover, it was restricted to binary classification (positive and negative) without any consideration of neutral and ambiguous categories, which are essential in opinion mining nuances.

The literature that has been reviewed highlights the fact that sentiment analysis still heavily relies on the traditional machine learning models. Naive Bayes is still considered a robust baseline because it is easy to implement and fast, but its feature independence assumption can negatively affect its performance on complex text. Support Vector Machines are very accurate most of the times especially when combined with data augmentation or oversampling process, though they are prone to noisy data. Logistic Regression is both efficient and interpretable with competitive results once it is tuned. Decision Trees and Random Forest provide robustness and are particularly effective on noise or unstructured data but are slightly less accurate than SVM. In general, these models are good but in the majority of studies, simple text representations are used and the idea is to achieve binary classification. It has been observed that more sophisticated methodologies are needed that can include richer feature embeddings, aspect-level sentiment, and use a wider variety of evaluation metrics.

In a study presented by Singhal et al. [55], Support Vector Machines (SVM) classifier was compared to Naive Bayes, Logistic Regression, and Random Forest in the process of detecting

fake reviews based on BERT embeddings. The Amazon product reviews dataset was used as the research has labeled information about the reviews based on whether they are genuine or fake. The BERT embedding of reviews preprocessed the reviews, which led to better understanding of the semantics of words in the context of reviews by the models. The models were trained and tested with identical data and resulted in direct performance comparison.

The findings showed that SVM had better results with an accuracy of 87.81 % compared to the other models. It means that SVM was the most effective model used in the case of detection of fake reviews due to its characteristics such as ability to seek optimal hyperplane on a high-dimensional space. Although SVM performed well, the authors did not compare the method to more advanced algorithms such as XGBoost or LightGBM, which were deemed to have better results with regards to classification. Also, hyperparameter tuning was not investigated during the research, which might have provided even more optimization of the models and possibly their better work. Such constraints imply that additional research should consider such extra models and adjustment methods in detecting fake reviews.

In spite of the positive outcomes, the research had significant limitations. The absence of the error analysis and feature importance rating was one of the major gaps in research. They are essential in explaining the model performance, rationalizing the misclassification, and seeing which features added the largest value to the model predictions. Moreover, the research has been done on a synthetic data which was only an approximation to the real-world data. Consequently, the results do not necessarily provide an overview of the issues that occurred when using the model on real Shopee data, potentially bringing in more noise and complexities. These gaps indicate future research opportunities, especially in terms of applying the model to real data and the more thorough analysis of the model mistakes and feature play.

3.2 Deep Learning Models

As unstructured user feedback, in the form of app reviews, social media comments, and digital ratings increases, deep learning has also become highly effective as a sentiment analysis tool. Deep learning models, unlike conventional machine learning models, have the ability to automatically learn relevant patterns and textual relational dependency, thus deep learning models are ideal in the analysis of user's opinions. Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks have demonstrated to perform better in recognition of sentiments especially when used to train large and noisy data such as mobile app reviews. The following section gives an extensive literature review of recent articles that have used deep learning in sentiment analysis problems on

different forms of user-generated review information. All the studies have been presented with their methodology, dataset, performance, and research limitations.

Ahammad et al. [56], developed a complex sentiment analysis of the user reviews of AI-powered mobile applications of the Google Play store. The authors have examined the performance of these six types of deep learning architectures: CNN, LSTM, BiLSTM, GRU, BiGRU, and a hybrid model of CNN+LSTM. Their procedure was quite comprehensive with a lot of preprocessing in form of cleaning, tokenization, and vectorization, training, and testing over the entire data. The data contained 122,231 user reviews and scores gathered on the most famous AI-powered apps: ChatGPT, Lensa, Bing, and Alexa, and each review was manually assigned one of three categories: positive, neutral, and negative. CNN + LSTM hybrid architecture model exhibited the highest accuracy of about 92.1 %, which is closely followed by LSTM that averaged 91.97 %, and the other resulting model (BiLSTM, GRU, and CNN) had relatively lower accuracy. The study provides a powerful analysis of the comparison of deep learning architectures in sentiment classification but it focuses mostly on accuracy. It does not provide detailed analysis by sentiment classes (which would include precision, recall, and F1-score), nor does it extend to any comparison to transformer-based models, like BERT, or to lightweight models that could be used on a mobile device, creating future opportunities to explore.

To evaluate the effectiveness of various models in sentiment classification of reviews in mobile applications on the Google Play Store, a comparative study was undertaken by Samanmali and Rupasingha [57]. The authors compared three methods, namely, the shallow Artificial Neural Network (ANN), Long Short-Term Memory (LSTM) network and Support Vector Machine (SVM). In their approach they used some common steps of data processing like tokenization, TF-IDF based vectorization and class balancing so as to provide equal representation of all classes of sentiment. The data of nearly 33,000 user reviews of 15 most popular mobile applications were processed as positive or neutral or negative. The LSTM model was performing better than the ANN and SVM model and showed the best ability to achieve high accuracy scores (of around 80-90%) across various apps, as well as it obtained the highest value of F1-score overall. However, there was no explicit description of hyperparameter optimization and no examination into performance of the models on long form reviews. Moreover, it lacked any embedding-based features like Word2Vec or contextual features like BERT, which could have provided the semantic understanding of the model.

In the study by Amirkhalili and Wong [58], the research is based on a sentiment analysis of user applications of mobile banking applications through the iOS App Store and Google Play

Store. To accomplish this task, the authors have used a combination of traditional machine learning algorithms, such as Multinomial Naive Bayes algorithm, and Long Short-Term Memory (LSTM) neural networks to detect sentiment. Also, they incorporated Latent Dirichlet Allocation (LDA) to do topic modeling to capture thematic constructions, as well as, sentiment patterns. The data consisted of about 2,025 manually categorized reviews of Canadian banks that have mobile applications, which were gathered through five large mobile banking platforms. The results showed that the LSTM model had nearly 82 % accuracy in the reviews of the iOS app as compare to Naive Bayes generating an accuracy of about 77 % accuracy in the Google Play reviews. The paper demonstrated the advantage of LSTM in picking up subtle patterns of sentiment in the text. A few limitations have been identified, the size of the dataset used was not very large, and there were no results on performance metrics by class e.g., precision, recall, or F1-score. Moreover, it was region-limited, only Canadian banking apps were considered, and it did not imply the use of a transformer-based model, which would be a better benchmark in terms of comparison.

In a recent study by Zaman et al. [59], user opinions were examined regarding the usage of mobile Extended Reality (XR) experiences in the Google Play Store. The study used deep learning classifications to conduct sentiment analysis on reviews with the aim of identifying the insights relating to quality. Preprocessing was performed by using a structured Knowledge Discovery in Databases (KDD) pipeline of steps including, tokenization, removing stop-words and stemming to readying the text before feeding them into the models. The list of data was made up of user reviews taken in the year 2025, and it was able to target the XR-related mobile apps. According to the study, the neural network models being implemented produced better results in sentiment classification compared to the standard machine learning baselines. The results summary, however, failed to mention specific performance measures such as accuracy, precision, and recall or break them down according to sentiment class. Furthermore, the study has a considerable limitation of the lack of clarity of the specific deep learning network used or if it was LSTM, CNN, GRU, or some other type, and a lack of comparison of the results with the current state-of-the-art models, which makes the interpretability and reproducibility of the study findings limited.

A research paper written by Ranjan and Mishra [60] assessed the use of sentiment analysis to determine the efficiency of an LSTM-based model, augmented with GloVe embeddings, as compared to conventional machine-learning models and ensemble methods like bagging. The main emphasis of the study was the analysis of user sentiment of the evaluations of Google Apps with a special attention being paid to that of the university students. The dataset included

a massive corpus of Google App Reviews in a multitude of languages and, in addition, they introduced a specialized dataset of Student App Reviews (SAR) that represent the academic environment. Using an LSTM model with GloVe word embeddings performed the best among all the tested approaches, with the accuracy of 95.2% and the F1-score of 88. The more traditional classification methods, such as SVM, Logistic Regression, and bagging also obtained 93-94% accuracy, but deep learning approach performed a little bit better. Although the research has been conducted with great results, it has certain limitations. It does not evaluate the generalization of the model to other apps or other domains, which is critical in the real-world deployment. Furthermore, the work does not include a comparative performance with transformer-based models, i.e., BERT or RoBERTa, the state of the art in sentiment classification tasks nowadays.

The presented studies show that LSTM-based models, and especially their augmented versions with pre-existing word embeddings (GloVe in particular) can consistently have high performance when it comes to sentiment analysis of mobile app reviews and have demonstrated accuracy of 82-95 %. Moreover, structures such as CNN+LSTM enhance that performance even higher, as they merge benefit of spatial and temporal feature extraction. These models are very robust to handling noisy user-created written text and are good at learning contextual and sequential dependencies, which are crucial in interpreter sentiments as they are expressed in review of the app.

The study conducted by Basri, Nur Faizal, and Ema Utami. [61], entitled "Application of Word2Vec and LSTM Models in Sentiment Analysis of Mobile Legends User Reviews", was devoted to using deep learning to analyze user reviews of the Mobile Legends game in terms of sentiment. The research posed the question of how deep learning models can enhance sentiment classes by examining the reviews of the users of one of the most popular mobile games in Indonesia.

The methodology aspect of the research involved a combination of Word2Vec embeddings and Long Short-Term Memory (LSTM) networks. Words were converted to vectors representation using word2Vec to extract the semantic meaning within the words. An LSTM model was trained on these embeddings to predict how positive or negative the reviews are. It used supervised learning to train the model, though the reviews were already labelled with the sentiment. The research applied a dataset of 100,000 reviewers of the Mobile Legends game, which was compiled in the Google Play Store. These reviews were mainly done in the bahasa Indonesia, a language that the majority of the user base of the game would understand. The

large volume of data offered a variety of sentiments since it would be appropriate to use it in training the model.

The findings were encouraging as Word2Vec-LSTM model reached a sentiment classification accuracy of 87.88%. It was revealed in the analysis that most of the user reviews were positive with words such as good, exciting, and awesome being frequently used in the reviews thus showing that most users reacted well to the game. Nevertheless, the study had very significant gaps. The performance of Word2Vec-LSTM model was good but not compared with BERT or RoBERTa which is proved to be performing better in the sentiment analysis task. Also, the research was not focused on the impacts of various preprocessing methods and the processing of noisy data that could have impacted the performance and generalization of the model.

Although each of these studies has its strengths, a number of limitations still exist in common. In particular, there is too much emphasis on accuracy as the key performance measure, and little results of precision, recall and F1-scores, which are necessary in imbalanced data sets. Also, there are some studies that apply the transformer-based models like BERT, which have achieved state-of-the-art performance in NLP tasks to form a valuable performance benchmark. The absence of fine-grained or aspect-level sentiment analysis is also another gap that is needed to detect sentiments attached to certain attributes or elements of an app. Moreover, most of the articles reviewed use single-language and single-domain data, limiting their utility to other linguistic and contextual environments.

Finally, the comparison shows that although deep learning, and especially its LSTM and hybrid variants, demonstrates consistent and accurate sentiment classification in mobile review applications, there is a great potential to improve these systems. The further studies can be enhanced by incorporating transformer models, multilingual data sets, and aspect-level sentiment data that will be more scalable and feature-oriented sentiment inferences on diverse app ecosystems.

