Algerian Republic Democratic and Popular
Ministry of Higher Education and Scientific Research

UNIVERSITY MUSTAPHA STAMBOULI
OF MASCARA

Faculty of Exact Science
Department of Computer Science

DOCTORAL LMD THESIS

Machine Learning Tools In Sentiment Analysis of Arabic Social Media

---

Author:
Laouni MAHMOUDI

Supervisor:
Pr. Mohammed SALEM

Submitted in fulfillment of the requirements for the degree of Doctorat LMD

Thesis Jury

Institution	Grade	Name	Role
University of MASCARA	MCA	Rochdi BACHIR BOUIADJRA	Chair
University of MASCARA	Professor	Mohammed SALEM	Supervisor
University of Granada, Spain	Professor	Antonio MORA GARCIA	Co-Supervisor
University of MASCARA	MCA	Mohammed DEBAKLA	Examiner
University of Saida	MCA	Mohamed Elhadi Rahmani	Examiner

June 8, 2023

UNIVERSITY MUSTAPHA STAMBOULI
OF MASCARA

Abstract

Faculty Name
Department of Computer Science

Doctorat LMD

Machine Learning Tools In Sentiment Analysis of Arabic Social Media

by Laouni MAHMOUDI

This thesis focuses on sentiment analysis on Arabic social media, which is a challenging task due to the complex and nuanced nature of Arabic language. To address this challenge, the thesis proposes several contributions, including the development of a new Arabic word embedding model, enhancements to the BERT model for imbalanced text classification.

The proposed Arabic word embedding model is designed to capture the unique semantic relationships and nuances of Arabic language, which can improve the accuracy of sentiment analysis. The enhancements to the BERT model include modifications to the attention mechanism and training process, which can improve the performance of the model on Arabic text data. The inclusion of a balancing layer in BERT approaches is aimed at addressing the issue of imbalanced data, which is common in sentiment analysis tasks.

The thesis presents experimental results that demonstrate the effectiveness of the proposed contributions in improving the accuracy and performance of sentiment analysis on Arabic social media. The proposed model outperforms the BERT baseline and achieves state-of-the-art results on several benchmark datasets.

Keywords: NLP, Text Classification, Sentiment Analysis, Social Media, Arabic language, Word embedding, Balancing

Acknowledgements

I would like to begin by expressing my heartfelt gratitude to the members of the jury for their invaluable feedback and support throughout the evaluation process. I extend my deepest appreciation to them.

- Dr.Rochdi BACHIR BOUIADJRA (Chair) from the University of MASCARA
- Pr.Mohammed SALEM (Supervisor) from the University of MASCARA
- Pr.Antonio MORA GARCIA (Examiner) from the University of Granada, Spain
- Dr.Mohammed DEBAKLA (Examiner) from the University of MASCARA
- Dr.Mohamed Elhadi Rahmani (Examiner) from the University of Saida

In particular, I would like to extend a special thank you to my supervisor : **Pr.Mohammed SALEM** , whose guidance, support, and mentorship have been invaluable throughout my thesis research. Their expertise and insights have been instrumental in helping me to successfully complete my research and contribute to the field.

I would like to take this opportunity to express my heartfelt gratitude to the University of Mascara for giving me a second chance in my academic career. I am also grateful to the faculty and staff of the Department of Computer Science for their unwavering support, guidance, and encouragement throughout my academic journey. Their expertise, knowledge, and dedication have been instrumental in helping me to develop the skills and knowledge necessary to succeed in the field of computer science.

Finally, I would like to express my appreciation to my family and friends for their love, support, and encouragement throughout my academic journey. Their unwavering belief in my abilities has been a constant source of motivation and inspiration, and I am grateful for their presence in my life.

Once again, I would like to express my sincere gratitude to the University of Mascara, the Department of Computer Science, and my supervisor for their support and guidance throughout my academic journey.. . .

Dedicated to the loving memory of my dear parents. Though they are no longer with me, their unwavering love and support continue to inspire and guide me each day. This thesis is a testament to their enduring legacy and the invaluable lessons they imparted upon me. I am forever grateful for the sacrifices they made to provide me with the opportunities I have today.. . .

