People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University of 8 May 1945 – Guelma
Faculty of Mathematics, Computer Science and Natural Sciences
Department of Computer Science

Master's Thesis

Field: Computer Science

Option: Computer Systems

Sentiment Analysis of Learners' Comments

Presented by: **Boulares Madjda**

Members of the Jury:

Mm. MADI Leyla	President
Dr. MEHENAOUI Zohra	Supervisor
Mr. HADJERIS Mourad	Examiner

June 2025

Abstract

The rapid expansion of online learning platforms has enabled students to express themselves freely through comments, offering valuable insights into their educational experiences.

The objective of this work is to propose an automatic sentiment analysis approach applied to learners' comments, in order to determine whether the expressed sentiments are positive, negative, or neutral.

To achieve this, a preprocessing phase was carried out, including text cleaning, tokenization, and automatic labeling of comments using the TextBlob library. The annotated data was then used to train an LSTM (Long Short-Term Memory) model, a recurrent neural network architecture particularly well suited for natural language processing.

The proposed approach was validated using a dataset collected from the Mark My Professor platform, which contains over 5,200 reviews written by learners about their courses and instructors. Mark My Professor is a platform dedicated to the evaluation of teachers and educational content by students.

The proposed model was evaluated using performance metrics such as accuracy, confusion matrix, recall, and F1 score. The results obtained show promising potential, but improvements are still needed, particularly due to the data imbalance that may have affected the model's learning process.

Keywords: Online learning, Sentiment analysis, TextBlob, LSTM, Artificial intelligence, Natural language processing, Classification, Student feedback.

Résumé

L'essor des plateformes d'apprentissage en ligne a permis aux étudiants de s'exprimer librement à travers des commentaires, offrant ainsi une source précieuse d'informations sur leur expérience pédagogique.

L'objectif de ce travail est de proposer une approche automatique d'analyse des sentiments appliquée aux commentaires des apprenants, afin de déterminer si les sentiments exprimés sont positifs, négatifs ou neutres.

Pour cela, une phase de prétraitement a été mise en œuvre, incluant le nettoyage des textes, la tokenisation, ainsi que l'étiquetage automatique des commentaires à l'aide de la bibliothèque TextBlob. Les données ainsi annotées ont ensuite servi à l'entraînement d'un modèle de type LSTM (Long Short-Term Memory), une architecture de réseau de neurones récurrent particulièrement adaptée au traitement du langage naturel.

L'approche proposée a été validée à l'aide d'une base de données recueillie sur la plateforme Mark My Professor, qui regroupe plus de 5 200 avis rédigés par des apprenants sur leurs cours et enseignants. Mark My Professor est une plateforme dédiée à l'évaluation des enseignants et des contenus pédagogiques par les étudiants.

L'évaluation du modèle proposé a été réalisée à l'aide d'indicateurs de performance tels que la précision, la matrice de confusion et le rappel et le F1 score. Les performances obtenues montrent un potentiel intéressant, mais des améliorations restent nécessaires, notamment en raison avec le déséquilibre des données qui a pu affecter l'apprentissage du modèle.

Mots clés : Apprentissage en ligne, Analyse des sentiments, TextBlob, LSTM, Intelligence artificielle, Traitement du langage naturel, Classification, Commentaires étudiants.

الملخص

ساهم ازدهار منصات التعليم الإلكتروني في تمكين الطلاب من التعبير بحرية من خلال التعليقات، مما وفر مصدرًا ثمينًا للمعلومات حول تجاربهم التعليمية.

يهدف هذا العمل إلى اقتراح نهج آلي لتحليل المشاعر يُطبّق على تعليقات المتعلمين، بهدف تحديد ما إذا كانت المشاعر المعبّر عنها إيجابية، سلبية أو محايدة.

ولتحقيق ذلك، تم تنفيذ مرحلة تمهيدية شملت تنظيف النصوص، والتجزئة ،(Tokenization) بالإضافة إلى التصنيف التلقائي للتعليقات باستخدام مكتبة .TextBlob وقد استُخدمت البيانات المصنفة بعد ذلك لتدريب نموذج من نوع LSTM (الذاكرة القصيرة والطويلة المدى)، وهي بنية من الشبكات العصبية المتكررة المصممة خصيصًا لمعالجة اللغة الطبيعية.

تمت عملية التحقق من فعالية النهج المقترح باستخدام قاعدة بيانات تم جمعها من منصة My Mark Professor، والتي تحتوي على أكثر من 5200 تقييم كتبها الطلاب حول أساتذتهم ودوراتهم. وتُعد هذه المنصة مخصصة لتقييم الأساتذة والمحتوى التربوي من طرف الطلبة.

تم تقييم النموذج المقترح باستخدام مؤشرات أداء مثل الدقة ،(Accuracy) ومصفوفة الالتباس، والاسترجاع ،(Recall) ودرجة .F1 وقد أظهرت النتائج المحققة أداءً واعدًا، إلا أن هناك مجالًا للتحسين، خصوصًا بسبب عدم توازن البيانات، الأمر الذي قد يكون قد أثر على جودة تعلم النموذج.

الكلمات المفتاحية: التعليم الإلكتروني، تحليل المشاعر، ،TextBlob ،LSTM الذكاء الاصطناعي، معالجة اللغة الطبيعية، التصنيف، تعليقات الطلاب.

Acknowledgments

I would like to open these acknowledgments by expressing my deepest gratitude to ***Dr. Zahra Mehenaoui****, who honored me with her guidance throughout this thesis. Her scientific rigor, insightful supervision, and constant encouragement were invaluable to me. She was always available to guide me with patience and generosity, and her trust in my abilities motivated me to surpass myself at each stage of this journey. This work would not have reached its current level of quality without her support and expertise.*

I extend my sincere thanks to the ***jury members*** *and the* ***teaching staff****, who devoted their time and energy to evaluating this work. Their constructive feedback and thoughtful observations helped me to refine my ideas and gave me a new perspective on the subject matter. I am equally thankful for the knowledge they shared throughout my academic path, which laid the foundation for this research.*

My warmest thoughts go to my ***family****, whose love, patience, and unwavering moral support have been a constant source of strength and comfort. During moments of uncertainty and exhaustion, their encouragement was my anchor. Their belief in me, even when I doubted myself, was what gave me the determination to complete this project. This accomplishment is as much theirs as it is mine.*

To my ***classmates and friends****, I will always remember the moments of cooperation, mutual help, and shared laughter that made this journey more enriching and memorable. Your companionship turned this academic experience into a human adventure filled with learning and kindness.*

Finally, I would like to thank all those who, whether through a word, a gesture, or a silent presence, contributed in some way to the success of this work. This thesis is also a tribute to those quiet yet meaningful human connections that give value and purpose to every step we take.

thank you.

Dedication

To you Dad

My beloved father ***MOUHAMED****, a symbol of wisdom and perseverance. Thank you for your strength, your wisdom, and your endless love. You've always been my guide, my protector, and my hero. This is for everything you are and everything you've given me.*

To you Mom

My dear mother ***SALIHA*** *an endless source of tenderness and strength.Thank you for your unconditional love, your patience, and the warmth you bring into my life. You've always been my safe place, my biggest supporter, and the heart of our home. This is for everything you've done and continue to do..*

To my brother BABAY , and sisters ZOUHRA, BOUCHRA, KHAOULA, and My little kids ELINE, DJANA thank you for your presence, encouragement, and those spontaneous moments of joy that eased the weight of fatigue. You were my silent strength in the background.
To my loyal friends without particularly my dearest ASSALA , for your laughter, your solid support, and your comforting words during difficult moments. Thank you for your humanity and sincerity.
And to all those, near or far, known or unknown, who sowed seeds of courage along my path: this humble accomplishment is dedicated to you with respect, affection, and heartfelt gratitude.

Boulares Madjda.

Contents

List of Figures

List of Tables

General Introduction

In recent years, the shift from traditional classroom instruction to digital formats has made online learning a central component of modern education systems. By offering flexibility, accessibility, and personalized learning experiences, e-learning environments allow learners to engage with educational content anytime and anywhere. This trend has been accelerated by technological advancements and global health challenges, further emphasizing the importance of integrating digital tools into pedagogical practices (Bond et al., 2020).

With in these digital environments, the analysis of learners' emotions and sentiments plays an increasingly critical role. Identifying how students feel about the content, the learning platform, or their instructors provides valuable insights into their overall learning experience. Research has demonstrated that accounting for emotional states significantly enhances emotional engagement, motivation, and even academic achievement (D'Mello and Graesser, 2012) and (Ocumpaugh et al., 2015). Emotionally aware systems can help tailor content, prevent disengagement, and create more supportive learning environments.

In this context, the present work proposes an automated sentiment analysis approach applied to student feedback collected from an online course evaluation platform. The goal is to develop a model capable of classifying learners' comments into three sentiment categories: positive, negative, or neutral. Our approach involves two main phases: first, automatic sentiment labeling using the TextBlob library; then, training a neural network model based on the Long Short-Term Memory (LSTM) architecture, which is well-suited for handling sequential text data. To evaluate our system, we used a dataset collected from the Mark My Professor website, which includes over 5,200 reviews written by students about their courses and instructors (Bouacida, 2018).

This thesis is organized into four chapters. The first chapter presents the theoretical foundations of sentiment analysis, including its application areas and common methodologies in natural language processing (NLP). The second chapter introduces neural networks, with a focus on the LSTM model and its effectiveness for text-based classification tasks. The third chapter details the design of the system, from data preprocessing to model architecture. Finally, the fourth chapter

presents the implementation and evaluation, including experimental results and performance analysis.

Chapter 1

Sentiment analysis

1.1 Introduction

The exponential growth of textual data produced through digital platforms—ranging from product reviews and social media to student evaluations—has made it increasingly important to extract and understand user sentiment. Sentiment Analysis (SA), a subfield of Natural Language Processing (NLP) and Artificial Intelligence (AI), is dedicated to identifying and categorizing opinions and emotions within textual content (Liu, 2020).

In the education sector, sentiment analysis has become a valuable tool for gaining insights into students' learning experiences. Student feedback provides a unique and often underutilized perspective on teaching quality and classroom engagement. By analyzing this feedback, educational institutions can make informed decisions to improve learning outcomes and adapt pedagogical strategies (Zhou and Ye, 2023),(Bhagat et al., 2024) Early approaches to sentiment analysis often relied on traditional machine learning techniques such as Naive Bayes, Support Vector Machines (SVM), and ensemble classifiers (Mabunda et al., 2021). While effective to a degree, these models faced challenges in handling complex language patterns and context. The advent of deep learning has brought a significant leap in sentiment analysis capabilities. Models such as Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks have demonstrated superior performance in capturing syntactic and semantic relationships in text (Nguyen et al., 2018)(Sangeetha and Prabha, 2021) .

More recently, attention mechanisms have emerged as a powerful enhancement to deep learning architectures. Multi-head attention, in particular, allows the model to focus on different parts of a sentence simultaneously, leading to more accurate and interpretable predictions. Studies have shown that LSTM models integrated with multi-head attention outperform other configurations in educational

sentiment classification tasks (Bhagat et al., 2024).

This chapter presents an overview of sentiment analysis and its applications, particularly in the context of education.

1.2 Definition and Scope of Sentiment Analysis

1.2.1 What is Sentiment Analysis

Sentiment Analysis, often referred to as opinion mining, is a branch of Natural Language Processing (NLP) that focuses on identifying and interpreting the emotional tone behind written content. Its primary objective is to classify text into sentiment categories such as positive, negative, or neutral(Liu, 2020). In more advanced use cases, sentiment analysis can also detect subtler emotional nuances or focus on specific aspects or features mentioned in the text.

As the volume of digital content continues to grow, sentiment analysis has become increasingly important across domains including marketing, politics, healthcare, and education. Within academic environments, it is particularly useful for analyzing students' written feedback to evaluate the quality of instruction, identify areas for improvement, and support data-driven decision-making (Zhou and Ye, 2023),(Bhagat et al., 2024).

The process of sentiment analysis generally involves several key stages: preprocessing the text to remove noise, transforming the textual data into numerical features, and then applying machine learning or deep learning algorithms to perform classification. Traditional lexicon-based approaches rely on sentiment dictionaries, while modern techniques employ advanced models such as LSTM networks and transformer-based architectures to better understand context and semantic relationships (Nguyen et al., 2018).

Sentiment analysis can be categorized into different **levels**:

Document-Level Sentiment Analysis :

Document-level sentiment analysis focuses on determining the overall emotional tone or opinion expressed throughout an entire document. It assumes the text conveys a single dominant sentiment, typically applicable to user reviews, articles, or feedback where only one opinion is being communicated. This level is useful when the goal is to capture the general attitude of the author without diving into finer details(Balaji et al., 2017).

Sentence-Level Sentiment Analysis :

This approach examines individual sentences to identify their specific sentiment, categorizing each as positive, negative, or neutral. It is beneficial in cases where different parts of a text may express varying sentiments. Sentence-level analysis provides more precision than document-level, allowing for a better understanding of mixed opinions within the same content(Balaji et al., 2017).

Aspect-Level Sentiment Analysis :

Aspect-level sentiment analysis delves deeper by identifying sentiments related to particular attributes or features mentioned in a text. Instead of analyzing the entire sentence or document, it focuses on evaluating the sentiment towards specific topics (e.g., “interface”, “content quality”). This level is especially valuable in feedback where users may like some aspects but dislike others(Balaji et al., 2017).

Word-Level Sentiment Analysis :

At the word level, the analysis aims to detect the emotional value carried by individual words or terms. This granularity helps build sentiment lexicons and enhances machine learning models by recognizing context-sensitive sentiment expressions. It also serves as a foundation for more complex levels of sentiment analysis(Balaji et al., 2017).

1.2.2 Subjectivity vs. Objectivity in Text

A key concept in sentiment analysis is the ability to distinguish between subjective and objective language. This distinction is vital because sentiment classification mainly targets subjective content, which reflects personal opinions, emotions, or attitudes. On the other hand, objective language presents factual information without expressing any personal feelings(Liu, 2020).

Subjective statements typically contain evaluative or emotional expressions, such as ”The professor was inspiring,” which shows the speaker’s personal viewpoint. In contrast, an objective sentence like ”The course had ten sessions” simply conveys information without emotional context.

In the context of education, particularly when analyzing student feedback, this differentiation becomes more complex. Student responses often contain both factual details and emotional impressions. According to Zhou and Ye (Zhou and Ye, 2023), student feedback tends to be rich in subjective expressions that highlight satisfaction, frustration, or suggestions for improvement, making it an ideal source for sentiment analysis.

Earlier sentiment analysis systems faced challenges in filtering out objective content, which often led to reduced accuracy. However, the development of more sophisticated models such as LSTM and attention-based neural networks has enhanced the ability to capture contextual meaning and better separate subjective elements from factual ones(Bhagat et al., 2024),(Sangeetha and Prabha, 2021).

Nguyen et al.(Nguyen et al., 2018) note that preprocessing techniques that identify and minimize objective content can lead to improved classification performance. In doing so, models focus more on the emotionally charged parts of the text, leading to more accurate sentiment predictions.

Many recent frameworks include subjectivity detection as a preliminary step in the sentiment analysis pipeline. This approach helps filter out neutral content, allowing the classifier to better focus on the emotional tone of the text. In educational datasets, where comments may range from neutral observations to highly opinionated feedback, this step becomes particularly useful.

In conclusion, the ability to effectively distinguish between subjective and objective text is a crucial step in building reliable sentiment analysis systems, especially in domains such as education, where user-generated feedback may include a wide variety of expression styles and tones.

