DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA
UNIVERSITY OF SAAD DAHLEB BLIDA 1
Faculty of Science Department of Computer Science

Thesis

Presented for obtaining the Master's degree
In computer science
Option: Intelligent Systems Enginnering

Realised by:

Bouziane Abdelillah and Ladjenef Mohamed

Sentiment analysis in Arabic dialect

Supervisor : Dr.Bouziane Abdelghani

Examinator: Dr.Bouras Dalila
President: Dr.Tebbi Hanane

2024/2025

Acknowledgements

First and foremost, all praise is due to Allah, the Lord of the universe, who granted us patience and strength throughout these long years and blessed us with success in completing this work.

We would also like to express our sincere gratitude and appreciation to our supervisor, Mr. Bouziane Abdelghani, for his guidance throughout the entire project, and for all the valuable advice, support, and resources that greatly contributed to the realization of this work.

We extend our heartfelt thanks to all the professors at SAAD DAHLEB University, especially those in the Computer Science Department, who have been part of our academic journey from the beginning. We are particularly grateful to Mrs. Boumahdi, who is in charge of the AI specialization in the Master's program, for her dedication and commitment to our academic and professional development.

We extend our heartfelt thanks to all the professors at SAAD DAHLEB University, especially those in the Computer Science Department, who have been part of our academic journey from the beginning. We are particularly grateful to Mrs. Boumahdi, who is in charge of the AI specialization in the Master's program, for her dedication and commitment to our academic and professional development.

We thank all the staff and faculty members without exception. We also wish to express our appreciation to everyone who contributed, directly or indirectly, to the completion of this work.

Finally, we are deeply thankful to our beloved parents for their unconditional love, sacrifices, and constant support. Our thanks also go to our brothers, sisters, and friends for their emotional and financial encouragement throughout the years of study, as well as to all those whose names are not mentioned here but who supported us along the way.

Thank you all.

Dedication

To my beloved mother, Fatiha,

whose endless love, prayers, and strength have been the light guiding me through every challenge. Your sacrifices and tenderness are the roots of every success I achieve.

To my sweet sisters asmaa and wissam ,

To my dear father, Mokhtar,

whose wisdom, hard work, and unwavering support have been my foundation. Thank you for teaching me the value of patience and perseverance.

To my closest friends,

Mohamed Ayoune and Ismail Mahmoud,

thank you for your constant encouragement, your presence through the ups and downs, and the countless memories that made this journey bearable and unforgettable.

To my binôme and teammate,

Ladjenef Mohamed,

thank you for your dedication, your partnership, and all the hard work we shared together to bring this project to life.

This work is dedicated to you all—with love, respect, and deep gratitude.

Bouzian Abdelillah

To my dear parents — may God prolong their lives;

To my beloved sister Nadia;

To my friend and teammate Abdo;

To all my friends and colleagues;

To every seeker of knowledge;

To everyone who has known me, whether closely or from afar;

To all of you, I dedicate this humble work.

Ladjenef Mohamed

Abstract

Sentiment analysis is a key area of natural language processing that aims to automatically identify and interpret opinions and emotions expressed in text. With the growing volume of user-generated content on social media, analyzing sentiment has become increasingly valuable for researchers and organizations.

This study focuses on sentiment analysis in Arabic, with a particular emphasis on dialectal variations. Due to the complexity of the Arabic language—its morphology, diverse dialects, and lack of annotated resources—this task presents unique challenges. The work explores various text representation techniques and evaluates a range of machine learning and deep learning models to determine suitable approaches for Arabic sentiment classification.

Through systematic experimentation, this research highlights the impact of text encoding methods and model choice on sentiment classification performance in Arabic, offering insights for future studies in this under-resourced linguistic domain.

Keywords:

Sentiment Analysis, Machine learning, Deep learning,.

Résumé

L'analyse des sentiments est un domaine essentiel du traitement automatique du langage naturel, visant à identifier et interpréter automatiquement les opinions et les émotions exprimées dans les textes. Avec la croissance du contenu généré par les utilisateurs sur les réseaux sociaux, cette analyse devient de plus en plus précieuse pour les chercheurs et les organisations.

Ce travail porte sur l'analyse des sentiments en arabe, en mettant l'accent sur les particularités des dialectes. En raison de la richesse morphologique de la langue arabe, de la diversité de ses dialectes et du manque de ressources annotées, cette tâche présente plusieurs défis. Cette étude explore différentes méthodes de représentation des textes et évalue plusieurs modèles d'apprentissage automatique et profond pour la classification des sentiments.

Les expérimentations menées permettent de mieux comprendre l'influence des techniques de représentation et du choix des modèles sur la performance de l'analyse des sentiments en arabe.

Mots-clés :

Analyse des sentiments, Apprentissage automatique, Apprentissage profond.

الملخص

تحليل المشاعر هو أحد المجالات الأساسية في معالجة اللغة الطبيعية، ويهدف إلى التعرف على الآراء والمشاعر الواردة في النصوص وتحليلها بشكل آلي. ومع ازدياد حجم المحتوى الذي ينشئه المستخدمون على وسائل التواصل الاجتماعي، أصبحت هذه التحليلات أكثر أهمية للباحثين والمؤسسات.

يركز هذا البحث على تحليل المشاعر باللغة العربية مع اهتمام خاص باللهجات المختلفة. وتُواجه هذه المهمة تحديات متعددة بسبب البنية الصرفية الغنية للغة العربية، وتنوع لهجاتها، ونقص الموارد الموصوفة. تستعرض هذه الدراسة تقنيات متنوعة لتمثيل النصوص وتقيّم نماذج متعددة من التعلم الآلي والتعلم العميق بهدف تصنيف المشاعر.

تُساهم هذه الدراسة في توضيح تأثير طرق التمثيل واختيار النماذج على أداء تصنيف المشاعر في النصوص العربية.

الكلمات المفتاحية:

معالجة اللغة الطبيعية، تحليل المشاعر، التعلم الآلي، التعلم العميق.

Contents

Contents

List of Figures:

List of Tables:

List of Acronyms

Acronym	Meaning
TF-IDF	(Term Frequency-Inverse Document Frequency
BoW	Bag of Words
W2V	Word2Vector
SVM	Support Vector Machines
DT	Decision Tree
NLP	Natural Language Processing
DNN	Deep Neural Networks
ML	Machine Learning
DL	Deep Learning
B-LSTM	Bidirectional-Long Short-Term Memory
LSTM	Long Short-Term Memory
SA	Sentiment Analysis

General Introduction

- **Context and Motivation**

Sentiment analysis, also known as opinion mining, is a rapidly growing field within natural language processing (NLP) that focuses on identifying and extracting subjective information from textual data. With the exponential increase in user-generated content on social media, forums, and review platforms, understanding public sentiment has become crucial for businesses, governments, and researchers.

- **Problem Statement**

Despite significant progress, accurately analyzing sentiment in text—especially in languages with limited resources such as Arabic—remains a challenging task. The complexity of natural language, the presence of sarcasm, ambiguity, and the diversity of linguistic expressions contribute to these challenges.

- **Importance of the Study**

The ability to automatically analyze and interpret sentiment from large volumes of text data provides valuable insights for decision-making, brand monitoring, and public opinion analysis. Improving sentiment analysis techniques, particularly for underrepresented languages, can have a significant impact on various domains.

- **Objectives**

The objectives of this work are to review the fundamental concepts and techniques in sentiment analysis, explore and compare different text representation methods (TF-IDF, Bag of Words, Word2Vec), evaluate the performance of various machine learning and deep learning models for sentiment classification, and highlight the challenges and propose solutions for sentiment analysis in Arabic text.

- **Scope and Focus**

This work focuses on the application of sentiment analysis to Arabic text, using both traditional and modern text representation techniques. The study aims to assess the effectiveness of different embeddings and classifiers, particularly in the context of Arabic dialects.

Thesis Organization

To achieve our stated objectives, we have organized our thesis as follows:

- **Chapter 1: State of the Art**

  This chapter surveys the main concepts, challenges, and techniques in sentiment analysis, with a particular focus on the Arabic language and its dialects. It highlights the unique linguistic issues, reviews key text representation and modeling approaches, and summarizes important related work in the field.

- **Chapter 2: Methodology and Experimental Framework**

  This chapter details the methodology for Arabic sentiment analysis, including data collection, preprocessing, feature extraction, and model design.

- **Chapter 3: Implementation and Results**

  Our primary objective is to propose a sentiment analysis model for Arabic dialect. In this chapter, we focus on the implementation of our model. We begin by presenting the programming language, development environment, and tools used. Next, we provide a detailed overview of the various stages of our work, from the choices of the hyperparameters to the evaluation of the models used. We conclude this section by discussing the results and determining the most effective model.

Chapter 1: State of the Art

Chapter 1: State of the Art

Introduction:

This chapter surveys the main concepts, challenges, and techniques in sentiment analysis, with a particular focus on the Arabic language and its dialects. It highlights the unique linguistic issues, reviews key text representation and modeling approaches, and summarizes important related work in the field.

1 Sentiment Analysis:

Is a subfield of Natural Language Processing (NLP) concerned with identifying, extracting, and classifying opinions or emotions expressed in textual data. It plays a crucial role in understanding public attitudes toward products, services, events, or social issues, particularly in domains such as marketing, politics, and healthcare. Sentiment analysis can be conducted at various levels—document, sentence, or aspect—and involves techniques ranging from rule-based approaches to advanced machine and deep learning models. The rise of online platforms and user-generated content has made sentiment analysis an essential tool for analyzing large-scale text data in real time [1].

2 Sentiment Analysis in Arabic and Dialects:

We are going to talk about sentiment analysis in Arabic, applications and challenges:

2.1 Arabic language

Arabic is one of the most popular languages that are spoken by millions of people all around the world, the Arabic language consists of two forms**:** [2]

- **Modern Standard Arabic**

Modern Standard Arabic (MSA) is the standard form of the Arabic language that is used in formal papers, schoolbooks, education, TV news, newspapers, street signs, etc. Modern Standard Arabic has a written standard format while the dialects are not. As mentioned earlier, the Arabic language attracted researchers due to the increased use of this language over the internet. Most of the researches that targeted the Arabic language focused on the Modern Standard Arabic form of the Arabic language;

- **Arabic Dialect**

Arab World consists of 22 countries. Each of these Arab countries has a special Arabic dialect that their population used for daily conversations and talk.

2.2 Applications

The applications of sentiment analysis:

Domain application	Large language models
- Business review analysis - Financial prediction - Social text analysis - Government intelligence - Healthcare domain - Recommendation system	- ChatGPT - ERNIE Box - PaLM

Table 1: application of sentiment analysis. [3]

2.3 Challenges

The challenges of sentiment analysis in the Arabic language are: [4]

- **Morphological analysis:**

Morphological analysis is an important phase in Sentiment analysis. Its main purpose is to decompose words into morphemes and to associate each morpheme with a morphological information such as stem, root, POS (Part Of Speech), and affix; This complexity requires the development of appropriate systems that are able to deal with tokenization, spell checking, stemming, lemmatization, pattern matching, and part-of-speech tagging;

- **Dialectal Arabic:**

For communication purposes, Arabic speakers usually use colloquial Arabic rather than MSA. There are around 30 major Arabic dialects that differ from MSA and from each other phonologically, morphologically, and lexically. Furthermore, Arabic dialects have no standard orthographies and no language academies. Therefore, using tools and resources designed for MSA to process Arabic dialects generates considerably low performance. Recently, researchers have started developing parsers for specific dialects such as CALIM;

- **Arabizi:**

Arabizi, Arabish, or Romanized Arabic refers to a system of writing Arabic using Latin characters. It is widely used to write MSA as well as Arabic dialects in social media platforms. Dealing with this form of writing has been the subject solely of studies that aim at detecting and converting Arabizi into Arabic;

- **Named entity recognition:**

In Arabic, large portions of Arabic names are associated with positive adjectives. For example, the first name ‘‘” corresponds سعيدto the adjective ‘‘” which means ‘‘happy”. In addition, Arabic سعيدproper nouns are not capitalized as in Latin languages, a fact which complicates the identification of named entities. For this reason, a system of Named Entity Recognition is crucial in analyzing Arabic texts and distinguishing between entity names and sentiment words.

3 Feature Extraction Techniques:

In this section, we review core techniques used in sentiment analysis and demonstrate how these techniques are integrated to form a complete analytical pipeline.

3.1 Overview of Feature Extraction Techniques:

Word embedding techniques transform text into numerical vectors that can be used by machine learning models. Common approaches include: [5]

TF-IDF (Term Frequency-Inverse Document Frequency): Measures the importance of a word in a document relative to a corpus, making it highly effective for feature extraction;

Bag of Words (BoW): Converts text into fixed-length vectors based on word frequency, providing a simple yet robust representation;

Word2Vec: Captures semantic meanings by converting words into dense vector representations, thus enhancing sentiment analysis tasks

Machine Learning Algorithms in Sentiment Analysis:

Several machine learning algorithms are commonly used for sentiment classification: [6]

Support Vector Machines (SVM): Particularly powerful in high-dimensional spaces, capable of modeling complex class boundaries with kernel tricks;

Decision Trees (DT) and Random Forests (RF): Known for their interpretability and ability to capture non-linear relationships in data;

XGBoost: A powerful gradient boosting algorithm that often achieves state-of-the-art results.

Chapter 2: Methodology and Experimental Framework

3.2 Deep Learning for NLP:

Deep learning models have advanced sentiment analysis by capturing complex patterns in text: [7]

- **Deep Neural Networks (DNN):** Multi-layer perceptrons that can model non-linear relationships.
- **Bidirectional Long Short-Term Memory (B-LSTM):** Recurrent neural networks that capture dependencies in both directions, useful for sequential data like text.

4 Related Work:

4.1 Introduction

Sentiment analysis can be conducted at different levels of granularity, each with its own challenges and applications. At the **document level**, the goal is to determine the overall sentiment expressed in a complete piece of text, such as a review or an article. This level offers a broad perspective but may miss subtle sentiment variations within different parts of the document. **Aspect-level sentiment analysis** goes deeper by identifying sentiments associated with specific features or components of a product, service, or topic—for instance, distinguishing between positive opinions about a product's price and negative opinions about its battery life. Lastly, **sentence-level (or comment-level) analysis** focuses on identifying the sentiment of individual sentences or short user-generated texts, such as tweets or comments. This level is particularly relevant in social media contexts, where texts are short, informal, and often written in dialects.

To place our research in context, we present in **Table 4** a selection of related studies from each of these levels. These works are selected based on their relevance to our methodology, particularly in terms of feature extraction techniques (e.g., TF-IDF, Bag of Words, Word2Vec, transformer embeddings) and models (e.g., SVM, CNN, RNN, Bi-LSTM). In addition, some of the selected studies target Arabic or Algerian dialects, which are directly aligned with the focus of our thesis.

Chapter 2: Methodology and Experimental Framework

Task Level	Reference & Year	Language / Dialect	Feature Extraction	Models & Techniques	Dataset & Focus	Results & Conclusions
Document-level	Pang & Lee (2002) [8]	English	TF-IDF, BoW	SVM	Movie reviews (binary classification)	Achieved ~82% accuracy with SVM; showed TF-IDF + SVM outperforms baselines.
Document-level	Nassar & Sezer (2022) [9]	Arabic	TF-IDF	Artificial Neural Network (ANN)	Generic Arabic texts; document-level sentiment	ANN achieved **94% F1-score**; showed TF-IDF combined with neural models is effective.
Aspect-level	Hu & Liu (2004) [10]	English	TF-IDF	Rule-based, phrase-level classification	Customer reviews; aspect-level sentiment	Identified sentiment toward product features; foundational for fine-grained analysis.
Sentiment-level	Bouziane et al. (2024) [11]	Algerian Dialect	Word2Vec, BoW	Bi-LSTM	Algerian social media posts	Achieved **0.94 accuracy**, **0.93 precision**, **0.91 recall**, **0.92 F1-score**; demonstrated deep learning's effectiveness on dialectal Arabic.
Sentiment-level	Mazari & Djeffal (2021) [12]	Algerian Dialect	TF-IDF, Word2Vec	SVM, Naïve Bayes, CNN, RNN	11,760 comments in Arabic/Arabizi	Word2Vec + CNN outperformed others; showed embeddings improve sentiment classification.
Sentiment-level	Benmounah et al. (2023) [13]	Algerian Dialect (Aradziri)	Transformer embeddings (DziriBERT)	Fine-tuned Transformer (DziriBERT)	YouTube comments in Algerian dialect	Achieved **>90% accuracy**; validated transformer models for sentiment in low-resource dialects.

Table 2:Related Work Overview . [4]

4.2 Discussion

To contextualize our research within existing literature, **Table 4** presents a selection of studies that align closely with the scope and methodology of this work. These studies are categorized according to the level of sentiment analysis—**document-level**, **aspect-level**, and **sentence-level** (or comment-level)—and involve languages ranging from English to Modern Standard Arabic and Algerian dialects.

At the document level, Pang and Lee [8] demonstrated the effectiveness of TF-IDF and SVM for binary sentiment classification in English movie reviews. In the context of Arabic, Nassar and Sezer [9] applied TF-IDF with an Artificial Neural Network (ANN), achieving high F1-scores for document-level sentiment classification. For aspect-level sentiment analysis, the foundational work by Hu and Liu [10] extracted opinions related to product features using rule-based methods.

At the sentence level, several studies focus specifically on Arabic dialects. Bouziane et al. [11] used Bi-LSTM with Word2Vec on Algerian dialect social media posts and achieved high performance across all evaluation metrics. Mazari and Djeffal [12] also targeted Algerian sentiment using a combination of Word2Vec and various classifiers, highlighting the benefit of embeddings in dialectal contexts. Finally, Benmounah et al. [13] explored transformer-based embeddings with a fine-tuned DziriBERT model for Algerian dialect sentiment classification, reaching over 90% accuracy and confirming the strong potential of large language models in this domain.

The results from these studies support the idea that combining appropriate feature extraction methods with deep learning architectures significantly enhances sentiment classification—particularly in morphologically rich and under-resourced languages like Arabic. This motivates our own methodological choices and validates the importance of our contribution to Arabic sentiment analysis research.

Chapter 2: Methodology and Experimental Framework

5 Conclusion

In this chapter, we presented the fundamental concepts and background necessary to understand sentiment analysis. We began by introducing sentiment analysis as a field within Natural Language Processing, outlining its objectives and growing significance across various domains such as business, politics, and social media.

We then focused specifically on sentiment analysis in the Arabic language and its dialects, highlighting the unique linguistic challenges posed by Arabic morphology, dialectal diversity, and the lack of standardized resources. These factors significantly affect the effectiveness of traditional text processing and classification techniques when applied to Arabic content, especially in informal, user-generated contexts.

Next, we reviewed key **feature extraction techniques** such as TF-IDF, Bag of Words, and Word2Vec, and explored their roles in transforming textual data into machine-readable formats. These methods provide the foundation for the models we later evaluated. We also examined machine learning and deep learning models—ranging from classical algorithms like SVM and Decision Trees to more advanced neural architectures like DNN and Bi-LSTM—which are widely used for sentiment classification tasks.

Finally, we provided a comparative overview of existing works at the **document**, **aspect**, and **sentence levels** of sentiment analysis. Through a structured summary table, we analyzed studies that share methodological similarities with our own research, including works focused on Arabic and Algerian dialect sentiment classification. These studies confirm the growing importance of using contextual embeddings and deep models in under-resourced linguistic environments.

This foundation sets the stage for the following chapter, where we detail our experimental methodology—including dataset description, preprocessing steps, feature extraction choices, and model design—tailored specifically to Arabic dialect sentiment analysis.

Chapter 2: Methodology and Experimental Framework

Chapter 2: Methodology and Experimental Framework

1 Introduction:

This chapter details the methodology for Arabic sentiment analysis, including data collection, preprocessing, feature extraction, and model design.

2 Dataset description:

The primary dataset used in this study is **`araCovid_sentiment.tsv`**, which contains 4,128 Arabic text samples: [14]

- A unique index column (`Unnamed: 0`)
- The comment text (`Text`), which is unique for each entry
- The sentiment label (`sentiment`), with three possible classes: Positive, Negative, and Neutral.

3 Basic Data Analysis Workflow:

- Loading the data
- Inspecting the data
- Visualizing distributions.

3.1 Loading the data:

To begin our analysis, we loaded the dataset using the pandas library. The data file, `araCovid_sentiment.tsv`, is a tab-separated values (TSV) file encoded in UTF-8.

Chapter 2: Methodology and Experimental Framework

3.2 Inspecting the Data:

Inspecting the data:

	Unnamed: 0	Text	sentiment
0	0	علاج ڤايروس الكورونا* كل يوم الصبح الواحد ي*...	Positive
1	1	هههههه هذا مكان اخطر من كورونا فايروس. 😂😂🔥 .	Positive
2	2	خايفه يجيبوا لقاح الكورونا وما يكفينا ويقوموا ...	Positive
3	3	الطواقي وشخبار الطرد لحل يطلعون اشاعات هههههه...	Positive
4	4	@Waleed_Albashaa هههههههه انتظر بعد ما تخلص كو...	Positive

Figure 1:The dataset.