Contents

List of Figures

List of Algorithms

List of Tables

List of Abbreviations

ALBERT	**A** **L**ite **BERT** **L**earning of **L**anguage **R**epresentations
BERT	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
BoW	**B**ag of **W**ord
CNN	**C**onvolutional **N**eural **N**etwork
GloVe	**G**lobal **V**ector **R**Representation
GRU	**G**ated **R**ecurrent **U**nit
GPT	**G**enerative **P**re-trained **T**ransformer
KNN	**K**-**N**earest **N**eighbors
LR	**L**ogistic **R**egression
LSTM	**L**ong **S**hort-**T**erm **M**emory
ML	**M**achine **L**earning
NLP	**N**atural **L**anguage **P**rocessing
NB	**N**aive **B**ayes
RF	**R**andom **F**orest
RNN	**R**ecurrent **N**eural **N**etwork
RoBERTa	**Ro**bustly **BERT** **A**pproach
SA	**S**entiment **A**nalysis
SVM	**S**upport **V**ector **M**achine
T5	**T**ext-to-**T**ext **T**ransfer **T**ransformer
TF	**T**erm **F**requency
TF-IDF	**T**erm **F**requency-**I**nverse **D**ocument **F**requency
Undersampling	
TL	**Tomek Links**
ENN	**E**uclidean **N**earest **N**eighbors
OSS	**One-Sided Selection**
CNN	**Condensed Nearest Neighbors**
Oversampling	
ROS	**R**andom **O**versampling
SMOTE	**S**ynthetic **M**inority **O**versampling **T**echnique
ADASYN	**A**daptive **S**ynthetic **S**ampling **M**ethod
B-SMOTE	**Borderline SMOTE**
Hybrid	
SMOTE-ENN	**SMOTE** combined with **ENN**
SMOTE-Tomek	**SMOTE** combined with **Tomek Links**
ROS-RUS	**Random Oversampling and Random Undersampling**

Chapter 1

introduction

1.1 Background

Natural Language Processing (NLP) (Mahmoudi2022) is a multidisciplinary field that focuses on developing algorithms and tools for processing, analyzing, and understanding human language. In recent years, there has been a growing interest in NLP techniques for analyzing Arabic text data, especially for sentiment analysis. Sentiment analysis(Zhang2019) aims to identify and extract subjective information, such as opinions and emotions, from text data. This information can be used to understand public sentiment and opinions on various topics, such as politics, entertainment, and social issues.

The rapid growth of social media platforms in the Arab world has led to an increased interest in sentiment analysis on Arabic social media. With platforms like Twitter, Facebook, and Instagram being extensively used, there is a vast amount of user-generated content available for analyzing public sentiment and opinions. Sentiment analysis, also known as opinion mining, is a branch of natural language processing that focuses on automatically extracting subjective information, such as sentiment, emotion, and opinion, from text data. By analyzing and categorizing text into different categories like positive, negative, or neutral, sentiment analysis provides valuable insights into public sentiment on various topics. It plays a crucial role in diverse applications such as brand monitoring, reputation management, market research, and political analysis. However, analyzing sentiment in Arabic text data presents unique challenges due to the complexity of the Arabic language and the use of dialects and slang. As a result, there is a growing need to develop effective NLP techniques that can handle these challenges and improve the accuracy and reliability of sentiment analysis on Arabic social media(Alotaibi2022).

One of the challenges in developing NLP techniques for analyzing Arabic text data is the problem of representation. Conventional methods such as bag-of-words and n-gram models have limitations in capturing the semantic relationships between words, and can result in high-dimensional and sparse representations. To address this issue, researchers have explored the use of word embeddings such as GloVe, which have shown promise in improving representation by capturing semantic relationships between words in a low-dimensional vector space. However, the effectiveness of these methods on Arabic text data has not been thoroughly investigated, and there is a need to explore new approaches for representing Arabic text that can improve the accuracy and robustness of NLP models.

NLP techniques for sentiment analysis on Arabic social media present unique challenges due to the complexity of the Arabic language and the use of dialects and slang. There is a growing need to develop effective NLP techniques that can handle these challenges and improve the accuracy and reliability of sentiment analysis on

Arabic social media. Moreover, the problem of representation is a significant challenge that needs to be addressed to improve the accuracy and robustness of NLP models for Arabic text data.

1.2 Research Problem

Machine learning algorithms often face the challenge of effectively representing textual data in a way that captures its meaning and context. Traditional methods such as bag-of-words and n-gram models have limitations in capturing the semantic relationships between words, leading to high-dimensional and sparse representations (Patil2022). To address this issue, word embeddings such as GloVe have shown promise in improving representation by capturing semantic relationships between words in a low-dimensional vector space. However, the effectiveness of such techniques on Arabic text data remains understudied. Hence, there is a need to explore new approaches for representing Arabic text that can improve the accuracy and robustness of NLP models.