1.2.3 Types of Sentiment Analysis

Sentiment analysis can be implemented at various levels of detail, depending on the specific goals of the analysis and the structure of the data. The most commonly used types include binary classification, multiclass classification, and aspect-based sentiment analysis (ABSA). Each type offers different levels of insight into user opinions and emotional expressions.

Binary Sentiment Classification

The binary approach to sentiment analysis is one of the simplest forms, aiming to label text as either positive or negative. This method has been widely applied in areas such as customer reviews or feedback analysis, where the objective is to understand whether overall sentiment is favorable or not(Liu, 2020). While this method is computationally efficient and straightforward to implement, it often fails to capture neutral expressions or more complex emotional tones.

Multi-Class Sentiment Classification

To address the limitations of binary models, multiclass sentiment classification includes additional sentiment categories, often incorporating a neutral class. Some models even extend this further to cover emotional states such as joy, anger, or sadness. This form of classification provides a more comprehensive view of the emotional landscape in textual data. For example, Bhagat et al.(Bhagat et al.,

2024) used a three-way sentiment classifier, positive, negative, and neutral, to analyze student feedback and demonstrated that multiclass models offer better granularity when interpreting educational data.

Aspect-Based Sentiment Analysis (ABSA)
Aspect-Based Sentiment Analysis goes beyond simply identifying the overall sentiment by focusing on specific targets or topics within a text. It enables the detection of sentiments related to individual components of a service or experience. For instance, in a student comment such as "The course was well-structured, but the assignments were overwhelming," ABSA would differentiate between the sentiment toward the course structure (positive) and the assignments (negative). As Zhou and Ye(Zhou and Ye, 2023) point out, this approach is particularly useful in analyzing complex feedback where multiple aspects are evaluated simultaneously.

The increasing complexity of natural language has prompted the adoption of advanced models such as LSTM with attention mechanisms and Transformer-based architectures. These techniques allow models to understand the context of each word and better associate sentiment with the appropriate aspect or emotion (Nguyen et al., 2018),(Sangeetha and Prabha, 2021).Consequently, both multi-class and aspect-based approaches benefit significantly from such modern deep learning enhancements.
In summary, the choice of sentiment analysis type depends on the desired depth of insight. While binary models are useful for quick evaluations, multi-class and aspect-based methods provide a more detailed understanding, especially in domains such as education where opinions are diverse and multifaceted.

1.3 Applications of Sentiment Analysis

1.3.1 Applications of Sentiment Analysis in Business and Marketing

In the commercial landscape, sentiment analysis plays a central role in helping businesses interpret customer opinions and adapt their strategies accordingly. With the growth of social media, e-commerce platforms, and review sites, companies have access to an abundance of user-generated content. Analyzing this data allows them to track brand reputation, evaluate customer satisfaction, and guide product development with greater precision(Liu, 2020).

One of the key business applications is brand perception monitoring. By analyzing sentiments from online platforms like Twitter, Amazon, or TripAdvisor,

organizations can gauge how their products or services are being received in real-time. Positive or negative trends in sentiment can influence branding strategies, marketing campaigns, and public relations efforts. As noted by Zhou and Ye.(Zhou and Ye, 2023), the ability to quickly respond to emerging sentiment patterns gives companies a competitive advantage in maintaining their public image.

Another crucial area is product and service improvement. Sentiment analysis allows businesses to pinpoint which features customers praise and which ones draw criticism. This is where aspect-based sentiment analysis (ABSA) becomes particularly effective—it helps link emotions and evaluations to specific product components, such as usability, pricing, or customer support(Sangeetha and Prabha, 2021). These insights enable product teams to prioritize enhancements that align with customer expectations.

In addition, competitive analysis can be enriched through sentiment mining. By examining customer reviews and social media posts about rival brands, companies can better understand market gaps and consumer pain points. Nguyen et al.(Nguyen et al., 2018) emphasized how deep learning models can help extract subtle insights from comparative statements, allowing firms to tailor their offerings and position themselves strategically.

Sentiment analysis also enhances customer relationship management (CRM). For instance, integrating sentiment-aware tools into customer support systems allows agents to prioritize and handle emotionally sensitive interactions more effectively. Machine learning-based classification systems can flag negative or urgent queries, enabling faster and more empathetic responses(Bhagat et al., 2024).

Finally, marketing analytics increasingly rely on sentiment insights to assess the impact of campaigns. Real-time tracking of audience reactions enables dynamic adjustments to advertising content and communication strategies. This responsiveness leads to better alignment between brand messaging and public sentiment, improving overall campaign effectiveness(Mabunda et al., 2021).

1.3.2 In Education (Feedback Analysis)

In the education sector, sentiment analysis has become a powerful tool to better understand students' learning experiences and perceptions of teaching quality. Academic institutions increasingly collect large volumes of student feedback through course evaluations, online learning platforms, and institutional surveys. Manually analyzing this data is not only time-consuming but also lacks objectivity and scalability. Sentiment analysis provides a scalable, automated way to extract insights from this feedback and guide data-driven decision-making (Zhou and Ye, 2023),(Bhagat et al., 2024).

Student comments often contain nuanced expressions of satisfaction, frustration, or constructive criticism. By applying sentiment classification techniques, educational administrators can detect positive and negative trends related to teaching methods, course materials, classroom engagement, or learning environments. For instance, Bhagat et al.(Bhagat et al., 2024)demonstrated the effectiveness of a multi-head attention-based LSTM model in accurately classifying student sentiments into positive, neutral, and negative categories across thousands of feedback samples.

Moreover, aspect-based sentiment analysis (ABSA) is especially valuable in education, as it allows systems to identify sentiments related to specific aspects of a course—such as the instructor's clarity, course content relevance, assessment methods, or workload. This finer level of analysis enables educators to focus on what matters most to students and make targeted improvements (Sangeetha and Prabha, 2021),(Nguyen et al., 2018).

Studies such as those by Mabunda et al(Mabunda et al., 2021) and Sindhu et al.(Sindhu et al., 2019) have shown that sentiment analysis can reveal hidden patterns in feedback that may not be obvious through simple rating scales. These insights support continuous teaching improvement, personalized learning, and enhanced student satisfaction. Furthermore, integrating real-time sentiment analysis into e-learning platforms allows for adaptive responses, such as recommending resources or alerting instructors when negative trends emerge(Capuano et al., 2021).

1.3.3 In Social Media and Politics

Social media platforms like Twitter, Facebook, and Reddit have become rich sources of public opinion, where users express their views on a wide range of topics, including political events, public policies, and social issues. Sentiment analysis in social media enables researchers, marketers, and policymakers to monitor public sentiment at scale and in real-time (Liu, 2020),(Zhou and Ye, 2023).

In politics, sentiment analysis is widely used to track public reactions to speeches, debates, campaigns, or legislation. It helps political analysts gauge the popularity of candidates, measure approval or disapproval of policies, and identify influential issues during elections. For example, Altrabsheh et al.(Altrabsheh et al., 2014) highlighted the value of real-time sentiment tracking in political decision-making and public communication.

Social media sentiment analysis has also been used to detect public mood during crises, such as pandemics or social movements. By analyzing emotional tone, governments and organizations can respond more effectively to public concerns, reduce misinformation, and tailor communication strategies(Singh et al., 2020),(Onan, 2021) .

One of the major challenges in this context is dealing with informal language, sarcasm, slang, and rapidly changing trends. However, advanced deep learning models such as CNNs, LSTMs, and transformers with attention mechanisms have significantly improved accuracy in understanding sentiment in short, noisy texts(Nguyen et al., 2018) and (Sangeetha and Prabha, 2021).

Overall, sentiment analysis in social media and politics allows for deeper understanding of collective emotions, supports public engagement strategies, and facilitates better governance through timely feedback interpretation.

1.4 Sentiment Analysis Approaches

1.4.1 Lexicon-Based Approaches

Lexicon-based sentiment analysis is one of the earliest and most interpretable techniques used to determine the sentiment polarity of a text. This approach relies on a predefined set of words—referred to as a sentiment lexicon—where each word is associated with a sentiment value, typically positive, negative, or neutral(Liu, 2020). The core idea is to identify sentiment-bearing words in a given text and use their assigned polarity to determine the overall sentiment.

There are two main strategies in lexicon-based analysis:

Dictionary-based methods, which use manually or semi-automatically curated lists of sentiment words (e.g., SentiWordNet, AFINN, or the NRC lexicon).

Corpus-based methods, which expand sentiment lexicons based on their contextual use in a large dataset, often by analyzing word co-occurrence patterns.

Lexicon-based methods are widely used due to their simplicity, transparency, and domain-independence. They do not require labeled training data and are particularly useful in scenarios where data annotation is scarce or expensive. However, these methods often struggle with contextual ambiguity, sarcasm, negation handling, and domain-specific language, which can significantly impact performance(Zhou and Ye, 2023).

In educational sentiment analysis, some early works applied lexicon-based approaches to student comments but faced limitations in capturing subtle emotions and aspect-specific sentiments. Aung and Myo(Aung and Myo, 2017), for instance, employed a lexicon-based method to classify student sentiments but did not evaluate the model using standard metrics, highlighting a common challenge with this approach. Similarly, Rani and Kumar(Rani and Kumar, 2017) proposed a rule-based sentiment system for teaching and learning improvement, but their model lacked adaptability to dynamic language use.

While lexicon-based approaches are less effective than machine learning or deep learning techniques in complex and context-rich datasets, they still provide valuable baselines and are often used in hybrid models to improve overall system performance(Sangeetha and Prabha, 2021).

1.4.2 Machine Learning-Based Approaches

Machine learning (ML) techniques have been widely used in sentiment analysis to classify text based on labeled training data. These approaches rely on extracting numerical features from text and feeding them into supervised algorithms that learn to distinguish between different sentiment classes, such as positive, negative, or neutral(Liu, 2020).

Common classifiers in this category include Support Vector Machines (SVM), Naive Bayes (NB), Decision Trees (DT), Random Forests (RF), and Logistic Regression (LR). These models typically use feature representation techniques like Bag of Words (BoW) or TF-IDF (Term Frequency–Inverse Document Frequency) to convert textual data into vectors that capture word frequencies or importance(Pacol and Palaoag, 2021).

ML-based approaches offer advantages such as fast training times, interpretability, and modularity, making them suitable for real-time or large-scale applications. However, they often struggle with complex language structures, such as sarcasm or idioms, and perform poorly when the dataset is small or imbalanced(Zhou and Ye, 2023).

In the context of education, several studies have utilized ML models to classify sentiments in student feedback. For example, Mabunda et al.(Mabunda et al., 2021) employed random forest and SVM classifiers to evaluate teaching quality from student comments. Similarly, Lalata et al. (Lalata et al., 2019) applied ensemble learning to enhance performance, showing that combining multiple learners improves robustness over single models.

Despite their popularity, ML models rely heavily on manual feature engineering and require extensive preprocessing. Their performance often serves as a baseline when compared to more advanced deep learning approaches.

1.4.3 Deep Learning-Based Approaches

Deep learning (DL) has revolutionized sentiment analysis by enabling models to automatically learn hierarchical representations of text from raw data. Unlike traditional ML models, deep learning architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs)—including Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU)—can capture both

syntactic and semantic patterns in sequences (Nguyen et al., 2018) ,(Sangeetha and Prabha, 2021).

LSTM, in particular, has proven effective in handling long-range dependencies, making it suitable for analyzing long comments or reviews. CNNs are often used to extract local features in text and have been applied successfully in hybrid DL models. Recent innovations include the use of attention mechanisms, which allow models to focus on the most relevant parts of the input sequence. Bhagat et al.(Bhagat et al., 2024) showed that combining LSTM with multi-head attention significantly improves classification accuracy in student feedback analysis, achieving over 95% accuracy on their dataset.

Capuano et al.(Capuano et al., 2021) also demonstrated the effectiveness of attention-based hierarchical RNNs for analyzing MOOC forum posts. Their approach enabled more accurate detection of emotions and opinions in educational discussions, providing actionable insights for instructors and administrators.

DL models require large amounts of labeled data and computational resources but often outperform traditional ML approaches in terms of accuracy, generalization, and context-awareness.

1.4.4 Hybrid Methods

Hybrid approaches combine the strengths of machine learning, deep learning, and lexicon-based techniques to enhance sentiment analysis performance. These systems often aim to balance efficiency, accuracy, and interpretability, especially in domains like education where data characteristics can vary widely.

One common hybrid strategy is to combine lexicon-based scoring with ML classifiers. For example, sentiment scores from a predefined lexicon can be used as features in an ML model to improve prediction quality. Another strategy integrates BoW or TF-IDF features with deep learning outputs, allowing the model to capture both shallow and deep patterns.

Studies by Sindhu et al.(Sindhu et al., 2019) and Sangeetha et Prabha (Sangeetha and Prabha, 2021) explored fusion models that combine attention mechanisms, word embeddings, and traditional classifiers, showing enhanced results for teaching evaluation and student feedback classification.

Bhagat et al.(Bhagat et al., 2024)implemented a hybrid framework combining ensemble classifiers, CNN-LSTM, single-head attention, and multi-head attention, comparing their performance across multiple datasets. Their findings confirmed that hybrid models outperform standalone ones, particularly when dealing with real-world feedback that contains both structured and unstructured content.

These hybrid methods are increasingly being adopted in educational analytics, marketing, and healthcare domains, as they provide flexibility and adaptability across languages, platforms, and contexts.

1.5 Text Preprocessing Techniques

Text preprocessing is a crucial step in sentiment analysis pipelines, especially when working with raw, unstructured data such as user reviews, social media posts, or student feedback. The quality of preprocessing directly influences the accuracy and performance of machine learning and deep learning models. This section outlines and discusses essential preprocessing techniques used in sentiment analysis, supported by recent research.

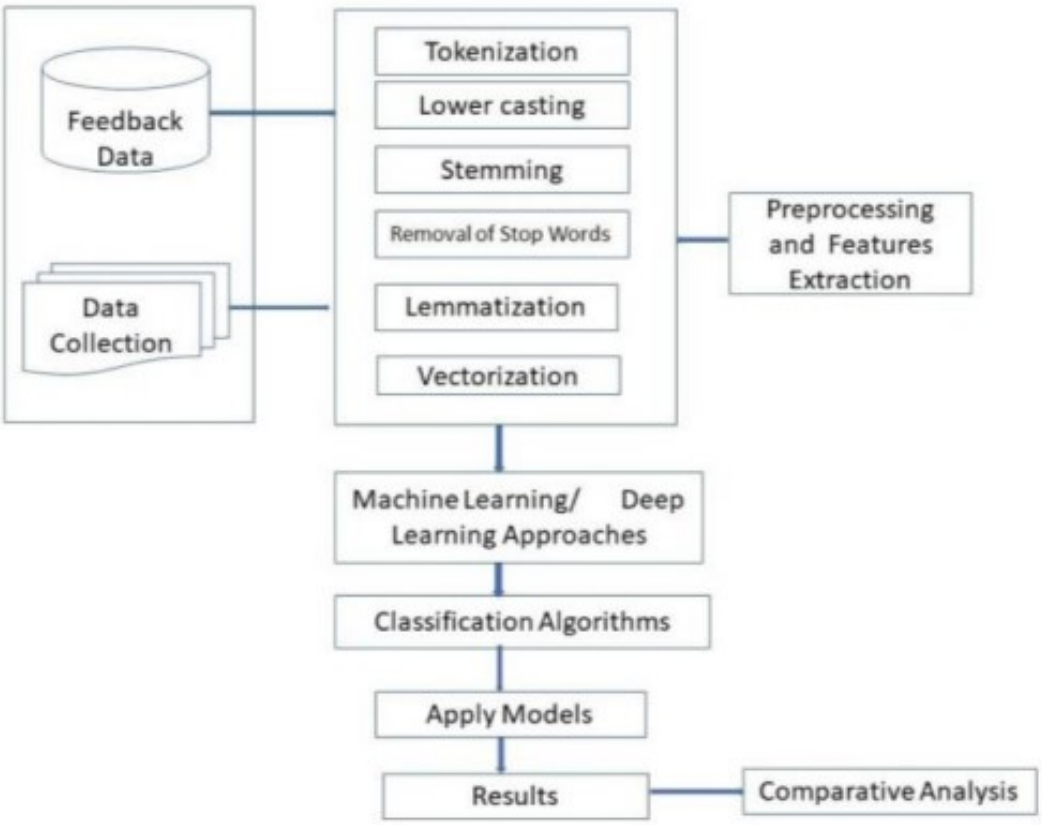

Figure 1.1: The methodology used for proposed models (Bhagat et al., 2024)

1.5.1 Tokenization

Tokenization is the process of dividing a text into individual units called tokens, which may represent words, characters, or subwords. This is often the first step in text preprocessing. Tokenization helps models interpret sentences as structured sequences, making them suitable for vectorization and further analysis(Liu, 2020).