- **Shape:** The dataset consists of 4,128 rows and 3 columns, representing 4,128 unique comments.

- **Columns:** The dataset columns are:

  - Unnamed: 0: An integer index column, likely generated during file saving/loading.
  - Text: The comment text, with all entries unique.
  - Sentiment: The sentiment label for each comment.

- **Data Types:**

  - Unnamed: 0: Integer (row index).
  - Text and sentiment: Object (string).

- **Missing Values:** No missing values are present in any column.

- **Text Column:** All comments are unique, indicating no duplicate comments.

- **Sentiment Column:** Contains 3 unique sentiment classes. The most frequent class is Neutral (1,600 out of 4,128, about 39%).

- **Index Column:** The Unnamed: 0 column ranges from 0 to 4,127, confirming it serves as a row index.

Conclusion:

The dataset is clean, with no missing values or duplicate comments. There is a class imbalance, with Neutral being the most common sentiment. The data is ready for further analysis.

3.3 Visualizing distributions:

This section provides visual insights into the dataset, highlighting sentiment distribution, comment length, and text characteristics. These analyses help identify patterns and potential challenges for model development.

3.3.1 Analysis of the "Sentiment Distribution" Graph:

The bar chart shows the distribution of sentiment labels in the dataset:

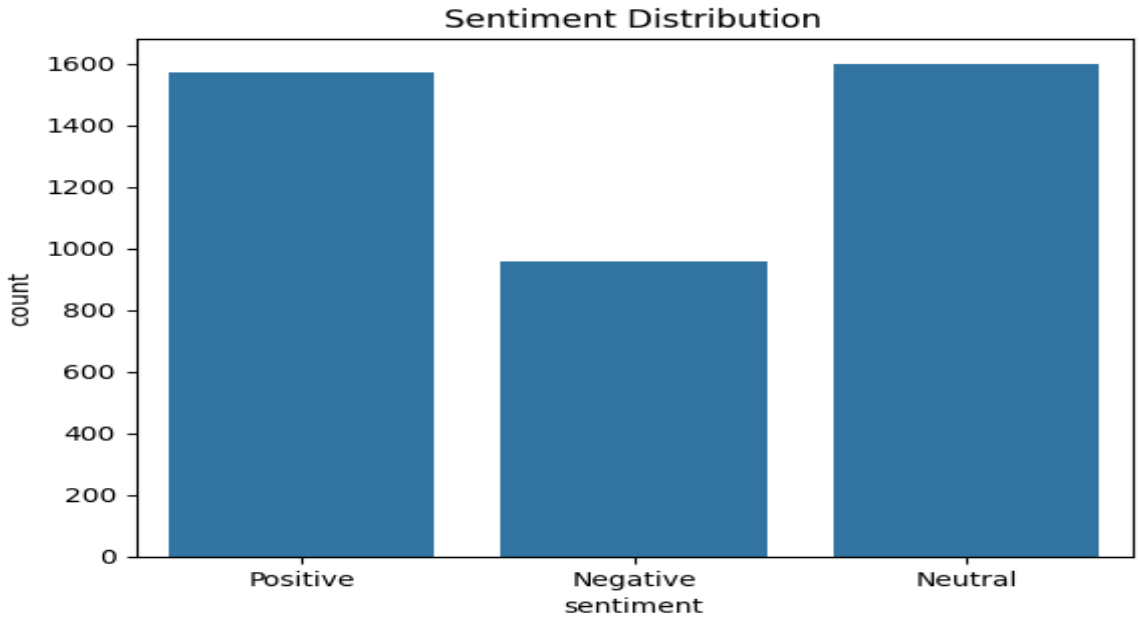


Figure 2:the distribution of sentiment labels in the dataset

Neutral and Positive sentiments are almost equally represented, each with about 1,600 comments. Negative sentiment is less frequent, with fewer than 1,000 comments. This indicates a class imbalance: Neutral and Positive are overrepresented compared to Negative. The dataset overall contains a good number of samples for each class.

3.3.2 Analysis of the "Length of the Comments" Graph:

The bar chart shows the distribution of Length of the Comments:

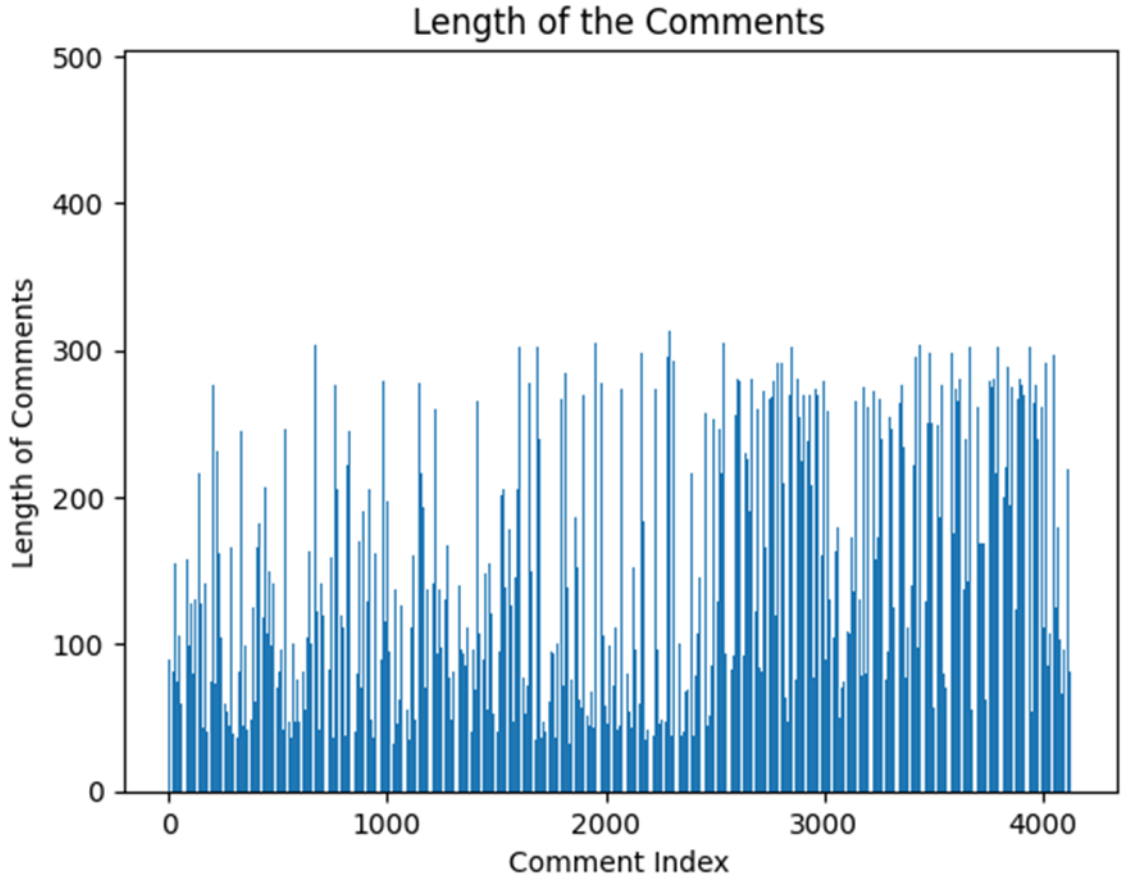


Figure 3:the distribution of Length of the Comments.

The "Length of the Comments" graph illustrates the distribution of comment lengths in the dataset, measured in characters. The chart reveals considerable variability, with comment lengths ranging from very short (under 50 characters) to quite long (over 300 characters). Most comments cluster below 200 characters, while occasional spikes indicate the presence of unusually long comments. The average comment length is approximately 150 characters.

3.3.3 Analysis of the Text Length by Sentiment Boxplot:

The boxplot visualizes the distribution of text lengths for each sentiment class (Positive, Negative, Neutral):

Chapter 2: Methodology and Experimental Framework

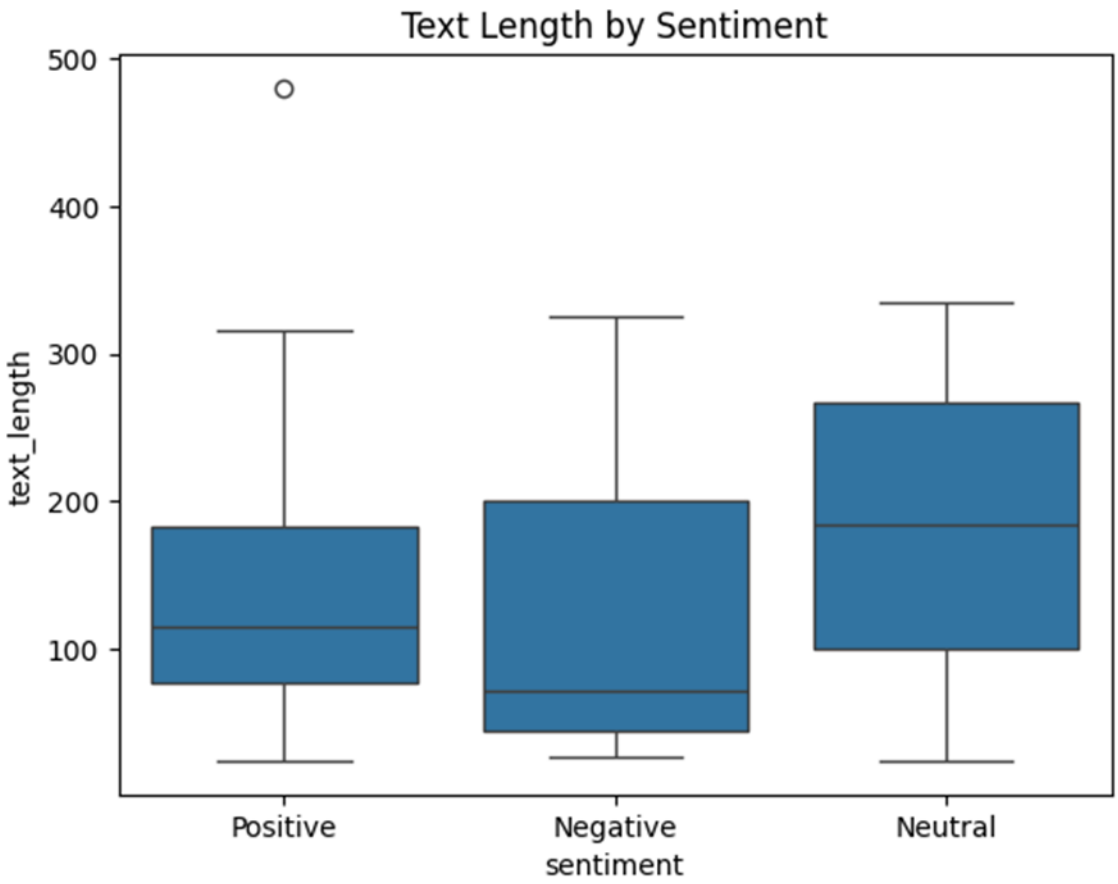
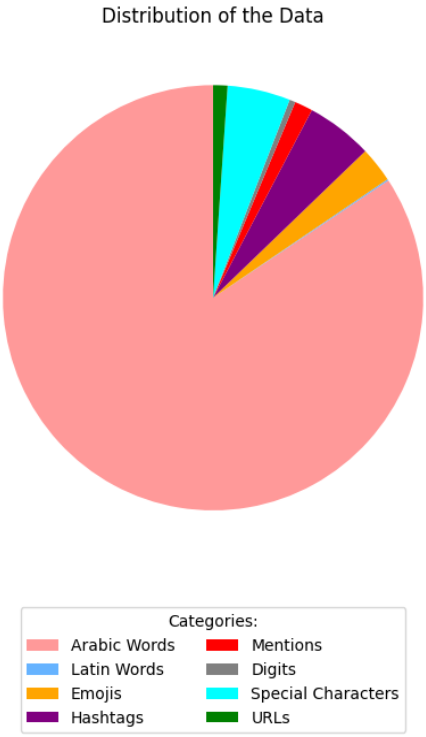


Figure 4:the distribution of text lengths for each sentiment class

The boxplot shows that the length of comments varies noticeably by sentiment class. Neutral comments are typically the longest, with both a higher median and greater variability in length, suggesting that neutral statements tend to be more detailed or descriptive. Negative comments have a moderate median length and a wide range, while positive comments are generally the shortest and most consistently sized. The presence of outliers, particularly among positive comments, indicates that a few comments in this class are unusually long. Overall, the distribution implies that users express neutral sentiments with more elaboration, whereas positive and negative sentiments are frequently expressed more concisely.

3.3.4 Analysis of Comment Content Distribution:

Analysis of Comment Content Distribution:



Element Type	Percentage (%)
Arabic Words	85.57%
Latin Words	6.39%
Emojis	0.58%
Hashtags	1.01%
Mentions (@user)	0.27%
Digits	1.17%
Special Characters	5.01%
URLs	1.10%

Figure 5:Analyses of Comment Content Distribution

The dataset is primarily composed of Arabic words, with smaller proportions of Latin words, special characters, and social media elements like hashtags, mentions, and emojis. There are 20,510 unique words in the dataset, reflecting a rich and diverse vocabulary. This highlights the need for robust Arabic NLP preprocessing.

Chapter 2: Methodology and Experimental Framework

4 Data Preprocessing:

Data preprocessing consists of:

4.1 Data cleaning:

Before performing sentiment analysis, it is essential to clean the raw text data to improve the quality and reliability of the results. The data cleaning process in this study is structured as follows:

- **Pseudocode for Data Cleaning Process**
  - **Input:** Raw dataset containing Arabic text comments.
  - **Process:** For each comment in the dataset:
    - **Remove URLs:**
      - Remove hashtags (words starting with #).
      - Remove user mentions (words starting with @).
      - Remove emojis and special characters.
      - Remove numbers with three or more consecutive digits.
      - Remove extra spaces and trim leading/trailing spaces.
      - Remove non-Arabic words, keeping only Arabic characters.
      - *Remove stop words (Arabic) using NLTK.*
    - **Remove any rows where the cleaned text is empty.**
  - **Output:** Cleaned dataset with only meaningful Arabic text, ready for further analysis.

Chapter 2: Methodology and Experimental Framework

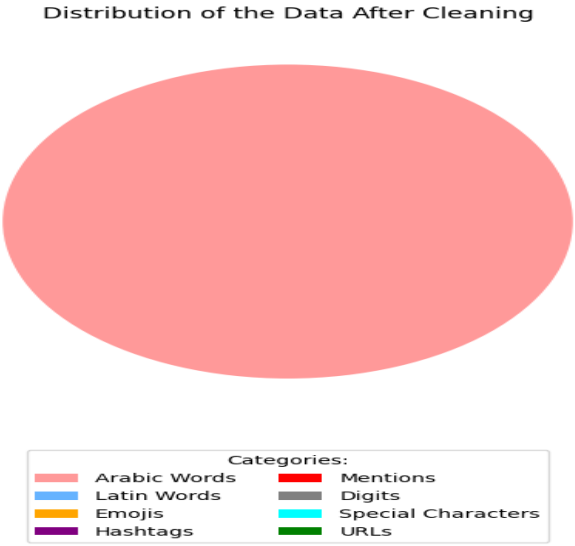


Figure 6:Distribution of data after cleaning

After cleaning the dataset, the vocabulary size has been reduced, and we now have 12,665 unique words. This reflects the effectiveness of the cleaning process in removing noise and irrelevant content, ensuring that the analysis focuses on meaningful Arabic text.

Below are some examples of the data before and after cleaning:

Column	Original Text	Cleaned Text
0	علاج ڤايروس الكورونا كل يوم الصبح الواحد يقرط ثلات حبات ثوم، ومع إنو الثوم ما إله علاقة بڤايروس الكورونا بس هكذا بتضمن لا حدا يقرب عليك ولا حدا يبوسك . 😂😂 😅😅😅 الله يبعد مننا ومنكم البلاء يارب 😂	علاج ايروس الكورونا كل يوم الصبح الواحد يقرط ثلات حبات ثوم ومع نو الثوم ما له علاقة بايروس الكورونا بس هكذا بتضمن لا حدا يقرب عليك ولا حدا يبوسك الله يبعد مننا ومنكم البلا يارب
4	@Waleed_Albashaa هههههههه انتظر بعد ما تخلص كورونا عادي 😂	هههههههه انتظر بعد ما تخلص كورونا عادي
10	@hamedalhinai02 هههه بعد كورونا إن شاء الله اما حالياً ملتزمين بالتباعد 😁 👍🏻	هههه بعد كورونا ن شا الله اما حاليا ملتزمين بالتباعد
20	@MaiBadran4 هههههههههه بعد كورونا ن شاء الله	هههههههههه بعد كورونا ن شا الله
30	. من كتر أخبار أعداد الموتي بسبب القتل والحرب و الكورونا مذيعة قناة العربية غلطت و قالت :- {{{ سنموت بعد الفاصل }}} 😂 https://t.co/5kYeIqyJ0k	من كتر خبار عداد الموتي بسبب القتل والحرب و الكورونا مذيعة قناة العربية غلطت و قالت سنموت بعد الفاصل
40	بسبب الخوف والرعب من فيروس كورونا / اخطأت وقالت سنموت بعد الفاصل 😁 https://t.co/pERviPvm6O	بسبب الخوف والرعب من فيروس كورونا اخطت وقالت سنموت بعد الفاصل
50	المصابين بفايروس كورونا عفاهم الله :- انتم تؤجرون على البلاء وتؤجرون على الصبر وتؤجرون على المرض وتؤجرون على الألم فلا تحزنوا ولا تكرهوا شيئاً قد اختاره الله لكم ، فأنتم بين بشارة وكفارة وطهارة بإذن الله #الحالات_اليوم	المصابين بفايروس كورونا عفاهم الله انتم تجرون على البلا وتجرون على الصبر وتجرون على المرض وتجرون على اللم فلا تحزنوا ولا تكرهوا شيا قد اختاره الله لكم فنتم بين بشارة وكفارة وطهارة بذن الله

Figure 7:the dataset before and after cleaning.

The text cleaning involved removing emojis, user mentions, URLs, decorative symbols, stop words and normalizing Arabic phrases for clarity and consistency.

4.2 Preprocessing the Sentiment Labels:

Chapter 2: Methodology and Experimental Framework

Before training and evaluating models using different feature extraction techniques such as TF-IDF, Bag of Words (BoW), and Word2Vec (W2V), it is essential to preprocess the sentiment labels to ensure they are compatible with machine learning and deep learning frameworks. In the original dataset, sentiment annotations were textual: "Positive", "Negative", and "Neutral". These categorical labels need to be transformed into a numerical format to be usable by the models.

- **Steps for Sentiment Label Preprocessing:**
  - **Label Encoding:**
    - Each sentiment category was mapped to a unique integer:
      - Negative → 0
      - Neutral → 1
      - Positive → 2
    - Consistency Check:A check was performed to ensure all labels were correctly mapped, and no unexpected values or inconsistencies were present in the dataset.
  - **One-Hot Encoding (for deep learning models):** For compatibility with categorical loss functions like categorical_crossentropy, the integer-encoded labels were further converted into one-hot vectors.
    - **For example:**
      - 0 → [1, 0, 0]
      - 1 → [0, 1, 0]
      - 2→ [0, 0, 1]
- **Applicability Across Feature Representations:** This label preprocessing step was applied consistently across all experiments using:
  - TF-IDF features
  - Bag of Words features
  - Word2Vec embeddings

The main benefits of this preprocessing are that it ensures numerical consistency required by most machine learning and deep learning algorithms, prevents errors that may arise from varying label formats, and facilitates the computation of evaluation metrics such as accuracy, precision, and confusion matrix.

This preprocessing standardizes the sentiment labels and prepares them for seamless integration into both classical and neural network-based classification pipelines.

Chapter 2: Methodology and Experimental Framework

4.3 Tokenization Process:

After cleaning the text data, the next essential step is tokenization which is splitting each comment into individual words (tokens). Tokenization is a foundational preprocessing step in NLP, enabling further analysis such as feature extraction and modeling (show figure):

	Text	cleaned_text	tokenized_text
0	علاج ڤايروس الكورونا* كل يوم الصبح الواحد يقرط* ثلات حبات ثوم، ومع إنو الثوم ما إله علاقة بڤايروس الكورونا بس هكذا بتضمن لا حدا يقرب عليك ولا حدا يبوسك . 😂😂😂 😅😅😅 الله يبعد منا ومنكم البلاء يارب	علاج ايروس الكورونا يوم الصبح الواحد يقرط ثلات حبات ثوم ومع نو الثوم علاقة بايروس الكورونا بتضمن حدا يقرب حدا يبوسك الله يبعد منا ومنكم البلا يارب	[علاج, ايروس, الكورونا, يوم, الصبح, الواحد, يقرط, ثلات, حبات, ثوم, ومع, نو, الثوم, علاقة, بايروس, الكورونا, بتضمن, حدا, يقرب, حدا, يبوسك, الله, يبعد, منا, ومنكم, البلا, يارب]
4	@Waleed_Albashaa هههههههه انتظر بعد ما تخلص كورونا عادي 😂	هههههههه انتظر تخلص كورونا عادي	[هههههههه, انتظر, تخلص, كورونا, عادي]
10	@hamedalhinai02 هههه بعد كورونا إن شاء الله اما حالياً ملتزمين بالتباعد 😁👍🏻	هههه كورونا شا الله اما حاليا ملتزمين بالتباعد	[هههه, كورونا, شا, الله, اما, حاليا, ملتزمين, بالتباعد]

Figure 8:Tokenization

In this study, we used the Natural Language Toolkit (NLTK) library for tokenization.