3.3 Large Language Models

As Natural Language Processing advances at a steady pace, deep contextual understanding of language in the form of Large Language Models (LLMs) such as BERT, RoBERTa, and DeBERTa has changed sentiment analysis. Compared to the traditional machine learning or even pre-trained deep learning models, LLMs pre-trained on enormous corpora provide better generalization, which makes them very useful with user-generated tasks like written reviews of an app. This following section will discuss the recent research works that use LLMs in the

field of sentiment analysis especially in the mobile or review-based nature. Each research is investigated in regard to methodology, data source, findings, and research gaps.

The aims of the study by Eser and Şahin [62] were to compare the effectiveness of different transformer-based approaches to sentiment analysis of user reviews of the Spotify app. The tested models such as the BERT, DistilBERT, RoBERTa and XLM-RoBERTa were implemented and trained to classify the reviews as positive, neutral or negative in the context of sentiment categories. The investigators preprocessed the data using extensive steps including removal of emojis, fixing of typos and tokenization, prior to training of the models. Moreover, they obtained the textual sentiment scores using VADER sentiment analysis and then provided a related view in user rating categories as well as trends. The database included the reviews of the Spotify app based in the Google play store. DistilBERT had the greatest accuracy and recall 71.68%, whereas XLM-RoBERTa obtains the highest F1-score 69.24%, so it has a good balance between accuracy and recall. Nevertheless, the study had certain limitations: their overall performance was relatively poor in comparison to state-of-the-art benchmarks, in addition, models were not able to work properly on the neutral and underrepresented classes. Moreover, the study did not look into multilingual input and real-time deployment of models, which are essential with respect to deployment on realistic and dynamic apps.

Motger et al., [63] proposed T-FREX which was a transformer-based method that had the objective to extract features of mobile apps reviews using encoder only transformers like BERT. Instead of using word-based syntactic rule as in traditional rule-based approaches, T-FREX molds the task as a token-level classification allowing the model to find app features in raw user reviews. The approach contains a pre-training step on large corpus, and then instance selection should be done to balance the trade-off between cost of fine-tuning and recall and precision. This was an industrial-scale and crowdsourced dataset based on ground truth with accurate annotations referring to real Google Play Store mobile application reviews. Unlike baseline methods, T-FREX showed improvements in feature-level precision and recalls and, therefore, achieved compositional gains across the range of downstream tasks of sentiment analysis. The study did not dwell on the actual feature sentiment classification or polarity extraction of the extracted features though. Consequently, it does not analyze the sentiment aspect directly, or discuss the possibility of applying fines-grained sentiment labeling on a per-feature basis in the feature extraction stage, although it does extend the preprocessing and feature extraction stage with LLMs.

Nkhata et al. [64] suggested a hybrid model that includes BERT with a bidirectional LSTM (BiLSTM) layer with the goal of improving the tasks of binary and fine-grained sentiment

classification. It was inspired by the motivation to combine the excellent contextual understanding provided by BERT and the sequential dependencies in both forward and backward directions provided by BiLSTM. In order to overcome the class imbalance problem, the research used data augmentation methods like SMOTE and NLPAUG. The model was trained and tested in two benchmark data sets namely IMDB dataset that measures a binary sentiment categorization and the SST-5 dataset measuring a five-level sentiment category. The performance was impressive it scored 97.67 % accuracy on IMDB, which is 0.27 % higher than the state of the art, and 59.48 % on SST-5, higher than a standard BERT base by 3.6 %. Nevertheless, the major weakness of the article is its focus on the domain. The model was not applied to mobile app review data sets that more often contain other forms of noise and user language. In addition, the work did not investigate sentiment tagging in neutral or ambivalence reviews that are prevalent in mobile platforms. The model, however, has a high promise of being adapted to cross-domain application, especially on sentiment-laden spaces such as app stores.

Chi et al. [65] built on the BERT embeddings and incorporated Temporal Convolutional Networks (TCN), as well as, Bidirectional LSTM (BiLSTM) and Self-Attention to develop an advanced sentiment analysis model that provides a rich text semantic representation of the text data. This hybrid architecture was developed to facilitate the integration of the contextual richness of BERT, temporal structure learning of TCN, sense of time sequence by BiLSTM, and the ability to focus and enrich the self-attention, in order to improve the sensitivity of the classification of sentiments. The model was tested on the dataset consisting of human-labeled hotel reviews gathered across several resources, such as TripAdvisor and Booking.com. The performance of the models indicated that this integrated model performed as forecasted and consistently surpassed baseline BERT models, especially in accuracy and F1-score, proving the hypothesis of the usefulness of combining deep contextual and sequential learning layers. Nonetheless, the research was restricted to the hospitality sector and its use with other kinds of user-generated reviews, including feedback on mobile apps, was not subjected to exploration. Also, no aspect of the research had investigated how efficiently the model could work in inference and was also lacking in aspect-level sentiment analysis which is essential in real life and also in providing insights to the user.

Another hybrid framework that was developed is the Instruct-DeBERTa by Jayakody et al., [66] framework, which is aimed at carrying out Aspect-Based Sentiment Analysis (ABSA), and it performs aspect term extraction (ATE) as well as aspect-level sentiment classification (ASC). The strategy uses DeBERTa-V3, and it is fine-tuned by supervised learning methods

that are made specifically applicable to multiple ABSA tasks. The method was trained and achieved validation on both multilingual and multi-domain SemEval datasets (related to reviews in restaurants and laptops). The model showed superior performance in targeting a specific aspect (i.e., opinion targets) and classifying sentiment polarity with respect to these targets among the existing ABSA models whose core is the BERT family. These findings demonstrate the power of DeBERTa in terms of its disentangled attention structure and convolved positional encodings in fine-grained sentiment detection. Nevertheless, the framework was not tested on mobile app review datasets which are based on domain-specific language, informal language, and ambiguity of user intent. Therefore, the effectiveness of Instruct-DeBERTa in dealing with the real-life app review scenarios has not been tested yet, as this represents the potential limitation of transferability of this model to actual use.

Lee et al. [67] deployed sentiment analysis application in user reviews based on optimal Korean specific language models of KoBERT, KoGPT-2, and KoBART. Their study mainly aimed at evaluating the emotions written in the user reviews especially those in the Korean language. The authors within their models trained the hyperparameters to better perform in sentiment classification and follow common text preprocessing procedures by performing text tokenization and filtering stop words. These preprocessing operations were also critical in prepping mutilated text in a form that is suitable to the models.

The data that was utilized in the research comprised user reviews available on different platforms, but the overall amount of reviews analyzed was not mentioned. Through the utilization of KoBERT and KoGPT-2 and KoBART, the research proved the usefulness of models with Korean specific terms in sentiment analysis tasks. The sentiment of the user reviews was well classified through the optimized models and this proves the potential of these models in sentiment analysis in the Korean language.

The research was however not without gaps. Although it concentrated on Korean language models, it failed to research the use of multilingual models and models trained on a different language that would have improved the applicability of the model to global usage. Moreover, these Korean-specific models did not compare their performance with other popular powerful models (e.g. BERT or RoBERTa), which also could have resulted in more informative comparisons regarding their performance and functionality across languages and applications. Nugroho et al. [68] in their study named BERT Fine-Tuning for Sentiment Analysis on Indonesian Mobile App Reviews had the objective of determining the usefulness of fine-tuning bert-based sentiment analysis tasks to linguistic contexts, namely, Indonesian users of mobile applications. The study aimed at improving the sentimental evaluation of user feedback of

Google Play Store. The authors used two pre-trained BERT models, one multilingual model and another pre-trained model using Indonesian text. These models were tuned by a collection of Indonesian user comments on the 10 best mobile applications of 2020. The process of fine-tuning has included two labeling strategies, such as score-based and lexicon-based, and hyperparameter tuning was applied in order to fine-tune the performance of the models.

The dataset of the study was a collection of user reviews of the Google Play Store of the 10 most popular mobile applications in Indonesia in 2020. The reviews were in Bahasa Indonesia, and therefore represented the language of the main user population, which is why this dataset is very relevant to test the effectiveness of language-specific fine-tuning of BERT. They observed that the fine-tuned Indonesian-specific BERT model obtained an accuracy of 84% when trained using lexicon-based labeling compared to training multilingual BERT or classical machine learning models. The outcome showed that domain-specific pre-training proved to be beneficial when it comes to sentiment analysis, especially when we need to learn the specifics of the Indonesian language when it comes to user reviews.

However, the study had certain gaps. Although it targeted Indonesian user reviews, it did not discuss how it can apply multilingual sentiment analysis, which would have widened the scope of the model to be predicted in other languages. Also, the research lacked the comparison of the BERT model with other developed models like RoBERTa or DeBERTa, which were proven to perform well in sentiment analysis tasks as well. Moreover, the paper did not consider the problem of working with noisy or informal language that is prevalent in user-generated content and might interfere with the effectiveness of sentiment classification when applied to a real environment.

Large Language Models (LLMs) including BERT, RoBERTa, DeBERTa, and other hybrids have shown high levels of performance in their sentiment categorization as well as aspect-based sentiment classification in different realms of reviews. Reported accuracies vary between about 71 to 97 %, depending on the complexity of the data sets and their levels of classification tasks (e.g. binary or multi-class sentiment). These models present impressive advantages: they use potent contextual representations, they have pre-trained embeddings, which generalize across tasks, and they are sensitive to learning the fine-grained opinion about a certain aspect within the user-generated text. Nevertheless, there are still a number of weaknesses. Most studies have not used or tested these models on mobile app review data, which has a distinctive linguistic and domain-specific challenge. The models in some cases only concentrate on extracting features but not necessarily on the sentiment polarity classification and thus their

utility in providing sentiment insights are limited. Moreover, multilinguality may be stated to be present but the capacity may not be used or tested in some applications. Few studies have been able to formulate aspect-level sentiment extraction into a solid end-to-end pipeline to suit mobile app feedback. Such observations imply that the baseline developed by LLMs is solid, and there are obvious possibilities to expand into mobile-specific, multilingual, and aspect-aware sentiment systems.

To summarize, the use of various models, including both traditional Machine Learning (ML) and more sophisticated Deep Learning (DL) and Large Language Models (LLMs), has been highly useful in conducting a sentiment analysis of the Mobile Legends user reviews. Research on classical ML algorithms, including Support Vector Machines(SVM) and Naive Bayes, has proved the success of such features extraction techniques as TF-IDF and Word2Vec in sentiment classification. Although these methods are efficient in their own sense, they tend to be incapable of fully capturing the complexity and subtlety of informal user-created content, like slang and emotional style.

Sentiment analysis tasks have found better performance with Deep Learning (DL) models focused on Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs) that can capture long range dependencies and local features of the text. The combination of the models and the pre-trained embeddings such as GloVe and Word2Vec improved their usage in sentiment classification. Nevertheless, the DL models were still struggling with dealing with noisy data, as well as the cost of training it on massive datasets.

The advent of Large Language Models (LLMs) and in particular BERT and its derivations, such as BERT-like RoBERTa, KoBERT, and others has shifted sentiment analysis to a new dimension, introducing pre-trained models capable of more complex, context-aware tasks. LLMs have performed better in numerous studies than traditional ML and DL models and have high accuracy, can work well with multilingual data, subtle language contexts, and informal language use. Such models also take advantage of the benefits of transfer learning to pre-optimize, improving their performance over particular tasks, even in the absence of large data specific to the task.