For example, the sentence "The course was excellent." would be tokenized as ["The", "course", "was", "excellent", "."]. Tokenization can be rule-based, language-specific, or subword-based (as used in models like BERT). According to Nguyen et al.(Nguyen et al., 2018), accurate tokenization is especially important in student feedback, where informal language and punctuation are common.

Advanced tokenization methods such as Byte-Pair Encoding (BPE) or WordPiece are now commonly used in transformer models, offering robustness for unknown or rare words(Onan, 2021).

1.5.2 Stopword Removal

Stopwords are commonly used words in a language (e.g., the, is, and, was) that do not contribute significant semantic value to the text. Removing them reduces noise and dimensionality in the data(Pacol and Palaoag, 2021).

In sentiment analysis, however, care must be taken, as some stopwords may carry subtle sentiment depending on context. For example, words like not or never significantly impact sentiment polarity. Zhou and Ye (Zhou and Ye, 2023) note that improper stopword removal can lead to loss of sentiment-bearing information, particularly in educational datasets where negations are frequent.

Therefore, some models apply custom stopword lists, or avoid stopword removal altogether, especially in deep learning models where context is learned automatically.

1.5.3 Lemmatization and Stemming

Both lemmatization and stemming aim to reduce words to their base or root form:

Stemming uses crude heuristics to remove word suffixes (e.g., learning → learn), sometimes leading to non-dictionary forms.

Lemmatization, by contrast, uses vocabulary and morphological analysis to return dictionary root forms (e.g., better → good).

Stemming is faster but less accurate; lemmatization is more linguistically correct but computationally intensive. According to Mabunda et al.(Mabunda et al., 2021), lemmatization improves classification performance in feedback analysis tasks by reducing vocabulary size without sacrificing semantic precision.

Bhagat et al.(Bhagat et al., 2024)highlight that lemmatization is particularly effective in pretraining deep models like LSTM or attention networks, where semantically correct roots enhance contextual embeddings.

1.5.4 Text Normalization

Text normalization involves converting text into a consistent format to reduce variations caused by capitalization, punctuation, or informal writing. It typically includes:

- Lowercasing all text.
- Removing special characters, emojis, or numbers.
- Expanding contractions (e.g., can't → cannot).
- Handling spelling variations or slang (e.g., u → you).

This step is especially critical in social media or student-generated content, where language is often informal and inconsistent. Capuano et al.(Capuano et al., 2021) emphasized that normalization improves sentiment detection in MOOCs where students use diverse writing styles.

Advanced normalization may also include spell correction or phonetic matching, as explored by Sindhu et al.(Sindhu et al., 2019), who worked with noisy student feedback data.

1.5.5 Dealing with Imbalanced Data

n real-world sentiment datasets, especially in education, class imbalance is a common issue—for instance, there may be more positive comments than negative or neutral ones. This imbalance can lead to biased models that perform poorly on minority classes (e.g., negative sentiment).

To address this, several techniques are used:

- Oversampling (e.g., SMOTE) and undersampling to balance class distribution(Sangeetha and Prabha, 2021).
- Class weighting in loss functions during training (especially in deep learning).
- Data augmentation strategies such as synonym replacement, back-translation, or paraphrasing(Nguyen et al., 2018).

Bhagat et al.(Bhagat et al., 2024)adopted balanced datasets by selectively sampling equal proportions of sentiments, ensuring robust model performance across all classes. Similarly, Onan (Onan, 2021)demonstrated that class-balancing significantly improved F1-scores in sentiment classification for MOOC evaluations.

Handling imbalance is essential for building fair and generalizable models, particularly when sentiment analysis is used in decision-making contexts like education or public policy.

1.6 Feature Extraction and Representation

The transformation of raw textual input into structured numerical features is a fundamental step in sentiment analysis. Since most machine learning and deep learning models cannot directly process text, this step ensures that language is converted into a format that models can learn from. Over time, feature extraction techniques have progressed from basic count-based models to advanced embeddings that capture semantic relationships and contextual meaning.

1.6.1 Bag of Words (BoW)

The Bag of Words model is a traditional and straightforward method that represents text by converting it into a vector of word frequencies. It counts how often each word appears in a document while disregarding grammar, context, and word order.

Although simplistic, BoW has shown usefulness in foundational sentiment analysis models. In educational research, Pacol and Palaoag (Pacol and Palaoag, 2021) applied BoW alongside machine learning algorithms and observed decent performance in analyzing student evaluations. However, this approach can lead to very large and sparse feature matrices and lacks the ability to understand meaning or similarity between words(Zhou and Ye, 2023).

1.6.2 Term Frequency–Inverse Document Frequency (TF-IDF)

TF-IDF is an improvement over BoW that assigns higher importance to words that are frequent in a single document but rare across the entire dataset. This helps reduce the impact of commonly occurring but less informative terms.

Nguyen et al.(Nguyen et al., 2018) applied TF-IDF in sentiment classification of Vietnamese student feedback and reported notable gains in accuracy over BoW. Similarly, Mabunda et al.(Mabunda et al., 2021)combined TF-IDF with traditional classifiers and showed that it improved their system's ability to identify both positive and negative sentiments in student-generated text.

1.6.3 Word Embeddings (Word2Vec, GloVe, FastText)

Word embeddings represent words as dense, continuous vectors in a multi-dimensional space. Unlike BoW or TF-IDF, which rely on frequency, embeddings capture meaning based on how words appear in context. Words with similar usage patterns are mapped to nearby points in the vector space.

Among the most widely used embeddings:

• Word2Vec captures word meaning using skip-gram and CBOW architectures.
• GloVe builds vectors from global co-occurrence statistics.
• FastText enhances Word2Vec by including subword information, making it effective for noisy or morphologically complex texts.

Sangeetha and Prabha(Sangeetha and Prabha, 2021) reported that FastText embeddings significantly improved sentiment classification of student feedback by capturing informal language patterns. Bhagat et al. (Bhagat et al., 2024) also integrated Word2Vec with LSTM networks and attention mechanisms, achieving high accuracy on educational datasets.

1.6.4 Contextual Embeddings (e.g., BERT)

While traditional embeddings generate a single vector per word, contextual embeddings such as those produced by BERT adapt the vector representation based on the word's surrounding context. This allows the model to differentiate between multiple meanings of the same word depending on usage.

For instance, the word "grade" in "I got a good grade" versus "The slope's grade is steep" would have different embeddings in BERT. Onan (Onan, 2021) leveraged BERT to analyze sentiment in MOOC evaluations and found it especially useful in capturing subtle variations in tone. Capuano et al.(Capuano et al., 2021) used attention-based architectures with contextual embeddings to examine student forum discussions, leading to enhanced interpretability and precision.

Contextual embeddings have quickly become the preferred approach in sentiment analysis tasks involving complex or informal text, despite their higher computational requirements.

1.7 Classification Models for Sentiment Analysis

1.7.1 Traditional Models (SVM, Naive Bayes, Random Forest)

Traditional machine learning algorithms have played a foundational role in the development of sentiment analysis systems. These models rely on feature-based representations—such as Bag of Words or TF-IDF—and use statistical learning to assign sentiment labels (e.g., positive, negative, or neutral) based on patterns found in labeled training data. While they have limitations in handling complex linguistic nuances, traditional models remain widely used due to their simplicity, speed, and interpretability, especially in low-resource settings.

- **Support Vector Machines (SVM)**

Support Vector Machines (SVM) are among the most popular classifiers in sentiment analysis due to their high accuracy and ability to generalize well in high-dimensional spaces. SVM works by identifying the optimal hyperplane that separates different sentiment classes with the maximum margin.

In the work by Nguyen et al.(Nguyen et al., 2018) , SVM was used as a baseline for classifying Vietnamese student comments. When combined with TF-IDF features, the model achieved satisfactory results, although deep learning models eventually outperformed it. Similarly, Pacol and Palaoag(Pacol and Palaoag, 2021) used SVM to classify student feedback and observed strong performance, especially when data was clean and well-preprocessed.

SVM is particularly effective when the dataset is balanced and when feature selection is carefully performed. However, its performance may degrade with large, noisy, or highly imbalanced datasets unless proper tuning or kernel optimization is applied (Zhou and Ye, 2023).

• **Naive Bayes (NB)**

Naive Bayes is a probabilistic classifier based on Bayes' theorem. It assumes feature independence, which rarely holds in natural language but still performs surprisingly well in many text classification tasks, including sentiment analysis.

In the study by Mabunda et al.(Mabunda et al., 2021), Naive Bayes was evaluated alongside SVM and Random Forest for classifying textual feedback from students. While it had faster training times and performed well on smaller datasets, its accuracy was lower compared to SVM and ensemble methods.

Rani and Kumar(Rani and Kumar, 2017)also applied a Naive Bayes-based approach in developing a sentiment analysis system for evaluating teaching quality. They noted that the model's simplicity and speed made it a good choice for real-time applications, but it often failed to capture complex patterns due to its strong independence assumption.

Despite its limitations, Naive Bayes is still favored in scenarios where computational efficiency is critical and labeled data is limited.

• **Random Forest (RF)**

Random Forest is an ensemble learning technique that combines multiple decision trees to produce more accurate and stable predictions. It is particularly effective in handling high-dimensional data and noisy features.

Mabunda et al.(Mabunda et al., 2021) found that Random Forest outperformed both SVM and Naive Bayes in classifying student feedback data. The model showed robustness to outliers and required less feature scaling. Similarly, Lalata et al.(Lalata et al., 2019) +reported improved sentiment classification when using ensemble methods like Random Forest over single classifiers.

One major advantage of Random Forest is its ability to handle imbalanced datasets better than most standalone classifiers, especially when paired with sampling strategies or feature weighting.

1.7.2 Deep Learning Models (CNN, LSTM, Bi-LSTM, GRU)

Deep learning models offer enhanced textual representation by automatically learning relevant features from sequences of words. Unlike traditional approaches that depend on handcrafted feature engineering, deep neural networks are capable of capturing semantic, syntactic, and temporal relationships in text data.

- **Convolutional Neural Networks (CNN)**

Although originally developed for image processing, CNNs have also proven effective in capturing local patterns in text, such as in short phrases or expressions. In a study by Nguyen et al.(Nguyen et al., 2018), a CNN-LSTM hybrid model was applied to student comment datasets and delivered competitive results. However, it was outperformed by attention-enhanced LSTM models.

- **Long Short-Term Memory (LSTM)**

LSTM networks are designed to process sequential data while retaining long-term dependencies. Their memory cell architecture allows them to model context effectively within text. According to Bhagat et al.(Bhagat et al., 2024), an LSTM model combined with a multi-head attention layer achieved an impressive 95.56% accuracy in classifying student comments. This hybrid architecture outperformed other models, particularly when dealing with lengthy and complex sentences.

- **Bidirectional LSTM (Bi-LSTM)**

Bi-LSTM processes sequences in both forward and backward directions, enhancing contextual understanding. Sangeetha and Prabha (Sangeetha and Prabha, 2021) integrated Bi-LSTM with attention mechanisms to analyze student feedback, which resulted in improved detection of aspect-specific sentiments.

- **Gated Recurrent Units (GRU)**

GRUs are a streamlined variant of LSTM that maintain comparable performance with reduced computational complexity. Though less frequently used in the reviewed literature, Capuano et al.(Capuano et al., 2021)demonstrated the GRU's effectiveness in educational settings, such as MOOC forums, particularly when combined with hierarchical attention mechanisms.

1.7.3 Attention Mechanisms

Attention mechanisms have revolutionized the field of Natural Language Processing by allowing models to focus on the most informative parts of a text rather than treating all words equally. Various types of attention have emerged, ranging from simple attention layers to advanced architectures like Transformers.

• **Basic Attention**

Simple attention assigns varying weights to different parts of the input, helping the model concentrate on more sentiment-rich words. Sangeetha and Prabha (Sangeetha and Prabha, 2021) showed that integrating this mechanism into a Bi-LSTM improved the model's ability to recognize subtle emotional cues in educational feedback.

• **Self-Attention**

Self-attention enables each word to weigh its relationship with all other words in a sequence, making it particularly powerful for capturing global dependencies. Capuano et al.(Capuano et al., 2021)used hierarchical self-attention to process online forum discussions, enhancing the model's understanding of discourse-level sentiment.

• **Multi-Head Attention**

This advanced mechanism, central to Transformer architectures, processes multiple attention distributions in parallel. Bhagat et al. (Bhagat et al., 2024) demonstrated that incorporating multi-head attention with an LSTM significantly improved performance, especially in handling lengthy and ambiguous student responses.

1.8 Evaluation Metrics

Evaluating sentiment analysis models requires appropriate metrics that reflect both the model's predictive performance and its ability to handle imbalanced data or subtle sentiment nuances. Several standard metrics are used, each offering different perspectives on the model's performance.

• **Accuracy**

Accuracy is one of the most frequently used evaluation metrics, especially in balanced datasets. It measures the ratio of correctly predicted instances to the total number of predictions made. While simple and intuitive, accuracy can be misleading in imbalanced datasets where one class dominates(Wang et al., 2020). Accuracy is most informative when classes are equally represented, but may overestimate performance in skewed sentiment distributions(Zhang et al., 2018).

• **Precision, Recall, and F1-Score**

- To overcome the limitations of accuracy in imbalanced datasets, precision, recall, and F1-score provide a more nuanced view:
- Precision measures how many of the instances predicted as positive are actually correct.
- Recall (or Sensitivity) evaluates how many actual positive cases were correctly identified.
- F1-score is the harmonic mean of precision and recall, balancing both concerns.

These metrics are especially useful in multi-class or fine-grained sentiment classification tasks, where false positives and false negatives carry different weights(Liu, 2012), (Medhat et al., 2014).

For example, in student feedback analysis, high recall ensures that all negative comments are identified, while precision ensures those flagged are truly problematic(Sindhu et al., 2019).

• Confusion Matrix

The confusion matrix is a tabular representation of predicted vs. actual classes. It helps to identify the types of errors made by the classifier—false positives, false negatives, true positives, and true negatives—providing deeper insight into model performance(Rana and Cheah, 2016).

It is especially helpful when evaluating multi-class sentiment models, such as positive, neutral, and negative sentiment classes (Behera et al., 2021).