- **Pseudocode for Tokenization**

  - **Input:** Cleaned dataset containing Arabic text comments.
  - **Process:**
    - Import the NLTK library and download the 'punkt' tokenizer models.
    - For each comment in the cleaned dataset:
    - Use **word_tokenize** from NLTK to split the comment into tokens (words).
    - Store the list of tokens in a new column.
    - Inspect the results by displaying samples of the original text and their corresponding tokens.
  - **Output:** Dataset with an additional column containing tokenized list of words of Arabic comments.

Chapter 2: Methodology and Experimental Framework

4.4 Feature Extraction Techniques:

Feature extraction transforms raw text into numerical representations suitable for machine learning models and deep learning. In this study, we explore three main techniques:

4.4.1 TF-IDF (Term Frequency-Inverse Document Frequency):

Assigns weights to words based on their frequency in a document relative to their frequency across all documents, highlighting important terms.

TF-IDF was chosen because it effectively highlights the most informative and distinctive words in each comment, reducing the influence of common or less meaningful terms. This helps the model focus on features that are more likely to contribute to accurate sentiment classification, especially in a dataset with a rich and diverse vocabulary like ours. Additionally, TF-IDF is computationally efficient and widely used in text classification tasks, making it a strong baseline for comparison with more advanced embedding techniques, (show figure)

	الصبر	الصين	الكورونا	الله	اللهم	اللي	الموت	الوبا	ان	شي	فيروس	كورونا	يارب	يروح	يلعن
0	0.0	0.0	0.757605	0.27518	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.591871	0.0	0.0
1	0.0	0.0	0.000000	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.000000	0.0	0.0
2	0.0	0.0	1.000000	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
3	0.0	0.0	0.000000	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0
4	0.0	0.0	0.000000	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.000000	0.0	0.0

Figure 9:TF-IDF in the dataset.

In this work, the vectorizer was configured with **max features set to 12665** to match the vocabulary size after cleaning, ensuring efficiency and focus on the most relevant terms. The tokenized text column was fit and transformed, resulting in a sparse TF-IDF matrix where each row represents a comment and each column corresponds to a unique word.

The resulting TF-IDF features served as input for traditional machine learning and deep learning models in the sentiment classification task.

- **Pseudocode for TF-IDF Feature Extraction:**
  - **Input:** Tokenized text data
  - **Process:**
    - Initialize a TF-IDF vectorizer with the desired maximum vocabulary size.
    - Fit the vectorizer on the cleaned text data to learn the vocabulary and IDF weights.
    - Transform the cleaned text into a TF-IDF feature matrix.
  - **Output:** A sparse matrix where each row represents a comment and each column represents a unique word, with values indicating the TF-IDF weight of each word in each comment.

4.4.2 Bag of Words (BoW):

Represents text by counting the occurrence of each word in the vocabulary, disregarding grammar and word order.

BoW was chosen because it offers a straightforward and interpretable way to represent text data, making it easy to understand which words contribute most to sentiment classification. Despite its simplicity and the fact that it ignores word order and context, BoW often serves as a strong baseline in text classification tasks. It is computationally efficient and well-suited for datasets with a large and diverse vocabulary, such as ours. (show figure).

	الصبر	الصين	الكورونا	الله	اللهم	اللي	الموت	الوبا	ان	شي	فيروس	كورونا	يارب	يروح	يلعن
0	0	0	2	1	0	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Figure 10:BoW in the dataset.

Bag of Words (BoW) was implemented to convert the cleaned Arabic text into a matrix of token counts. This approach captures the frequency of each word in every comment, providing a simple yet effective representation for traditional machine learning models and deep learning models.

- **Pseudocode for Bag of Words Feature Extraction:**
  - **Input:** Tokenized text data
  - **Process:**
    - Initialize **a CountVectorizer.**
    - Fit the vectorizer on the cleaned text to build the vocabulary.
    - Transform the cleaned text into a Bag of Words feature matrix.
  - **Output:** A matrix where each row represents a comment and each column represents a unique word, with values indicating the count of each word in each comment.

This BoW matrix was then used as input for various machine learning algorithms to evaluate its effectiveness compared to TF-IDF and Word2Vec embeddings.

4.4.3 Word2Vec with AraVec:

Uses pre-trained AraVec embeddings to map Arabic words into dense vectors that capture semantic and syntactic relationships, allowing for more complex text display.

Word2Vec was chosen because it generates distributed word representations that capture the context and meaning of words, which is especially valuable for morphologically rich languages like Arabic. By using the pre-trained AraVec model, which is trained on large-scale Arabic Twitter data, we can obtain high-quality embeddings that reflect real-world language usage and relationships between words.

In this work, the AraVec model (`full_grams_cbow_300_twitter.mdl`) was loaded using the Gensim library. The model provides 300-dimensional embeddings for a wide range of Arabic words. Two types of features were extracted:

Average Word Vectors: For classical machine learning models, each comment is represented by the average of its word vectors.

Embedding Matrix: For deep learning models, an embedding matrix is constructed and used in Keras Embedding layers.

- **Pseudocode for Word2Vec (AraVec) Feature Extraction:**

  - **Input:**
    - tokenized text data
    - Pre-trained AraVec Word2Vec model

  - **Process:**
    - Load the AraVec model to obtain the embedding dimension (300).
    - Build a vocabulary from the dataset.
    - For each word in the vocabulary, find its vector from AraVec if available.
    - Construct an embedding matrix where each row corresponds to a word's vector.
    - For classical ML: Compute the average vector for each comment.
    - For deep learning: Tokenize and pad sequences, then use the embedding matrix in the Embedding layer.

  - **Output:** For ML: Matrix of average word vectors for each comment.

    For DL: Embedding matrix and padded sequences, ready for use in neural network models.

5 Model Design and Implementation:

This section describes the design and implementation of the models used for Arabic sentiment analysis. Both traditional machine learning and deep learning approaches were explored to evaluate their effectiveness on the dataset.

5.1 Machine Learning Models

Several classical machine learning algorithms were implemented and compared:

5.1.1 Support Vector Machine (SVM):

SVM is effective for high-dimensional data and is widely used in text classification tasks, as it works by finding the optimal hyperplane that separates data points of different classes with the maximum margin. SVMs are robust to overfitting, particularly in high-dimensional spaces common in text data, and they perform well with sparse feature representations such as TF-IDF and Bag of Words. Additionally, SVMs are computationally efficient for medium-sized datasets and provide strong baseline performance. (show figure).

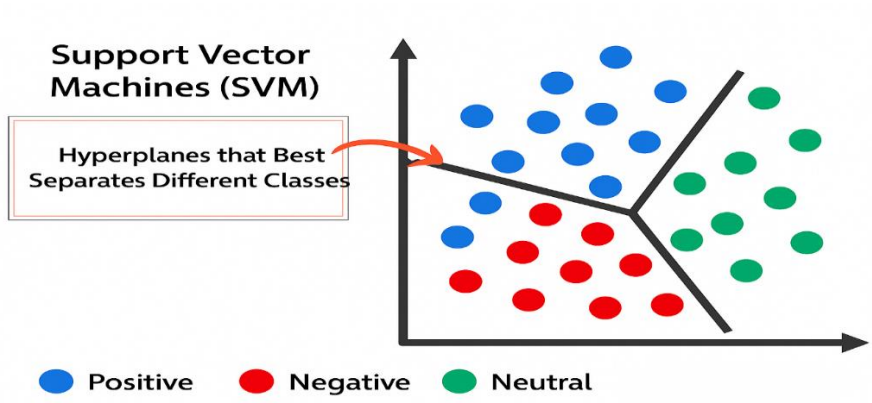


Figure 11:SVM model

For consistency, the same SVM model was applied across all feature extraction techniques (TF-IDF, BoW, and Word2Vec). The process involved initializing a linear SVM, training it on the training set, and evaluating its performance on the test set. The pseudocode is as follows:

- **Pseudocode for SVM Model:**
  - **Input:** Training features and labels, test features
  - **Process:**
    - Initialize the SVM model
    - Train the model on the training data
    - Use the trained model to predict labels for the test data
  - **Output:** Predicted labels for the test data.

5.1.2 Decision Tree:

Decision Trees are simple and interpretable models that split the data based on feature values, creating a tree-like structure of decisions. Each internal node represents a test on a feature, each branch represents an outcome of the test, and each leaf node represents a class label.

Decision Trees were chosen because they are easy to understand and visualize, making them useful for interpreting model decisions. Additionally, they can handle both numerical and categorical data without requiring feature scaling, and are capable of capturing non-linear relationships between features and target classes. (show figure).

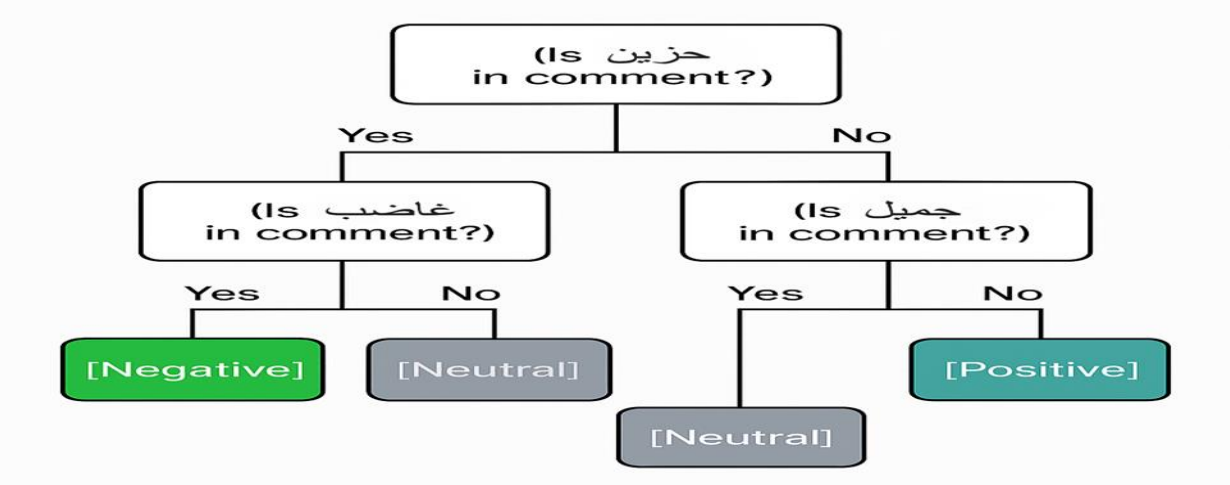


Figure 12:Decision Tree model.

For consistency, the same Decision Tree model was applied across all feature extraction techniques (TF-IDF, BoW, and Word2Vec).

- **Pseudocode for Decision Tree Model:**
  - **Input:** Training features and labels, test features
  - **Process:**
    - Initialize the Decision Tree model
    - Train the model on the training data
    - Use the trained model to predict labels for the test data
  - **Output:** Predicted labels for the test data

5.1.3 Random Forest:

Random Forest is an ensemble method that builds multiple decision trees and combines their outputs to improve generalization and reduce overfitting. (show figure).

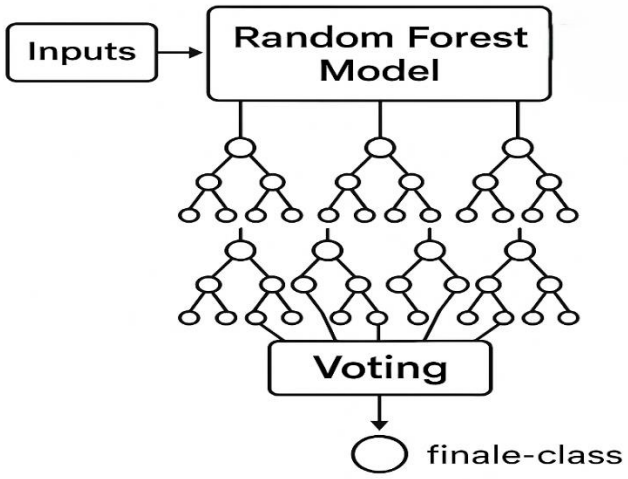


Figure 13:Random Forest model.

Random Forests were chosen for several reasons. They are robust to noise and overfitting, particularly when dealing with high-dimensional data. Additionally, Random Forests can effectively handle large feature spaces and offer the advantage of providing feature importance scores, which aid in model interpretability. The ensemble nature of Random Forests typically results in better performance compared to a single decision tree.

For consistency, Random Forest models were applied across all feature extraction techniques (TF-IDF, BoW, and Word2Vec). However, some differences in hyperparameters were necessary to account for the varying dimensionality and sparsity of the feature sets.

- **Pseudocode for Random Forest Model:**
  - **Input:** Training features and labels, test features
  - **Process:**
    - Initialize the Random Forest model with suitable hyperparameters for the feature type
    - Train the model on the training data
    - Use the trained model to predict labels for the test data
  - **Output:** Predicted labels for the test data

5.1.4 XGBoost:

XGBoost is a powerful gradient boosting algorithm that often achieves state-of-the-art results in classification tasks. It builds trees sequentially, where each new tree corrects errors made by previous ones.

XGBoost is highly efficient and scalable, making it suitable for large datasets and high-dimensional feature spaces. It incorporates regularization to reduce overfitting and improve generalization. Additionally, XGBoost provides built-in support for handling missing values and offers advanced features such as early stopping and parallel processing.

In our implementation, we set the `eval_metric='mlogloss'` parameter in the XGBoost classifier to ensure that the model's performance was evaluated using the multiclass logarithmic loss metric during training. This metric is particularly suitable for multiclass classification problems, as it provides a more informative assessment of model performance compared to simple accuracy, especially when dealing with imbalanced classes.

For consistency, XGBoost models were applied across all feature extraction techniques (TF-IDF, BoW, and Word2Vec).

Pseudocode for XGBoost Model:

- **Input:** Training features and labels, test features
- **Process:**
  - ❖ Initialize the XGBoost model with suitable hyperparameters for the feature type
  - ❖ Train the model on the training data
  - ❖ Use the trained model to predict labels for the test data
- **Output:** Predicted labels for the test data

These models were trained using the extracted features (TF-IDF, BoW, or Word2Vec) and evaluated using standard metrics such as accuracy, precision, recall, and F1-score.

5.2 Deep Learning Models:

To capture more complex patterns in the text, deep learning models were also implemented:

5.2.1 Deep Learning Models for TF-IDF and BOW:

Deep Learning Models for TF-IDF and BOW:

5.2.1.1 Deep Neural Network (DNN)

DNN is a feedforward neural network consisting of multiple dense fully connected layers. For both TF-IDF and BoW features the following pseudocode was applied :

- **Input:** Feature vectors generated by TF-IDF or BoW, with dimensionality equal to the size of the vocabulary.
- **Process:**
  - Dense layer with ReLU activation and L2 regularization.
  - Dropout layer.
  - Another dense layer with ReLU activation and L2 regularization.
  - Another Dropout layers.
  - Final dense output layer with 3 units (for three classes) and SoftMax activation, with L2 regularization.
- **Output:** Predicted class probabilities for each input sequence.

5.2.1.2 Bidirectional Long Short-Term Memory (B-LSTM):

B-LSTM models are designed to capture sequential dependencies and contextual information in text by processing input as ordered sequences of word embeddings. However, TF-IDF and Bag of Words (BoW) representations transform text into high-dimensional, sparse vectors that discard word order and sequence information. As a result, these representations are incompatible with B-LSTM architectures, which require input in the form of word index sequences or embeddings that preserve the original word order. Therefore, B-LSTM models

cannot be effectively applied to TF-IDF or BoW features, and are instead used with embedding-based representations such as Word2Vec.

5.2.2 Deep Learning Models for Word2Vector:

Deep Learning Models for Word2Vector:

5.2.2.1 Deep Neural Network (DNN)

For the Word2Vec (W2V) features the following pseudocode was applied :

- **Input:** Sequences of word indices with length `max_len`.
- **Process:**
  - Embedding layer (`Embedding`) with input_dim=max_words, output_dim=embedding_dim, and input_length=max_len.
  - Global average pooling layer (`GlobalAveragePooling1D`) to reduce the sequence dimension.
  - Dense layer with ReLU activation and L2 regularization.
  - Dropout layer.
  - Dense layer with ReLU activation and L2 regularization.
  - Another Dropout layer.
  - Final dense output layer with 3 units (for three classes) and softmax activation, with L2 regularization.
- **Output:** Predicted class probabilities for each input sequence.

5.2.2.2 Bidirectional Long Short-Term Memory (B-LSTM):

For B-LSTM features the following pseudocode was applied :

Input:
Sequences of integer word indices with fixed length `max_len`, where each word corresponds to a token in the vocabulary.

Process:

- **Embedding Layer**: Transforms input sequences into dense vectors using a pre-trained Word2Vec embedding matrix. Parameters: `input_dim=max_words`, `output_dim=embedding_dim`, `weights=[embedding_matrix]`, `input_length=max_len`, `trainable=True`.
- **Dropout Layer**: Randomly drops 30% of the embedding outputs to reduce overfitting.
- **First Bidirectional LSTM Layer**: Applies a Bi-LSTM with 100 units in both directions, outputs a sequence. Dropout is set to 0.3 and recurrent dropout to 0.2.

- **Dropout Layer**: Again, 30% dropout after the first Bi-LSTM to regularize.
- **Second Bidirectional LSTM Layer**: Applies another Bi-LSTM with 64 units, but returns only the final output (not the full sequence). Dropout = 0.3, recurrent dropout = 0.2.
- **Dense Output Layer**: Fully connected layer with 3 units and softmax activation to classify into one of three sentiment classes: positive, negative, or neutral.

Output:
A probability distribution across the 3 sentiment classes for each input sequence.

The models were compiled with the Adam optimizer, sparse categorical cross-entropy loss, and accuracy as the evaluation metric. Early stopping was employed to monitor validation accuracy and restore the best weights if performance did not improve for several epochs. A custom callback was used to evaluate test performance during training.

The learning rate was dynamically adjusted using both a reduction-on-plateau strategy and an exponential decay schedule to facilitate efficient convergence and prevent overfitting.

6 Conclusion:

In conclusion, this chapter has outlined the methodology adopted for Arabic sentiment analysis, covering the processes of data collection, preprocessing, feature extraction, and model design. These steps lay the foundational framework necessary for building an effective sentiment classification system. In the following chapter, we proceed to the implementation phase, where the proposed models are developed and trained. Additionally, we will present and discuss the experimental results, evaluating the performance of the models and analyzing their effectiveness in accurately capturing sentiment in Arabic textual data.

Chapter 3:

Implementation and Results

Chapter 3: Implementation and Results

1 Introduction:

Our primary objective is to propose a sentiment analysis model for Arabic dialect. In this chapter, we focus on the implementation of our model. We begin by presenting the programming language, development environment, and tools used. Next, we provide a detailed overview of the various stages of our work, from the choices of the hyperparameters to the evaluation of the models used. We conclude this section by discussing the results and determining the most effective model.

2 Environment and work tools:

In this section, we present the development environment and the tools used for the implementation of our system. We describe the choices of programming languages and the libraries employed, as well as the platforms and software that facilitated the development and testing of our solution.

2.1 Environment Hardware

The hardware used for testing is summarized in the following table:

- **PC**:

Machine	**specifications**
Device	Dell XPS Laptop
Processor	Intel Core i7, 12th Generation (Performance Series)
RAM	16 GB DDR5
Graphics	Integrated Intel Iris Xe Graphics
Storage	1 TB NVMe SSD
Operating System	Windows 11

Table 3:PC specification.

2.2 Programming language and software:

- **Python**: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed [15].

- **Visual Studio Code IDE:** Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE [16].

2.3Libraries

the libraries we have imported into our project are defined in the following:

- **NumPy:** NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more [17].

- **Pandas:** pandas is a fast, powerful, flexible and easy to use open-source data analysis and manipulation tool, built on top of the python programming language [18].

- **Keras** Keras is a deep learning API designed for human beings, not machines. Keras focuses on debugging speed, code elegance & conciseness, maintainability, and deplorability. When you choose Keras, your codebase is smaller, more readable, easier to iterate on [19].

- **Tensorflow** : TensorFlow isan open-source software library developed by Google for machine learning and artificial intelligence. It's primarily used for training and deploying machine learning models, especially deep neural networks. It is versatile and can be used for various tasks like natural language processing, image recognition, and more [20].