Overall, the traditional models have a good reliable baseline, but the deep learning models and LLMs have their evident benefit, especially when dealing with large, complex data sets, such as the one in the present Mobile Legends user reviews. Future research can work further on combining these advanced models with specific domain optimizations, covering real-time deployment, multilingual capability, and how to detect sarcasm. With the ongoing development

of deep learning and LLM, they will become a pioneer in sentiment analysis in the gaming sector and other variants.

Table 3.1: Literature review summary table

Machine Learning Models	Study	Model	Dataset	Key Findings	Limitations / Gaps
	Jannah et al. [47]	SVM, Naive Bayes	Synthetic Shopee beauty product reviews	SVM achieved 86.14% accuracy; Naive Bayes 85.53%; TF-IDF + lemmatization effective	Synthetic dataset limits generalizability; no error analysis or feature importance; neutral class not fully addressed
	Juliane et al. [48]	Naive Bayes	Twitter comments	High training accuracy (93.33%), validation 86.66%; simple preprocessing effective	Small dataset; only NB tested; no comparison with other ML models
	Azeraf et al. [49]	Neural Naive Bayes (NNB)	IMDb movie reviews	Reduced error 4.5x vs NB; better on context-dependent and long reviews	Only binary sentiment; no comparison with SVM/LR; single dataset limits generalization
	Fashion e-commerce study [50]	SVM + SMOTE-Tomek	Fashion product reviews	Accuracy 92%, F1 0.89 with oversampling; baseline accuracy 68%	Binary classification only; neutral sentiment not studied; no comparison under same oversampling plan
	Wirawan et al. [51]	SVM + EDA	Instagram cyberbullying comments	Accuracy improved to 92% (+2.5% from baseline SVM); domain-specific improvement	No comparison with LR or Decision Tree baselines
	Özmen & Gündüz [52]	SVM, Logistic Regression, Decision Tree, Random Forest	Turkish e-commerce reviews	SVM performed best; LR decent across categories	No hyperparameter tuning; no interpretability analysis
	Danyal et al. [53]	Linear SVM, Multinomial NB, Logistic Regression, Passive Aggressive	IMDb 50K, Sentiment Polarity Dataset	LR slightly better (90.27%) than SVM (89.48%); simple linear models still effective	Only accuracy reported; no aspect-level sentiment; no multilingual or domain-specific testing

	Khan et al. [54]	SVM, Random Forest	News headlines, blog articles, social media posts	SVM slightly better (80.39%) vs RF (78.56%); RF more robust to noise	Only binary classification; no segment-level polarity; only TF-IDF features used
	Singhal et al. [55]	SVM, Naive Bayes, Logistic Regression, Random Forest (with BERT embeddings)	Amazon product reviews (fake vs genuine)	SVM best (87.81%); BERT embeddings improved semantic understanding	Synthetic/approximate data; no hyperparameter tuning; no error analysis or feature importance
Deep Learning Models	Ahammad et al. [56]	CNN, LSTM, BiLSTM, GRU, BiGRU, CNN+LSTM	122,231 AI app reviews (Google Play)	CNN+LSTM: 92.1%, LSTM: 91.97%; hybrid CNN+LSTM performed best; deep models learn contextual patterns well	Focuses on accuracy only; no per-class metrics; no transformer comparison; not lightweight for mobile
	Samanmali & Rupasingha [57]	ANN, LSTM, SVM	~33,000 reviews from 15 popular apps	LSTM: 80–90%; LSTM best across apps; highest F1-score	No hyperparameter tuning; no embedding/contextual features; long reviews not analyzed
	Amirkhalili & Wong [58]	LSTM, Naive Bayes	2,025 mobile banking reviews (Canada)	LSTM: 82%, NB: 77%; LSTM captures subtle sentiment patterns better	Small dataset; region-specific; no per-class metrics; no transformer benchmark
	Zaman et al. [59]	Neural networks (type unspecified)	XR app reviews (Google Play, 2025)	Not specified; deep learning better than ML baselines	Model type unclear; no detailed metrics; no SOTA comparison
	Ranjan & Mishra [60]	LSTM + GloVe embeddings	Google Apps reviews + Student App Reviews (SAR)	Accuracy: 95.2%, F1-score: 88; LSTM + embeddings outperforms traditional ML models	No transformer comparison; generalization to other apps/domains not evaluated
	Basri et al. [61]	Word2Vec + LSTM	100,000 Mobile Legends reviews (Bahasa Indonesia)	87.88%; most reviews positive; model captures semantic meaning via embeddings	No BERT/RoBERTa comparison; impact of preprocessing/noisy data not explored

Large Language Models	Eser & Şahin [62]	BERT, DistilBERT, RoBERTa, XLM-RoBERTa; VADER for sentiment score	Spotify app reviews from Google Play Store	DistilBERT: highest accuracy & recall (71.68%); XLM-RoBERTa: highest F1-score (69.24%)	Poor performance on neutral/underrepresented classes; no multilingual or real-time deployment
	Motger et al. [63]	T-FREX (token-level classification with BERT encoder)	Crowdsourced industrial dataset from Google Play Store	Improved feature-level precision and recall compared to baselines	Does not classify sentiment on extracted features; no aspect-level sentiment
	Nkhata et al. [64]	Hybrid: BERT + BiLSTM; data augmentation (SMOTE, NLPAUG)	IMDb (binary), SST-5 (five-level) datasets	IMDb: 97.67% accuracy; SST-5: 59.48% accuracy; better than standard BERT	Focused on movies, not app reviews; neutral/ambivalent sentiment not handled
	Chi et al. [65]	BERT embeddings + TCN + BiLSTM + Self-Attention	Hotel reviews from TripAdvisor, Booking.com	Outperformed baseline BERT in accuracy & F1; rich semantic representation	Limited to hospitality sector; no aspect-level sentiment or real-time deployment
	Ullah, W et al. [66]	Instruct-DeBERTa fine-tuned on SemEval datasets	Restaurant & laptop reviews (multilingual, multi-domain)	High performance in aspect term extraction & sentiment classification	Not tested on mobile app reviews; informal language handling unclear
	Lee et al. [67]	KoBERT, KoGPT-2, KoBART	User reviews in Korean	Effective Korean-specific models; better sentiment classification	No multilingual testing; no comparison with general LLMs (BERT, RoBERTa)
	Nugroho et al. [68]	Fine-tuned BERT (multilingual + Indonesian) with score-based and lexicon-based labeling	Google Play Store reviews of top 10 Indonesian apps	Indonesian BERT: 84% accuracy (lexicon-based)	No multilingual expansion; comparison with RoBERTa/DeBERTa missing; noisy/informal text not fully addressed

Chapter 4

Proposed Approach

4.1 Introduction

This chapter presents the methodological approach proposed to obtain the objectives of this research study. This paper aims to systematically analyze and compare the performance of classical Machine Learning (ML) classifiers against a latest pre-trained Transformer-based Large Language Model (LLM) with user-generated mobile gaming reviews. To achieve this goal, a structured methodological pipeline was developed. It covers dataset retrieval, extensive text processing, exploratory visual analytics, feature generation and selection process, model construction, LLM fine-tuning and hyperparameter tuning performed via gradient descent.

There is a trade-off between the classification objective, which may be more suited to modern LLM approaches., Our research goal is to prove that whether such LLM-based solutions are objectively better in terms of both accuracy and robustness than traditional ML classifiers in processing real-world noisy Google Play Store reviews [69]. Mobile gaming reviews may be informal, multilingual and highly variable in the linguistic styles, provide meaningful insights on both generalization performance and their applicability of ML and LLM models for our proposed problem statement.

In addition, all the methodological steps from preprocessing steps of data cleaning to final model training are performed on each model with scientific rigour, reproducibility and transparency. In summary this chapter gives full details of the approach proposed, which is the basis of experimental results discussed in later chapters.

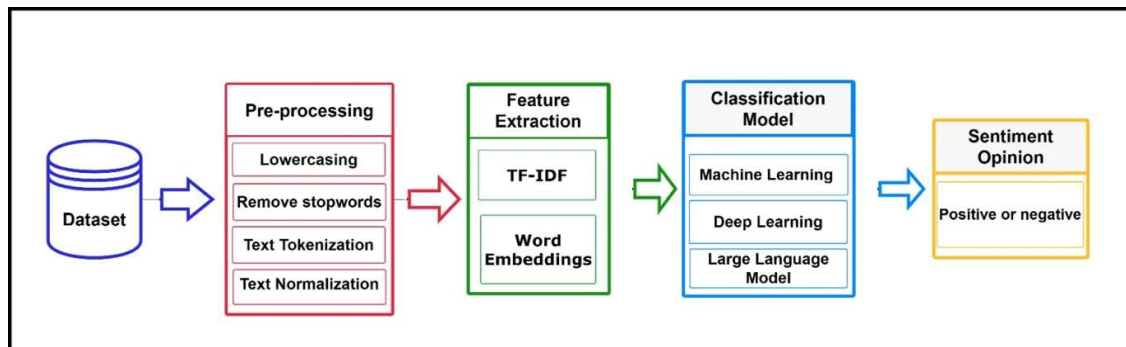


Figure 4.1: Model Training Pipeline of Proposed Methodology

4.2 Dataset Information

This study uses the Mobile Legends Google Play Reviews dataset [69]. The dataset contains 52,000 user reviews scraped from the Google Play Store. It has six columns, most of these columns are review text, rating, user id and date. The time range spans:

Start: February 16, 2025

End: June 10, 2025

Google Play Store reviews offer rich insights into user sentiment, perceptions of game balance and updates, user frustrations e.g., lag, bugs, monetization concerns, satisfaction drivers e.g., graphics, gameplay, rewards.

4.3 Preprocessing

A rigorous preprocessing pipeline was applied to convert raw user reviews into model-ready numerical representations. Text cleaning was performed to eliminate noise and standardize the user-generated text before any modelling. First, we lowered cased all text to eliminate case-based inconsistencies after that URLs, emojis, special characters and numerals were deleted using regular expressions. The cleaned text was tokenized with the function `word_tokenize` from NLTK to divide sentences into their tokens. Most frequent stop words such as “the,” and “is” were eliminated by the NLTK stopwords corpus to exclude non-informative terms. After that duplicate words and missing values were handled and punctuation marks and special symbols were deleted. Lastly, we lemmatized (WordNetLemmatizer) to bring words down to their base form e.g., “running” → “run” providing a more regular and meaningful representation for downstream machine-learning models. Sentiment labels as shown in Table 4.1 were generated based on Google Play Store rating logic.

Table 4.1: Class Label distribution based on ratings

Rating	Class Label
1-2 stars →	Negative
4-5 stars →	Positive

Thus, a new column label was added with categorical sentiment values. For ML pipelines, the notebook explicitly created as text → cleaned user review and label → sentiment category. These columns were used in all models SVM, Logistic Regression, RF and LLM

preprocessing. A word cloud was generated for frequently occurring cleaned tokens to visually display dominant themes in user reviews shown in Figure 4.2.



Figure 4.2: Word Cloud of Most Frequent Terms in Mobile Legends Reviews

A distribution analysis was performed using a Bar chart shown in Figure 4.3 to show the count of positive, neutral, and negative reviews.