• ROC Curve and AUC Score

The Receiver Operating Characteristic (ROC) curve plots the true positive rate against the false positive rate at various thresholds. The Area Under the Curve (AUC) summarizes this curve into a single value, representing the model's ability to distinguish between classes.

AUC is often favored in binary sentiment classification tasks where model discrimination is crucial, especially in medical or finance-related text analytics(Liu, 2012).

1.9 Challenges in Sentiment Analysis

Although sentiment analysis has advanced significantly with the integration of deep learning and linguistic techniques, various challenges continue to impact the reliability and generalizability of models. These challenges are especially prominent in real-world applications like educational feedback, social media interactions,

and multilingual communication. This section outlines four major obstacles frequently discussed in the literature.

1.9.1 Detecting Sarcasm and Irony

Sarcasm and irony are particularly difficult to detect in sentiment classification because the literal meaning of a sentence often contradicts its intended sentiment. For instance, the expression "Oh, wonderful, another pop quiz!" could appear positive to a model relying on surface-level sentiment words, even though the actual meaning is negative.

Standard machine learning models, including many neural networks, often misinterpret such cases due to the lack of contextual or pragmatic understanding. Zhang et al.(Zhang et al., 2018) and Cambria et al.(Cambria et al., 2017) emphasize that sarcasm often requires not only linguistic context but also external knowledge, speaker intent, or social cues. Recent work incorporating context-aware models and external knowledge bases shows promise, but this remains a major area of ongoing research.

1.9.2 Multilingual and Code-Switching Complexity

Most sentiment analysis systems are developed in English, yet multilingual content and code-switching—the practice of mixing two or more languages within a sentence—are common in global user data. For example, phrases like "C'est trop cool ce movie" pose difficulties for language-specific models.

According to Wang et al.(Wang et al., 2020) and Onan (Onan, 2021), multilingual and mixed-language data disrupt common preprocessing tasks like tokenization and word vectorization. Traditional models and monolingual embeddings often fail to manage such variation effectively. Although multilingual transformer models (e.g., mBERT) and translation-based approaches are increasingly adopted, they still struggle with preserving language-specific nuances and cultural sentiment expressions.

1.9.3 Domain-Specific Dependency

A frequent issue in sentiment analysis is domain dependency, where a model trained on one type of text (e.g., product reviews) performs poorly on another domain (e.g., student feedback or political commentary). Words can carry different sentiments depending on context; for example, "cold" could be negative in a food review but neutral in a weather-related sentence.

As reported by Sindhu et al.(Sindhu et al., 2019) and Zhou Ye (Zhou and Ye, 2023), domain-specific models in education perform significantly better than

general-purpose ones. To mitigate domain shift, researchers employ domain adaptation, such as fine-tuning pre-trained models on in-domain datasets or leveraging domain-specific embedding.

1.9.4 Class Imbalance in Sentiment Datasets

Many real-world sentiment datasets are imbalanced, meaning that one sentiment class—typically positive—is far more represented than others. This imbalance causes models to be biased toward the dominant class, reducing their ability to identify minority sentiments such as negative or neutral feedback.

To address this, Bhagat et al.(Bhagat et al., 2024)and Sangeetha and Prabha(Sangeetha and Prabha, 2021)explored methods like oversampling, undersampling, and class-weighting during training. They also experimented with data augmentation and synthetic data generation to balance sentiment classes. While these techniques improve performance, they can also introduce noise or lead to overfitting if not implemented properly.

1.10 Conclusion

Understanding sentiments and opinions conveyed through text has become increasingly important across various fields. This chapter provided an overview of the fundamental stages of sentiment analysis, including text cleaning, feature representation, and the application of classification models. Although the integration of deep learning techniques and attention-based architectures has led to significant performance improvements, several issues—such as detecting sarcasm, handling multilingual content, and addressing domain-specific variations—persist. Ongoing research is vital to develop more adaptable and context-sensitive models capable of functioning effectively in real-world scenarios.

Chapter 2

Sentiment analysis with Long Short Term Memory LSTM

2.1 Introduction

Long Short-Term Memory (LSTM) is a powerful deep learning architecture designed to model sequential data and has become a cornerstone in the field of sentiment analysis(Hochreiter and Schmidhuber, 1997). As a variant of Recurrent Neural Networks (RNNs), LSTM was developed to overcome the limitations of traditional RNNs, particularly the vanishing and exploding gradient problems(Nguyen et al., 2018). What makes LSTM unique is its ability to retain long-term dependencies and contextual information over time, which is especially valuable when analyzing human language—where the meaning of a word often depends on the words that precede it(Murthy et al., 2020) & (Behera et al., 2021).

In sentiment analysis, where the objective is to detect the emotional tone behind text data, capturing the flow and structure of language is essential. LSTM achieves this through its memory cells and gating mechanisms (input, forget, and output gates), which allow the model to selectively store and update information across a sequence(Behera et al., 2021) & (Nguyen et al., 2018). This capability has made LSTM particularly effective for tasks like classifying opinions in social media posts, product reviews, and student feedback(Hochreiter and Schmidhuber, 1997).

In recent years, LSTM has outperformed many traditional machine learning techniques, showing remarkable accuracy in recognizing nuanced sentiments, even in complex or lengthy sentences. Its flexibility to be integrated with other models, such as CNNs or SVMs, further expands its potential in real-world applications (Behera et al., 2021). This chapter provides an in-depth exploration of how LSTM is applied to sentiment analysis, its architectural components, strengths, limita-

tions, and its overall impact on understanding subjective text data(Behera et al., 2021) & (Nguyen et al., 2018).

2.2 Introducing recurrent neural networks (RNN)

Recurrent Neural Networks (RNNs) form a category of neural networks specifically designed for processing sequential data. Unlike traditional networks, they feature recurrent connections that allow them to retain and integrate contextual information over extended periods(Sherstinsky, 2020).

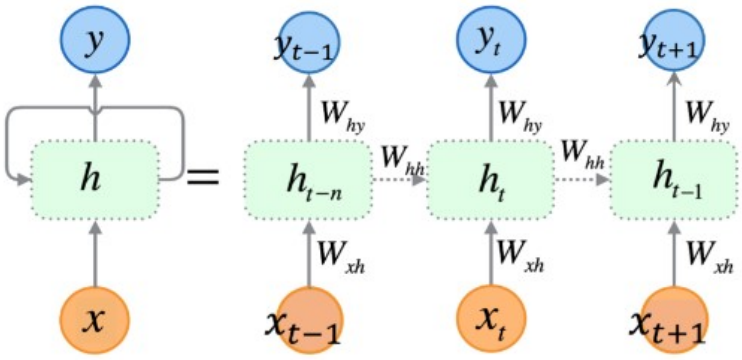

Figure 2.1: The recurrent neural network.

2.2.1 Mathematical operation

At each time step t, the RNN receives an input X_t and combines it with the previous hidden state H_{t-1} to produce a new hidden state H_t. This process is described by the following equation(Schmidt, 2019):

$$H_t = \varphi_h \left(X_t W_{xh} + H_{t-1} W_{hh} + b_h \right) \tag{2.1}$$

- X_t : Input vector at time t.
- H_{t-1} : Previous hidden state (memory of past inputs).
- W_{xh} : Weight matrix connecting the input to the hidden state.
- W_{hh} : Weight matrix connecting the previous hidden state to the new one.

- b_h : Bias term.
- φ_h : Activation function (typically tanh or sigmoid).

Next, the hidden state H_t is used to produce an output O_t through an additional layer, as described by the following equation(Schmidt, 2019):

$$O_t = \varphi_o \left(H_t W_{ho} + b_o \right) \tag{2.2}$$

From a depth perspective, neural networks that consist of more than one nonlinear hidden layer are commonly classified as deep neural networks (DNNs). On the other hand, those with just a single nonlinear hidden layer—like the one depicted in Figure 2.2—are generally referred to as shallow neural networks (Gao et al., 2019).

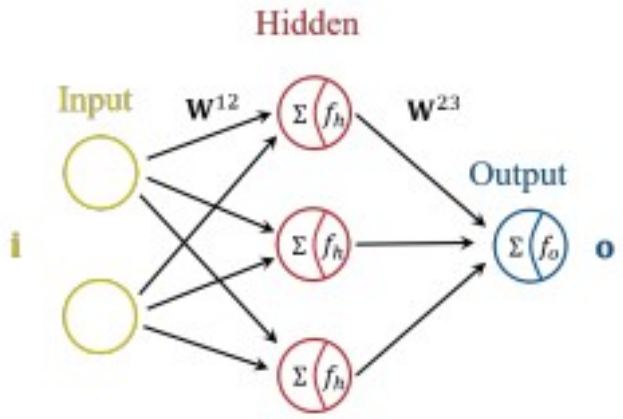

Figure 2.2: An example of an ANN

The weights W_{12} and W_{23} scale the outputs from the previous layers, while the element-wise activation functions f_h and f_o in the hidden and output layers perform nonlinear transformations. The network as a whole defines a composite function:

$$o = f_o(W_{23}\, f_h(W_{12}\, i)) \tag{2.3}$$

Given a training dataset $\{i, t\}$, the goal of training an artificial neural network (ANN) is to approximate the mapping between input i and target t by adjusting the weights W_{12} and W_{23} to best fit this composite function(Bengio et al., 1994).

Similar to traditional neural networks, the training process of a recurrent neural network (RNN) relies on the backpropagation of error. However, because RNNs process data sequentially, gradient calculations must account not only for the current state but also for past states in the sequence. This temporal dependency can result in an exponential decay of gradient values, especially when they are smaller than one. This issue, known as the vanishing gradient problem, poses a significant challenge for learning long-term dependencies in sequential data(Cherif, 2022).

In response to this limitation, Hochreiter and Schmidhuber (1997) introduced the Long Short-Term Memory (LSTM) network. This model is built upon a distinctive internal architecture that includes specialized components known as gates. These gates serve to regulate how information is stored, retained, and passed along through time steps, enabling the network to handle long sequential data more effectively(Hochreiter and Schmidhuber, 1997).

2.2.2 Fundamental Problem of Traditional RNNs

Challenges of RNNs with Long-Term Dependencies:
Recurrent neural networks face significant challenges when handling long-term temporal dependencies. One of their primary limitations is the difficulty in propagating error signals across time steps during the learning process. This time-dependent backpropagation can lead to a sharp decline in gradient values when weights are small, or conversely, to an overwhelming increase in gradient values when weights are large. Such fluctuations severely hinder the learning process, especially when it comes to memorizing information over extended time periods. Research by Hochreiter in 1991 revealed that these instabilities make it nearly impossible to learn long-term relationships when the time gap between an input and its corresponding output becomes too large (Hochreiter and Schmidhuber, 1997).

LSTM it's A Robust Solution to the Issue In response to the challenges faced by traditional recurrent networks, Hochreiter and Schmidhuber introduced a novel architecture in 1997, known as Long Short-Term Memory (LSTM). This advanced model was developed to maintain a consistent error flow throughout the learning process, preventing both vanishing and exploding gradients. The core of this architecture is the memory cell, a crucial element designed to store information over extended periods. This memory cell is supported by a stable recurrent connection, referred to as the Constant Error Carousel (CEC), which has a fixed weight of 1. This setup enables the retention of information over time without any loss (Hochreiter and Schmidhuber, 1997).

2.3 Long Short Term Memory (LSTM)

The Long Short-Term Memory (LSTM) architecture is a refined variant of recurrent neural networks (RNNs), specifically developed to address the difficulty of capturing long-term dependencies in sequential data. Traditional RNNs often struggle with either vanishing or exploding gradients during training, making it hard to retain information across extended sequences. LSTM addresses this issue through a unique structural design that ensures a stable propagation of error signals via its memory cells, enabling the model to effectively learn from long-range temporal patterns.(Houdt et al., 2020)

A standard LSTM unit contains a memory cell that preserves its internal state across time steps. This cell is controlled by a set of multiplicative gates—namely the input gate, forget gate, and output gate. These gates manage how information is handled within the cell by determining what should be added to memory, what should be discarded, and what should be output to affect the network's predictions.(Houdt et al., 2020)

An LSTM network is composed of a sequence of interconnected memory blocks. Each block has a sophisticated internal structure that enables it to retain, discard, or update information depending on the input data sequence it receives (Moukodouma et al., 2024) .LSTM networks are well-suited for learning from long sequences, even when important information is spread across many time steps. They maintain reliable and consistent memory over time. In contrast to conventional RNNs, which face challenges when dealing with extended delays between inputs and outputs, LSTMs are capable of capturing long-term temporal dependencies. This is largely due to their internal structure known as the Constant Error Carousel (CEC), which ensures a stable flow of gradients by preventing them from either vanishing or exploding. This feature enables the model to preserve information for longer durations. Because of this capability, LSTMs are widely used in applications that require understanding long-range dependencies, such as speech recognition, machine translation, and the analysis of time series data(Houdt et al., 2020).

2.3.1 LSTM Architecture

The Long Short-Term Memory (LSTM) network is a refined variant of recurrent neural networks (RNNs), specifically designed to overcome the challenges conventional RNNs face when modeling long-term dependencies. At the core of the LSTM lies a memory cell that preserves information across time steps, controlled by three nonlinear gates—input, forget, and output—that govern how information is stored, discarded, or exposed (Mahadevaswamy and Swathi, 2023).

There are four layers instead of just one, and they interact in a very unique manner. The recurring module in LSTM is illustrated in Figure 2.3.

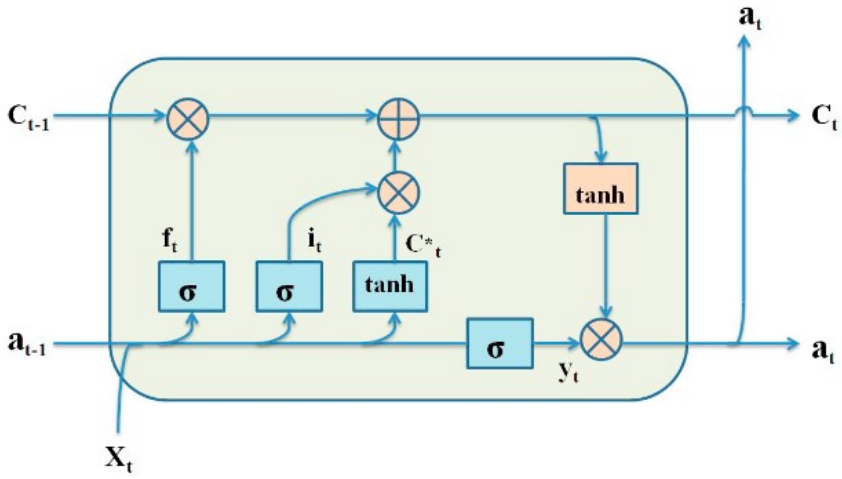

Figure 2.3: Recurring module in LSTM

The cell state, typically illustrated as the horizontal flow at the top of the LSTM unit, functions somewhat like a conveyor belt. It moves along with minimal linear transformations, allowing information to flow relatively unchanged. However, LSTMs can selectively update this state by adding or removing content through components known as gates. These gates incorporate a combination of sigmoid and tanh activations along with element-wise multiplication to manage the information flow. An LSTM cell includes three primary gates: the input gate, the forget gate, and the output gate (Yu et al., 2019).