- **SeaBorn:** Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics [21] **.**

- **NLTK:** NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum [22].

3.1.1 Evaluation Metrics:

- **Accuracy:** Measures the proportion of correctly classified samples among all samples.

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

where **TP** = True Positives, **TN** = True Negatives, **FP** = False Positives, **FN** = False Negatives.

- **Precision:** Indicates the proportion of true positive predictions among all positive predictions made by the model.

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

- **Recall:** Reflects the proportion of true positive predictions among all actual positive samples.

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

- **F1-score:** The harmonic means of precision and recall, providing a balanced measure of model performance, especially in the presence of class imbalance.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.1.2 Use of Confusion Matrices:

- Confusion matrices were generated for each model to visualize the distribution of true and predicted labels across sentiment classes.
- The confusion matrix provides detailed insight into model performance, highlighting which classes are most often confused and where misclassifications occur.
- The combination of these evaluation strategies ensures a comprehensive and fair assessment of all models, allowing for meaningful comparison and discussion of results.

3.2 Hyperparameter choices and Model configuration:

This section details the hyperparameter choices and model configurations used for both traditional machine learning and deep learning models across different embedding techniques. Parameter selection was guided by iterative experimentation to balance model complexity, accuracy, and computational efficiency.

3 Training and Evaluation Setup:

To assess the performance of the sentiment analysis models, a rigorous training and evaluation setup was employed. The following procedures and metrics were used:

3.3 Procedures and Metrics:

To assess the performance of the sentiment analysis models, a rigorous training and evaluation setup was employed. The following procedures and metrics were used:

3.3.1 Train/Test Split and Cross-Validation:

To evaluate the performance of both machine learning and deep learning models, the dataset was split into training and testing sets using an 80/20 ratio.

Model Type	Training	Validation	Test
Neural Network, BiLSTM	72%	8%	20%
SVM, DT, RF, XGBoost	80%	—	20%

Table 4:Data splitting.

A fixed random seed ensured reproducibility. For deep learning models (Neural Network and BiLSTM), an additional 10% of the training data was set aside for validation during training to help monitor model performance and prevent overfitting. Traditional machine learning models (SVM, Decision Tree, Random Forest, XGBoost) were trained using the full training set without a validation split.

3.3.2 Parameter Choices for TF-IDF and BoW:

3.3.2.1 Machine Learning Models:

- **SVM:**
  - Kernel: Linear (after comparing with RBF and polynomial kernels).
  - Other parameters: Defaults.
- **Decision Tree:**
  - All parameters: Defaults.
  - Note: No restrictions to provide a clear baseline.

- **Random Forest:**
  - Number of estimators (trees):
    - TF-IDF: 71.
    - BoW: 100.
  - Other parameters: Defaults.

- **XGBoost:**
  - Parameters: Defaults
  - Evaluation metric: mlogloss for multiclass classification.

3.3.2.2 Deep Learning Models

- **Neural Network (DNN):**
  - Architecture: Two hidden layers 32 and 8 units.
  - Dropout: 0.5
  - L2 regularization: 0.0001
  - Learning rate:
    - TF-IDF: 0.00025
    - BoW: 0.0003
  - Batch size: 32
  - Loss function: categorical_crossentropy
  - Optimizer: Adam
  - ReduceLROnPlateau: Factor 0.5, patience 3 epochs
  - Early stopping: Monitors validation accuracy, restores best weights

3.3.3 Parameter Choices for Word2Vec (W2V):

3.3.3.1 Machine Learning Models:

- SVM, Random Forest, XGBoost, Decision Tree.
  - Parameters: Defaults except SVM uses linear kernel.
  - No explicit hyperparameter tuning. results reflect baseline performance with average Word2Vec embeddings.

3.3.3.2 Deep Learning Models:

- Bidirectional LSTM (B-LSTM).
  - Embedding layer: Pre-trained AraVec weights.

- LSTM layers: Two stacked Bidirectional LSTM with 100 and 64 units.
- Dropout: 0.3 with recurrent dropout 0.2.
- Optimizer: Adam with learning rate 0.001
- Loss function: `categorical_crossentropy`
- Activation function: Used default activations tanh for the main cell activation and sigmoid for the recurrent gate activation
- Batch size: 64.
- Early stopping.
- ReduceLROnPlateau.

- **DNN Model**
  - Embedding layer: Pre-trained AraVec weights.
  - Pooling: GlobalAveragePooling1D.
  - Dense layers: 128 and 64 units with ReLU and L2 regularization, each followed by dropout (0.5).
  - Optimizer: Adam (learning rate 0.001).
  - Loss function: `categorical_crossentropy`
  - Batch size: 64.
  - Early stopping.

All hyperparameters were determined through extensive experimentation and iterative testing. The final values reflect those that consistently led to the best test and validation results.

3.4 Results Overview:

A comparison of five models—SVM, Decision Tree, Random Forest, XGBoost, and Neural Network—was conducted for sentiment classification. The dataset includes three sentiment classes: Negative (0), Neutral (1), and Positive (2). Results are evaluated using accuracy, precision, recall, and F1-score.SVM, Decision Tree, Random Forest, XGBoost: All models achieved strong results, with Random Forest, XGBoost, and Neural Network providing the best overall performance.

3.4.1 TF-IDF:

3.4.1.1 TF-IDF in ML & DL:

- **SVM:** Accuracy: 0.89
  - Classification Report:
    - Negative (0): Precision 0.90, Recall 0.81, F1-score 0.86

- Neutral (1): Precision 0.81, Recall 0.96, F1-score 0.88
- Positive (2): Precision 0.97, Recall 0.86, F1-score 0.91

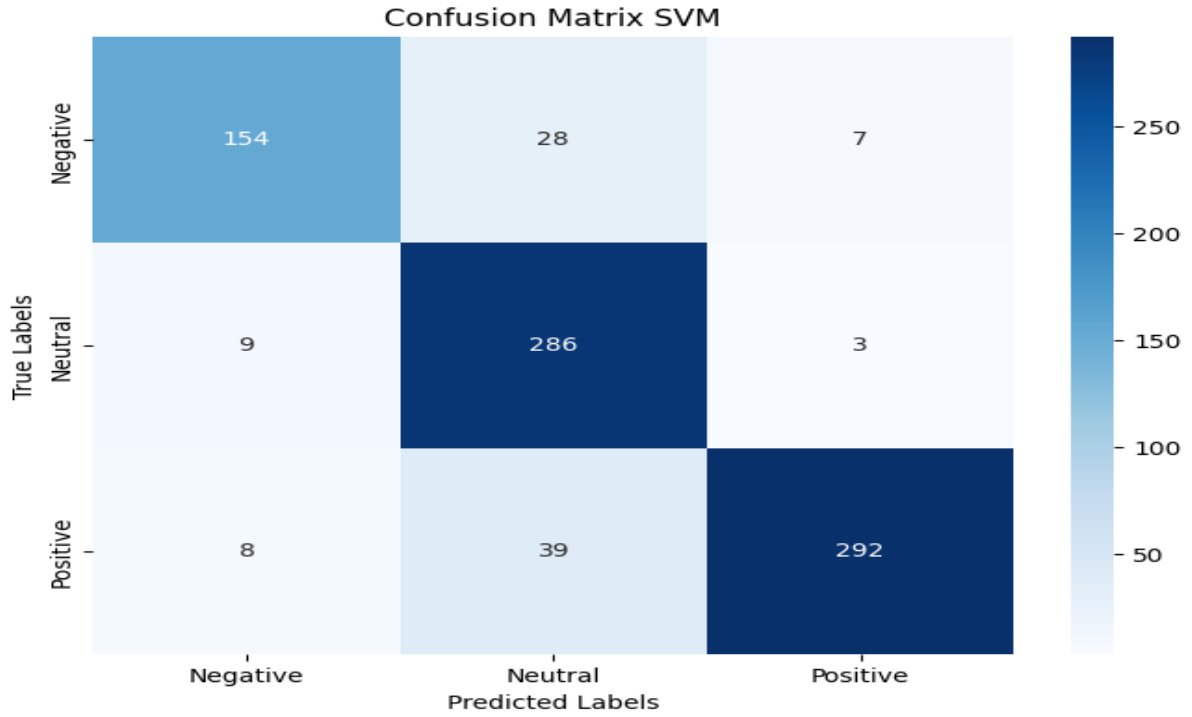


Figure 14:The SVM confusion matrix.

The SVM confusion matrix shows most predictions are correct, with few misclassifications mainly between neutral and negative or positive and neutral classes. The model rarely confuses positive with negative, confirming its strong and balanced performance across all sentiment categories.

- **Decision Tree:** Accuracy: 0.82
  - **Classification Report:**
    - Negative (0): Precision 0.84, Recall 0.79, F1-score 0.81.
    - Neutral (1): Precision 0.79, Recall 0.82, F1-score 0.80.
    - Positive (2): Precision 0.84, Recall 0.84, F1-score 0.84.

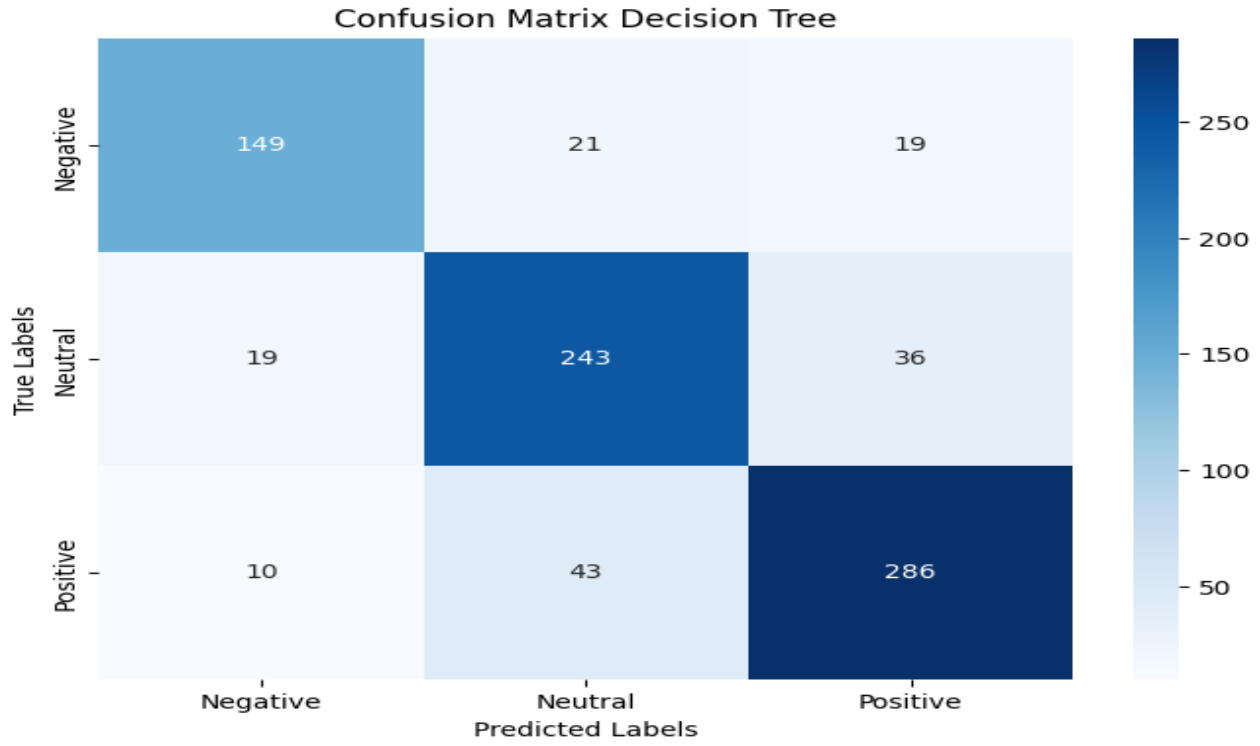


Figure 15:The Decision Tree confusion matrix.

The Decision Tree confusion matrix shows good alignment with true labels but reveals frequent confusion between neutral and the other classes, especially neutral versus positive. This underscores the model's interpretability but also its limited ability to distinguish subtle differences in sentiment compared to more advanced models.

- **Random Forest:** Accuracy: 0.86
  - **Classification Report:**
    - Negative (0): Precision 0.88, Recall 0.80, F1-score 0.84.
    - Neutral (1): Precision 0.79, Recall 0.91, F1-score 0.85.
    - Positive (2): Precision 0.93, Recall 0.86, F1-score 0.89.

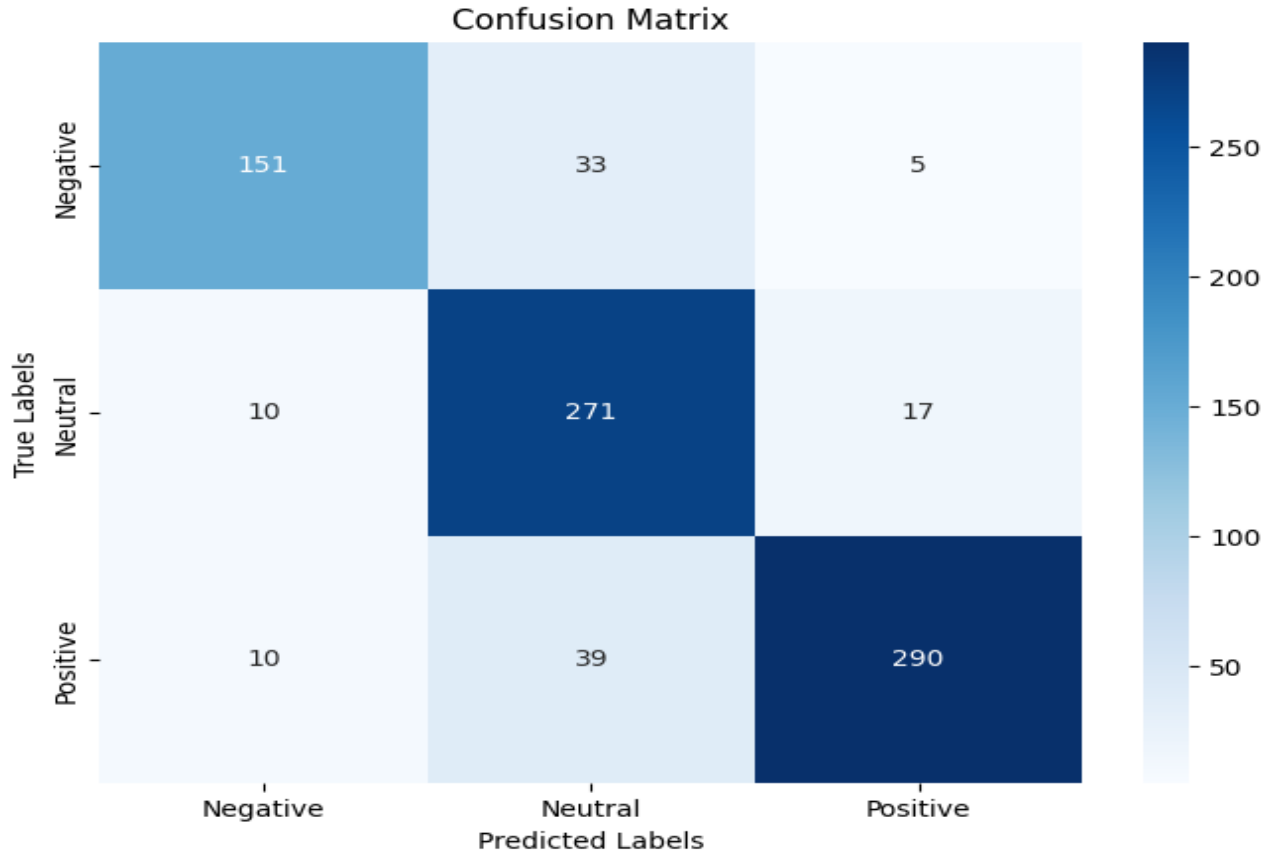


Figure 16:The Random Forest confusion matrix.

The confusion matrix indicates that most predictions are correct, with only moderate confusion between neutral and the other classes. This demonstrates the model's ability to generalize well and handle class imbalance, making it a reliable choice for sentiment classification tasks. Overall, Random Forest combines strong predictive power with interpretability, outperforming the Decision Tree and closely matching the performance of SVM and Neural Network models.

- **XGBoost:** Accuracy: 0.84
  - **Classification Report:**
    - Negative (0): Precision 0.88, Recall 0.78, F1-score 0.83.
    - Neutral (1): Precision 0.78, Recall 0.89, F1-score 0.83.
    - Positive (2): Precision 0.88, Recall 0.83, F1-score 0.86.

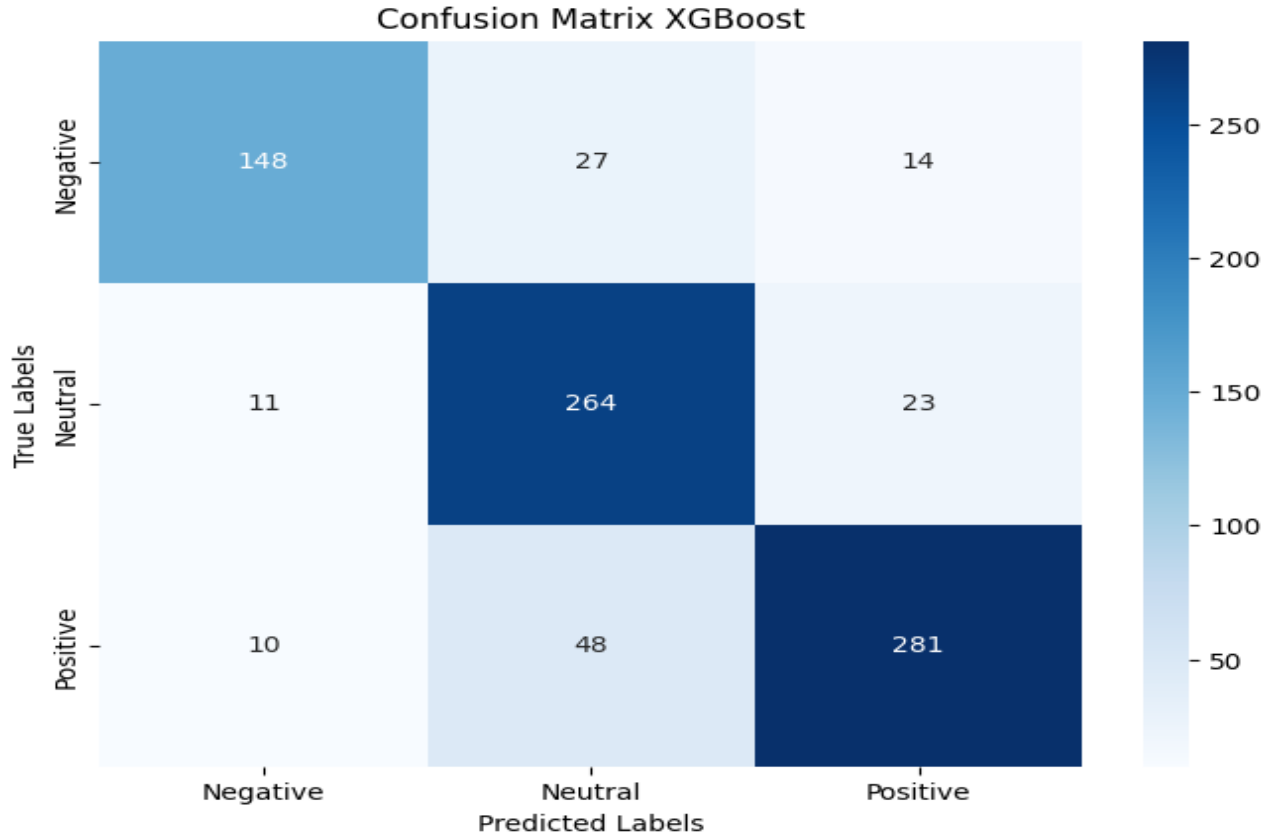


Figure 17:The XGBoost confusion matrix.

The XGBoost confusion matrix shows strong alignment with true labels, with most errors involving confusion between neutral and the other classes—especially neutral being misclassified as positive. Errors between negative and positive are rare, highlighting XGBoost's effectiveness in distinguishing sentiment classes, though neutral remains the most challenging to separate.