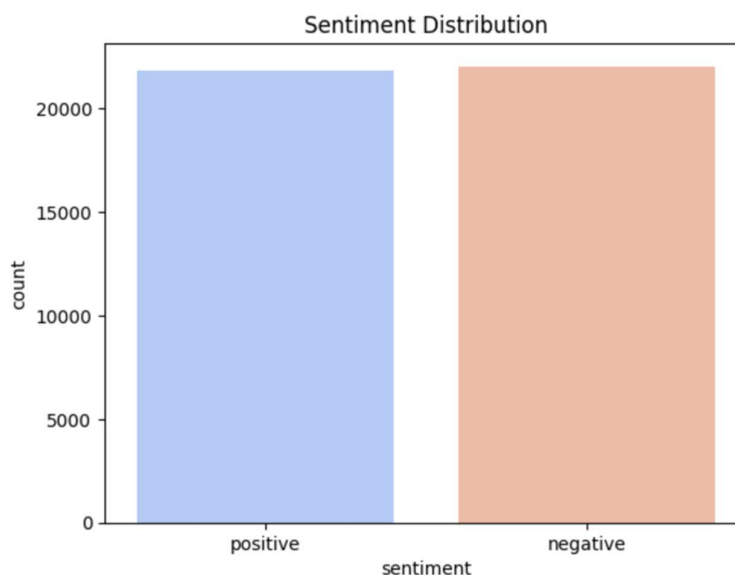


Figure 4.3: Bar Chart of Sentiment Category Distribution

Text was transformed into numeric features for machine-learning models using TF-IDF vectorization, which scored terms based on frequency in individual documents and rarity across the corpus. By choosing parameters such as `max_features` to cap the number of terms in the vocabulary, `ngram_range = (1,2)` to take unigrams and bigrams into account, `min_df = 5` to ignore terms that appear more rarely and gain a compact representation less sensitive to random noise. Lasso beta regression achieved both these objectives with good trade of reduces up to context. Grid Search was run using a TF-IDF + Logistic Regression to fine tune the modeling pipeline, with a systematic scan across combinations of vocabulary sizes (1000 features/words up to 3000) and regularization strengths $C = \{0.1, 1, 10\}$. By 3-cross-validation, GridSearchCV found values of hyperparameters that provide the best accuracy, leading to an experimentally verified and well-tuned configuration of feature-model for sentiment classification.

4.4 Machine Learning Models

In this section, we briefly describe all the machine-learning and deep-learning models that are trained in our experiments while focusing only on how they function and relate to TF-IDF or embedding-based sentiment classification.

4.4.1 Support Vector Machine (LinearSVC)

LinearSVC selects an optimal separating hyperplane to maximize the margin between positive and negative classes suiting for high-dimensional TF-IDF vectors.

Table 4.2: SVM model parameter

Parameter / Metric	Value
Vectorizer	TfidfVectorizer() (default parameters)
Train/Test Split	80% Train / 20% Test
Grid Search Parameters	C: [0.01, 0.1, 1, 10, 100] loss: ['hinge', 'squared_hinge']
Best Parameters (GridSearchCV)	C = 0.1, loss = 'squared_hinge'
Model Used	LinearSVC()

4.4.2 Logistic Regression

With its sigmoid / softmax function parameterization, Logistic Regression model can estimate class probabilities which is useful for TF-IDF features being linearly separable to distinguish classes.

Table 4.3: Logistic Regression model parameter

Parameter / Metric	Value
Vectorizer	TfidfVectorizer(max_features=10,000, ngram_range=(1, 2), stop_words='english')
Text Column	Cleaned
Label Column	Sentiment
Train/Test Split	80% Train / 20% Test (random_state=42)
Model	LogisticRegression()
C (Regularization Strength)	1.0
Penalty	l2
Solver	Liblinear
max_iter	2000

4.4.3 Random Forest Classifier

We used Random Forest Ensemble predictions across bootstrapped decision trees designed to decrease the high variance and improve robustness of noisy TF-IDF vectors.

Table 4.4: Random Forest model parameter

Parameter / Metric	Value
Vectorizer	TfidfVectorizer(max_features=15,000, ngram_range=(1,2), stop_words='english')
Text Column	Cleaned
Label Column	Sentiment
Train/Test Split	80% Train / 20% Test (random_state=42)

Model	RandomForestClassifier()
n_estimators	200
max_depth	None (fully grown trees)
min_samples_split	2
min_samples_leaf	1
random_state	42
n_jobs	-1 (uses all CPU cores)
Evaluation Metric	Test accuracy

4.4.4 LightGBM Classifier

LightGBM is another boosting algorithm based on gradient boosting that implements novel techniques such as GOSS and EFB, which can result in both fast training time but high accuracy on high-dimensional feature space like TF-IDF matrices.

Table 4.5: LightGBM model parameter

Parameter / Metric	Value
Vectorizer	TfidfVectorizer(max_features=15,000, ngram_range=(1,2), stop_words='english')
Text Column	Cleaned
Label Column	Sentiment
Train/Test Split	80% Train / 20% Test (random_state=42)
Model	LGBMClassifier()
n_estimators	500
learning_rate	0.05
max_depth	-1 (no depth limit)
num_leaves	64
subsample	0.8
colsample_bytree	0.8
objective	"binary" (since only two sentiment classes)
random_state	42
n_jobs	-1 (use all CPU cores)

4.4.5 Naïve Bayes Classifier

Multinomial Naïve Bayes calculates the probabilities of the class on term frequencies, and works well with sparse high-dimensional TF-IDF features. Its joint word with character-level n-grams preserve lexical and subword features shared across informal reviews.

Table 4.6: Naïve Bayes model parameter

Parameter / Component	Value
Vectorizer 1 (Word-level TF-IDF)	TfidfVectorizer(max_features=20,000, ngram_range=(1,2), stop_words='english')
Vectorizer 2 (Character-level TF-IDF)	TfidfVectorizer(analyzer='char', ngram_range=(3,5), max_features=20,000)
Merged Feature Space	hstack([word_TFIDF, char_TFIDF])
Text Column	Cleaned
Label Column	Sentiment
Train/Test Split	80% Train / 20% Test (random_state=42)
Model	MultinomialNB()
alpha (default)	1.0
fit_prior	True
Total Feature Dimensions	Word + Char TF-IDF combined $\approx$ up to 40,000 features

4.5 Large Language Model

4.5.1 DistilBERT-Base-Uncased

DistilBERT is a distilled version of BERT with ~97% of its effectiveness, smaller in size and faster training hence it can be applied to mid-size datasets. As the model is trained with small learning rates, dropout, label smoothing and early stopping, the framework excels in seizing contextual sentiment cues than traditional ML models.

Table 4.7: DistilBERT model parameter

Parameter / Component	Value
Base Model	distilbert-base-uncased
Task	Binary sentiment classification (positive / negative)

Input Text Column	Cleaned
Label Column	sentiment (after removing score = 3 and mapping $\leq 2 \rightarrow$ negative, $\geq 4 \rightarrow$ positive)
Train/Test Split	80% Train / 20% Test (seed=42)
Max Sequence Length	128 tokens
Tokenizer	AutoTokenizer.from_pretrained("distilbert-base-uncased") with padding = max_length, truncation = True
num_labels	2
dropout	0.20
attention_dropout	0.20
label_smoothing	0.10
Training Framework	Hugging Face Trainer
learning_rate	1e-5
warmup_ratio	0.10
lr_scheduler_type	Cosine
num_train_epochs	3
per_device_train_batch_size	16
per_device_eval_batch_size	16
gradient_accumulation_steps	2 (effective batch size 32)
weight_decay	0.15
eval_strategy	"steps" (every 100 steps)
save_strategy / save_steps	"steps", every 300 steps
metric_for_best_model	eval_accuracy (higher is better)
Early Stopping	EarlyStoppingCallback(early_stopping_patience=2)
Mixed Precision	fp16 = True

4.5.2 DeBERTa-v3-Base

DeBERTa-v3 promotes contextual understanding via disentangled attention and augmented mask decoding, which results in higher-quality semantics. Larger dropout rates, label smoothing and weight decay were utilized to prevent overfitting during fine tuning.

Table 4.8: DeBERTa-v3 model parameter

Parameter / Component	Value
Base Model	microsoft/deberta-v3-base
Task	Binary sentiment classification (positive / negative)
Input Text Column	Cleaned
Label Column	sentiment (class-encoded)
Train/Test Split	80% Train / 20% Test (seed=42)
Max Sequence Length	256 tokens
Tokenizer	AutoTokenizer.from_pretrained("microsoft/deberta-v3-base") with truncation=True, padding="max_length"
num_labels	2
hidden_dropout_prob	0.30
attention_probs_dropout_prob	0.30
label_smoothing_factor	0.15
Training Framework	Hugging Face Trainer
learning_rate	1e-5
warmup_ratio	0.10
weight_decay	0.18
lr_scheduler_type	Cosine
num_train_epochs	7
per_device_train_batch_size	8
per_device_eval_batch_size	16
gradient_accumulation_steps	2 (effective batch size 16)
eval_strategy	"steps", every 350 steps
save_strategy / save_steps	"steps", every 350 steps
logging_steps	80
metric_for_best_model	eval_accuracy (greater_is_better=True)
Early Stopping	EarlyStoppingCallback(early_stopping_patience=2)

Mixed Precision	fp16 = True
-----------------	-------------

4.5.3 RoBERTa-Base

RoBERTa further improves the BERT by discarding NSP, applying dynamic masking and training with optimized batch schedules for more powerful contextual encoding.

Table 4.9: RoBERTa model parameter

Parameter / Component	Value
Base Model	roberta-base
Task	Binary sentiment classification (positive / negative)
Input Text Column	Cleaned
Label Column	sentiment (class-encoded)
Train/Test Split	80% Train / 20% Test (seed=42)
Max Sequence Length	256 tokens
Tokenizer	AutoTokenizer.from_pretrained("roberta-base") with truncation=True, padding="max_length"
num_labels	2
hidden_dropout_prob	0.25
attention_probs_dropout_prob	0.25
label_smoothing_factor	0.12
gradient_checkpointing	True
Training Framework	Hugging Face Trainer
learning_rate	8e-6
warmup_ratio	0.10
lr_scheduler_type	Cosine
per_device_train_batch_size	16
per_device_eval_batch_size	16
gradient_accumulation_steps	2 (effective batch size 32)
num_train_epochs	8
weight_decay	0.15
eval_strategy	"steps" every 300 steps
save_strategy / save_steps	"steps", every 300 steps
logging_steps	60

metric_for_best_model	eval_accuracy (greater_is_better=True)
Early Stopping	EarlyStoppingCallback(early_stopping_patience=2)
Mixed Precision	fp16 = True

4.5.4 BERT-Base-Uncased

BERT utilizes bidirectional Transformer encoders pre-trained with MLM and NSP, which allows the model to effectively learn contextual representation.