Figure 2.5 illustrates the structure of a standard LSTM block, highlighting its components such as the gates, input signal $x^{(t)}$, output $y^{(t)}$, activation functions, and peephole connections. The block's output is fed back into its own input as well as into each of the gates through recurrent connections.(Houdt et al., 2020) At each time step t, the LSTM block processes the current input x_t, the previous hidden state y_{t-1}, and the previous cell state c_{t-1}. These components interact through key operations we will see it(Houdt et al., 2020).

Each LSTM unit includes:

- A **memory cell** c_t that stores long-term information, The memory cell plays a central role in the functioning of LSTM networks. It is designed to store information over long durations and is controlled by multiplicative gates that regulate the flow of data. One of its key advantages is its ability to maintain

a consistent error signal, which helps prevent the vanishing or exploding gradient problems typically encountered in standard RNNs. This stability is enabled by a fixed recurrent connection known as the Constant Error Carousel (CEC), which allows error signals to be carried across time steps without diminishing(Houdt et al., 2020).

$$c_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{2.4}$$

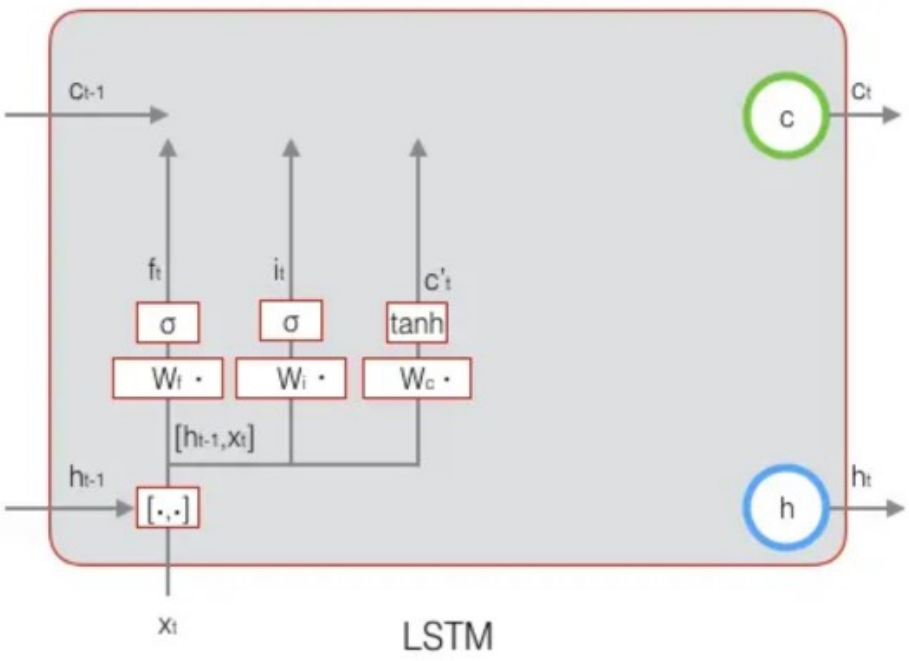

Figure 2.4: Memory cell (Qixuan, 2024)

- Three **gates**:

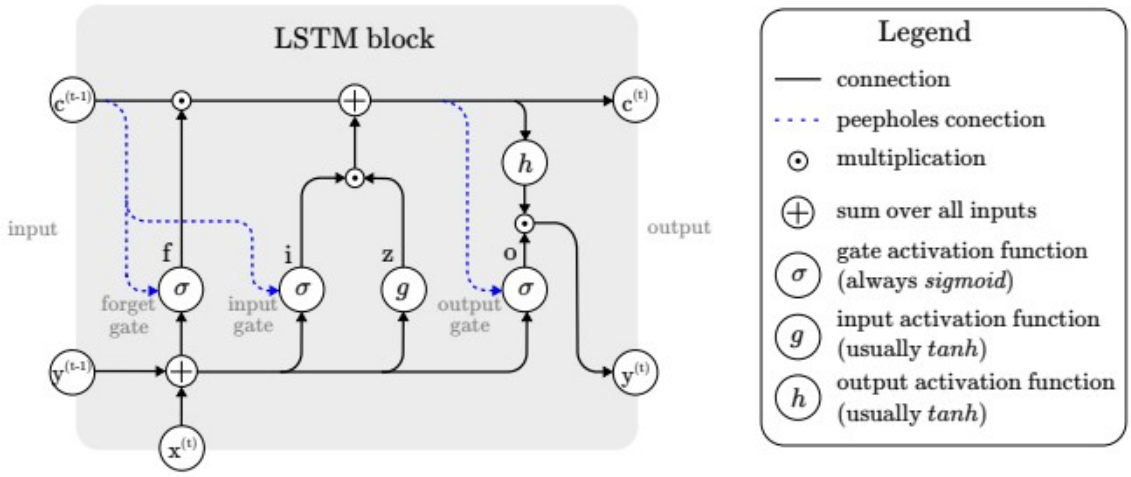

Figure 2.5: Architecture of a typical vanilla LSTM block(Houdt et al., 2020).

- Input gate (i_t)
- Forget gate (f_t)
- Output gate (o_t)

Input Gate: Controls how much of the new information flows into the memory cell:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.5}$$

Forget Gate: Determines which part of the previous memory should be discarded:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.6}$$

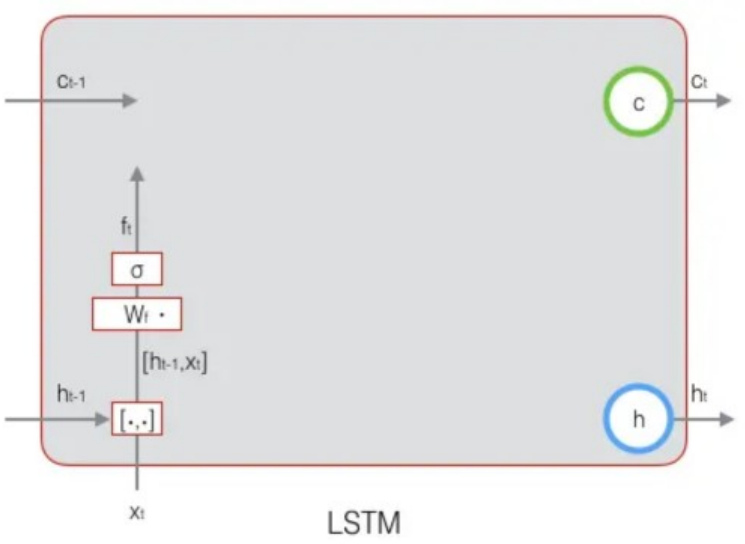

Figure 2.6: Forget Gate(Qixuan, 2024)

Output Gate: Regulates which part of the cell state is exposed as output:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.7}$$

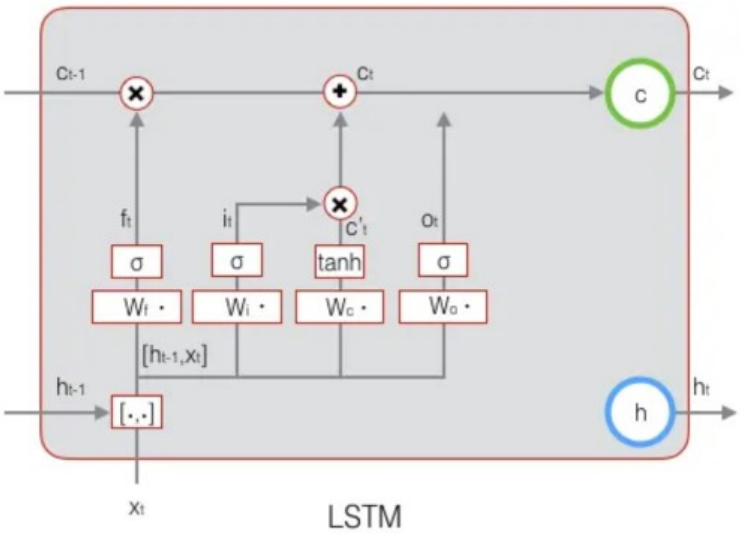

Figure 2.7: Output Gate(Qixuan, 2024)

- A **hidden state** (h_t) used as the unit's output,Computes the final hidden state based on the cell state and output gate.

$$h_t = o_t \circ \tanh(c_t) \tag{2.8}$$

Where:

- x_t : Input vector at time step t.
- h_{t-1} : Hidden state from the previous time step.
- c_{t-1} : Cell state from the previous time step.
- c_t : Candidate cell state.
- h_t : Current hidden state (also output).
- i_t : Input gate activation vector.
- f_t : Forget gate activation vector.
- o_t : Output gate activation vector.
- W_*, U_* : Weight matrices for input and recurrent connections.
- b_* : Bias vectors.
- σ : Sigmoid activation function.
- tanh : Hyperbolic tangent activation function.
- $\circ$: Element-wise (Hadamard) product.

2.4 Sentiment Analysis Using LSTM

A sentiment analysis model leveraging Long Short-Term Memory (LSTM) networks is introduced to interpret textual data, such as student feedback in E-learning environments. As a specialized form of Recurrent Neural Network (RNN), LSTM is well-suited for identifying long-range dependencies in sequential text. It addresses the shortcomings of conventional RNNs by incorporating memory cells, which are regulated by three types of gates—input, forget, and output—that manage how information is stored, discarded, and accessed. The model processes the input through a pipeline involving data cleaning, tokenization, label encoding, and word embedding. These embeddings convert text into dense vectors that capture

semantic meaning. Once the data is prepared, LSTM layers are used to learn temporal features, followed by a dense output layer with a SoftMax function to classify the sentiments. Performance improves with additional training epochs, reaching accuracy levels above 99% after ten iterations. This demonstrates the LSTM's strength in handling sentiment analysis tasks, particularly when working with large-scale and unstructured textual content (Murthy et al., 2020).

2.4.1 Steps in Sentiment Analysis Using LSTM

Data Preprocessing

Data preprocessing marks the first phase in sentiment analysis using LSTM. Initially, the text data is cleaned by converting it to lowercase and removing unnecessary punctuation. Following this, the text is broken down into smaller units (tokens) such as words or subwords. Each token is then assigned a unique integer value through encoding. Word embeddings are generated to represent words as dense vectors, which capture their semantic meanings. This transformation allows the model to better understand the relationships between words, facilitating more efficient text processing.(Murthy et al., 2020).

Model Input:

The preprocessed text is transformed into a sequence of word embeddings. Each word in the sentence is represented by a vector that encapsulates its semantic meaning. These sequences of embeddings are then passed into the LSTM network, maintaining the structure and order of words in the sentence. This sequential processing enables the LSTM model to interpret information in a manner similar to how humans understand sentences by considering the entire context.(Murthy et al., 2020).

LSTM Processing

LSTM networks are designed to handle sequential data like sentences by using memory cells that preserve information across time steps. The network utilizes three gates: the input gate, which controls the flow of information into the memory cell; the forget gate, which determines what information is discarded; and the output gate, which dictates the output based on the stored information. These gates work together to enable the LSTM to learn dependencies and store relevant

information throughout the sequence.(Murthy et al., 2020).

Sentiment Prediction

After processing the sequence, LSTM produces a hidden state at each time step, which represents the accumulated knowledge. The final hidden state, which encodes the entire context of the sequence, is passed through a SoftMax function to predict the sentiment. Based on the output from SoftMax, the model classifies the sentiment as positive, negative, or neutral, with the probability distribution indicating the confidence in each class.(Murthy et al., 2020).

Training

The LSTM model is trained using backpropagation through time (BPTT), adjusting its parameters to minimize the error between predicted and actual sentiment labels. Optimization techniques such as gradient descent are used to update the model's weights iteratively, allowing the network to capture the complex patterns and relationships present in the data.(Murthy et al., 2020).

Classification

Once trained, the model can classify unseen text. The output of the SoftMax layer provides a probability distribution across the sentiment classes (positive, negative, or neutral). The sentiment with the highest probability is selected as the final classification. This process enables the LSTM model to make accurate predictions based on the contextual and sequential nature of the text.(Murthy et al., 2020).

2.5 Conclusion

In summary, the LSTM model has proven to be a highly effective approach for sentiment analysis, particularly for analyzing large, sequential text data. Its architecture, which is designed to retain long-term dependencies, makes it especially useful for understanding sentence context, a critical element for accurate sentiment classification. By processing text sequentially, LSTM models are capable of detecting subtle emotions within more intricate feedback, such as that found in product reviews or educational comments. However, despite its advantages, LSTM models present challenges, including high computational costs and difficulties with rare

or highly contextual errors. Nevertheless, with ongoing advances in deep learning and the availability of large datasets, LSTM remains a key tool in sentiment analysis, providing valuable insights that can influence business strategies, educational improvements, and more.

Chapter 3

Design

3.1 Introduction

This chapter presents the design of a sentiment analysis system whose main objective is to automatically analyze student comments using a deep learning model, specifically a Long Short-Term Memory (LSTM) neural network. The system aims to classify comments into three sentiment categories: positive, negative, and neutral.

3.2 Architecture of the Proposed Approach

The system employs a modular framework encompassing the entire processing pipeline, from data ingestion to sentiment prediction and evaluation.

3.2.1 Module Overview

- **Data Acquisition:**Retrieving Excel files stored on Google Drive.
- **Text Preprocessing:** Standardizing and cleaning input text through tokenization and normalization.
- **Automatic Annotation:** Leveraging lexicon-based methods (TextBlob) for sentiment labeling.
- **Dataset Balancing:**Ensuring uniform distribution across sentiment categories.

- **Model Training:** Building and training an LSTM-based classification model.
- **Evaluation and Reporting:**Measuring performance using standard classification metrics.

The proposed approach integrates classical Natural Language Processing (NLP) techniques with deep learning to form a robust sentiment analysis pipeline. Initially, student feedback comments are collected and preprocessed to normalize the text. The comments are then automatically labeled using a lexicon-based method (TextBlob), which estimates the polarity of each statement. To enhance model fairness, class balancing is applied to equalize the representation of each sentiment category.

Following this, the text data is tokenized and transformed into padded sequences of integers, preparing it for ingestion by a bidirectional LSTM network. The model learns contextual representations of sequences by processing them in both forward and backward directions, allowing it to capture intricate linguistic patterns. The output is fed into dense layers for final sentiment classification.

Model training is conducted using an optimized configuration with early stopping to prevent overfitting. Finally, evaluation is performed using accuracy, precision, recall, and F1-score to assess model performance across all sentiment classes. The architecture of the proposed model is illustrated in Figure 3.1.

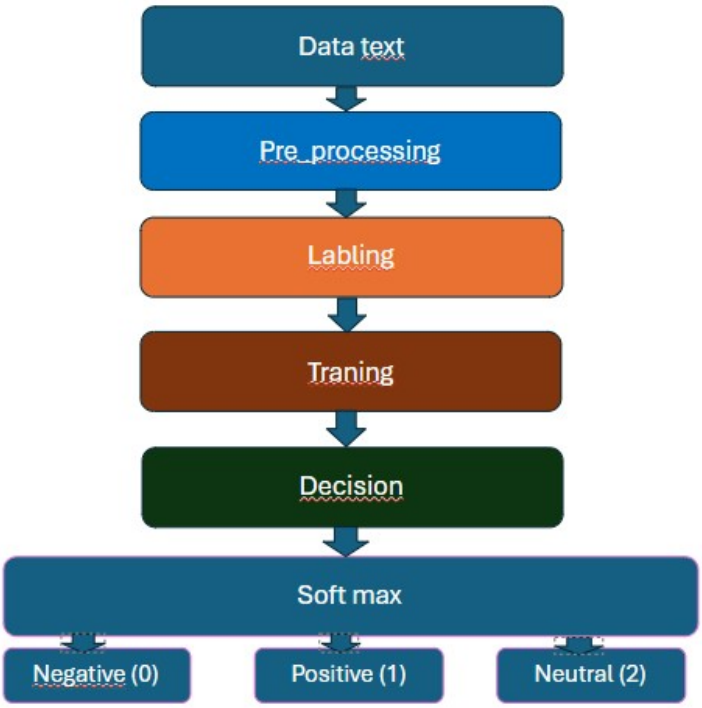

Figure 3.1: Architecture of proposed network used

3.3 Approach Description

3.3.1 Text Preprocessing

Text preprocessing is a fundamental step in preparing raw textual data for machine learning and deep learning. It transforms unstructured comments into clean, consistent text suitable for analysis. Below, we provide detailed explanations of each substep, along with illustrative examples. Preprocessing is a crucial step in any natural language processing pipeline. Its goal is to clean and normalize the input text, ensuring it is suitable for downstream machine learning and deep learning tasks. The steps followed during pre-processing are illustrated in Figure 3.2.