Another major challenge in text classification is dealing with imbalanced datasets (Wang2019), where the number of examples in each class significantly differs. This can lead to biased models that perform poorly on minority classes, which are often the ones of interest. This problem is particularly common in sentiment analysis tasks, where the distribution of sentiment classes can be skewed towards positive or negative. Several techniques have been proposed to address this issue, including resampling techniques, cost-sensitive learning, and deep learning-based approaches. Therefore, there is a need to develop effective strategies to deal with imbalanced datasets and enhance the performance of text classification models, especially in the context of sentiment analysis on Arabic social media.

In this context, this thesis proposes a novel representation of Arabic text data by combining GloVe embeddings with a Bag of Roots technique. This representation captures both surface-level and semantic information, leading to better performance in sentiment analysis when compared to existing methods. The proposed approach has the potential to enhance the accuracy and robustness of NLP models by addressing the problem of representation.

Furthermore, this thesis proposes a novel solution to address the problem of imbalanced datasets in text classification by incorporating a balancing layer into the BERT architecture. The balancing layer is incorporated after the embedding step, which improves the models' performance, and after the BERT representation step, which enhances the results further. The proposed approach demonstrates that deep learning models can effectively handle imbalanced data, leading to enhanced performance on minority classes. The approach has the potential to be applied to various text classification tasks and contribute to the development of more effective models in NLP.

1.3 Objectives and Scope

The main objective of this thesis is to develop a new representation for Arabic text data that can enhance text classification performance, particularly in the domain of sentiment analysis on social media. The proposed representation will aim to capture both surface-level and semantic information, and overcome the limitations of conventional methods such as bag-of-words and n-gram models. To achieve this,

we will investigate the effectiveness of combining GloVe representations with a Bag of Roots technique. The Bag of Roots technique is a novel approach that extracts morphological roots from Arabic words and uses them to capture the underlying semantic structure of the text. We hypothesize that combining this technique with GloVe representations can result in a more effective and robust representation of Arabic text data.
In addition, we aim to address the problem of imbalanced datasets in text classification by improving the performance of deep learning models. Imbalanced datasets can lead to biased models that perform poorly on minority classes. To address this problem, we propose to enhance the BERT architecture by adding a balancing layer that can effectively deal with imbalanced datasets. We will evaluate two balancing techniques: oversampling and undersampling. Oversampling involves replicating examples from the minority class, while undersampling involves randomly selecting examples from the majority class. We hypothesize that adding a balancing layer to the BERT architecture can improve the performance of deep learning models on imbalanced datasets, particularly in the context of sentiment analysis on Arabic social media. The proposed approach can contribute to the development of more accurate and reliable NLP models for analyzing Arabic text data on social media platforms.

1.4 Contributions

In this thesis, we make three main contributions. Firstly, we introduced a new method to represent Arabic text data that combines GloVe embeddings with a Bag of Roots technique. The use of GloVe embeddings enables the representation to capture semantic information, while the Bag of Roots technique enhances the model's ability to capture surface-level features. We evaluated The approach on a sentiment analysis task and demonstrated that the proposed method outperforms existing approaches. Specifically, the proposed method achieved higher accuracy, precision, recall, and F1 score than other methods. The results suggest that this new representation has the potential to be applied to other Arabic NLP tasks and improve their performance.

This new representation is particularly significant for Arabic text data as it presents unique challenges due to the morphology and complexity of the language. The Bag of Roots technique addresses these challenges by breaking down the text into its roots and considering them as separate entities. This approach is especially effective for Arabic text, where a single word can have multiple roots. By considering the roots of each word in the text, the model is able to capture a greater range of surface-level features. The combination of this approach with GloVe embeddings allows for the capturing of both surface-level and semantic information, resulting in a more comprehensive representation of the text.
The two other contributions of our thesis addresse the issue of imbalanced datasets in text classification. In real-world scenarios, it is common to encounter datasets with a severe imbalance between classes, where some classes have significantly fewer samples than others. This results in a bias towards the majority class during training, leading to poor performance on the minority classes. To tackle this issue, we propose a novel solution by incorporating a balancing layer into the BERT architecture.

The balancing layer is added after the embedding step, which enhances the model's ability to handle imbalanced data. It generates a weight for each class based on its

frequency in the training data, which is then used to reweight the loss function during training. By incorporating this layer, we demonstrate that deep learning models can effectively handle imbalanced data, leading to improved performance on minority classes. In addition, we also investigate the performance of the balancing layer when it is added after the BERT representation step, which further enhances the results.

Our proposed approach has the potential to be applied to various text classification tasks beyond sentiment analysis. By addressing the issue of imbalanced datasets, it can contribute to the development of more effective models in NLP. This approach also highlights the importance of considering the class distribution in the training data, as it can significantly affect the model's performance. Our thesis provides new insights and methods for improving the performance of text classification models on imbalanced datasets.