- **Neural Network:** Accuracy: 0.86
  - **Classification Report:**
    - Negative (0): Precision 0.85, Recall 0.80, F1-score 0.83
    - Neutral (1): Precision 0.82, Recall 0.88, F1-score 0.85
    - Positive (2): Precision 0.91, Recall 0.88, F1-score 0.89

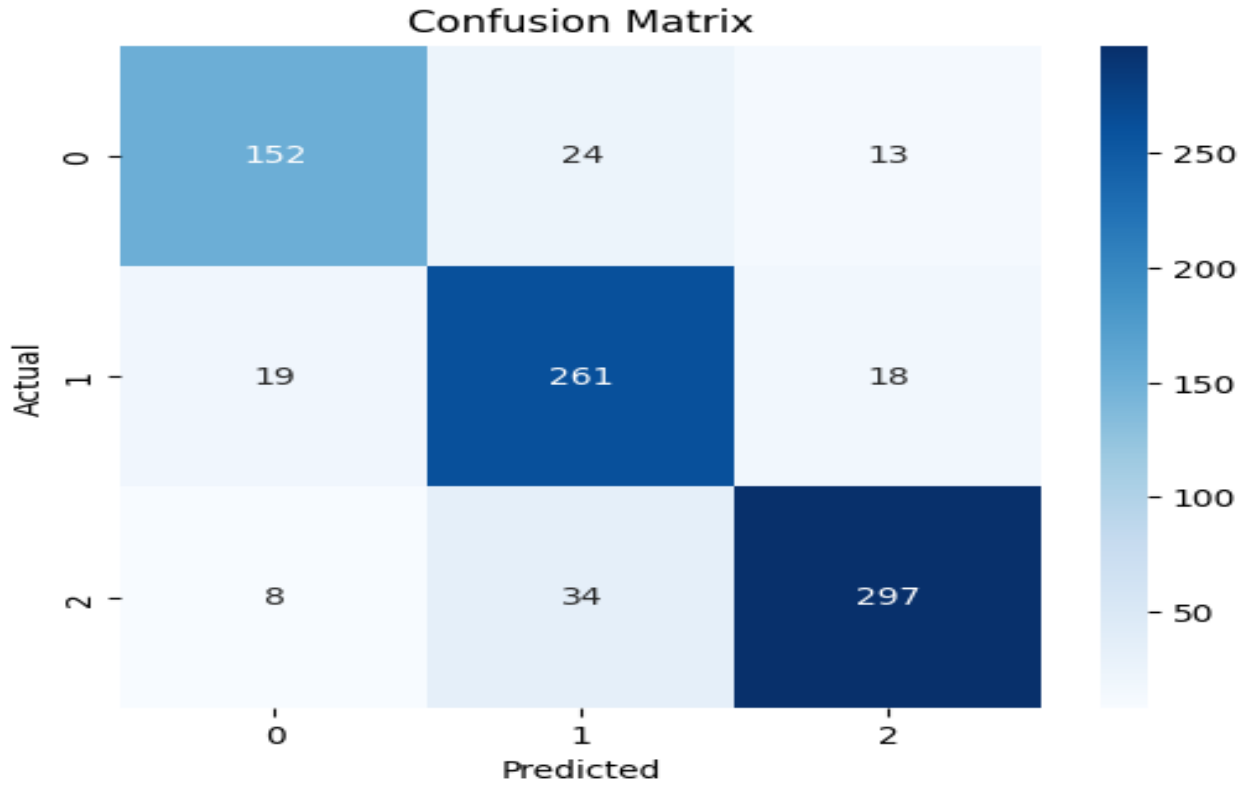


Figure 18:The Neural Network confusion matrix.

The confusion matrix shows that the Neural Network accurately predicts most cases, with most errors involving neutral comments being confused with negative or positive. Direct confusion between negative and positive is rare, highlighting the model's strong ability to distinguish clear sentiment classes while neutral remains the most ambiguous. Overall, the Neural Network maintains balanced and robust performance across all sentiment categories.

3.4.1.2 Learning Curves Summary:

The graph below gives a quick look at how the Neural Network learned over 21 epochs. On the left, you can see that both training and validation accuracy improved quickly and then leveled off, showing the model learned well and didn’t overfit. The test accuracy (red dashed line) closely matches the validation curve, which means the model performs just as well on new data.

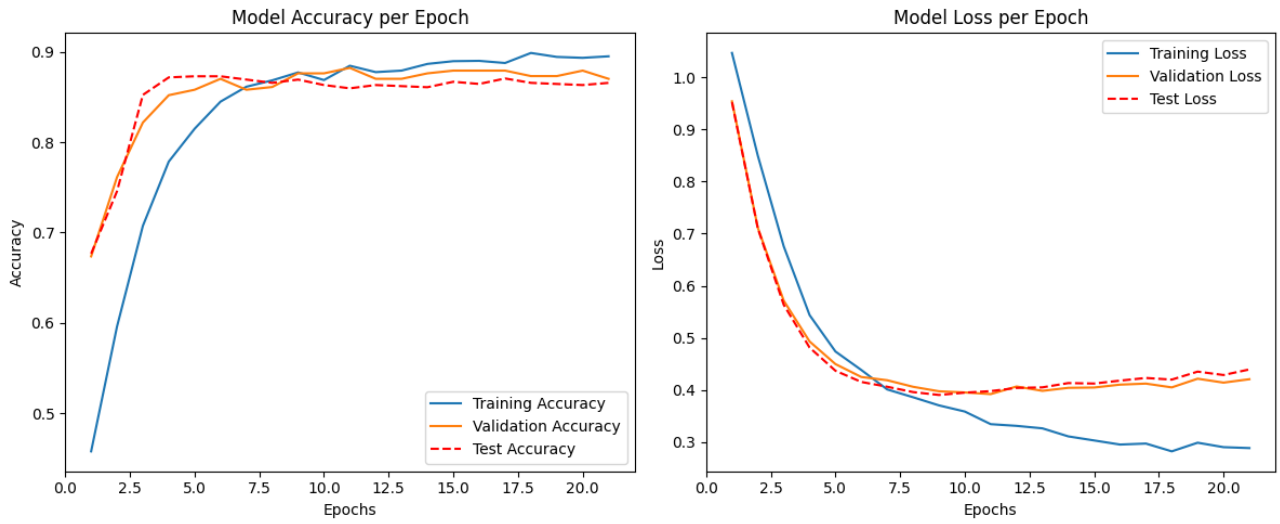


Figure 19:Model Acuracy and Model Lost per Epoch.

On the right, the loss curves for training, validation, and test sets all drop sharply at first and then flatten out together. This parallel movement means the model is stable and reliable, with no signs of overfitting or underfitting. In short, these curves show that the Neural Network learned efficiently and delivers consistent, trustworthy results for sentiment classification.

3.4.1.3 Training Time Comparison:

The bar chart visually below compares the training times of five models: SVM, Decision Tree, Random Forest, XGBoost, and Neural Network—when using TF-IDF features. SVM stands out with a much longer training time 291.3 seconds, far exceeding the other models. Decision Tree, Random Forest, and XGBoost all train significantly faster, with times clustered between 11 and 17 seconds. Neural Network has a moderate training time of 26.4 seconds, slightly higher than the ensemble methods but still much lower than SVM. Decision Tree is the quickest, while Random Forest and XGBoost have nearly identical training times.

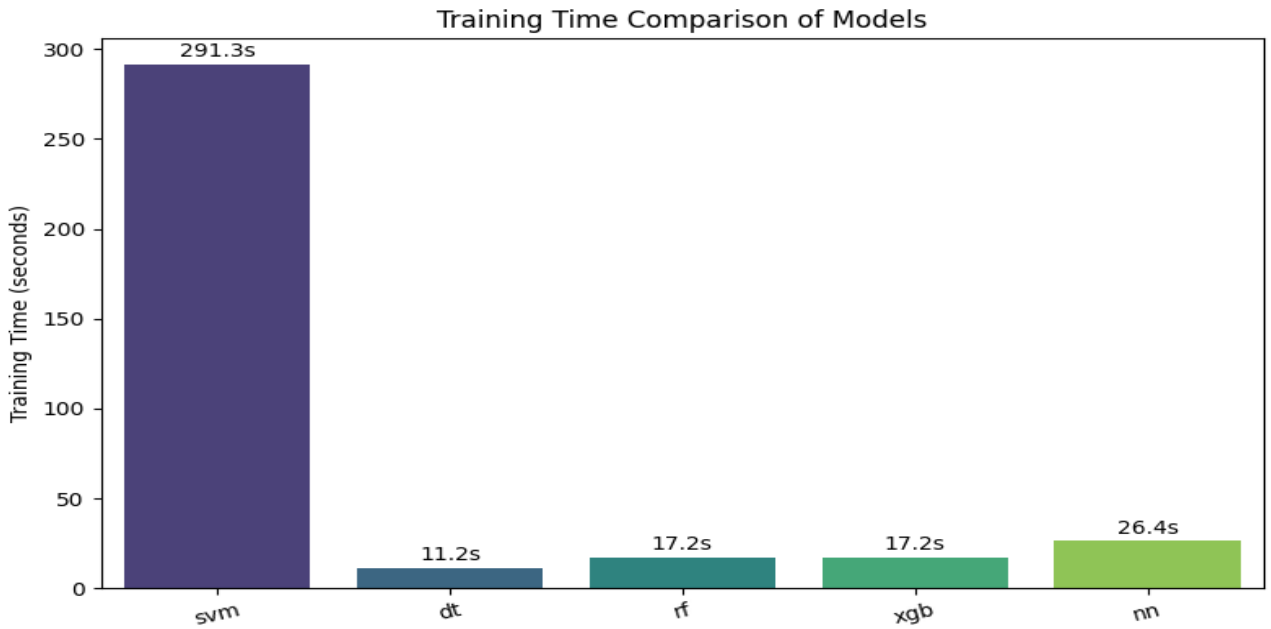


Figure 20:Training Time Comparison of Model.

This stark contrast highlights the computational cost of SVM in high-dimensional spaces, while ensemble methods and neural networks provide a better balance between speed and performance for practical sentiment analysis tasks.

3.4.1.4 TF-IDF Resault Discussion :

Model	Accuracy	Precision (avg)	Recall (avg)	F1-Score (avg)	Notable Observations
SVM	0.89	0.89	0.88	0.88	Highest overall accuracy; good balance; longest training time
Decision Tree	0.82	0.82	0.81	0.82	Fastest to train; struggles with neutral class
Random Forest	0.86	0.87	0.86	0.86	Strong, consistent; handles class imbalance well
XGBoost	0.84	0.85	0.83	0.84	Good balance; slight confusion in neutral class
Neural Network	0.86	0.86	0.85	0.86	Stable learning curves; no overfitting; moderate training time

Table 5 :Test Comparison of Models with TF-IDF.

In conclusion, the results show that while all five models performed well, Random Forest, SVM, and the Neural Network stood out as the top performers. SVM delivered the highest accuracy for positive sentiment, while Random Forest offered strong, consistent results across all sentiment classes and managed data imbalance effectively. The Neural Network also showed excellent performance, with smooth learning curves and no signs of overfitting, making it both accurate and stable.

However, performance isn't the only factor to consider—training time matters too. While SVM achieved impressive results, it also had the longest training time due to the complexity of processing high-dimensional data. In contrast, Decision Tree trained the fastest but fell short in classification performance, especially for neutral sentiment. Random Forest and XGBoost struck a good balance, offering strong accuracy with relatively quick training times. The Neural Network had a moderate training time, making it a practical choice when you want deep learning benefits without excessive computational cost.

In the end, if fast training and deployment are your main concerns, Decision Tree or Random Forest may be the better fit. But for tasks where accuracy and robustness are critical—and time or resources are less of a constraint—SVM and Neural Network are excellent options. Choosing the right model ultimately depends on balancing performance needs with practical limitations.

3.4.2 Bag of Words:

3.4.2.1 Bag of Words in ML & DL:

- **SVM:** Accuracy: 0.87

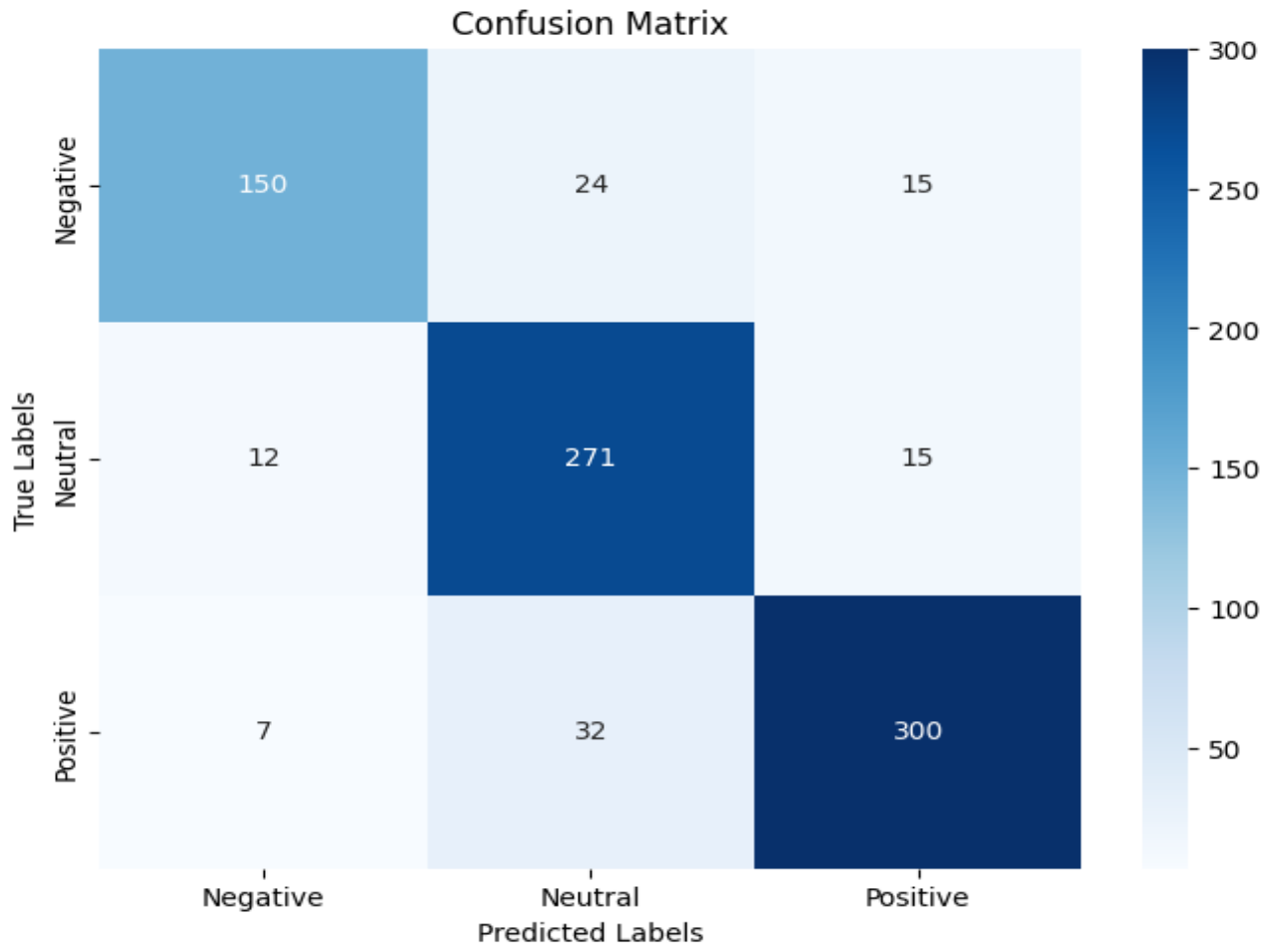


Figure 21:Confusion Matrix of SVM.

The confusion matrix for the SVM model indicates that the majority of predictions are accurate, with only a small number of errors occurring primarily between the neutral and negative, or neutral and positive classes. The model seldom confuses positive with negative sentiments, demonstrating its reliable and well-balanced performance across all sentiment categories.

- **Classification Report:**

  - Negative (0): Precision 0.83, Recall 0.83, F1-score 0.83.
  - Neutral (1): Precision 0.83, Recall 0.90, F1-score 0.86.
  - Positive (2): Precision 0.93, Recall 0.87, F1-score 0.90.

- **Decision Tree:** Accuracy: 0.85

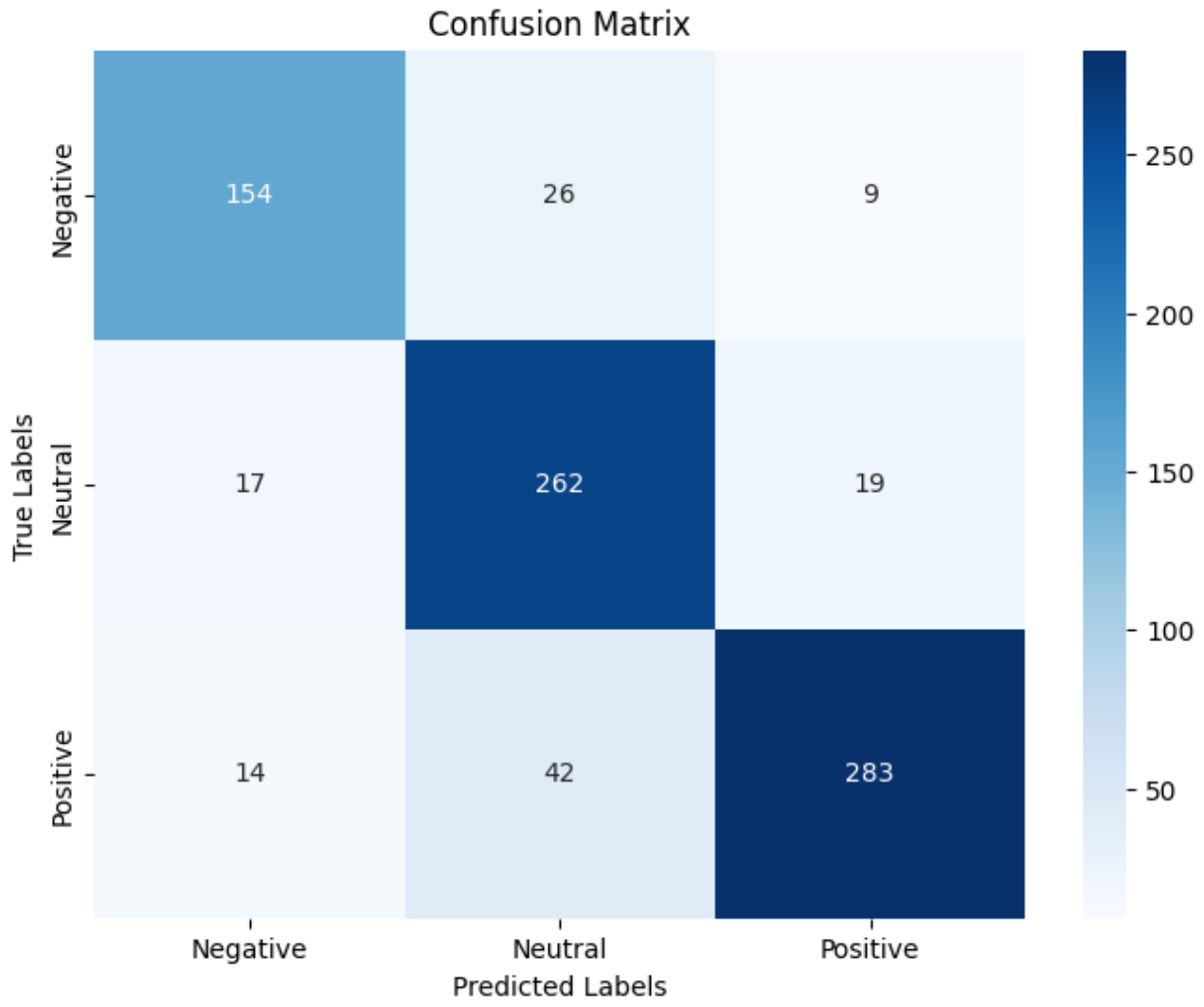


Figure 22:Confusion Matrix of Decision Tree.

The confusion matrix for the Decision Tree model reveals that while many predictions match the actual labels, there is significant overlap between the neutral class and the others—particularly between neutral and positive. This reflects the model's strength in interpretability but also exposes its limitations in capturing nuanced sentiment distinctions when compared to more sophisticated models.