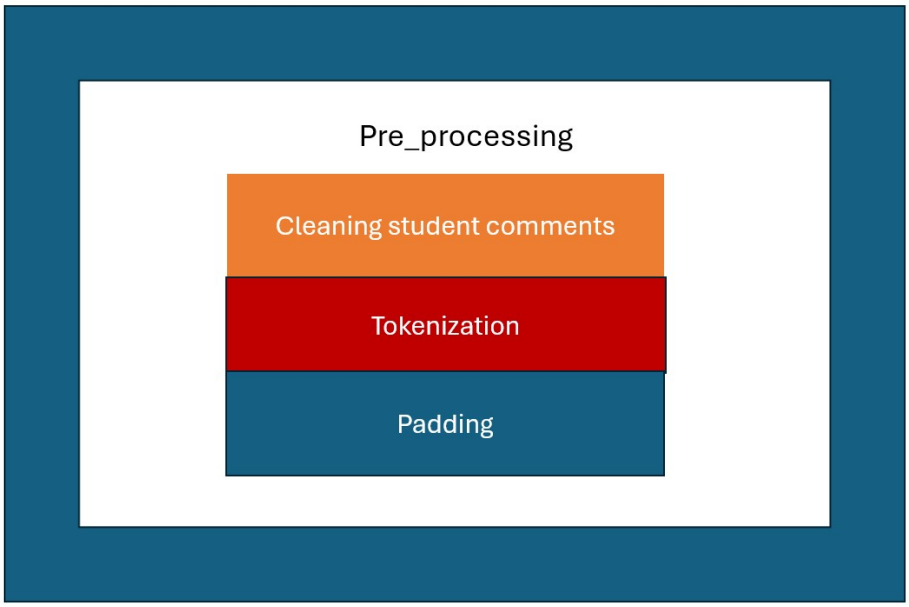

Figure 3.2: Architecture of Preprocessing

Cleaning Student Comments

This stage involves removing extraneous symbols and normalizing the case of characters.
Example:
Original: "This COURSE is amazing!!! 100% recommended :)"
Cleaned: "this course is amazing recommended"
The raw text data is cleaned by removing unwanted elements such as:

- Punctuation marks and special symbols
- Numerical digits
- Uppercase letters (converted to lowercase)
- Stopwords (optional, for some experiments)

This step helps standardize the comments and reduce noise.

Tokenization

Tokenization is the process of splitting text into meaningful units—tokens—such as words or subwords. This enables the transformation of text into sequences that models can process.
Steps:

- Use of a Keras Tokenizer to assign a unique integer index to each word.
- Construction of a vocabulary dictionary.

Example 1:
Cleaned Text: ”i loved the platform recommended”
Tokens: [”i”, ”loved”, ”the”, ”platform”, ”recommended”] Token IDs: [2, 15, 5, 46, 98] Tokenization splits text into individual words (tokens) that can be mapped to integers.
Example 2:
Cleaned Text: ”this course is amazing recommended”
Tokens: [”this”, ”course”, ”is”, ”amazing”, ”recommended”] Token IDs: [1, 23, 5, 102, 87]
After cleaning, the text is split into individual tokens (words or subwords). Each token is mapped to an integer using a tokenizer, which creates a vocabulary dictionary based on frequency.

Padding

Neural networks require inputs of uniform shape. Padding ensures that all sequences have the same length.
Steps

- Determine the maximum sequence length in the dataset.
- Add zeros to shorter sequences until they match the maximum length.

Example 1:
Token Sequence: [2, 15, 5, 46, 98]
Max Sequence Length: 7
Padded Sequence: [2, 15, 5, 46, 98, 0, 0]
All sequences are padded to ensure equal length input for the model.
Example 2: Original Token IDs: [1, 23, 5, 102, 87]
Padded (max length = 7): [1, 23, 5, 102, 87, 0, 0]
Since neural networks require input sequences of uniform length, all tokenized sequences are padded with zeros to match the length of the longest comment in the dataset.

3.3.2 Labeling the dataset

The proposed sentiment analysis framework integrates a lexicon-based annotation strategy with a deep learning classification model to enhance the reliability of

sentiment predictions. Initially, the raw textual data undergoes a comprehensive preprocessing phase, including the removal of user tags, hyperlinks, punctuation, and numerals, followed by lowercasing, stopword elimination, and optional stemming or lemmatization.

Once cleaned, each text sample is passed through TextBlob, a rule-based sentiment analyzer that computes a polarity score ranging from – 1 to +1.
TextBlob is a high-level natural language processing (NLP) library for Python that simplifies the implementation of various text analysis tasks, including part-of-speech tagging, noun phrase extraction, text classification, translation, and notably, sentiment analysis. It is built on top of more complex NLP toolkits such as NLTK and Pattern, offering a user-friendly API that allows developers and researchers to perform powerful text operations with minimal code.

Based on predefined thresholds (e.g., polarity > 0.1 as positive, < -0.1 as negative, and in between as neutral), sentiment labels are automatically assigned.

These generated labels are then used to train a Long Short-Term Memory (LSTM) neural network, which is well-suited for handling sequential data due to its ability to retain contextual information over time. The text inputs are tokenized and transformed into padded sequences, then mapped into a dense vector space using an embedding layer.

The LSTM model learns to capture semantic patterns and sentiment cues from the input sequences, using the TextBlob-provided labels as supervisory signals.

This hybrid methodology effectively addresses inconsistencies found in manually annotated datasets, reduces subjective bias, and leads to improved classification performance.

By combining automated lexicon-based annotation with data-driven learning, the system offers a scalable and robust solution for sentiment analysis tasks (Aljedaani et al., 2022).

3.3.3 Training

In this project, we implemented a deep learning model based on a Bidirectional Long Short-Term Memory (BiLSTM) architecture to perform sentiment analysis on student comments. The model was designed to process and classify text data into three sentiment categories: positive, negative, and neutral. It begins with an embedding layer that converts each word into a dense vector representation, limited to the top 10,000 most frequent words in the corpus. This is followed by a Bidirectional LSTM layer with 128 units, which captures contextual information from both directions of the sequence. To reduce overfitting, dropout layers were added after the LSTM (0.5) and dense (0.3) layers. A fully connected dense layer

with 64 neurons and ReLU activation is included before the output layer, which uses a softmax activation function to return probabilities for the three sentiment classes. The model was compiled using the Adam optimizer and trained with categorical crossentropy as the loss function. Early stopping was applied to monitor the validation loss and stop training if no improvement was observed for two consecutive epochs. The dataset was split into 80% training and 20% testing subsets, and training was performed over a maximum of 16 epochs with a batch size of 16.

3.3.4 Decision

LSTM networks are particularly effective for handling sequential data due to their ability to maintain memory over long input sequences. This makes them well-suited for analyzing the context and structure of natural language. The LSTM model design consists of the following components:

- **Embedding Layer:** Converts token indices into dense vectors of dimension 128.
- **Bidirectional LSTM:** Captures both forward and backward dependencies in the sequence.
- **GlobalMaxPooling1D:** Selects the most salient features from LSTM outputs.
- **Dense Layers:** Facilitates the final sentiment classification via a softmax output.

The layer-by-layer architecture of the LSTM model is illustrated in Figure 3.3.

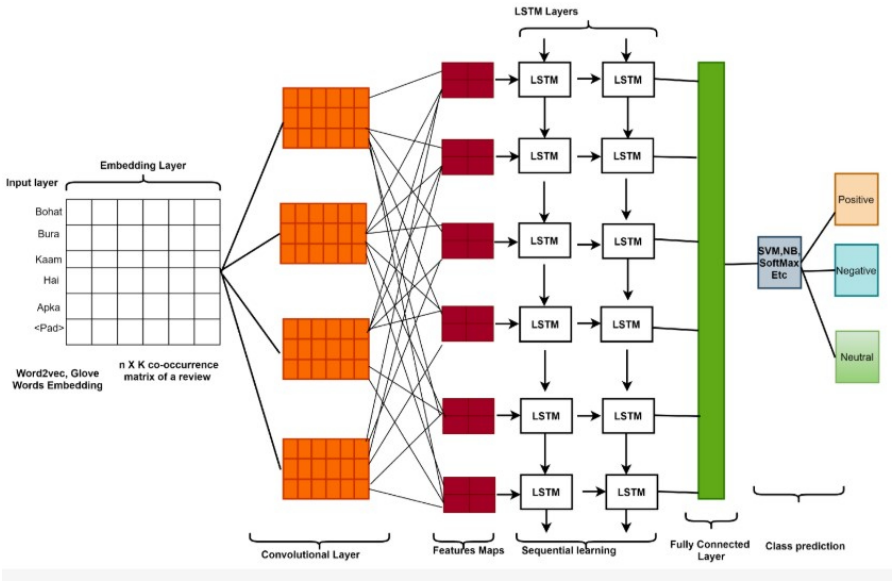

Figure 3.3: LSTM framework of sentiment analysis(Khan et al., 2022).

3.4 Conclusion

The proposed system illustrates the design of a deep learning pipeline for sentiment analysis, tailored to educational comments. Leveraging LSTM architecture and automatic labeling, the framework enables efficient classification and lays the groundwork for further refinement using more sophisticated models such as CNN-LSTM or transformer-based networks.

Chapter 4

Implementation

4.1 Introduction

After detailing the proposed approach in the previous chapter, this section presents the practical implementation of our sentiment analysis system. It describes the technical environment, the tools used, and the various stages involved in transforming the conceptual design into an operational model. The main objective remains the classification of learner comments collected from e-learning platforms based on their emotional polarity, using a deep learning architecture built around Long Short-Term Memory (LSTM) networks.

The first part of this chapter outlines the development environment, including the programming frameworks, software libraries, and hardware configuration. The second part focuses on the implementation of the data processing pipeline — from preprocessing and automatic labeling to feature extraction and model training. Finally, we present the evaluation of the trained model on real-world data, along with performance metrics that allow us to assess its effectiveness.

4.2 Environment

4.2.1 Cloud-Based Development Platform: Google Colaboratory

Google Colaboratory, commonly known as Google Colab, is a cloud-based programming platform developed by Google Research. It allows users to write and execute Python code directly from a web browser, without requiring any local configuration or installation. Colab offers an interactive notebook environment similar to Jupyter, enriched with features for collaboration, such as sharing notebooks via Google Drive and enabling real-time edits by multiple users (Naik, 2023).

One of Colab's main advantages is its free access to powerful hardware accelerators, including GPUs and TPUs, which are essential for training deep learning models efficiently. This makes it an ideal tool for students, researchers, and developers working on machine learning and data analysis tasks. Moreover, Colab comes preloaded with popular Python libraries such as TensorFlow, Keras, and scikit-learn, reducing setup time and allowing rapid development and experimentation.(Naik, 2023)

Figure 4.1 presents the interface of Colab.

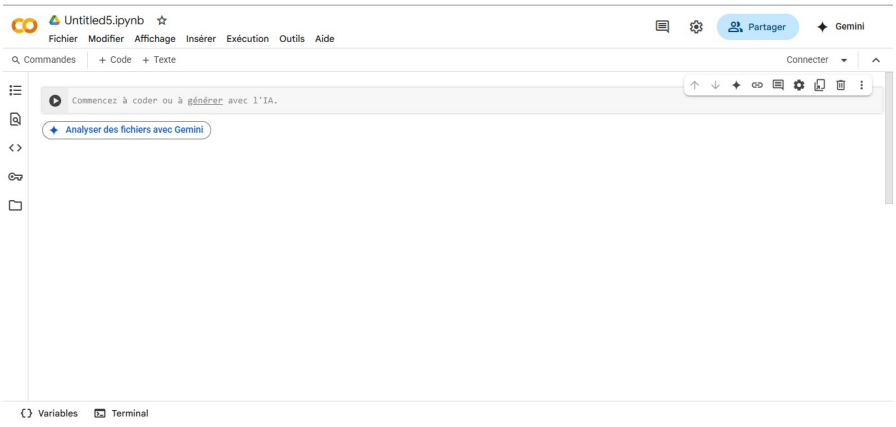

Figure 4.1: Colab interface

Colab is especially suitable for tasks involving data science, natural language processing, and deep learning. One of its major advantages is its free access to hardware acceleration, including Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs), which are essential for training complex neural networks such as LSTMs.

From a technical standpoint, Colab provides the following features:

Python Code Execution: Users can run code in separate cells, visualize output inline, and interactively debug or modify blocks of logic.

Collaboration and Sharing: Notebooks can be easily shared via Google Drive, with permissions for viewing or editing, enabling real-time teamwork.

Pre-installed Libraries: Colab comes with a wide range of libraries commonly used in machine learning, such as TensorFlow, scikit-learn, Pandas, NumPy, Matplotlib, and NLTK.

Visualization Tools: Interactive graphs and charts can be generated directly within the notebook using tools such as Matplotlib and Seaborn.

Custom Environment Support: Users can install additional Python packages as needed using pip commands within the notebook environment.

Google Colab served as the ideal development platform for the implementation

and testing of our LSTM-based sentiment analysis system, providing both computational efficiency and development flexibility.

4.3 Programming Language: Python

Python is a high-level, interpreted programming language widely used in the fields of artificial intelligence, data science, and natural language processing. Its readable syntax, extensive community support, and large ecosystem of libraries make it particularly well-suited for academic and industrial research.

In this project, Python served as the primary language for implementing the entire sentiment analysis system. It enabled seamless integration of data processing modules (via Pandas and NLTK), automatic labeling (via TextBlob), and deep learning model development (via TensorFlow and Keras). Python's flexibility and compatibility with cloud platforms like Google Colab make it a preferred choice for machine learning practitioners and researchers worldwide (Naik, 2023).

Python's extensive support for scientific computing—through libraries such as NumPy and Pandas—makes it particularly well-suited for data preprocessing. Moreover, the availability of frameworks like TensorFlow, Keras, and NLTK facilitates the implementation of complex natural language processing workflows and neural network architectures such as LSTM.

Given these advantages, Python was selected as the core language to implement the various components of our system: from data cleaning and sentiment labeling to model training and evaluation.

4.4 Libraries and Frameworks Used

The implementation of our sentiment analysis system required a range of libraries that support both natural language processing and deep learning. Each library played a specific role in building and training the model, preprocessing the data, or evaluating the results.

4.4.1 Pandas

Pandas is a core open-source Python library designed specifically for data manipulation and statistical analysis. It provides flexible and powerful data structures such as Series (one-dimensional) and DataFrames (two-dimensional) which allow for efficient handling, filtering, and transformation of structured data.