Overall, in this thesis, we make three main contributions.

1. **Introduction of a new method for representing Arabic text data**: We propose a novel approach that combines GloVe embeddings with a Bag of Roots technique to represent Arabic text. By leveraging GloVe embeddings, our representation captures semantic information, while the Bag of Roots technique enhances the model's ability to capture surface-level features. We evaluate our approach on a sentiment analysis task and demonstrate its superiority over existing methods. The proposed method achieves higher accuracy, precision, recall, and F1 score, indicating its potential for improving performance in other Arabic NLP tasks.

2. **Tackling imbalanced datasets in text classification**: Imbalanced datasets are common in real-world scenarios, where some classes have significantly fewer samples than others. This leads to biased training and poor performance on minority classes. To address this issue, we propose a novel solution by incorporating a balancing layer into the BERT architecture. The balancing layer generates weights for each class based on their frequency in the training data and reweights the loss function during training.
   a- Integrating this layer After Embedding: We demonstrate that deep learning models can effectively handle imbalanced data, leading to improved performance on minority classes.
   b- Integrating this layer After BERT representation: We also investigate the performance of the balancing layer when added after the BERT representation step, further enhancing the results.

1.5 Summary of Remaining Chapters

The following is a summary of the remaining chapters in this thesis:

- **Chapter 2: Sentiment Analysis on Social Media** This chapter will introduce social media and its impact on sentiment analysis. It will also address the challenges of sentiment analysis in Arabic social media and explore various techniques used in sentiment analysis.

- **Chapter 3: Text Representation in NLP** This chapter will examine different approaches to text representation in NLP, including word embeddings, bag-of-words, and TF-IDF. Additionally, it will cover the advantages and limitations of each method and their applications in sentiment analysis of Arabic text.
- **Chapter 4: A New Arabic Word Embeddings Representation for SA** In this chapter, we will present our proposed method for developing new Arabic word embeddings based on GloVe and Roots.
- **Chapter 5: Balancing Approaches** This chapter will discuss the challenges of imbalanced datasets in sentiment analysis and explore various balancing techniques such as oversampling, undersampling, and data augmentation.
- **Chapter 6: Improving Multi-Class Text Classification on Imbalanced Datasets** Chapter 5 will detail our two proposed approaches for improving imbalanced text sentiment analysis, including the integration of new Arabic word embeddings and balancing techniques with BERT.
- **Chapter 7: Conclusion** Finally, Chapter 6 will provide a summary of the main contributions of this thesis, discuss potential areas for future research, and address limitations of our proposed approaches.

Chapter 2

Sentiment Analysis on Social Media

2.1 Introduction

Social media has become an integral part of our daily lives, providing a platform for people to share their thoughts, opinions, and experiences (Ellison2011). The vast amount of user-generated content on social media has created a unique opportunity for businesses to gain insights into customer behavior and sentiment.
Sentiment analysis is a powerful tool that allows businesses to analyze and understand the emotional tone of social media posts and comments. Natural Language Processing (NLP) techniques are used to identify and extract subjective information from text data, enabling businesses to gain valuable insights into customer sentiment and behavior (Zhang2019).
The benefits of sentiment analysis of social media are numerous. By monitoring social media sentiment, businesses can improve the customer experience by quickly identifying and responding to negative feedback, protecting their brand reputation and increasing customer satisfaction and loyalty.
Sentiment analysis can also provide businesses with marketing insights, helping them to refine their marketing strategies and improve overall business performance. Additionally, sentiment analysis can be used for competitive intelligence, providing businesses with valuable insights into their competitors' strengths and weaknesses (Gao2021).
To successfully implement sentiment analysis, businesses must first identify the social media channels and platforms that are relevant to their industry and target audience. They must then develop a strategy for monitoring and analyzing social media sentiment, using tools such as sentiment analysis software and social media monitoring platforms.
However, there are also challenges associated with sentiment analysis of social media. For instance, sentiment analysis can be impacted by the nuances of language, slang, sarcasm, and cultural differences. As such, it is important for businesses to continually refine their sentiment analysis techniques to ensure accuracy and reliability (Smailović2021).
In conclusion, sentiment analysis of social media is a powerful tool that allows businesses to gain insights into customer behavior and sentiment, helping them to make data-driven decisions and improve overall business performance. While there are challenges associated with sentiment analysis, the benefits far outweigh the risks, making it an essential tool for businesses of all sizes.