Table 4.10: BERT-Base-Uncased model parameter

Component	Setting / Value
Base model	roberta-base
Task	Binary sentiment classification (positive / negative)
Input text column	Cleaned
Label column	sentiment (class-encoded)
Label rule	negative if score ≤ 3 , else positive
Train/test split	80% train / 20% test (seed=42)
Preprocessing	Lowercasing, emoji removal (regex), punctuation & special-char removal, whitespace normalization, NLTK stopword removal, duplicate removal on content
Tokenizer	AutoTokenizer.from_pretrained("roberta-base")
Max sequence length	256 tokens (padding="max_length", truncation=True)
num_labels	2
hidden_dropout_prob	0.25
attention_probs_dropout_prob	0.25
label_smoothing_factor	0.12
Metric	Accuracy (evaluate.load("accuracy"))
Learning rate	8e-6
Warmup ratio	0.10
LR scheduler	Cosine
Weight decay	0.15
Epochs	8
Per-device train batch size	16
Per-device eval batch size	16

Gradient accumulation steps	2 (effective batch size 32)
Evaluation strategy	"steps" (every 300 steps)
Save strategy / steps	"steps", every 300 steps
Logging steps	60
Early stopping	EarlyStoppingCallback(early_stopping_patience=2)
Mixed precision	fp16=True
Best-model criterion	load_best_model_at_end=True, metric_for_best_model="eval_accuracy", greater_is_better=True
Output directory	"sentiment_roberta"

4.5.5 ELECTRA-Large-Discriminator

Replaced Token Detection allows ELECTRA to be trained more effectively than traditional masked language models and retains the ability to model the entire distribution of linguistic context for words without making independence assumptions between predictions.

Table 4.11: ELECTRA-Large Discriminator model parameter

Component	Setting / Value
model_name	"google/electra-base-discriminator"
num_labels	2
max_length	256
padding	"max_length"
truncation	True
test_size	0.2
hidden_dropout_prob	0.30
attention_probs_dropout_prob	0.30
classifier_dropout	0.30
weight_decay	0.15
label_smoothing_factor	0.12
learning_rate	1.5e-5
warmup_ratio	0.15
lr_scheduler_type	"cosine"
num_train_epochs	8
per_device_train_batch_size	16

gradient_accumulation_steps	2
eval_strategy	"steps"
eval_steps	250
save_strategy	"steps"
save_steps	250
load_best_model_at_end	True
metric_for_best_model	"eval_accuracy"
greater_is_better	True
early_stopping_patience	2
output_dir	"sentiment_electra"
logging_steps	60
fp16	True

4.6 Deep Learning Models

4.6.1 Bi-LSTM

This model utilizes 100-dimensional GloVe embeddings and a bi-GRU layer to encode forward-backward context information from user reviews. It used a custom attention mechanism to focus on sentiment-aware tokens and applied dense layers in combination with dropout for stable classification.

Table 4.12: Bi-LSTM model parameter

Component	Setting/Value
MAX_WORDS	20000
MAX_LEN	120
test_size	0.2
random_state	42
Embedding Type	GloVe 6B
EMB_DIM	100
Trainable	False
Type	Bidirectional GRU
Units	64
Dropout	0.3

return_sequences	True
Type	Custom Layer
Dense Layer 1 units	64
Dense Layer 1 activation	"relu"
Dropout Layer rate	0.4
Output Layer units	1
Output Layer activation	"sigmoid"
Loss	"binary_crossentropy"
Optimizer	"adam"
Metrics	["accuracy"]
Epochs	12
batch_size	64
validation_split	0.2
Monitor	"val_loss"
Patience	2
restore_best_weights	True

4.6.2 CNN + BiGRU Hybrid Model

The hybrid model integrates multi-kernel 3,5,7 filters CNNs to capture local n-gram patterns and a BiGRU layer expressing long-range dependencies. After the global pooling, batch normalization and dropout are applied to stabilize learning along with the GloVe embeddings providing semantic classification.

Table 4.13: CNN + BiGRU Hybrid model parameter

Component	Setting/Value
MAX_WORDS	16000
MAX_LEN	120
oov_token	"<UNK>"
test_size	0.2
random_state	42
Embedding Type	GloVe 6B
EMB_DIM	100
trainable (Initial)	False

trainable (Second Stage)	True
SpatialDropout1D	0.25
Filter units	128 (x3)
Kernel size	3, 5, 7
activation	"relu"
Concatenate	N/A
MaxPooling1D size	2
Type	Bidirectional GRU
units	64
dropout	0.3
recurrent_dropout	0.3
return_sequences	True
Pooling Layers	GlobalMaxPooling1D, GlobalAveragePooling1D
Output Concatenate	N/A
Dense 1 units	128
Dense 2 units	64
kernel_regularizer	l2(1e-4)
BatchNormalization	N/A
Dropout 1 rate	0.5
Dropout 2 rate	0.35
Output Layer units	1
Output Layer activation	"sigmoid"
optimizer	AdamW
learning_rate	2e-4
weight_decay	1e-4
epochs	15
validation_split	0.2
learning_rate	1e-4
epochs	5
EarlyStopping patience	3
RestoreBestWeights	True
ReduceLROnPlateau patience	1
factor	0.4

min_lr	1e-6
--------	------

4.7 Summary

In this chapter, we outlined the complete procedure employed in sentiment classification of reviews on Mobile Legends dataset. After extensive pre-processing of dataset, three modelling levels were applied, (1) classical machine learning baselines with TF-IDF includes SVM, Logistic Regression, Naive Bayes, Random Forest, LightGBM, (2) deep-learning models incorporating semantic embeddings and sequence learning e.g., Bi-GRU with attention, CNN + BiGRU hybrid and (3) modern transformer-based LLMs DistilBERT; BERT-base; RoBERTa-base; DeBERTa-v3-base; ELECTRA-discriminator. Each model was tuned to reasonable values of hyperparameters and regularizations, and the performance was measured on held out test data in terms of accuracy on classification. By designing this multi-level methodological framework, we also provide strong empirical evidence to compare the three types of architectures i.e., traditional, deep and transformer-based architecture, which makes our results and analysis robust.

Chapter 5

Experimental Results and Discussion

5.1. Introduction

In this chapter, we describe and discuss the experimental results for machine learning (ML), deep learning (DL) and transformer-based models presented in this work. The experiments are performed on a large text dataset containing 41,957 samples and 8779 testing samples. The primary goal was to assess and compare predictive results of conventional machine learning models, neural network architectures and the latest transformer-based models for the target text classification task.

Across all three model categories using consistent evaluation metrics, including accuracy, precision, recall, and F1-score. Comparative tables and visualizations are used to jointly highlight performance trends, strengths, and limitations across ML, DL, and LLM approaches. This structure allows clear identification of performance gains achieved by increasingly sophisticated modeling paradigms. The chapter concludes with a detailed discussion of the findings, followed by an examination of the limitations of the proposed work and a concise summary of key insights.

5.2. Results of Individual Models

The model-wise performance is presented in terms of classifier accuracy with precision, recall and F1-score. These models are strong baselines and can be compared with state-of-the-art transformer-based architectures.

5.2.1. Traditional Machine Learning Models

Logistic Regression with Grid Search and TF-IDF data provided good results in the high dimensionality text datasets. The parameters Regularization parameter $C=1$ and a TF-IDF vocabulary size of 3,000 contributed to stable and balanced performance. The Support Vector Machine performed well. The best SVM model had an accuracy of 0.89, with both classes being equally represented by precision and recall. It was able to effectively address sparse representations of text by maximizing the margin between classes.

For LightGBM, the accuracy was 0.89 seems to perform well, however, it is slightly lower than that for linear models. Although LightGBM is good with structured data, its performance implies a weak use of context information from text features. Naïve Bayes was very simple but performed quite well (accuracy = 0.89), especially for negative class recall. But its strong independence assumption limited the overall power of classification.

The Random Forest algorithm achieved an accuracy of 0.88, because the ensemble learning it adopted but showing reduced effectiveness compared to linear classifiers due to the high dimensionality of TF-IDF features.

Table 5.1: Performance Metrics of ML Models

Model	Precision	Recall	F1-Score	Accuracy
Logistic Regression	0.89	0.89	0.89	0.89
SVM (Optimized)	0.90	0.89	0.89	0.89
LightGBM	0.89	0.89	0.89	0.89
Naïve Bayes	0.89	0.88	0.88	0.88
Random Forest	0.88	0.88	0.88	0.88

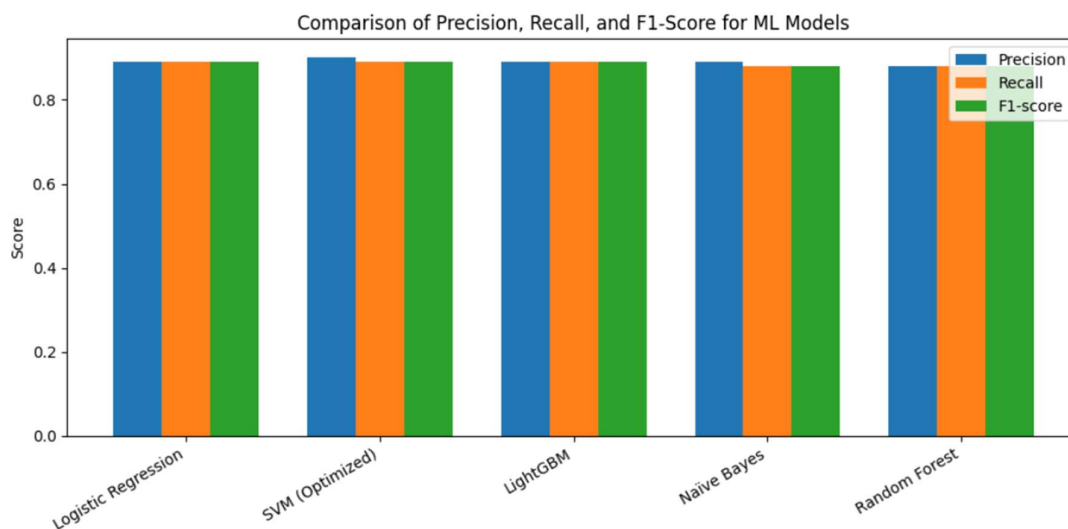


Figure 5.1 Comparison of precision, recall, and F1-score for machine learning models

The performance of the evaluated machine learning models in terms of precision, recall, and F1-score enables a better understanding of their classification capabilities beyond global accuracy. The focused SVM model obtained the most precision (0.90), This is an indication that SVM performs very well in separating class boundaries when used to model high-dimensional TF-IDF feature spaces.

5.2.2. Deep Learning Models

The BiLSTM model and the CNN + BiGRU had 0.8779 and 0.8743 accuracy, respectively shown in Table 5.2. Their accuracy and loss graphs are shown in Figure 5.2, and 5.3. The results show that the sequential neural architectures achieved better contextual understanding than the traditional ML models, yet they still underperformed in comparison to transformer-based models which can model long range dependencies.

Table 5.2: Performance results of Deep Learning Models

Model	Precision	Recall	F1-Score	Accuracy
Bi-LSTM	0.8798	0.8793	0.8780	0.8792
CNN + BiGRU Hybrid	0.8779	0.8779	0.8779	0.8778

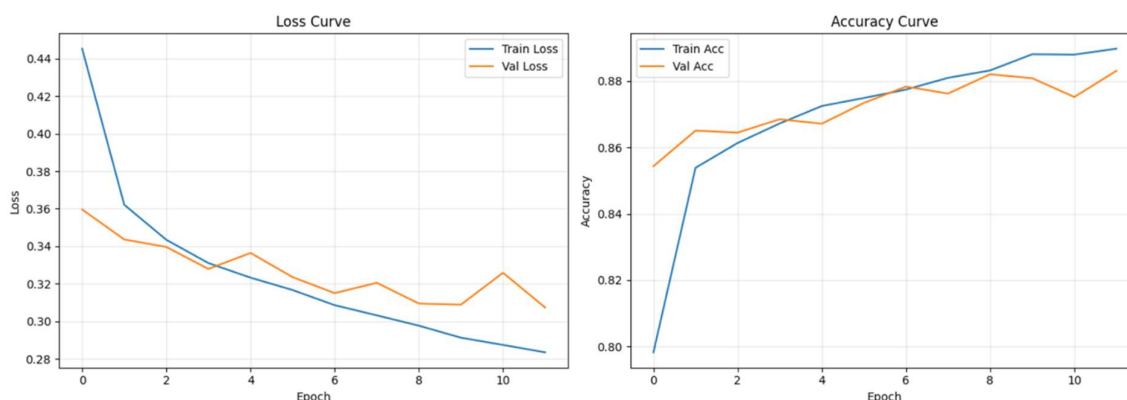


Figure 5.2. Training and validation loss and accuracy curves of the Bi-LSTM model.