Originally developed for financial data processing, pandas has since become a foundational tool in data science workflows. It supports a wide range of operations,

including reading and writing data from various formats (CSV, Excel, JSON), handling missing values, grouping and aggregating data, and integrating seamlessly with numerical libraries such as NumPy and visualization libraries like Matplotlib.

In the context of this project, pandas was used to load the dataset of learner comments, normalize text fields, label sentiment categories, and prepare the input structure required for subsequent processing by the machine learning model (McKinney, 2011).

4.4.2 NumPy

NumPy (Numerical Python) is a fundamental Python library for numerical computing, and it forms the core of scientific and machine learning ecosystems in Python. It provides a high-performance multidimensional array object, known as ndarray, along with a suite of functions to perform efficient mathematical operations on large datasets.

Originally introduced and detailed in Travis Oliphant's Guide to NumPy, the library was designed to extend Python with the capabilities of performing array-based computing efficiently. It supports vectorized operations, linear algebra routines, Fourier transforms, and advanced random number generation—all of which are essential in data science and machine learning workflows.

In the context of this project, NumPy was employed to support matrix operations, transform textual data into numerical formats, and manage arrays during model training and preprocessing. Its tight integration with other libraries such as Pandas, TensorFlow, and scikit-learn makes it an indispensable tool for implementing robust and scalable analytical systems (Oliphant, 2006).

4.4.3 NLTK (Natural Language Toolkit)

The Natural Language Toolkit (NLTK) is a comprehensive suite of Python libraries and educational resources developed to facilitate the teaching and implementation of natural language processing (NLP) techniques. First introduced by Bird, Loper, and Klein, NLTK was designed both as a research tool and as a pedagogical platform for students and practitioners in computational linguistics.

NLTK provides modular and well-documented interfaces to over 50 corpora and lexical resources such as WordNet, as well as a wide variety of text processing libraries for classification, tokenization, stemming, lemmatization, part-of-speech tagging, parsing, and semantic reasoning.

In this project, NLTK was essential for preprocessing learner comments. It enabled tasks such as tokenizing text into words, removing French stopwords, and applying lemmatization to normalize textual data before it was passed to the sentiment analysis model. Due to its extensibility and multilingual support, NLTK

remains one of the most widely used libraries in NLP research and education (Bird et al., 2006).

4.4.4 TextBlob

TextBlob is a high-level natural language processing library built on top of NLTK and Pattern. It offers a simple and intuitive API for performing common text processing tasks such as tokenization, part-of-speech tagging, noun phrase extraction, translation, and—most notably—sentiment analysis. In sentiment classification, TextBlob calculates two primary scores:

- **Polarity**, which ranges from -1 (completely negative) to +1 (completely positive),
- **Subjectivity**, which measures the degree of personal opinion versus factual information.

One of the main advantages of TextBlob is its ease of integration into sentiment analysis pipelines, especially for quick prototyping and semi-supervised classification. It uses a rule-based sentiment lexicon to assess the emotional content of a given sentence or document.

In the context of this project, TextBlob was used to automatically label learner comments with sentiment classes (positive, negative, neutral), based on their polarity scores. This enabled the generation of training data for the LSTM model without requiring manual annotation (Illia et al., 2021).

4.4.5 scikit-learn

scikit-learn is an open-source Python library designed for efficient implementation of machine learning algorithms and tools. Built on top of NumPy, SciPy, and matplotlib, it provides a consistent interface for a wide variety of supervised and unsupervised learning tasks.

scikit-learn includes modules for classification, regression, clustering, dimensionality reduction, model selection, and preprocessing. In the context of natural language processing (NLP) and sentiment analysis, it is particularly valuable for:

- Feature extraction techniques like Bag of Words (BoW) and TF-IDF
- Splitting datasets into training and test subsets
- Computing class weights to handle imbalanced data
- Evaluating models using metrics such as accuracy, precision, recall, and F1-score

Although originally not designed for deep learning, scikit-learn integrates seamlessly with libraries such as TensorFlow and Keras, providing a robust preprocessing and evaluation toolkit.

In our project, scikit-learn was used to extract textual features, manage data splits, compute balanced class weights, and generate detailed classification reports for evaluating the LSTM model's performance (Pölsterl, 2020).

4.4.6 TensorFlow and Keras

TensorFlow is an open-source deep learning framework developed by Google, designed to facilitate the construction, training, and deployment of machine learning models at scale. It offers a highly flexible architecture for numerical computation, supporting both low-level operations and high-level APIs.

Keras, integrated as the official high-level API of TensorFlow since version 2.0, provides a simplified and user-friendly interface for building and training neural networks. It abstracts much of the complexity of TensorFlow while preserving the ability to customize and optimize model behavior.

Together, TensorFlow and Keras allow for the implementation of complex models such as Long Short-Term Memory (LSTM) networks, which are particularly suited for processing and analyzing sequential data like text. Their extensive support for GPU acceleration, automatic differentiation, and modular design makes them ideal for sentiment analysis tasks in natural language processing (NLP).

In our project, TensorFlow and Keras were used to define a bidirectional LSTM architecture. The model was trained on sequences of learner comments and optimized using categorical crossentropy. The training process was monitored with early stopping and evaluated using accuracy and classification metrics. The integration of these two libraries enabled efficient and scalable model development in a cloud-based environment like Google Colab (Grattarola and Alippi, 2021).

4.4.7 Matplotlib

Matplotlib is a widely used open-source Python library designed for creating static, interactive, and animated visualizations. Originally developed to mimic the plotting capabilities of MATLAB, it has become a foundational tool for scientific computing and data visualization in Python.

The core of Matplotlib is its pyplot module, which provides a high-level interface for generating a wide range of plots, including line graphs, bar charts, scatter plots, histograms, and heatmaps. Its flexible architecture also allows for fine-grained control over plot elements such as axes, labels, legends, and colors, making it suitable for producing publication-quality figures.

In the context of this project, Matplotlib was used to visualize key training metrics of the LSTM model, such as accuracy and loss over training epochs. These plots provided valuable insights into the model's learning behavior and helped identify issues such as overfitting or underfitting during development(Tosi, 2009).

4.5 The dataset

4.5.1 Dataset Description

The dataset employed in this study consists of textual feedback collected from the Mark My Professor platform, which contains over 5,200 reviews written by learners about their courses and instructors Bouacida (2018). Mark My Professor is a platform dedicated to the evaluation of teachers and educational content by students.

The learners' comments are extracted from a file named dataE.xlsx, which was initially stored in a compressed archive (DATASETS.zip) on Google Drive. Each entry in the dataset corresponds to a comment written in English by a student on an online learning platform. These comments reflect the learners' experiences, opinions, or perceptions regarding different aspects of digital education.

Upon loading, the dataset was processed to standardize column names and extract the relevant text column (e.g., comment, commentaire, or similar). The raw comments underwent a cleaning phase, which involved removing punctuation, numeric values, hyperlinks, and user mentions, as well as converting all text to lowercase. This step ensures uniformity in the input data before further analysis.

Since the dataset originally lacked sentiment labels, a lexicon-based approach using the TextBlob library was applied to automatically assign sentiment categories. Each comment received a sentiment score (polarity) from TextBlob, which was then mapped to one of three classes:

Positive if polarity > 0.1

Negative if polarity $<$ -0.1

Neutral otherwise

This annotated dataset was then used to train a deep learning model. For that purpose, the textual data was tokenized using a Keras Tokenizer (with a vocabulary size of 10,000), converted into sequences of integers, and padded to equal length. The resulting inputs were paired with one-hot encoded sentiment labels to form the features (X) and targets (y) of the model.

In total, the dataset contains approximately 5000 student comments, and it is evenly distributed across the three sentiment classes, as shown by the bar chart generated in the exploratory phase. The final labeled version was exported to a CSV file (dataE-labellisee.csv) for reuse and evaluation.

4.5.2 Distribution of the dataset

To assess the performance of the proposed LSTM-based sentiment analysis model, the dataset was divided into three distinct subsets using a two-step stratified sampling approach. Initially, 10% of the data was set aside as a test set, ensuring it remained unseen during training and validation. The remaining 90% was then further split into 80% for training and 10% for validation. This resulted in an overall distribution of approximately 80% training, 10% validation, and 10% testing.

This division was implemented using the `train_test_split()` function from Scikit-learn, ensuring randomness while maintaining class balance. The detailed breakdown of this split is presented in Table 4.1.

Subset	**Percentage**	**Approx. Count**	**Purpose**
Training Set	80%	$\sim$ 4160	Model fitting
Validation Set	10%	$\sim$ 520	Parameter tuning
Test Set	10%	$\sim$ 520	Final evaluation

Table 4.1: Dataset split: training, validation and test sets

Purpose of Dataset Splitting

To ensure a reliable evaluation of the sentiment analysis model, the dataset was divided into three distinct subsets, each serving a specific function:

- **Training Set (80%) "Model fitting":**
  - Used to train the LSTM model by updating its internal weights.
  - The model learns to associate student comments with the correct sentiment labels.
  - These data are seen repeatedly during the training epochs.
- **Validation Set (10%) "Parameter tuning":**
  - Used to monitor the model's performance during training.
  - Helps tune hyperparameters (e.g., number of epochs, dropout rate).
  - Plays a key role in detecting overfitting.
- **Test Set (10%): "Final evaluation"**
  - Reserved for the final evaluation after training is complete.
  - Provides an unbiased estimate of model performance on unseen data.
  - Performance metrics such as accuracy, precision, recall, and F1-score are computed on this set.

4.5.3 Distribution of sentiments in the dataset

Table 4.2 and Figure 4.2 provides a detailed overview of this distribution.

Sentiment Class	Number of Comments
Positive (1)	2691
Neutral (2)	2048
Negative (0)	608

Table 4.2: Distribution of comments by sentiment class.

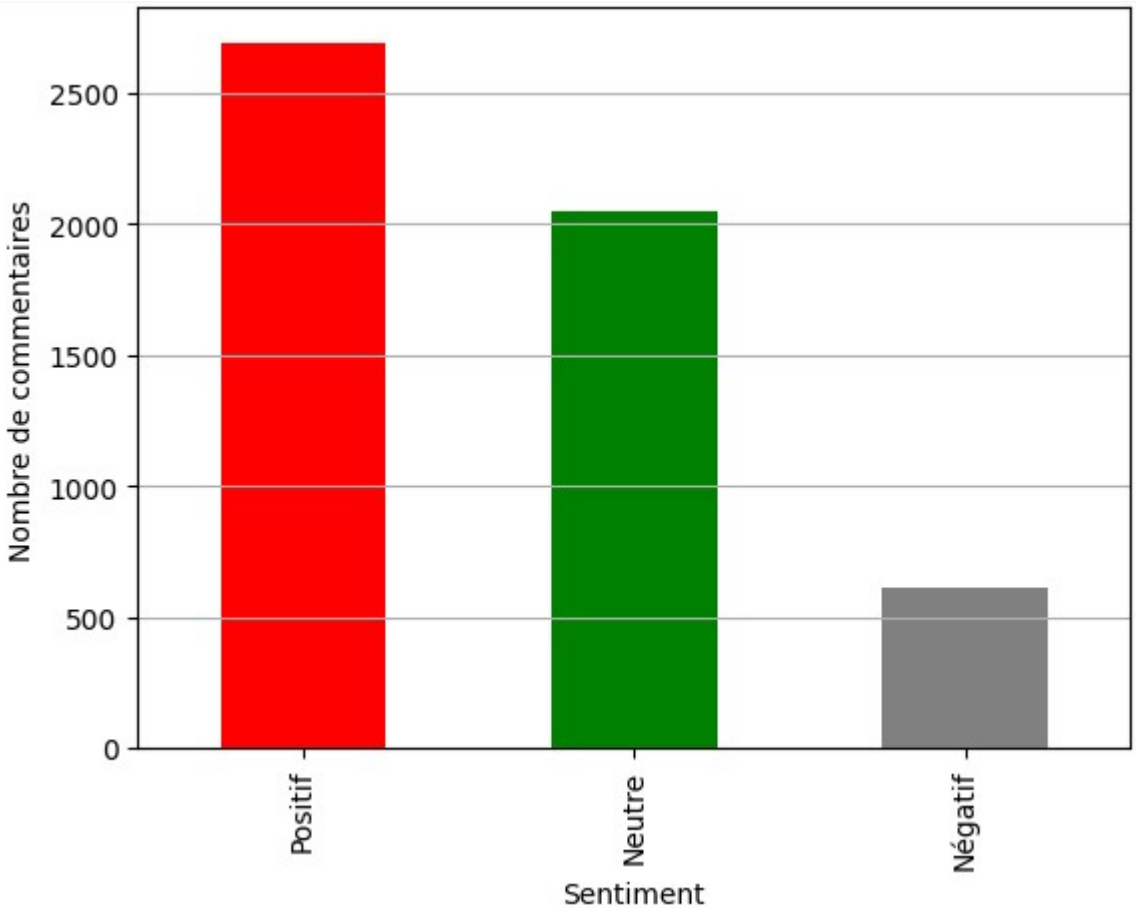

Figure 4.2: distribution of sentiment classes.

4.6 Training and Testing

This section outlines the training and evaluation process of the sentiment classification model. The dataset, composed of student comments, was first preprocessed and annotated automatically using TextBlob, a lexicon-based tool that assigns polarity scores to text. Based on these scores, comments were labeled as positive, negative, or neutral.

The core architecture employs a Bidirectional Long Short-Term Memory (Bi-LSTM) neural network, chosen for its ability to retain context in both forward and backward directions, making it well-suited for sequence modeling tasks like sentiment analysis. To ensure a balanced evaluation, the dataset was divided into three subsets: 80% for training, 10% for validation, and 10% for testing.

Each comment was tokenized and padded to a fixed length, then passed through an embedding layer to capture semantic relationships. The training process was guided by two key hyperparameters:

Batch Size: Defines the number of samples processed before the model's internal parameters are updated. A batch size of 16 was selected to balance memory usage and convergence speed.

Epochs: Represents the number of complete passes through the training dataset. The model was trained for up to 20 epochs, with early stopping enabled to prevent overfitting once the validation loss stopped improving.

The model's learning behavior was monitored using graphical plots of training and validation accuracy. Final performance was evaluated using a classification report and a confusion matrix (Figure 4.3), highlighting the model's predictive capability across the three sentiment categories. In the final training run, the model achieved a training accuracy of approximately 94% and a validation accuracy of around 74%, indicating a strong but not overfitted learning behavior (Figure 4.4).

Accuracy Type	Value
Training Accuracy	94%
Validation Accuracy	74%

Table 4.3: Accuracy results obtained during the different phases of the learning process.