2.2 Social media

Social media platforms are websites and mobile applications that allow users to create and share content with other users. These platforms have become a central part of modern communication, with billions of users around the world using them every day. Here are some of the most popular social media platforms:

Facebook - Facebook is the largest social media platform, boasting over 2.8 billion monthly active users worldwide. The platform allows users to create profiles, share updates, photos, and videos, and connect with friends and family. Users can create posts on their profiles and share them with their friends, who can react to and comment on them. The platform also offers features such as groups and pages, which allow users to connect with others who share similar interests or causes.

Facebook has become an integral part of many people's lives, serving as a platform for communication, entertainment, and information-sharing. The platform's popularity and reach have made it an attractive destination for businesses and organizations looking to engage with their target audience. Facebook offers a suite of advertising tools that allow businesses to create targeted ads and reach specific demographics. In addition, the platform offers insights and analytics that enable businesses to track their performance and optimize their campaigns.

Despite its popularity, Facebook has also faced criticism for issues such as data privacy, misinformation, and hate speech. The platform has taken steps to address these concerns, including implementing policies and tools to combat misinformation and hate speech, and providing users with more control over their privacy settings. Nevertheless, the platform remains a significant force in the social media landscape, connecting users around the world and providing a platform for expression and engagement.(Facebook2022).

FIGURE 2.1: Facebook Icon.

Twitter - Twitter is a popular microblogging platform that allows users to share their thoughts and ideas in short messages called tweets. With over 330 million monthly active users, Twitter has become a powerful tool for communication and information sharing. Users can post tweets of up to 280 characters, which can include text, photos, videos, and links. Twitter also provides a range of features to help users engage with their audience, including the ability to like, retweet, and reply to tweets.

One of the unique features of Twitter is the use of hashtags, which allows users to categorize their tweets and join conversations on specific topics. Hashtags are represented by the symbol followed by a keyword or phrase, such as BlackLivesMatter or COVID19. By using hashtags, users can connect with others who are interested in the same topics and participate in ongoing conversations.

In addition to individual users, Twitter is also widely used by businesses, news organizations, and public figures to reach their audiences and share information. Twitter provides a range of advertising and analytics tools to help businesses and

organizations measure the effectiveness of their tweets and engage with their followers. Overall, Twitter has become a major platform for communication, information sharing, and social engagement, with a global reach that continues to grow (Twitter2022).

FIGURE 2.2: Tweeter Icon.

Instagram - Instagram is a popular social media platform with over 1 billion monthly active users. It is a photo and video sharing platform that allows users to share visual content with their followers. Users can also follow other users and discover new content using hashtags and search.

Instagram's focus on visual content makes it a popular platform for artists, photographers, and businesses that rely on visual marketing. The platform offers a range of tools for editing and enhancing photos and videos, including filters, text overlays, and stickers. Additionally, Instagram allows users to share content on their stories, which are short-lived posts that disappear after 24 hours. This feature has become increasingly popular among users and businesses as a way to share more casual, behind-the-scenes content.

In recent years, Instagram has also introduced new features such as Instagram Reels, which allows users to create and share short-form videos, similar to TikTok. The platform has also added shopping features, allowing businesses to tag products in their posts and stories, making it easier for users to discover and purchase products (Obrist2021).

FIGURE 2.3: Instagram Icon.

LinkedIn -LinkedIn is a social networking platform designed for professionals and business-oriented individuals. It has over 740 million members in over 200 countries and territories worldwide. The platform enables users to create professional profiles that highlight their skills, education, and work experience. These profiles also serve as a means to showcase one's professional accomplishments and connect with other like-minded individuals.

One of LinkedIn's primary functions is to facilitate professional networking. Users can connect with other professionals in their industry or field, expand their network, and potentially find new job opportunities. The platform offers various features such as messaging, job postings, and groups that allow users to engage in meaningful conversations and build relationships with others.

In addition to networking, LinkedIn also offers several tools to help users enhance their professional skills and knowledge. These tools include LinkedIn Learning, a platform for online courses, and LinkedIn Live, which allows users to broadcast live video content to their followers. The platform also offers various resources such as job search tools, career advice, and industry insights, making it a valuable resource for professionals seeking to advance their careers (Mollick2021).

FIGURE 2.4: LinkedIn Icon.

YouTube - YouTube is a popular video sharing platform that has over 2 billion monthly active users. The platform allows users to upload videos, watch and share videos with others, and subscribe to channels. The platform provides a wide variety of video content, including music videos, vlogs, educational content, and entertainment videos. YouTube's search functionality and recommendation algorithm make it easy for users to discover new and relevant videos.