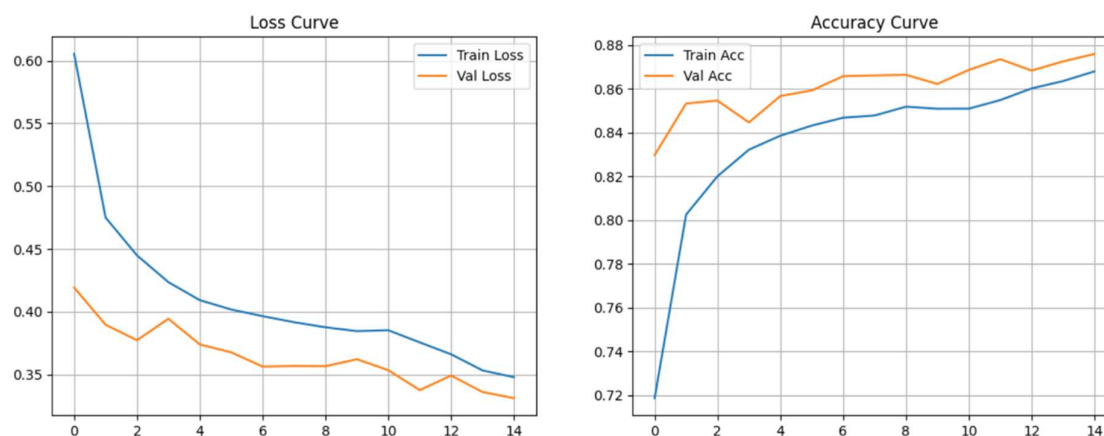


Figure 5.3. Training and validation accuracy and loss of the CNN-BiGRU model.

5.2.3. Results of Transformer-Based Models

Transformer-based models performed much better than classical ML and DL techniques because they could learn contextual semantics through self-attention mechanisms. The considered models are DistilBERT, RoBERTa, ELECTRA, and DeBERTa-v3. The performance results of LLMs are summarized in Table 5.3.

Table 5.3: Performance results of Transformer-Based Models

Model	Precision	Recall	F1-Score	Accuracy
Distilbert-Base-Uncased	0.9019	0.9014	0.9012	0.9013
DEBERTA-V3-BASE	0.9179	0.9164	0.9162	0.9164
Roberta-Base	0.9038	0.9038	0.9038	0.9037
Electra-Large-Discriminator	0.8990	0.8989	0.8988	0.8988

DistilBERT, a lightweight transformer model scored around 0.9013 accuracy, accuracy and loss curves are shown in Figure 5.4, surpassing all classical ML models with more computationally efficient. DEBERTA-V3 Figure 5.5 achieved the highest precision rate of 0.9164 due to robust pretraining and tokenization attack strategies optimization. ELECTRA obtained test accuracy of 0.8988 Figure 5.7, competitive with but slightly below RoBERTa in Figure 5.6 due to its discriminator-based training nature.

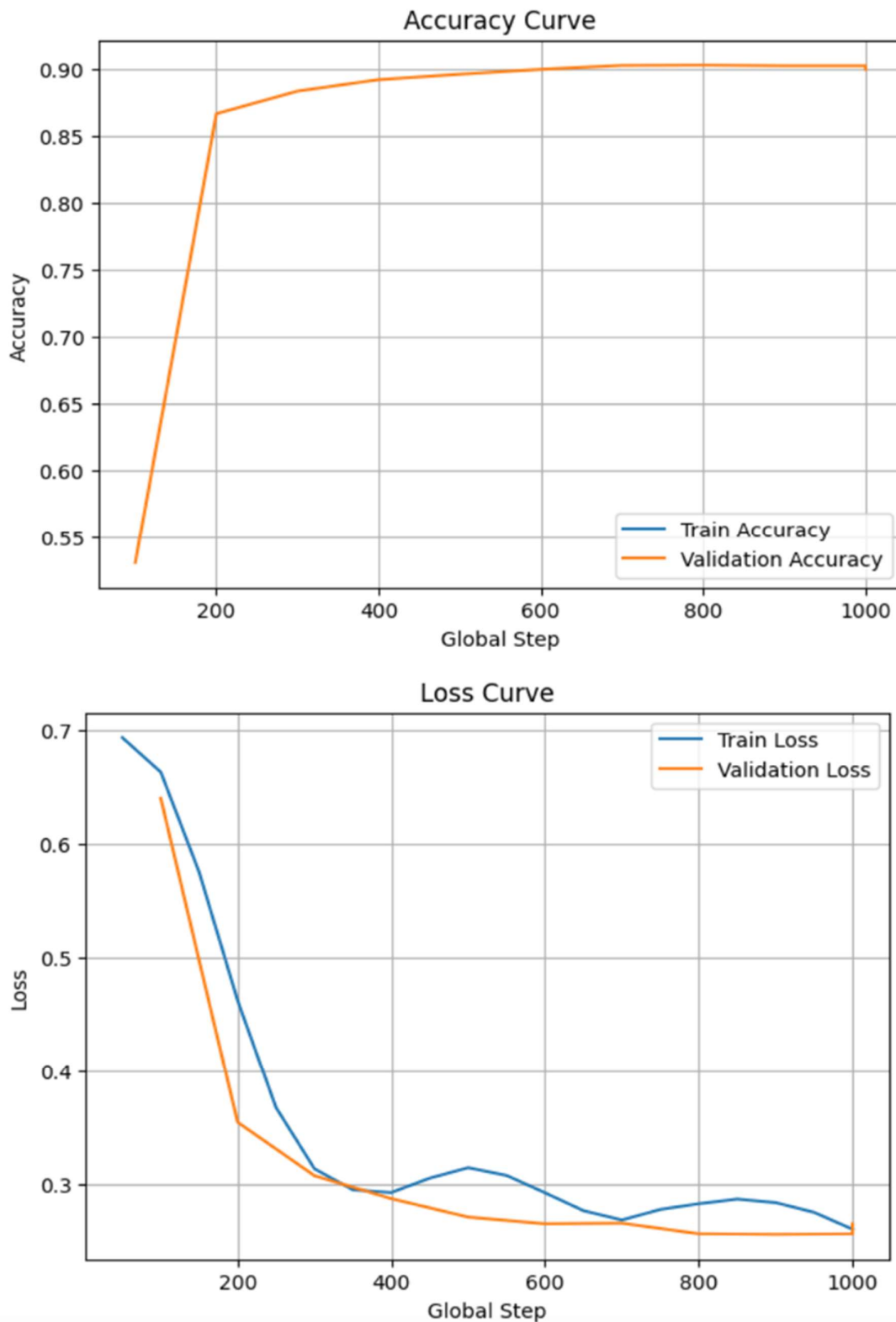


Figure 5.4. Training and validation accuracy and loss of the DistilBERT model.

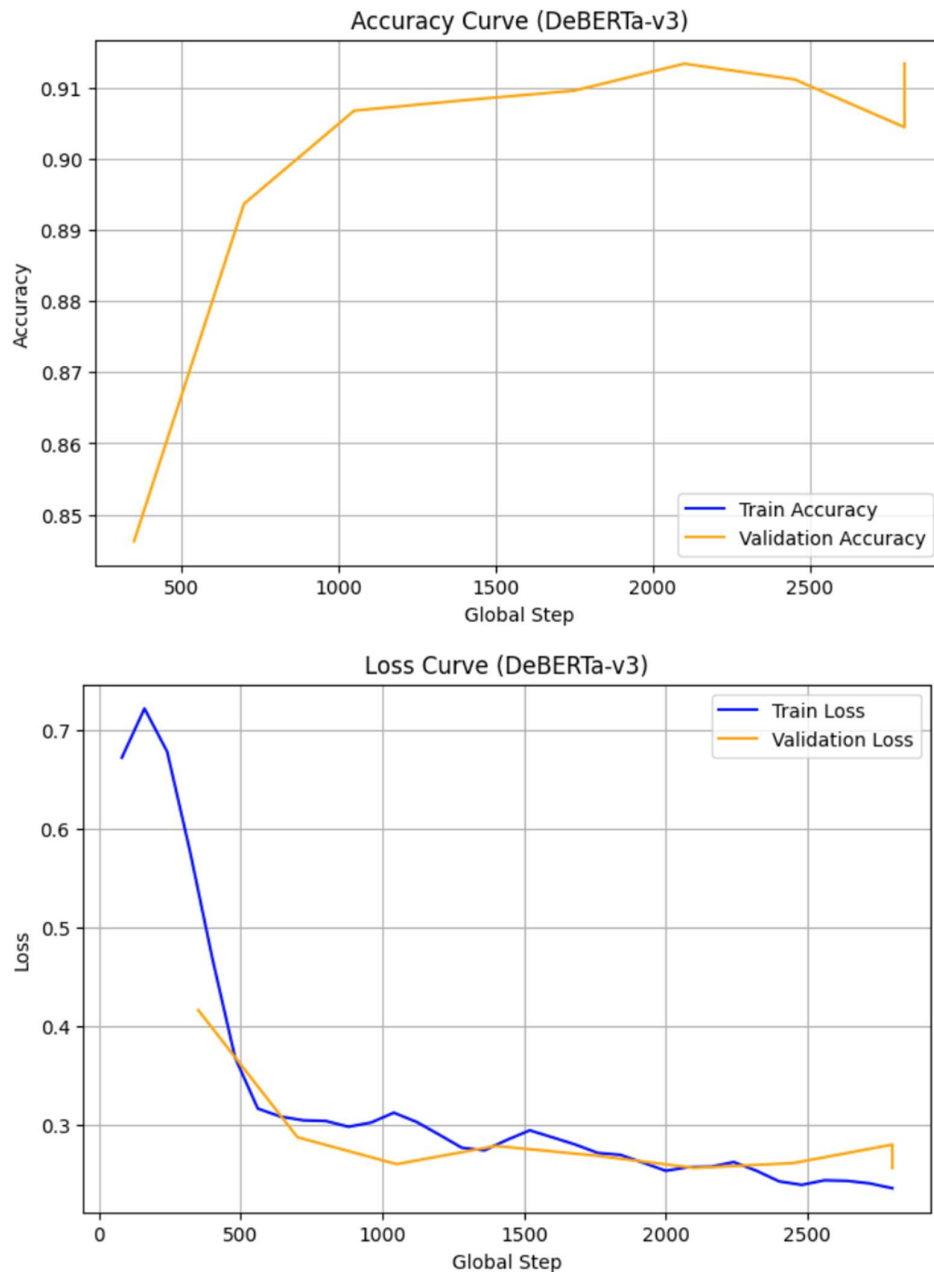


Figure 5.5. Training and validation accuracy and loss of the DeBERTa-v3 model.