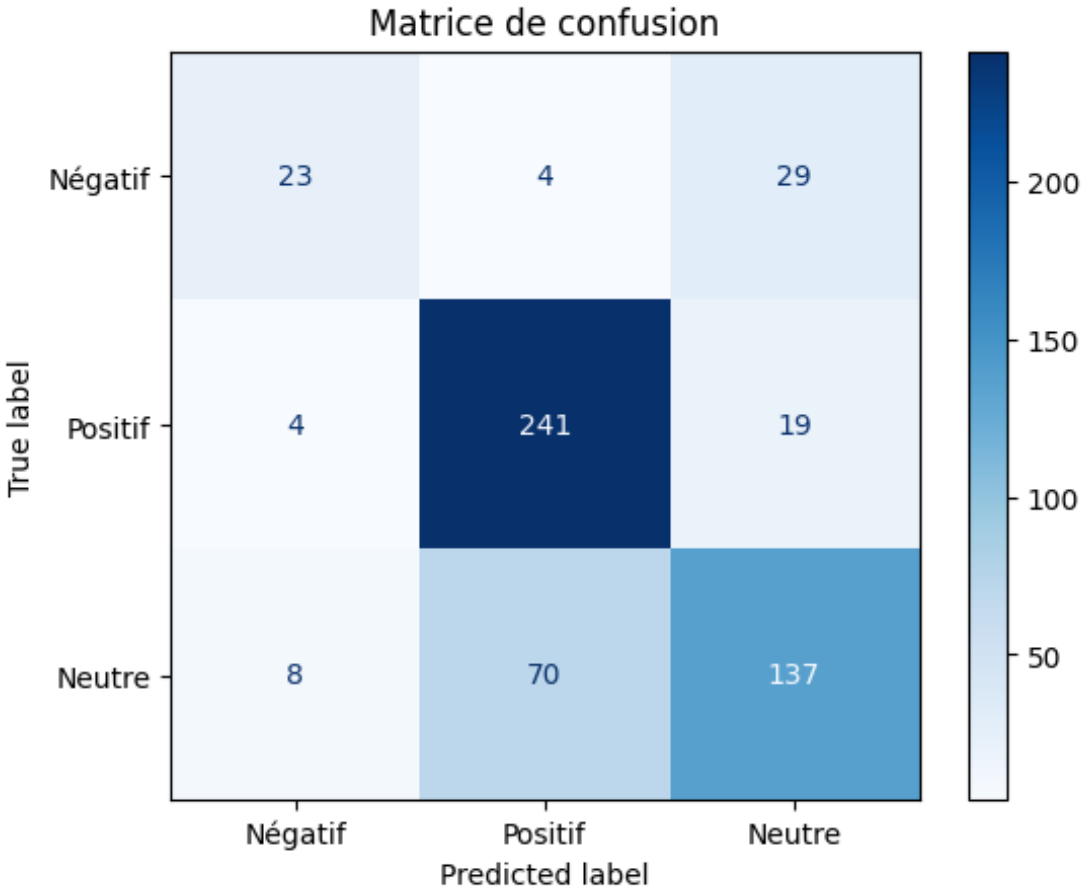

Figure 4.3: confusion matrix.

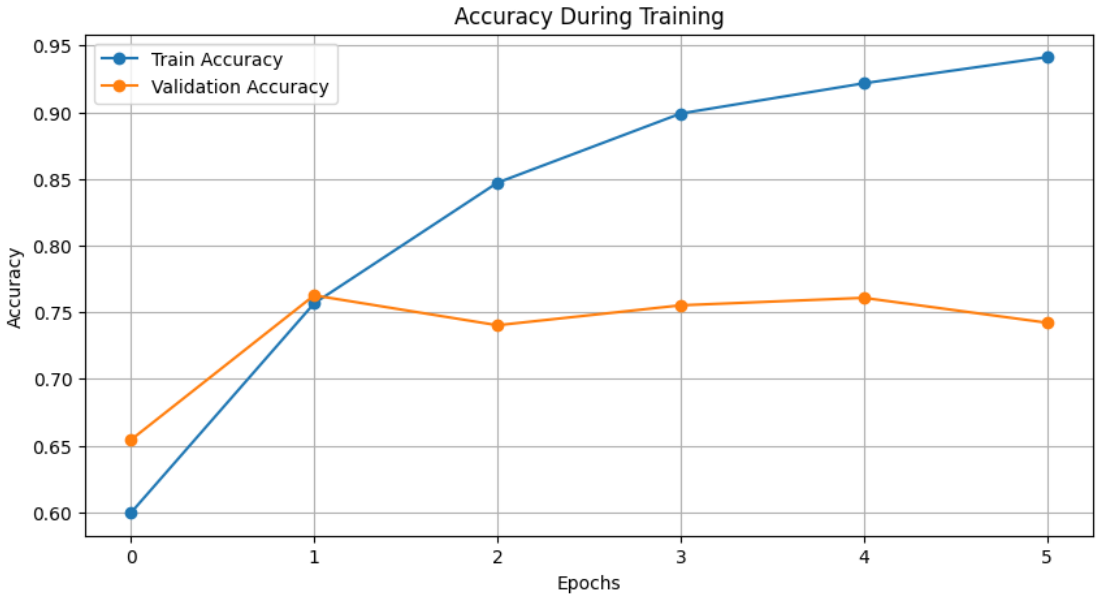

Figure 4.4: Train Accuracy and Validation Accuracy.

4.7 Obtained Results

In this section, we present the outcomes obtained using the LSTM model on the dataset composed of student comments. To evaluate the performance of the classifier, we relied on standard machine learning metrics based on the following fundamental concepts:

- **True Positive (TP)**: Cases where the model correctly predicts a given class (e.g., *Positive*) and the actual label is the same.
- **True Negative (TN)**: Cases where the model correctly predicts a class is not present (e.g., *Not Positive*) and the actual label confirms this.
- **False Positive (FP)**: Instances where the model incorrectly predicts a class when it is not actually present.
- **False Negative (FN)**: Instances where the model fails to predict the actual class and instead assigns a different one.

Using these indicators, we computed the following performance metrics:

- **Accuracy**: Represents the proportion of correct predictions made by the model across all data.

- **Recall**: Measures the model's ability to identify all relevant instances of a specific class.
- **F1-Score**: A harmonic mean of precision and recall, providing a balance between the two.

These evaluation metrics are presented in Equations 4.1, 4.2, 4.3, and 4.4, respectively.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.3}$$

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN} \tag{4.4}$$

The table Table 4.4 shows a summary of the LSTM model's performance in terms of training and testing accuracy, as well as performance metrics for each class.

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1-Score**
LSTM	0.7495	0.74	0.75	0.74

Table 4.4: Results of the LSTM classifier on the test set

The results obtained, with an accuracy of around 70%, can be largely explained by the imbalanced distribution of sentiments in the dataset. Specifically, the dataset includes 2,692 comments expressing positive sentiments, 2,048 neutral comments, and only 608 negative ones. This significant imbalance—particularly the underrepresentation of negative comments—has affected the model's learning process, making it less effective at detecting negative sentiments. Therefore, the model's performance could be significantly improved by addressing the imbalance among the sentiment classes in the training data.

General Conclusion

In today's digital age, online learning platforms have become a fundamental component of modern education. As students increasingly turn to these environments for accessible and flexible learning, understanding their emotional engagement has emerged as a key factor in maintaining motivation and reducing dropout rates.

Sentiment analysis plays a key role in online learning environments, as it helps to better understand learners' feelings, identify their needs, and improve both instructional content and teaching methods accordingly. By giving learners a voice through the analysis of their comments, it becomes possible to adapt educational strategies to enhance their engagement and success.

In this context, this work proposed an automatic sentiment analysis approach based on an LSTM model, capable of effectively processing textual sequences. Following a preprocessing and data annotation phase, the model was trained on a real dataset collected from the Mark My Professor platform. The results obtained are promising, despite certain limitations, particularly those related to the imbalance of sentiment classes within the data. This observation opens up opportunities for improvement, notably through resampling techniques or the integration of more balanced datasets.

Thus, this study highlights the relevance of artificial intelligence techniques—particularly recurrent neural networks—for automatically extracting valuable insights into learners' experiences in digital learning environments.

Based on the results achieved, it can be concluded that the main objectives of this research have been successfully achieved.

Bibliography

Aljedaani, W., Rustam, F., Mkaouer, M. W., Ghallab, A., Rupapara, V., Washington, P. B., Lee, E., and Ashraf, I. (2022). Sentiment analysis on twitter data integrating textblob and deep learning models: The case of us airline industry. *Knowledge-Based Systems*, 255:109780.

Altrabsheh, N., Cocea, M., and Fallahkhair, S. (2014). Sentiment analysis: Towards a tool for analysing real-time students feedback. In *2014 IEEE International Conference on Tools with Artificial Intelligence*, pages 419–423.

Aung, K. Z. and Myo, N. N. (2017). Sentiment analysis of students' comment using lexicon-based approach. In *2017 16th IEEE/ACIS International Conference on Computer and Information Science (ICIS)*, pages 149–154.

Balaji, P., Nagaraju, O., and Haritha, D. (2017). Levels of sentiment analysis and its challenges: A literature review. In *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*, pages 436–439. IEEE.

Behera, R. K., Jena, M., Rath, S. K., and Misra, S. (2021). Co-lstm: Convolutional lstm model for sentiment analysis in social big data. *Information Processing Management*, 58(1):102435.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Bhagat, B. P., Dhande-Dandge, S. S., and Raut, S. R. (2024). Sentiment analysis of student's comments using long short-term memory with multi-head attention. *IAES International Journal of Artificial Intelligence*, 13(4):4747–4756.

Bird, S., Loper, E., and Klein, E. (2006). Nltk: The natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72, Sydney, Australia. Association for Computational Linguistics.

Bond, M., Buntins, K., Bedenlier, S., Zawacki-Richter, O., and Kerres, M. (2020). Mapping research in student engagement and educational technology in higher education: A systematic evidence map. *International Journal of Educational Technology in Higher Education*, 17(1):1–30.

Bouacida, I. (2018). Sentiment analysis and opinion mining techniques for learning analytics. Master's thesis, Eötvös Loránd University, Budapest, Hungary.

Cambria, E., Schuller, B., Xia, Y., and Havasi, C. (2017). New avenues in opinion mining and sentiment analysis. *IEEE Intelligent Systems*, 28(2):15–21.

Capuano, N., Caballé, S., Conesa, J., and Greco, A. (2021). Attention-based hierarchical recurrent neural networks for mooc forum posts analysis. *Journal of Ambient Intelligence and Humanized Computing*, 12(11):9977–9989.

Cherif, F. (2022). Exploration des approches d'analyse des sentiments à base d'aspects ciblés. Unpublished work or institutional report.

D'Mello, S. and Graesser, A. (2012). Autotutor and affective learning. In *International Handbook of Emotions in Education*, pages 131–144. Routledge.

Gao, C., Yan, J., Zhou, S., Varshney, P. K., and Liu, H. (2019). Long short-term memory-based deep recurrent neural networks for target tracking. *Information Sciences*, 502:279–296.

Grattarola, D. and Alippi, C. (2021). Graph neural networks in tensorflow and keras with spektral [application notes]. *IEEE Computational Intelligence Magazine*, 16(1):99–106.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Houdt, G. V., Mosquera, C., and Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, 53(8):5929–5955.

Illia, F., Eugenia, M. P., and Rutba, S. A. (2021). Sentiment analysis on pedulilindungi application using textblob and vader library. In *Proceedings of the International Conference on Data Science and Official Statistics*, volume 2021, pages 278–288, Indonesia. BPS-Statistics Indonesia.

Khan, L., Amjad, A., Afaq, K. M., and Chang, H.-T. (2022). Deep sentiment analysis using cnn-lstm architecture of english and roman urdu text shared in social media. *Applied Sciences*, 12(5):2694.

Lalata, J. A. P., Gerardo, B., and Medina, R. (2019). A sentiment analysis model for faculty comment evaluation using ensemble machine learning algorithms. In *ACM International Conference Proceeding Series*, pages 68–73.

Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Liu, B. (2020). Text sentiment analysis based on cbow model and deep learning in big data environment. *Journal of Ambient Intelligence and Humanized Computing*, 11(2):451–458.

Mabunda, J. G. K., Jadhav, A., and Ajoodha, R. (2021). Sentiment analysis of student textual feedback to improve teaching. In *Interdisciplinary Research in Technology and Management*, pages 643–651. CRC Press.

Mahadevaswamy, U. B. and Swathi, P. (2023). Sentiment analysis using bidirectional lstm network. *Procedia Computer Science*, 218:45–56.

McKinney, W. (2011). pandas: a foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, 14(9):1–9.

Medhat, W., Hassan, A., and Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4):1093–1113.

Moukodouma, D. F. B., Denis, C., Mbourou, D. R. R., and Nkoulembene, C. A. (2024). Comparaison des modèles lstm et transformée en ondelettes-lstm pour la prédiction de la température dans une partie du bassin du congo. *International Journal of Innovation and Applied Studies*, 44(1):77–94.

Murthy, G. S. N., Allu, S. R., Andhavarapu, B., Bagadi, M., and Belusonti, M. (2020). Text based sentiment analysis using lstm. *International Journal of Engineering Research and Technology*, 9(05):299–303.

Naik, P. G. (2023). *Conceptualizing Python in Google Colab: Hands-on Practical Sessions*. Shashwat Publication, India.

Nguyen, V. D., Van Nguyen, K., and Nguyen, N. L. T. (2018). Variants of long short-term memory for sentiment analysis on vietnamese students' feedback corpus. In *2018 10th International Conference on Knowledge and Systems Engineering (KSE)*, pages 306–311. IEEE.

Ocumpaugh, J., Baker, R. S., and Rodrigo, M. M. T. (2015). *Baker Rodrigo Ocumpaugh Monitoring Protocol (BROMP) 2.0 Technical and Training Manual*.

Oliphant, T. E. (2006). *Guide to NumPy*, volume 1. Trelgol Publishing, USA.

Onan, A. (2021). Sentiment analysis on massive open online course evaluations: a text mining and deep learning approach. *Computer Applications in Engineering Education*, 29(3):572–589.

Pacol, C. A. and Palaoag, T. D. (2021). Enhancing sentiment analysis of textual feedback in the student-faculty evaluation using machine learning techniques. *European Journal of Engineering Science and Technology*, 4(1):27–34.

Pölsterl, S. (2020). scikit-survival: A library for time-to-event analysis built on top of scikit-learn. *Journal of Machine Learning Research*, 21(212):1–6.

Qixuan, Y. (2024). Three-class text sentiment analysis based on lstm. *arXiv preprint arXiv:2412.17347*.

Rana, T. A. and Cheah, Y.-N. (2016). A two-tier ensemble framework for sentiment analysis of social media texts. *Journal of Information Science*, 42(6):783–797.

Rani, S. and Kumar, P. (2017). A sentiment analysis system to improve teaching and learning. *Computer*, 50(5):36–43.

Sangeetha, K. and Prabha, D. (2021). Sentiment analysis of student feedback using multi-head attention fusion model of word and context embedding for lstm. *Journal of Ambient Intelligence and Humanized Computing*, 12(3):4117–4126.

Schmidt, R. M. (2019). Recurrent neural networks (rnns): A gentle introduction and overview. *arXiv preprint arXiv:1912.05911*.

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306.

Sindhu, I., Daudpota, S. M., Badar, K., Bakhtyar, M., Baber, J., and Nurunnabi, M. (2019). Aspect-based opinion mining on student's feedback for faculty teaching performance evaluation. *IEEE Access*, 7:108729–108741.

Singh, N. K., Tomar, D. S., and Sangaiah, A. K. (2020). Sentiment analysis: a review and comparative analysis over social media. *Journal of Ambient Intelligence and Humanized Computing*, 11(1):97–117.

Tosi, S. (2009). *Matplotlib for Python Developers*. Packt Publishing Ltd., Birmingham, United Kingdom.

Wang, Z., Ho, S.-B., and Cambria, E. (2020). Multi-level fine-scaled sentiment sensing with ambivalence handling. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 28(4):683–697.

Yu, Y., Si, X., Hu, C., and Zhang, J. (2019). A review of recurrent neural networks: Lstm cells and network architectures. *Neural Computation*, 31(7):1235–1270.

Zhang, L., Wang, S., and Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253.

Zhou, J. and Ye, J. (2023). Sentiment analysis in education research: A review of journal publications. *Interactive Learning Environments*, 31(3):1252–1264.