One of the key features of YouTube is its monetization system, which allows creators to earn money from their videos. Creators can monetize their videos through ads, sponsorships, merchandise sales, and other methods. This has led to the rise of many successful YouTube channels and the growth of the platform's creator community.

YouTube has also become a valuable resource for educational content. Many creators have created channels dedicated to teaching skills, providing career advice, and sharing knowledge on a variety of topics. The platform's popularity and accessibility have made it an effective tool for online learning and self-improvement(Miller2021).

FIGURE 2.5: Youtube Icon.

TikTok - TikTok is a social media platform that is focused on short-form video content. It has become one of the most popular social media apps with over 1 billion monthly active users. The app is designed for users to create and share short videos, typically around 15 seconds long, that are set to music or other audio. Users can also add various visual effects, filters, and stickers to their videos to make them more engaging.

One of the unique aspects of TikTok is its algorithm, which is designed to show users content that is relevant to their interests. The app uses machine learning algorithms to analyze user behavior and preferences, and then suggests content that is likely to be of interest to them. This has helped TikTok to become a popular platform for discovering new music, viral challenges, and trends.

TikTok has also become a platform for influencer marketing, with many brands using the app to promote their products and services. Influencers on the app have large followings and can have a significant impact on consumer behavior. Many brands have also created their own TikTok accounts to engage with users and promote their products in a more creative and engaging way (Oz2021).

FIGURE 2.6: Tik Tok Icon.

Snapchat - Snapchat is a popular multimedia messaging app that allows users to send photos and videos that disappear after being viewed. It has over 280 million monthly active users and is particularly popular among younger generations. In addition to private messaging, Snapchat also allows users to share "stories" with their friends, which are collections of photos and videos that are visible for 24 hours before disappearing. The platform also offers a range of filters, lenses, and stickers that users can use to enhance their snaps and express themselves creatively.

Snapchat has evolved to offer a range of additional features and content, such as news, entertainment, and original programming. The "Discover" section of the app features content from media outlets, including news stories, short-form videos, and articles, while the "Snap Map" allows users to see the location of their friends and discover events happening nearby. In recent years, Snapchat has also launched new features such as "Spotlight", which showcases user-generated short-form videos, and "Snap Originals", which are exclusive shows produced by Snapchat (Lai2021).

FIGURE 2.7: Snapshat Icon.

These platforms offer a variety of features and tools for users to communicate, connect, and share content with others, making them an integral part of modern communication and social interaction.

2.3 Arabic language in Social Media

Arabic is one of the most widely spoken languages in the world, and it is also one of the fastest-growing languages on social media platforms. With over 400 million Arabic speakers globally, there is a significant amount of social media content being created and consumed in Arabic (AbuJarour2021).
One of the most popular social media platforms for Arabic speakers is Facebook, which has over 164 million Arabic-speaking users. Instagram and Twitter are also

popular among Arabic-speaking users, with over 33 million and 25 million users respectively (Statista2021).
Arabic social media content ranges from personal updates and opinions to news and political commentary. Many Arabic-speaking users also use social media to connect with friends and family members who live in other countries.
One of the unique features of Arabic language on social media is the use of Arabic script. While many social media platforms support Arabic script, some users may prefer to use transliterated Arabic or a mixture of Arabic and English (Elshibly2021). Another important aspect of Arabic language on social media is the use of dialects. Arabic is a highly diverse language, with many different dialects and variations. Users may choose to write in their local dialect, which can vary significantly from Modern Standard Arabic, the standardized form of the language.

2.4 Natural Language Processing and Social Media

Natural Language Processing (NLP) is a field of computer science that focuses on the interaction between computers and human language. NLP techniques can be applied to social media to analyze and understand the vast amounts of text data generated by users (Huang2021). Here are some ways in which NLP is being used in social media:

1. Sentiment analysis: NLP can be used to analyze the sentiment of social media posts, determining whether the text is positive, negative, or neutral. This can be useful for brands to understand how their products or services are being perceived by customers.

2. Topic modeling: NLP can also be used to identify topics that are being discussed on social media. This can help businesses identify trends and understand what their customers are talking about (Schakel2020).

3. Named entity recognition: NLP can be used to identify and extract names of people, organizations, and locations from social media posts. This can be useful for businesses to understand who is talking about them and where they are located (**?**).

4. Chatbots: NLP is also used to power chatbots that can interact with customers on social media platforms. Chatbots can be used to answer frequently asked questions and provide customer support (Baptista2021).

5. Language translation: NLP can be used to automatically translate social media posts from one language to another. This can be useful for businesses that operate in multiple countries and want to engage with customers in their native language (Wang2018).