The DeBERTa-v3 model reached a maximum test accuracy of 0.9164, which is the best performance obtained in this study shown in Figure 5.5. Its decoupled attention mechanism and strengthened context encoding remarkably increased classification performance.

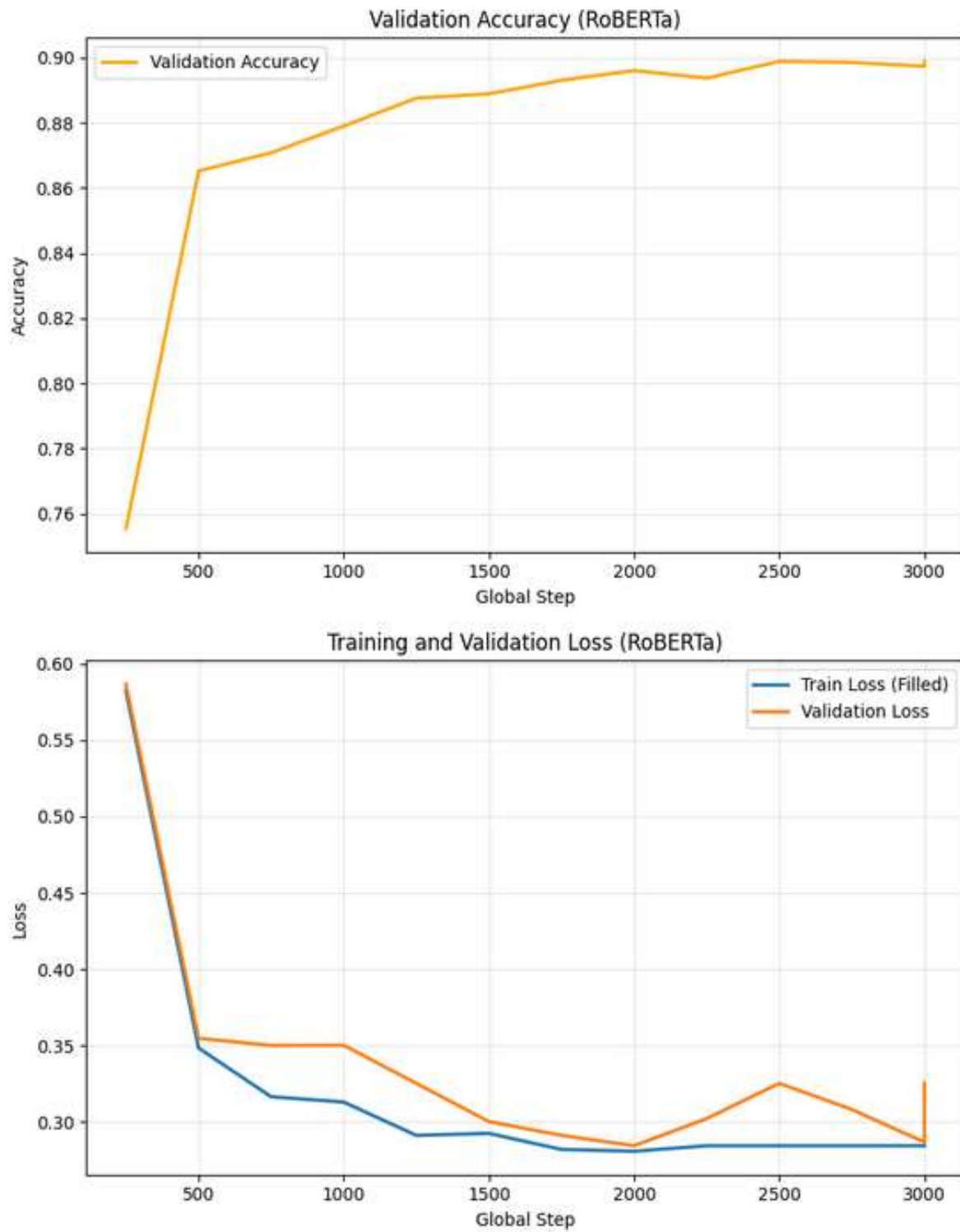


Figure 5.6. Training and validation accuracy and loss of the RoBERTa model.

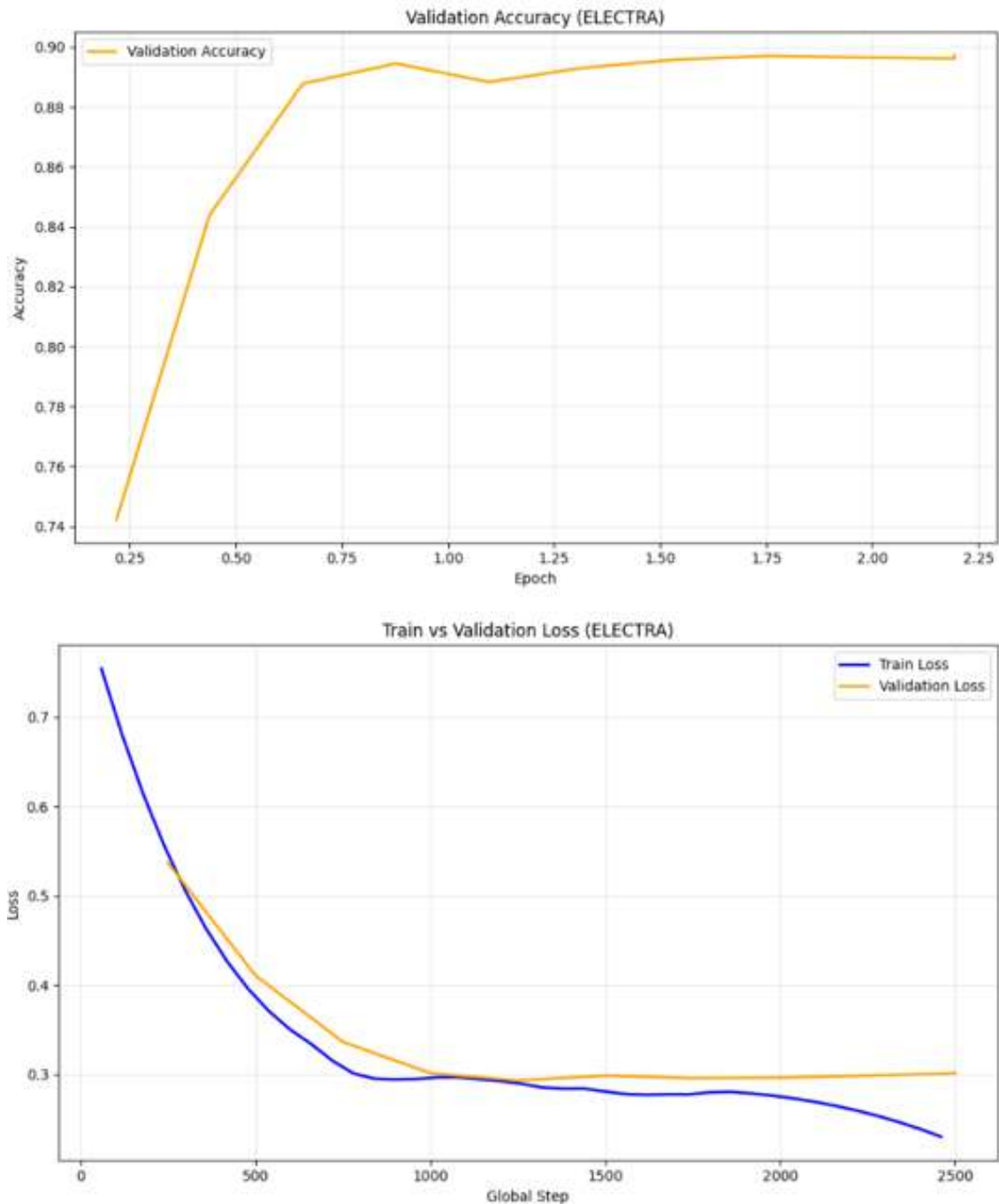


Figure 5.7. Training and validation accuracy and loss of the ELECTRA model.

5.3. Discussion of Findings

The experimental results show that transformer-based models are indeed more effective compared to traditional ML and DL models. Although efficient in handling the sparse TF-IDF features, linear models like logistic Regression and SVM also suffer from the inability to capture deep contextual relationships.

CNN and Bi-LSTM-based models inferred semantic meaning from word sequences and although they could not fully capture long-range dependencies, their performance was worse than transformers. DeBERTa-v3 is the best-performing model for all models, with accuracy = 0.9164, training & validation Loss of DeBERTa-v3 are presented in Figure 5.5. The result proves that disentangled attention and improved contextual embedding are beneficial to a better answer prediction.

The overall performance results of all models as shown in Figure 5.8 indicate that leveraging pretrained language models is crucial for achieving high performance in complex text classification tasks, particularly when contextual nuance plays a significant role.

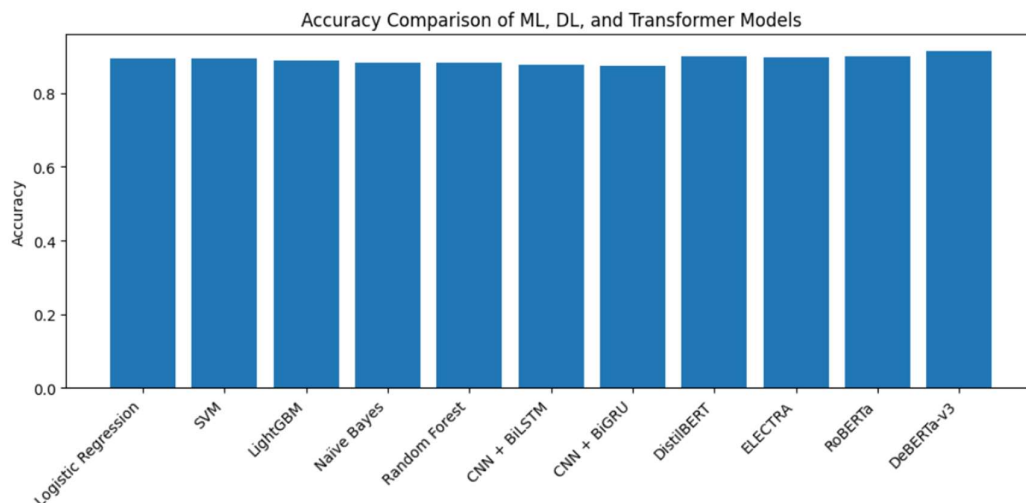


Figure 5.8. Accuracy comparison of machine learning, deep learning, and transformer-based models.

5.4. Limitations of the Proposed Work

Despite strong performance, several limitations remain. Firstly, pre-trained transformer models demand a lot of computational resources and they can't be used in real-time. Secondly, the dataset lacks temporal variation, preventing the application of sequence-based temporal modeling. Thirdly, the hyperparameter tuning of transformers is computational demanding and strict to training settings. Lastly, here we only consider the supervised learning and leave semi-supervised or unsupervised solutions which may further improve generalization.

5.5. Summary of the Chapter

This chapter presented an extensive experimental comparison of ML, DL and, LLM models for text classification. All models were evaluated with a common framework to provide fair and consistent comparison based on standard evaluation measurements. At baseline, traditional ML models LR and optimized SVM showed strong results of up to 89% accuracy on high performance using high dimensional textual representations. Deep architectures, such as Bi-LSTM and CNN-BiGRU, led to marginal gains in contextual learning but weaker identification accuracies ~88%, thereby demonstrating little capacity of modeling long-range dependencies.