- **Classification Report:**
  - Negative (0): Precision 0.83, Recall 0.81, F1-score 0.82
  - Neutral (1): Precision 0.79, Recall 0.88, F1-score 0.83
  - Positive (2): Precision 0.91, Recall 0.83, F1-score 0.87

- **Random Forest:** Accuracy: 0.86

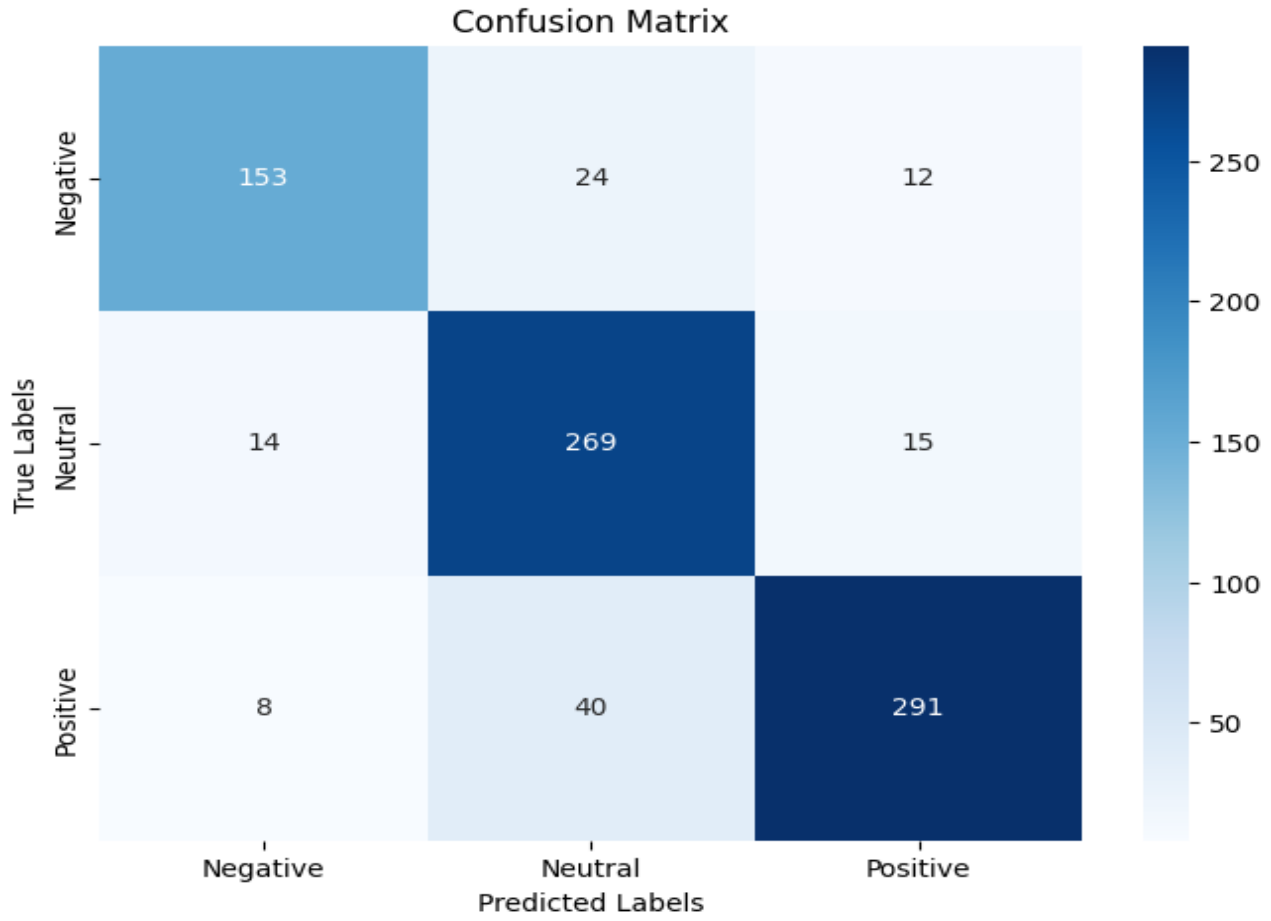


Figure 23:Confusion Matrix of Random Forest.

The Random Forest confusion matrix indicates that most predictions are correct, with moderate confusion between neutral and the other classes. This demonstrates the model's ability to generalize well and handle class imbalance, making it a reliable choice for sentiment classification tasks. Overall, Random Forest combines strong predictive power with interpretability, closely matching the performance of SVM and Neural Network models.

- **Classification Report:**
  - Negative (0): Precision 0.87, Recall 0.81, F1-score 0.84
  - Neutral (1): Precision 0.81, Recall 0.90, F1-score 0.85
  - Positive (2): Precision 0.92, Recall 0.86, F1-score 0.89

- **XGBoost:** Accuracy: 0.88

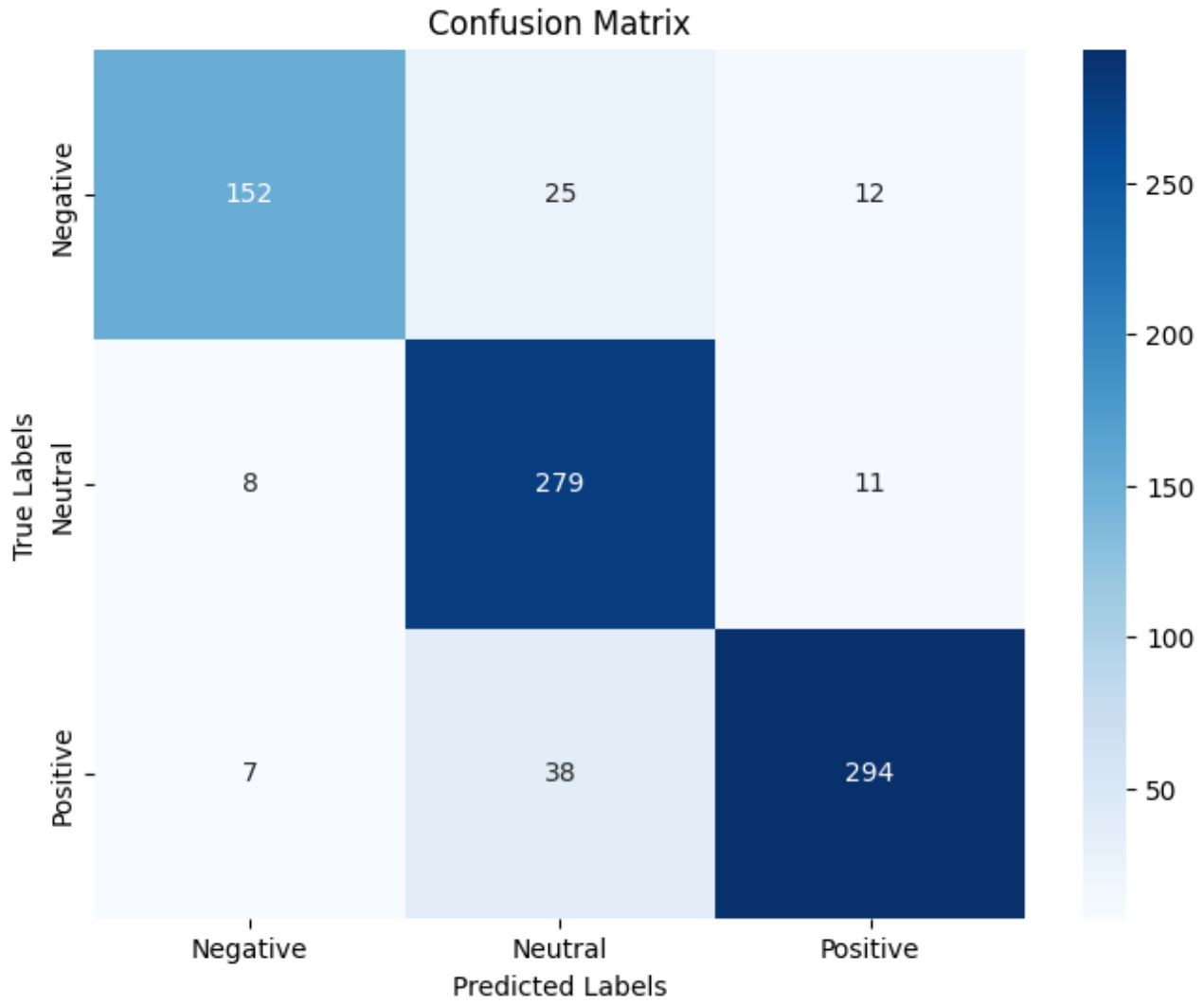


Figure 24:Confusion Matrix of XGBoost.

The XGBoost confusion matrix shows strong alignment with true labels, with most errors involving confusion between neutral and the other classes especially neutral being misclassified as positive. Errors between negative and positive are rare, highlighting XGBoost's effectiveness in distinguishing sentiment classes, though neutral remains the most challenging to separate.

- **Classification Report:**

  - Negative (0): Precision 0.91, Recall 0.80, F1-score 0.85
  - Neutral (1): Precision 0.82, Recall 0.94, F1-score 0.87
  - Positive (2): Precision 0.93, Recall 0.87, F1-score 0.90

- **Neural Network:** Accuracy: 0.87

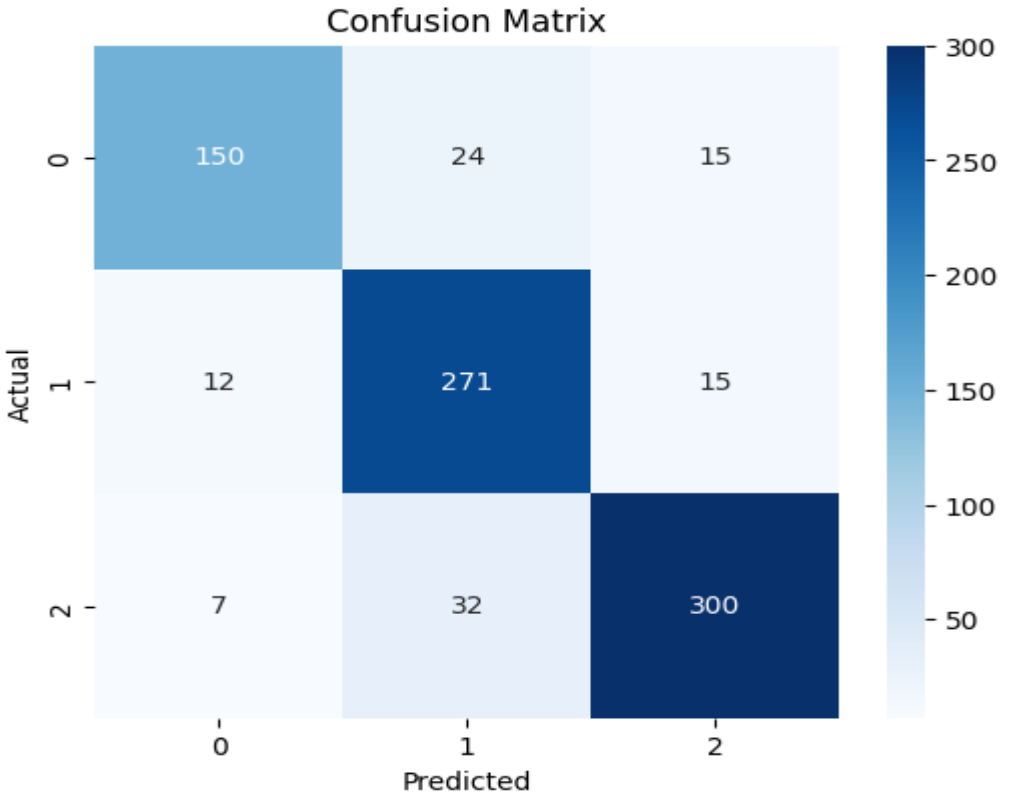


Figure 25:Confusion Matrix of Neural Network.

The confusion matrix shows that the Neural Network accurately predicts most cases, with most errors involving neutral comments being confused with negative or positive. Direct confusion between negative and positive is rare, highlighting the model's strong ability to distinguish clear sentiment classes while neutral remains the most ambiguous. Overall, the Neural Network maintains balanced and robust performance across all sentiment categories.

- **Classification Report:**
  - Negative (0): Precision 0.89, Recall 0.79, F1-score 0.84
  - Neutral (1): Precision 0.83, Recall 0.91, F1-score 0.87
  - Positive (2): Precision 0.91, Recall 0.88, F1-score 0.90

3.3.2.2 Learning Curves Summary:

The following graph displays the training, validation, and test accuracy and loss curves for the Neural Network model using Bag of Words features:

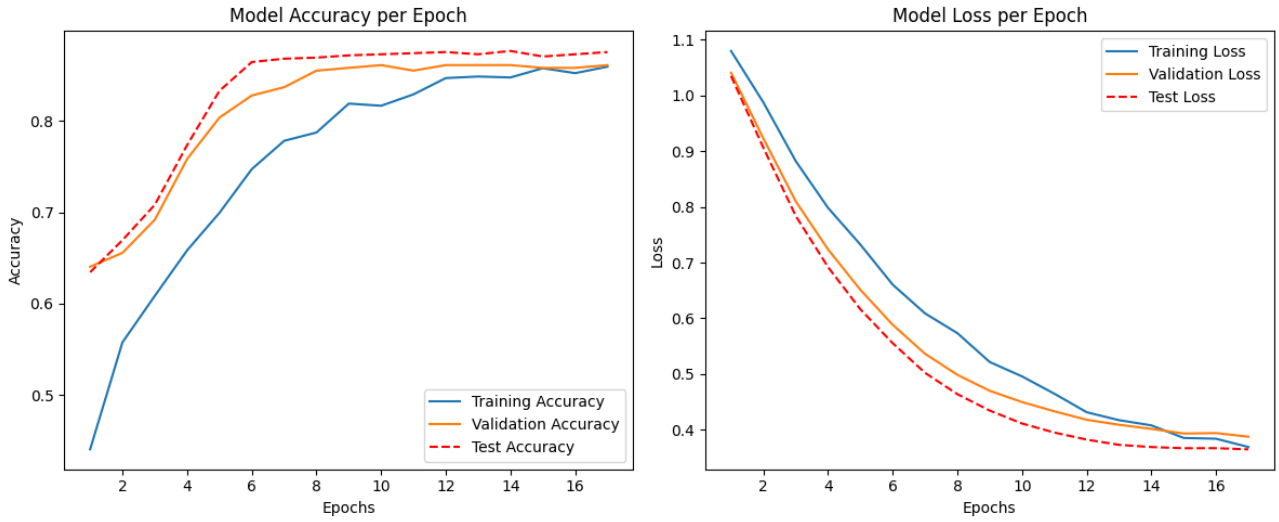


Figure 26:Model Accuracy and Model Loss per Epoch.

The Neural Network's accuracy and loss curves show rapid improvement followed by stable plateaus, with training, validation, and test metrics closely aligned. This indicates efficient learning, strong generalization, and no signs of overfitting or underfitting—making the model robust and reliable for sentiment classification using Bag of Words features.

3.3.2.3 Training Time Comparison:

This bar chart visually below compares the training times of five models—SVM, Decision Tree, Random Forest, XGBoost, and Neural Network—when using Bag of Words features. SVM stands out with a longer training time 492.4 seconds, far exceeding the other models. Decision Tree, Random Forest, XGBoost, and Neural Network all train much faster, with times ranging from about 10 to 17 seconds. Decision Tree is the quickest, while Random Forest and Neural Network have similar moderate training times. XGBoost is the fastest among the ensemble methods.

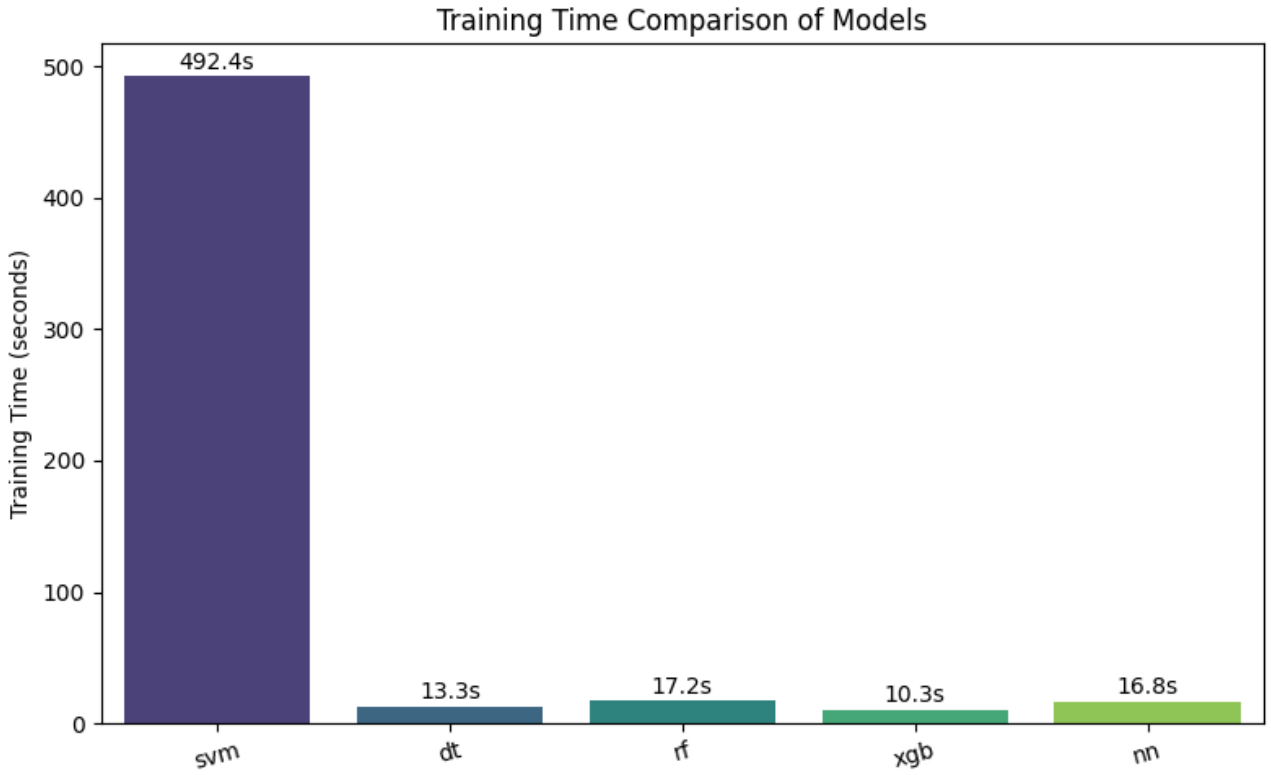


Figure 27:Training Time Comparison of Model.

This stark contrast highlights the computational cost of SVM in high-dimensional spaces, while ensemble methods and neural networks offer a better balance between speed and performance for practical applications.

3.3.2.4 BOW Resault Discussion :

Model	Accuracy	Precision (avg)	Recall (avg)	F1-Score (avg)	Notable Observations
SVM	0.87	0.86	0.87	0.86	Good balance; reliable performance; longest training time
Decision Tree	0.85	0.84	0.84	0.82	Fastest to train; more confusion in neutral class
Random Forest	0.86	0.86	0.87	0.86	Robust results; generalizes well; balanced classes
XGBoost	0.88	0.88	0.87	0.87	Best overall performance on BoW; handles neutral well
Neural Network	0.87	0.87	0.86	0.87	Smooth learning; balanced metrics; moderate training time

Table 6: Test Comparison of Models with BOW.

This analysis compared five popular models: SVM, Decision Tree, Random Forest, XGBoost, and a Neural Network—for Arabic sentiment classification using Bag of Words features. All models performed reasonably well, but a few stood out more clearly in terms of both accuracy and practical usability.

XGBoost, SVM, and the Neural Network achieved the highest accuracy scores (0.87–0.88) and consistently strong F1-scores across all sentiment classes. Among them, XGBoost delivered the best overall balance, especially in handling neutral and positive sentiments, which are often the hardest to distinguish. SVM also showed excellent performance, particularly for positive sentiment, but came with a significant drawback—training time. It took over 490 seconds to train, making it much slower than the other models. In contrast, Random Forest offered robust performance while training relatively quickly, striking a solid balance between accuracy and speed.

The Neural Network proved to be both effective and stable. Its learning curves show that it trained efficiently and generalized well, without overfitting. Though it wasn't the fastest model to train, its performance makes it a reliable option for tasks where deep learning advantages are desired.

On the other hand, the Decision Tree model was the fastest to train but lagged slightly in classification performance. It struggled more with neutral sentiment, often misclassifying it as either positive or negative, which limits its effectiveness in more nuanced real-world scenarios.

From a practical perspective, if speed and simplicity are your top priorities, Decision Tree or Random Forest are great options. If you're aiming for high accuracy and can afford a longer training time, SVM, XGBoost, or the Neural Network would be better choices. Ultimately, choosing the right model depends on balancing predictive performance with real-world constraints like training time and computational resources.