NLP is a powerful tool that can help businesses analyze and understand the vast amounts of text data generated by social media users. By using NLP techniques, businesses can gain valuable insights into customer behavior and preferences, and use this information to improve their products and services.

2.5 Sentiment Analysis and Social Media

Sentiment analysis is a technique that uses Natural Language Processing (NLP) to identify and extract subjective information from text data. In the context of social media, sentiment analysis is used to determine the emotional tone of social media posts and comments. Here are some ways in which sentiment analysis is used in social media (Pang2008):
1. Brand reputation management: Sentiment analysis can be used to monitor social media posts about a brand or product, and determine whether the sentiment is positive, negative, or neutral. This can help brands identify potential issues and respond in a timely manner to protect their reputation.
2. Customer service: Sentiment analysis can be used to monitor social media posts related to customer service, and quickly identify negative sentiment. This can help businesses respond quickly to customer complaints and improve their customer service.
3. Market research: Sentiment analysis can be used to monitor social media posts related to specific topics or products, and identify trends and insights that can be used for market research.
4. Political analysis: Sentiment analysis can be used to monitor social media posts related to political issues, and determine the sentiment of the public on specific topics. This can be useful for politicians and political campaigns to understand public opinion and tailor their messaging accordingly.
5. Social listening: Sentiment analysis can be used for social listening, which involves monitoring social media posts related to a brand, product, or topic. This can help businesses understand what their customers are saying about them and identify opportunities for improvement.

In conclusion, sentiment analysis is a powerful tool that allows businesses to analyze and understand the vast amounts of text data generated by social media users. By using sentiment analysis, businesses can gain valuable insights into customer sentiment and behavior, improve customer experience, manage their brand reputation, and refine their marketing strategies. With the rapid growth of social media and the increasing importance of customer feedback, sentiment analysis is becoming an essential tool for businesses of all sizes. By embracing sentiment analysis, businesses can stay ahead of the curve and make data-driven decisions to improve their overall performance.

2.6 Machine Learning and Sentiment Analysis

Machine learning is a subfield of artificial intelligence (AI) that involves developing algorithms and models that enable computers to learn from data and make predictions or decisions based on that data. In other words, it is a way of teaching computers to recognize patterns in data and use that knowledge to improve their performance over time. Machine learning is a rapidly growing field that is being used in a wide range of applications, including natural language processing, image recognition, fraud detection, and recommendation systems (Ng2017).
Sentiment analysis, on the other hand, is a specific application of machine learning that involves analyzing text data to determine the emotional tone or sentiment expressed within it. Sentiment analysis is a crucial tool for businesses and organizations that want to understand their customers' opinions and preferences. It is also

used by social media platforms to monitor public opinion and by news organizations to analyze the sentiment of their readers.
The importance of ML and SA in the modern world cannot be overstated. The vast amount of data generated daily is too much for humans to analyze manually, and machine learning and sentiment analysis offer a way to process this data efficiently and accurately. By using ML and SA, businesses and organizations can gain insights into their customers' opinions and preferences, monitor public opinion, and make data-driven decisions that improve their bottom line (Jones2021).

However, there are also challenges associated with ML and SA. One of the main challenges is bias. Machine learning models can be biased if the data used to train them is biased. This means that the models may make inaccurate predictions or decisions that reflect the bias in the data. Another challenge is the lack of transparency in machine learning models. Some models are so complex that it is difficult to understand how they make their decisions. This can lead to mistrust in the models and their results.
Despite these challenges, ML and SA offer a wealth of opportunities for businesses and organizations. By using these tools, they can gain valuable insights into their customers and stakeholders, improve their decision-making processes, and stay ahead of the competition. As such, it is essential for businesses and organizations to invest in ML and SA and stay up-to-date with the latest developments in these fields.
In summary, the modern world generates a vast amount of data that can be analyzed to gain valuable insights into various aspects of human behavior. Machine learning and sentiment analysis are two related fields that offer a way to process this data efficiently and accurately. While there are challenges associated with these fields, the opportunities they offer cannot be overstated. Businesses and organizations that invest in ML and SA will gain a competitive advantage and be better equipped to make data-driven decisions.