Models based on Transformers greatly outperformed those of the ML and DL models by usage of their self-attention architectures and the availability of pretrained contextual representations. DeBERTa-v3 performed the best among them, with an accuracy of 91.64% and stable, balanced classification according to confusion matrix analysis. Precision, recall, and F1-score estimations also affirms the classical ML models specifically SVM were robust but transformers superior predictive capacity.

In general, the results are very clear that attention-based transformer models outperform when capturing semantics and context in textual data. These findings confirm the proposed modeling method and contribute to the implications of the conclusions and future work presented in the next chapter, especially for optimizing, scaling, deploying models and how they can be used in real-world applications.

References

- [1] Ballas, V., Michalakakis, K., Alexandridis, G., & Caridakis, G. (2024). Automating mobile app review user feedback with aspect-based sentiment analysis. *Lecture Notes in Computer Science*, 179–193.
- [2] Maroof, A., Wasi, S., Jami, S. I., & Siddiqui, M. S. (2024). Aspect-based sentiment analysis for the service industry. *IEEE Access*, 12, 109702–109713.
- [3] Adiningtyas, H., & Auliani, A. S. (2024). Sentiment analysis for mobile banking service quality measurement. *Procedia Computer Science*, 234, 40–50.
- [4] Sivakumar, M., & Uyyala, S. R. (2021). Aspect-based sentiment analysis of mobile phone reviews using LSTM and fuzzy logic. *International Journal of Data Science and Analytics*, 12(4).
- [5] Qahar, A., Ruldeviyani, Y., Mukharomah, U. N., Fidyawan, M. A., & Putra, R. (2024). Factor analysis influencing mobile JKN user experience using sentiment analysis. *IAES International Journal of Artificial Intelligence*, 13(2), 1782–1793.
- [6] Liu, B. (2020). *Sentiment analysis: Mining opinions, sentiments, and emotions* (2nd ed.). Cambridge University Press.
- [7] Thelwall, M., Buckley, K., & Paltoglou, G. (2012). Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology*, 63(1), 163–173.
- [8] Rana, A., & Singh, S. (2014). Comparative analysis of sentiment orientation using SVM and Naïve Bayes techniques. *International Journal of Computer Applications*, 99(16), 1–6.
- [9] Liu, T. Y., Yang, Y., Wan, H., Li, H., & Zhang, X. (2018). Opinion mining and sentiment analysis in social media: Challenges and solutions. In *Proc. IEEE Int. Conf. Data Mining Workshops* (pp. 19–26).
- [10] Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2), 1–135.
- [11] Sammons, M., Christodoulopoulos, C., Kordjamshidi, P., Khashabi, D., Srikumar, V., & Roth, D. (2016). EDISON: Feature extraction for NLP, simplified. *Language Resources and Evaluation*, 4085–4092.
- [12] Christian, H., Agus, M. P., & Suhartono, D. (2016). Single document automatic text summarization using term frequency-inverse document frequency (TF-IDF). *ComTech Computer Mathematics and Engineering Applications*, 7(4), 285.

- [13] Mujahid, M., et al. (2021). Sentiment analysis and topic modeling on tweets about online education during COVID-19. *Applied Sciences*, 11(18), 8438.
- [14] Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 1, 173–180.
- [15] Tripathy, A., Agrawal, A., & Rath, S. (2016). Classification of sentiment reviews using n-gram machine learning approach. *Expert Systems with Applications*, 57, 117–126.
- [16] Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57, 345–420.
- [17] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- [18] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proc. 2014 Conf. Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543).
- [19] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135–146.
- [20] Peters, M., et al. (2018). Deep contextualized word representations. In *Proc. 2018 Conf. NAACL-HLT* (pp. 2227–2237).
- [21] Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 79–86).
- [22] Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML)* (pp. 137–142).
- [23] McCallum, A., & Nigam, K. (1998). A comparison of event models for naïve Bayes text classification. In *Proceedings of the AAAI Workshop on Learning for Text Categorization* (pp. 41–48).
- [24] Mahto, A. K. (2021, September 7). Demystifying Naïve Bayes: Simple yet powerful for text classification. *Medium*.
- [25] Mahto, A. K. (2021, September 7). Demystifying Naïve Bayes: Simple yet powerful for text classification. *Medium*.
- [26] ResearchGate. (n.d.). Decision tree used for sentiment classification. *ResearchGate*.

- [27] Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML)* (pp. 137–142).
- [28] Medium. (n.d.). Sentiment Analysis Using SVM. *ScrapeHero*.
- [29] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32.
- [30] Jurafsky, D., & Martin, J. H. (2023). *Speech and language processing* (3rd ed., draft). Pearson.
- [31] Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference on NAACL-HLT* (pp. 4171–4186).
- [32] Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1746–1751).
- [33] Data Science Today. (n.d.). Sentiment analysis: CNN vs LSTM. *Data Science Today*.
- [34] Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Proceedings of INTERSPEECH* (pp. 1045–1048).
- [35] GeeksforGeeks. (n.d.). Introduction to recurrent neural network. *GeeksforGeeks*.
- [36] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.
- [37] LSTM. (n.d.). LSTM. *Dive into Deep Learning*.
- [38] Cho, K., et al. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734).
- [39] ResearchGate. (n.d.). GRU model structure (GRU: Gated recurrent neural network). *ResearchGate*.
- [40] Brown, T., et al. (2020). Language models are few-shot learners. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 33, 1877–1901.
- [41] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)* (pp. 4171–4186).
- [42] DeLucia, S. A. (n.d.). Unleashing BERT: Transformer model for NLP. *Arize*.

- [43] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8), 9.
- [44] Liu, Y., et al. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.
- [45] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper, and lighter. *arXiv preprint arXiv:1910.01108*.
- [46] Yang, Z., et al. (2019). XLNet: Generalized autoregressive pretraining for language understanding. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)* (pp. 5754–5764).
- [47] Jannah, N. Z. B., & Kusnawi, K. (2024). Comparison of Naïve Bayes and SVM in sentiment analysis of product reviews on marketplaces. *Sinkron: Jurnal dan Penelitian Teknik Informatika*, 8(2), 727–733.
- [48] Juliane, C. (2021, November). Implementation of naive bayes algorithm on sentiment analysis application. In *2nd International Seminar of Science and Applied Technology (ISSAT 2021)* (pp. 193-200). Atlantis Press.
- [49] Azeraf, E., Monfrini, E., & Pieczynski, W. (2021). Improving usual Naive Bayes classifier performances with Neural Naive Bayes based models. *arXiv preprint arXiv:2111.07307*.
- [50] Sumantiawan, D. I., Suseno, J. E., & Syafei, W. A. (2023). Sentiment analysis of customer reviews using Support Vector Machine and SMOTE-Tomek Links for identifying customer satisfaction. *Jurnal Sistem Informasi Bisnis*, 13(1), 1–9.
- [51] Wirawan, A., & Cahyono, H. D. (2023). Easy data augmentation in sentiment analysis of cyberbullying. In *Proceedings of the 6th International Conference on Information and Communications Technology (ICOIACT)*, Yogyakarta, Indonesia (pp. 443–447).
- [52] Özmen, C. G., & Gündüz, S. (2025). Comparison of machine learning models for sentiment analysis of big Turkish web-based data. *Applied Sciences*, 15(5), 2297.
- [53] Danyal, M., Khan, S., Khan, M., Ghaffar, M., Khan, B., & Arshad, M. (2023). Sentiment analysis based on performance of linear support vector machine and multinomial Naïve Bayes using movie reviews with baseline techniques. *Journal on Big Data*, 5, 1.
- [54] Khan, T. A., Sadiq, R., Shahid, Z., Alam, M. M., & Su'ud, M. M. (2024). Sentiment analysis using support vector machine and random forest. *Journal of Informatics and Web Engineering*, 3(1), 67–75.

- [55] Singhal, S., Majeed, H. L., Ibrahim, H. M., Jatana, N., Gupta, C., Kuma, A., & Hassen, O. A. (n.d.). Sentiment analysis on Amazon reviews of mobile phones using machine learning. *Technology*, 15, 19.
- [56] Ahammad, S., Sinthia, S. A., Ahmed, M., Hossain, M., Asif, N. A. A., & Ikram, N. A. (2024). Deep learning-based sentiment analysis of user generated reviews of various AI powered mobile applications. In *Proceedings of the 2024 International Conference on Inventive Computation Technologies (ICICT)* (pp. 505–512).
- [57] Samanmali, P. H. C., & Rupasingha, R. A. H. M. (2024). Sentiment analysis on Google Play Store app users' reviews based on deep learning approach. *Multimedia Tools and Applications*, 83(36), 84425–84453.
- [58] Amirkhalili, Y., & Wong, H. Y. (2025). Banking on feedback: Text analysis of mobile banking iOS and Google app reviews. *arXiv preprint arXiv:2503.11861*.
- [59] Zaman, N., Goldberg, D. M., Qiao, Z., & Abrahams, A. S. (2025). More than meets the eye: Feature concerns and suggestions in mobile XR app reviews. *Decision Support Systems*, 114475.
- [60] Ranjan, S., & Mishra, S. (2022). Perceiving university students' opinions from Google app reviews. *Concurrency and Computation: Practice and Experience*, 34(10), e6800.
- [61] Basri, N. F., & Utami, E. (2025). Application of Word2Vec and LSTM models in sentiment analysis of Mobile Legends user reviews. *Sistemasi: Jurnal Sistem Informasi*, 14(2), 856–871.
- [62] Eser, G., & Sahin, C. (2024). Sentiment analysis and rating prediction for app reviews using transformer-based models.
- [63] Motger, Q., Miaschi, A., Dell'Orletta, F., Franch, X., & Marco, J. (2024). T-FREX: A transformer-based feature extraction method from mobile app reviews. In *Proceedings of the 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)* (pp. 227–238).
- [64] Nkhata, G., Gauch, S., Anjum, U., & Zhan, J. (2025). Fine-tuning BERT with Bidirectional LSTM for fine-grained movie reviews sentiment analysis. *arXiv preprint arXiv:2502.20682*.
- [65] Chi, D., Huang, T., Jia, Z., & Zhang, S. (2025). Research on sentiment analysis of hotel review text based on BERT-TCN-BiLSTM-attention model. *Array*, 25, 100378.
- [66] Ullah, W., Zhang, Z., & Stefanidis, K. (2023). Sentiment analysis of mobile apps using BERT. In *Proceedings of the International Conference on Industrial, Engineering and*

Other Applications of Applied Intelligent Systems (pp. 66–78). Cham, Switzerland: Springer Nature.

[67] Lee, M. A., Park, Y. J., Na, J. Y., & Sohn, C. B. (2023). Implementation of review sentiment analysis application using KoBERT, KoGPT-2, and KoBART optimized hyperparameters. *Journal of Digital Contents Society*, 24(11), 2831–2840.

[68] Nugroho, K. S., Sukmadewa, A. Y., Wuswilahaken, D. W. H., Bachtiar, F. A., & Yudistira, N. (2021). BERT fine-tuning for sentiment analysis on Indonesian mobile apps reviews. In *Proceedings of the 6th International Conference on Sustainable Information Engineering and Technology* (pp. 258–264).

[69] <https://www.kaggle.com/datasets/abiyyurasyiq/mobile-legends-google-play-reviews>
last accessed 15/12/2025