3.3.3 Word2Vec:

3.3.3.1 Word2ec in ML & DL:

- **SVM:** Accuracy: 0.77
  - **Classification Report:**
    - Negative (0): Precision 0.81, Recall 0.78, F1-score 0.79
    - Neutral (1): Precision 0.79, Recall 0.74, F1-score 0.76
    - Precision (2): 0.74, Recall 0.79, F1-score 0.76

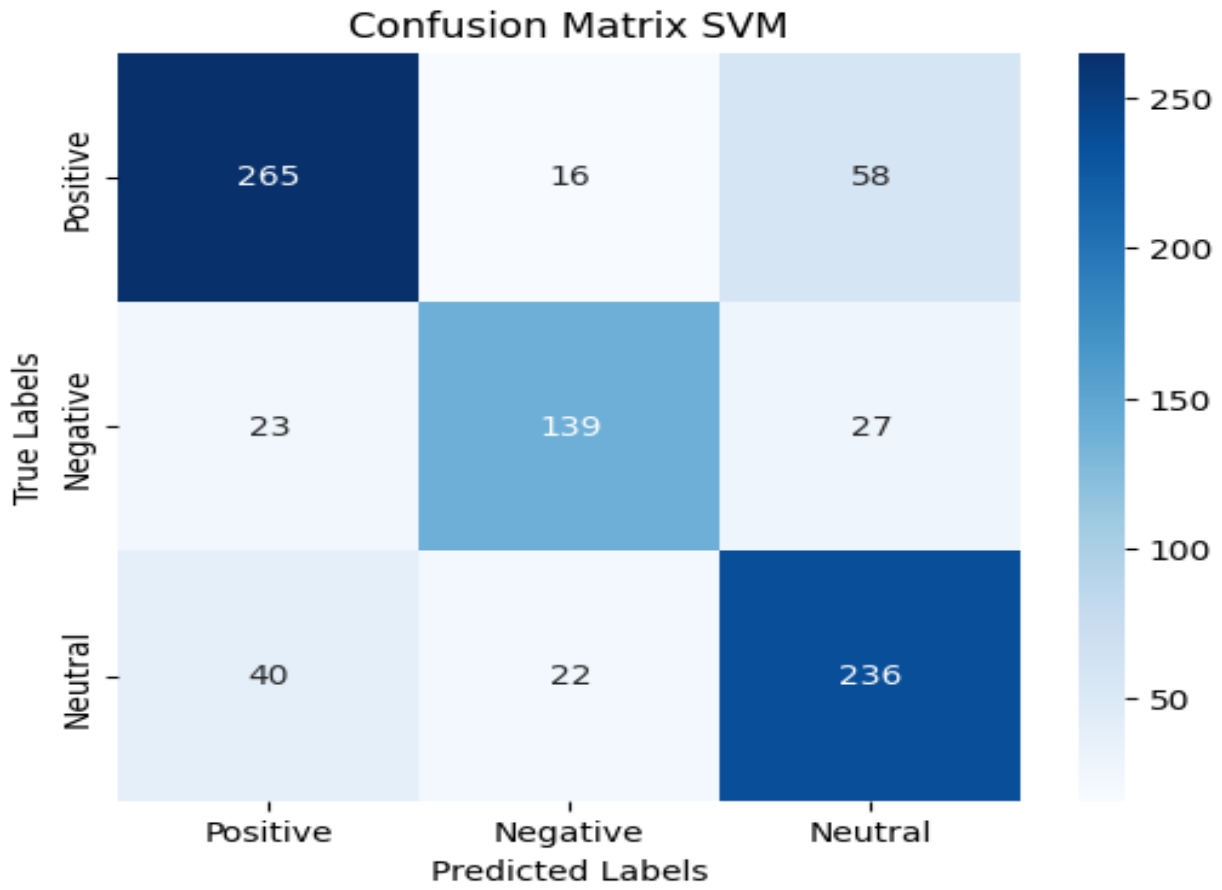


Figure 28:Confusion Matrix of SVM.

SVM with Word2Vec embeddings correctly predicts most cases but often confuses neutral with positive or negative sentiments. This highlights SVM's difficulty in distinguishing the neutral class.

- **Decision Tree:** Accuracy: 0.68

- **Classification Report:**
  - Negative (0): Precision 0.74, Recall 0.67, F1-score 0.70
  - Neutral (1): Precision 0.62, Recall 0.60, F1-score 0.61
  - Positive (2): Precision 0.66, Recall 0.74, F1-score 0.70

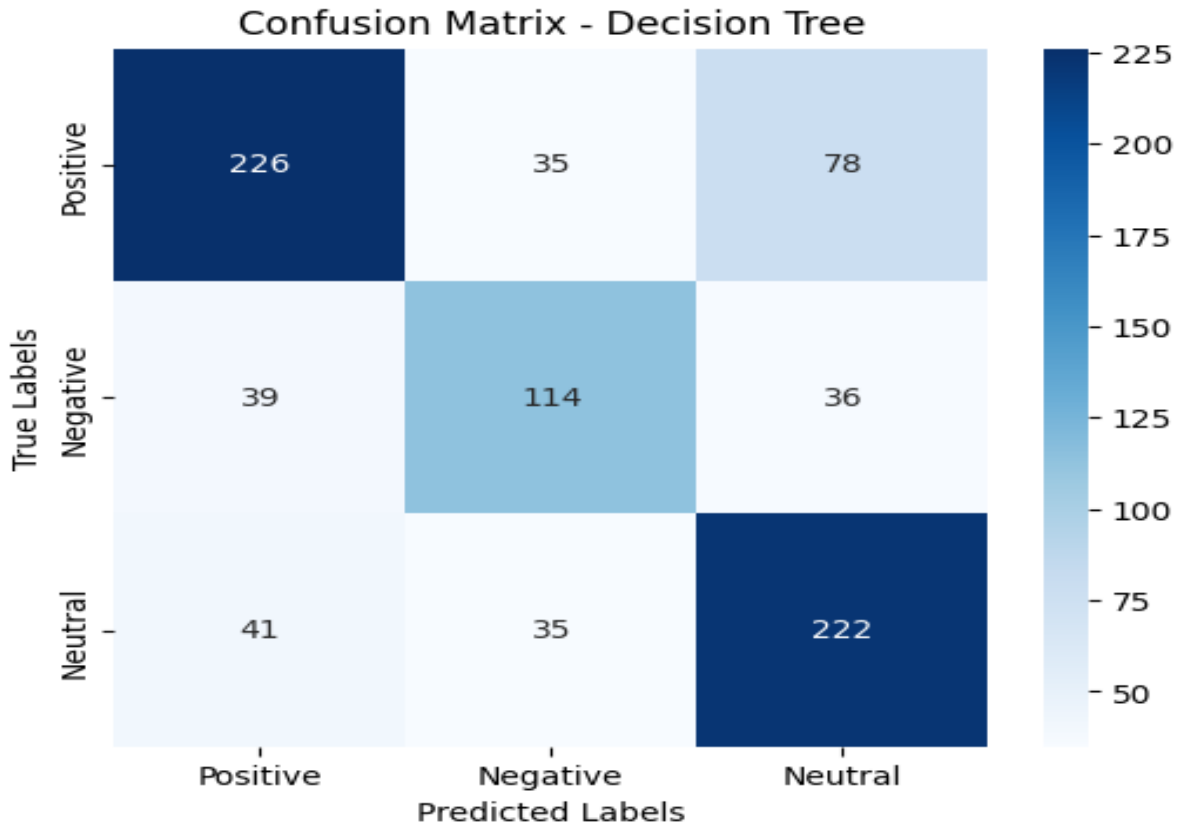


Figure 29:Confusion Matrix of Decision Tree.

The Decision Tree confusion matrix reveals frequent confusion between neutral and the other classes, especially neutral versus positive. This underscores the model's interpretability but also its limited ability to distinguish subtle differences in sentiment compared to more advanced models.

- **Random Forest:** Accuracy: 0.81
  - **Classification Report:**
    - Negative (0): Precision 0.86, Recall 0.82, F1-score 0.84
    - Neutral (1): Precision 0.87, Recall 0.70, F1-score 0.77

- Positive (2): Precision 0.74, Recall 0.87, F1-score 0.80

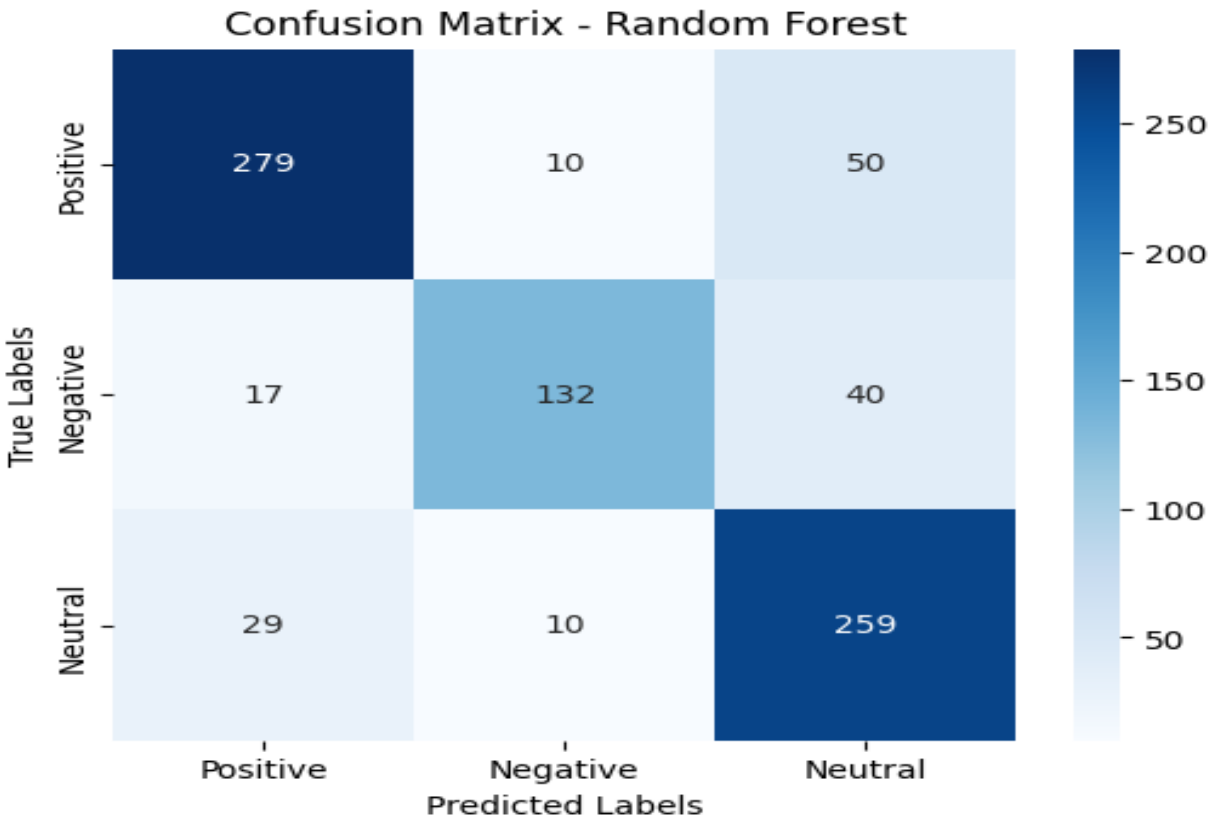


Figure 30:Confusion Matrix of Random Forest.

Random Forest achieves strong predictive performance with Word2Vec embeddings, correctly classifying most sentiments and showing only moderate confusion between neutral and other classes. It rarely confuses negative and positive sentiments, outperforming Decision Tree and SVM in both accuracy and class separation.

- **XGBoost:** Accuracy: 0.84
  - **Classification Report:**
    - Negative (0): Precision 0.87, Recall 0.86, F1-score 0.86
    - Neutral (1): Precision 0.86, Recall 0.76, F1-score 0.80
    - Positive (2): Precision 0.80, Recall 0.87, F1-score 0.83

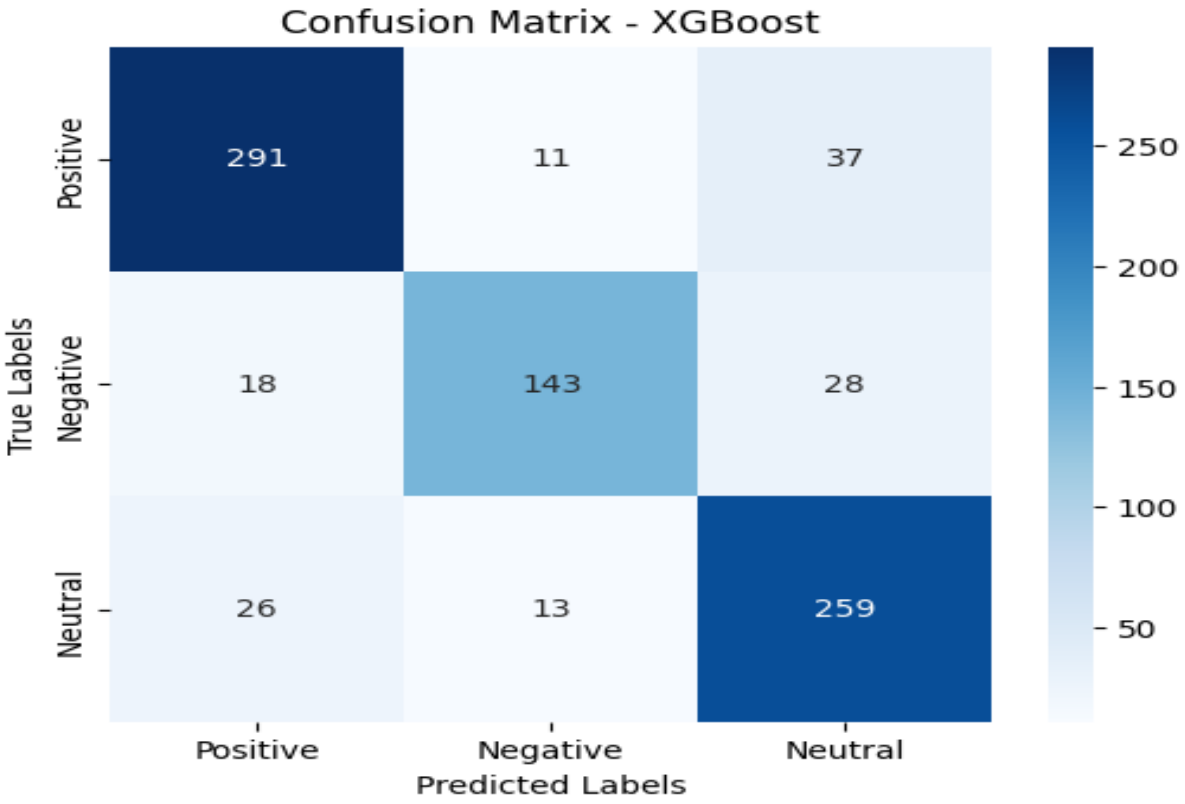


Figure 31:Confusion Matrix of XGBoost.

The XGBoost confusion matrix demonstrates strong alignment with true labels, with most errors involving confusion between neutral and the other classes—especially neutral being misclassified as positive. Errors between negative and positive are rare, highlighting XGBoost's effectiveness in distinguishing sentiment classes, though neutral remains the most challenging to separate.

- **Neural Network (DNN):** Accuracy: 0.85

  - **Final Classification Report:**

    - Negative (0): Precision 0.91, Recall 0.84, F1-score 0.87
    - Neutral (1): Precision 0.82, Recall 0.83, F1-score 0.83
    - Positive (2): Precision 0.82, Recall 0.89, F1-score 0.85

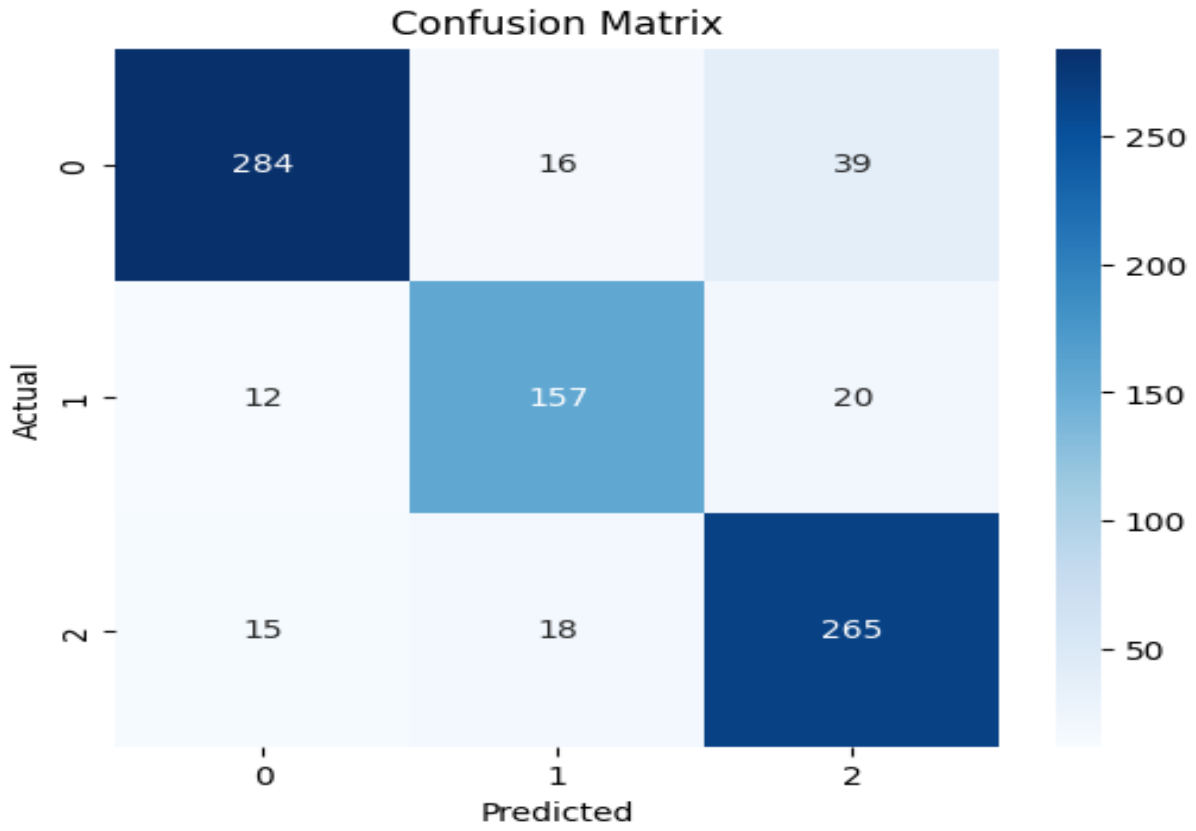


Figure 32:Confusion Matrix of DNN model.

The confusion matrix for the DNN model confirms its robust performance: most predictions are correct, with only moderate confusion between neutral and the other classes. The model rarely confuses negative and positive sentiments, indicating its effectiveness in distinguishing sentiment classes. This balanced performance across all categories demonstrates the DNN's suitability for sentiment analysis using Word2Vec embeddings.

- **B-LSTM:** Accuracy: 0.86

  - **Final Classification Report:**

    - Negative (0): Precision 0.93, Recall 0.83, F1-score 0.88
    - Neutral (1): Precision 0.82, Recall 0.85, F1-score 0.84
    - Positive (2): Precision 0.81, Recall 0.89, F1-score 0.85

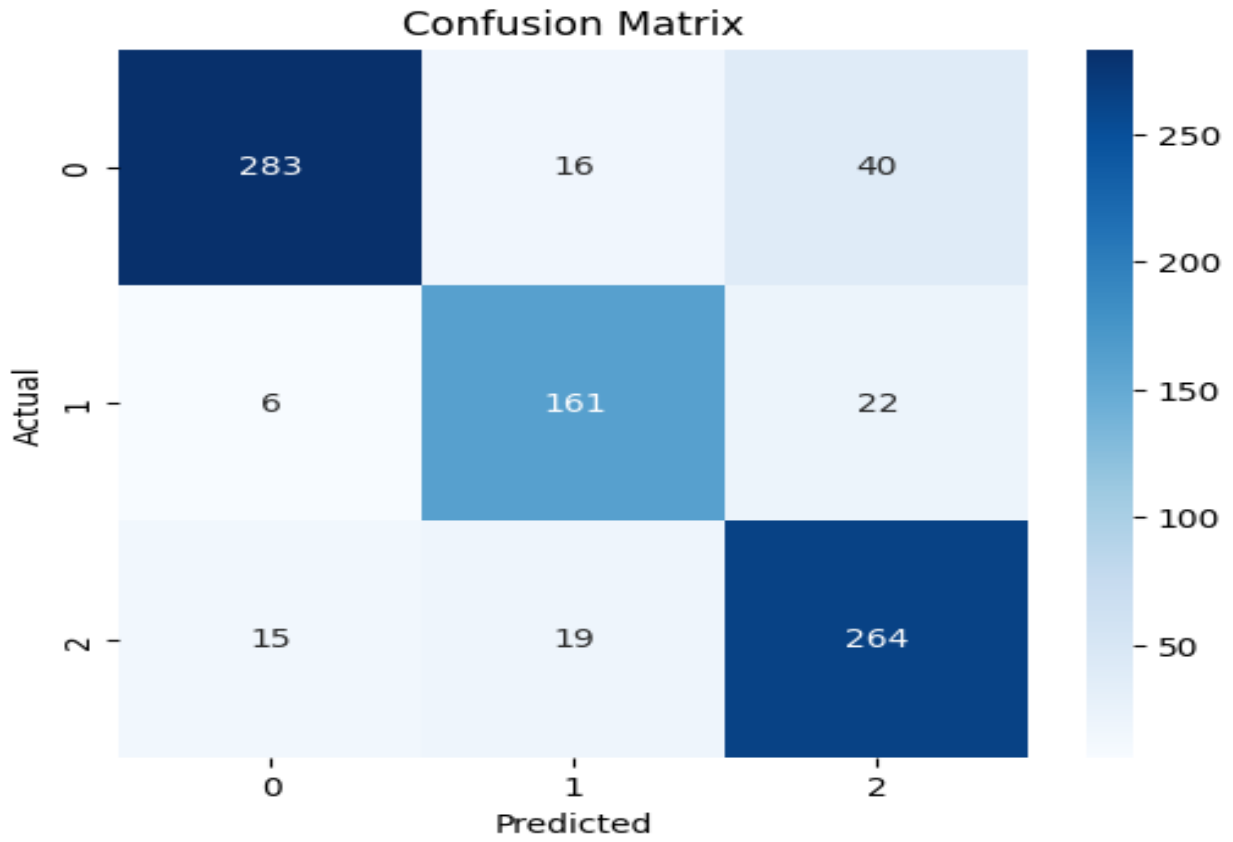


Figure 33:Confusion Matrix of B-LSTM Model.

The confusion matrix reveals that B-LSTM rarely confuses negative and positive sentiments, and handles neutral comments with greater accuracy than other models. This demonstrates the advantage of sequence modeling and context awareness in B-LSTM, making it the most effective approach for sentiment analysis using Word2Vec features in this study.

3.3.3.2 Learning Curves Summary for DNN &B-LSTM:

The following graph displays the training, validation, and test accuracy and loss curves for the Neural Network (DNN) & (B-LSTM) model using Word2Vec embeddings:

- **DNN**

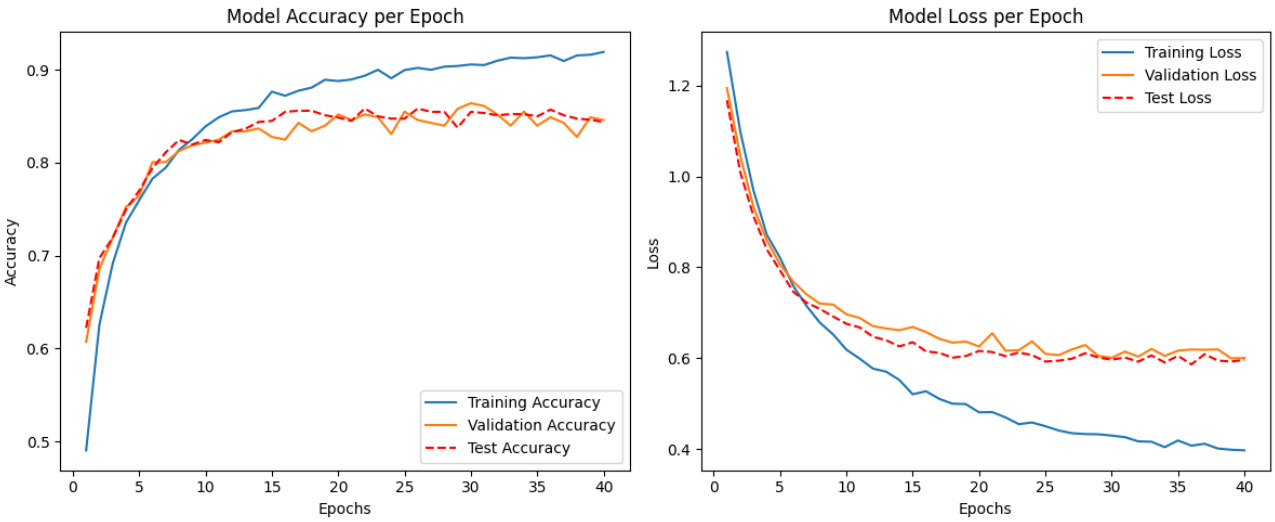


Figure 34:Model Accuracy and Model Loss per Epoch.

The DNN's accuracy and loss curves show rapid improvement followed by stable plateaus, with training, validation, and test metrics closely aligned. This indicates efficient learning, strong generalization, and no signs of overfitting or underfitting—making the model robust and reliable for sentiment classification using Word2Vec features.

- **B-LSTM**

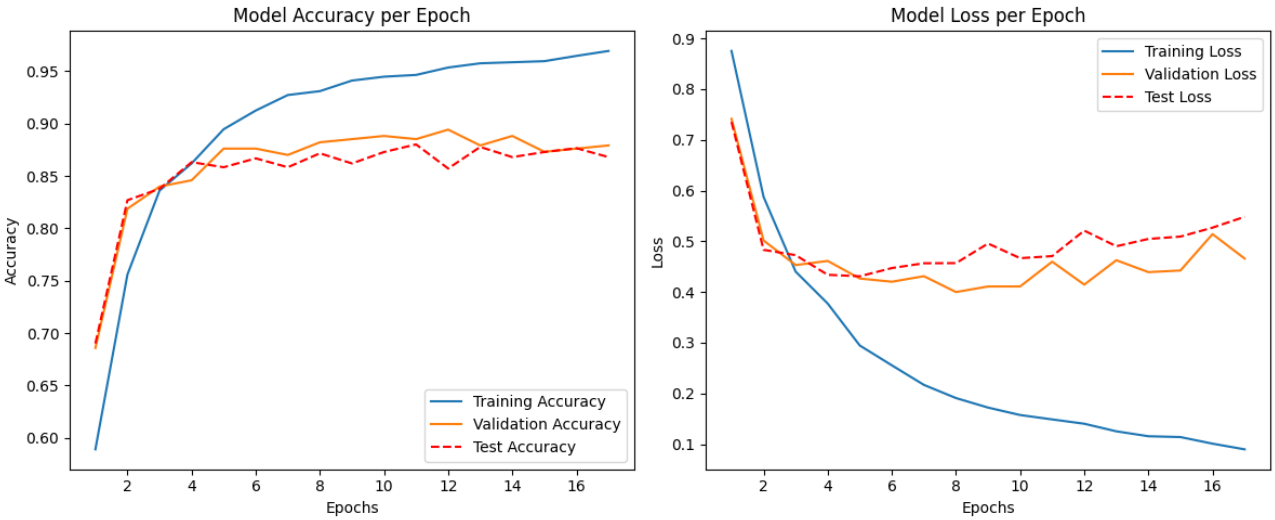


Figure 35:Model Accuracy and Model Loss per Epoch.

The B-LSTM's learning curves show rapid and stable improvement in both accuracy and loss for training, validation, and test sets. All curves plateau together, indicating the model learned efficiently and generalized well without overfitting. The close alignment of test and validation metrics demonstrates the B-LSTM's robustness and reliability for sentiment classification using Word2Vec embeddings. This confirms that sequence modeling with B-LSTM effectively captures contextual information, resulting in superior performance compared to other models.

3.3.3.3 Training Time Comparison:

This training time comparison for models below using Word2Vec embeddings is illustrated in the bar chart:

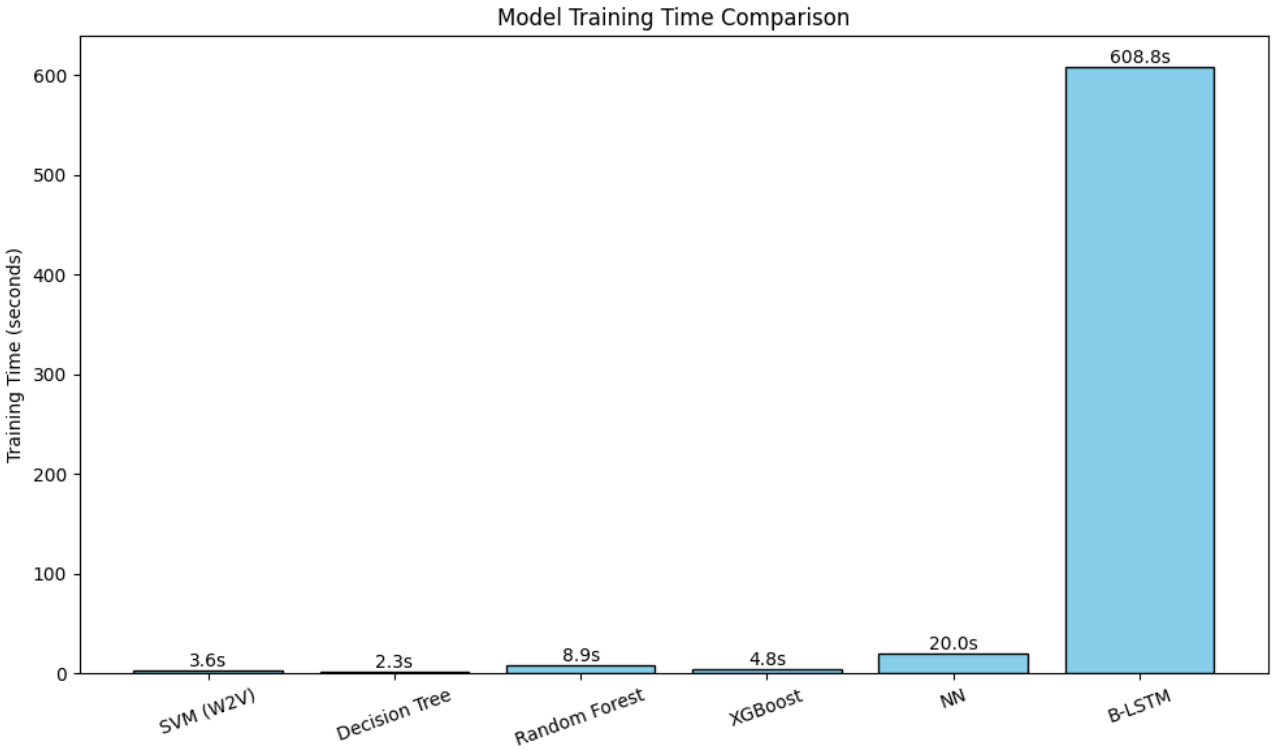


Figure 36:The Training Time comparaison for model.

The chart shows that traditional machine learning models—SVM, Decision Tree, Random Forest, and XGBoost—train relatively quickly, with times ranging from a few seconds to under a minute. The DNN model requires more time due to its increased complexity, but remains practical for most applications. In contrast, the B-LSTM model has the longest training time by a significant margin, reflecting the computational demands of sequence modeling and recurrent layers. This stark difference highlights the trade-off between model complexity and training efficiency: while B-LSTM delivers the best performance for maximum accuracy and contextual understanding, B-LSTM with Word2Vec embeddings is the optimal choice.

3.3.3.4 Word2Vector Result Discussion :

Model	Accuracy	Precision (avg)	Recall (avg)	F1-Score (avg)	Notable Observations
SVM	0.77	0.78	0.77	0.77	Lower performance; struggles with neutral class; fast training
Decision Tree	0.68	0.67	0.67	0.67	Weakest performance; high confusion in neutral sentiment
Random Forest	0.81	0.80	0.80	0.80	Good performance; moderate confusion; quick training
XGBoost	0.84	0.83	0.83	0.83	Strong results; handles neutral class better than other ML models
DNN	0.85	0.85	0.85	0.85	Balanced, robust; efficient learning curves; practical training time
B-LSTM	0.86	0.86	0.86	0.86	Highest accuracy for W2V; superior context handling; longest training time

Table 7: Test Comparison of Models with Word2Vector.

In this evaluation, six models—SVM, Decision Tree, Random Forest, XGBoost, DNN, and B-LSTM—were tested using Word2Vec embeddings (AraVec) to classify Arabic sentiments into negative, neutral, or positive. The results clearly show that deep learning models, particularly B-LSTM, outperform traditional machine learning approaches in both accuracy and handling subtle sentiment variations.

Among all models, B-LSTM achieved the highest accuracy (86%) and demonstrated a strong ability to distinguish between all sentiment categories, including the often-ambiguous neutral class. This performance reflects the model's strength in capturing context and word order—an advantage made possible by its sequence modeling capabilities. Similarly, the DNN model also delivered balanced and robust performance, coming close to B-LSTM while requiring less training time.

XGBoost and Random Forest followed closely in terms of accuracy, showing strong performance with quicker training times, making them appealing for real-time or resource-constrained scenarios. SVM and Decision Tree, while simpler and faster, struggled more with classifying neutral sentiments and had overall lower accuracy. When it comes to training time, the gap becomes more pronounced. Traditional models like Decision Tree, Random Forest, and XGBoost trained quickly—sometimes in

seconds—while DNN required more time but remained efficient. On the other hand, B-LSTM was by far the most computationally demanding, requiring substantially more time to train due to its recurrent architecture.

Ultimately, if maximum performance and contextual understanding are your top priorities, B-LSTM with Word2Vec is the best choice. However, if faster training or lower resource usage is essential, models like Random Forest or XGBoost offer a compelling trade -off between speed and accuracy. This comparison highlights that the best model depends not only on raw performance but also on the practical constraints and goals of the application.

3.4 Comparative Summary of Embedding Techniques and Model Performance:

A comparison of three word embedding techniques—TF-IDF, Bag of Words (BoW), and Word2Vec (AraVec)—combined with various machine learning and deep learning models, reveals distinct performance patterns.

For TF-IDF and BoW, ensemble models such as Random Forest and XGBoost consistently demonstrated strong performance, achieving accuracies around 88%, with robust precision and recall across all sentiment classes, but the SVM model achieved the highest accuracy at 89%.

The Neural Network (DNN) also performed competitively, particularly with BoW, showing balanced generalization and minimal overfitting. When using Word2Vec embeddings, deep learning models clearly outperformed traditional ones, with B-LSTM achieving the highest accuracy (86%) and best handling of contextual sentiment, especially for the often-confused neutral class.

While SVM and Decision Tree showed reasonable performance on TF-IDF and BoW, they lagged behind on Word2Vec due to limited contextual understanding. In terms of training time, traditional models trained fastest, while B-LSTM required significantly more time due to its sequential processing. Overall, the best model per technique is XGBoost for TF-IDF and BoW, and B-LSTM for Word2Vec.

Technique	Top Accuracy	Chosen Best Model	Why?
TF-IDF	**SVM (89%)**	**XGBoost (88%)**	Slightly lower accuracy, but faster, more robust, and consistent.
BoW	**XGBoost/Neural Net (88%)**	**XGBoost**	Strong performance, fast training, better generalization.
Word2Vec	**B-LSTM (86%)**	**B-LSTM**	Best contextual understanding, highest scores across metrics.

Table 8:Comparative Summary of Feature Extraction Techniques and Model Performance.

Taking into account both performance and linguistic representation, Word2Vec with B-LSTM stands out as the most effective combination, offering the best accuracy and contextual sensitivity, despite its higher computational cost.

4 Conclusion:

This chapter detailed the implementation and evaluation of multiple sentiment analysis models for Arabic dialects, using three distinct feature extraction techniques: TF-IDF, Bag of Words (BoW), and Word2Vec (AraVec). We outlined the development environment, tools, libraries, and provided a thorough justification for hyperparameter selection and model configurations.

Through rigorous experimentation, we compared traditional machine learning models (SVM, Decision Tree, Random Forest, XGBoost) and deep learning models (DNN, B-LSTM) across different embedding techniques. The results demonstrate that the choice of both embedding technique and model architecture significantly impacts performance.

For TF-IDF and BoW, ensemble methods such as Random Forest and XGBoost, along with SVM, delivered strong performance, achieving accuracies up to 89%. The Neural Network models also showed stable learning curves with minimal overfitting. For Word2Vec embeddings, deep learning approaches, particularly the B-LSTM model, outperformed traditional methods by effectively capturing contextual information and word order, which is critical for morphologically rich languages like Arabic.

Moreover, the comparative analysis highlighted the consistent challenge of accurately classifying neutral sentiment, underscoring the need for more advanced embeddings or context-aware models in future work. Training time was also shown to be an important factor, with B-LSTM delivering superior performance at the expense of higher computational cost, while models like Random Forest and XGBoost offered a practical trade-off between accuracy and efficiency.

In summary, this chapter's findings provide clear guidance on selecting suitable models and embedding techniques for Arabic sentiment analysis, demonstrating that combining deep contextual embeddings with sequence models offers significant performance advantages. These insights serve as a foundation for future improvements and motivate further research into more advanced language models, larger datasets, and real-world applications.

General Conclusion

This study provides a comprehensive evaluation of Arabic sentiment analysis using a range of feature extraction techniques and classification models. By systematically comparing traditional machine learning algorithms and deep learning architectures across TF-IDF, Bag of Words (BoW), and Word2Vec (AraVec) embeddings, we have demonstrated how the choice of representation and model architecture directly impacts sentiment classification performance for Arabic text.

Our experimental results show that traditional models like SVM, Random Forest, and XGBoost perform strongly with TF-IDF and BoW features, achieving accuracies up to 89%, while deep learning models, particularly the B-LSTM with Word2Vec embeddings, excel in capturing contextual and sequential information, making them highly effective for morphologically rich languages like Arabic. Notably, the B-LSTM achieved the best performance when deeper context was required, despite its higher computational cost. Meanwhile, ensemble methods such as Random Forest and XGBoost offered robust results and a practical trade-off between accuracy and training time, making them well-suited for real-world applications with limited resources.

A key insight from this research is the persistent challenge in accurately classifying neutral sentiment, which often overlaps with negative and positive classes. This highlights the limitations of static word embeddings and emphasizes the potential benefits of more advanced context-aware representations.

This work also sheds light on the practical aspects of deploying sentiment analysis systems for under-resourced languages: hardware constraints, training time, and the availability of high-quality dialectal data all play a significant role in model selection. The comprehensive comparison and detailed configuration guidelines presented here can serve as a practical reference for researchers and practitioners working on Arabic or similar languages with complex morphology and dialectal variations.

However, the study is not without limitations. The models were evaluated on a single dataset; expanding to larger and more diverse datasets would improve generalizability. Moreover, while the use of static embeddings like Word2Vec (AraVec) provided valuable insights, exploring more advanced contextual embeddings such as AraBERT, multilingual BERT, or other transformer-based architectures could further enhance performance by better handling semantic nuances and polysemy in Arabic text.

General Conclusion

Future research should focus on addressing these limitations by integrating domain adaptation techniques, testing on dialect-specific corpora, and investigating low-resource strategies such as transfer learning or few-shot learning. Combining contextual embeddings with advanced sequence models or transformer-based architectures holds promise for achieving more accurate and robust sentiment analysis in Arabic and other low-resource languages.

In summary, this study contributes a solid empirical foundation and practical recommendations for developing effective Arabic sentiment analysis systems. It demonstrates that a thoughtful combination of appropriate embeddings and model architectures, along with a clear understanding of computational trade-offs, is essential to achieve high performance in real-world applications.

Bibliography

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

[2] Arwa A. Al Shamsi, Sherief Abdallah. (2021). "Text Mining Techniques for Sentiment Analysis of Arabic Dialects: Literature Review", *Advances in Science, Technology and Engineering Systems Journal*, 6(1), 1013.

[3] Yanying Mao, Qun Liu, Yu Zhang. (2024). "Sentiment analysis methods, applications, and challenges: A systematic literature review". *Journal of King Saud University - Computer and Information Sciences*, 36(5).

[4] Naaima Boudad, Rdouan Faizi, Rachid Oulad Haj Thami. (2018). "Sentiment analysis in Arabic: A review of the literature". *Ain Shams Engineering Journal*, 9, 2481–2482.

[5] Ziad Doughan, Sari Itani, Samir Itani. (2025). "ArabSis: Arabic Corpus Sentiment Analysis". *IEEE Access*, 13, 81084.

[6] Cutler, A., Cutler, D. R., & Stevens, J. R. (2001). "Random forests". *Machine Learning*, 45(1), 157–176.

[7] Lulu, L., & Elnagar, A. (2018). "Automatic Arabic Dialect Classification Using Deep Learning Models". *Procedia Computer Science*, 142, 262–269.

[8] Pang, B., & Lee, L. (2004). "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts". *ACL*.

[9] Nassar, N., & Sezer, E. A. (2021). "Sentiment Analysis of Arabic Text Using Machine Learning Techniques". *International Journal of Advanced Computer Science and Applications*, 12(4), 565–571.

[10] Hu, M., & Liu, B. (2004). "Mining and Summarizing Customer Reviews". *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 168–177.

[11] Bouziane, A., Bouougada, B., Bouchiha, D., & Doumi, N. (2024). *Sentiment analysis of Algerian Arabic dialect on social media Using Bi-LSTM recurrent neural networks*. Journal of Engineering and Exact Sciences, 10(7), 20058.

[12] Mazari, A., & Djeffal, A. (2022). "Sentiment Classification in Algerian Dialect Using Hybrid Machine Learning and Word Embeddings". *International Journal of Computing and Digital Systems*, 11(1), 1–12.

[13] Benmounah, A., et al. (2023). "Sentiment Analysis of Algerian Dialect using DziriBERT: A Deep Learning Approach". *Conference on Arabic Language Processing (ALP)*.

[14] Dataset: **https://doi.org/10.48550/arXiv.2110.01948**

[15] Python: https://www.python.org/doc/essays/blurb/

Bibliography

[16] Visual Studio code:https://code.visualstudio.com/Search?q=what+is+visualstudio+code

[17] NumPy:https://numpy.org/doc/stable/user/whatisnumpy.html

[18] Pandas :https://pandas.pydata.org/

[19] Karas: https://keras.io/

[20] Tensorflow :https://www.tensorflow.org/learn

[21] Seaborn: https://seaborn.pydata.org/

[22] NLTK:https://www.nltk.org/