2.7 Machine Learning algorithm for Sentiment Analysis

sentiment analysis (SA) can be approached using a variety of machine learning (ML) techniques, which can be broadly categorized into three main groups:

2.7.1 Conventional Machine Learning Approaches

Conventional ML Approaches: These include algorithms such as Support Vector Machines (SVM) (Hasan2022), Logistic Regression (LR) (Smith2018), Naive Bayes (NB) (Liu2012), Decision Trees (DT) (Harrag2009), Random Forest, and others. These approaches typically require manual feature engineering and are less capable of handling the complexities of natural language processing (NLP) compared to deep learning and transformer-based approaches.
Here is a detailed algorithmic workflow for machine learning for sentiment analysis:

Algorithm 1: Machine Learning Algorithm for Sentiment Analysis

Input: labeled dataset
Output: ML Model

Step 1: *Dataset preprocessing*
Step 2: *Feature Extraction*
Step 3: *Model Training*
Step 4: *Model Testing*
Step 5: *Model Evaluation*
Step 6: *Model Deployement*

2.7.2 Deep Learning Approaches

Deep Learning Approaches: These approaches use neural networks with multiple layers to learn features automatically from the input data. Common architectures for SA include Recurrent Neural Networks (RNN) (Cai2018), Convolutional Neural Networks (CNN) (Lecun1998), Long Short-Term Memory (LSTM) (Brownlee2018), Bidirectional LSTM (BiLSTM) (Pei2022), and others. These approaches can handle complex NLP tasks and have shown impressive results in SA.

Algorithm 2: Sentiment Analysis using Deep Learning

Input: Text labeled
Output: Trained model

Data Collection;
  Gather dataset;
  Labeled The dataset;
Preprocessing;
  Clean and preprocess the dataset;
  Token, stem, and lemm;
Preparation;
  Convert preprocessed into num representation;
  Using bag-of-words, TF-IDF, or word embeddings;
  Choose a DL model;
Model Training;
  Train the DL model;
  Split data into training and test sets;
  Optimize model parameters using backpropagation and gradient descent;
Model Evaluation;
  Evaluate the trained DL model;

2.7.3 Transformer-Based Approaches

Transformer-Based Approaches: These are the latest and most powerful class of models for NLP tasks, including SA (Nair2021). They use attention mechanisms to encode the input text and capture long-range dependencies between words. Examples include BERT (Bidirectional Encoder Representations from Transformers) (Devlin2018), GPT (Generative Pre-trained Transformer) (Brown2020), and others.

These models have set new benchmarks in NLP tasks, including SA.

Algorithm 3: Sentiment Analysis using Transformer-based Models

Input: Text documents labeled
Output: Trained transformer-based model

```
Data Collection;    Gather a text dataset ;
    Labeled dataset;
Preprocessing;
    Clean and preprocess the text data;
    Tokenization, stemming, and lemmatization;
Preparation;
    Convert data into representation;
    Using transformer-based models such as BERT or GPT;
Model Training;
    Train the transformer-based model;
    Split data into training and test sets;
    Optimize model parameters using backpropagation and gradient descent;
Evaluation;
    Evaluate the trained model;
    Measure acc, pre, recall, F1score, and other metrics;
```

Each approach has its strengths and weaknesses, and the choice of approach depends on the specific task and the available resources. Conventional ML approaches can be simpler and faster to train, but may require more feature engineering and may not perform as well as deep learning or transformer-based approaches. Deep learning approaches are more complex and computationally expensive, but can handle more complex tasks and generally perform better than conventional ML approaches. Transformer-based approaches are the state-of-the-art in NLP tasks, including SA, but can require extensive training and fine-tuning, as well as significant computational resources.

In recent years, sentiment analysis on social media using machine learning has gained significant attention due to its potential to provide valuable insights into public opinion. With the help of machine learning algorithms, we can now analyze large volumes of social media data, detect sentiment and emotions behind the text, and use this information to make informed decisions. However, there are still many challenges that need to be addressed to improve the accuracy and effectiveness of sentiment analysis.

One of the key challenges in this area is developing better word representations that can capture the subtle nuances of language and context. Another challenge is dealing with imbalanced text classification problems, where there is an unequal distribution of sentiment labels in the dataset. This can lead to biased models that are not representative of the true sentiment distribution in the data.

To address these challenges, researchers are exploring new techniques such as deep learning and transfer learning, which can help improve the accuracy of sentiment analysis models. Other approaches include developing more robust and diverse training datasets, as well as using active learning methods to improve the quality of the annotations.

2.8 Conclusion

In conclusion, social media has become a ubiquitous part of people's lives and a rich source of information for researchers in various fields. Arabic language in social media presents unique challenges due to its complexity and the presence of dialects. Natural Language Processing (NLP) techniques have been developed to analyze Arabic text and perform sentiment analysis (SA) on social media data. Machine learning (ML) models have been employed to classify the sentiment of Arabic social media data, and different word embedding techniques have been proposed to improve the performance of these models.
However, there is still a need for more robust and accurate models that can handle the challenges of Arabic language in social media, such as dialects, text representation, and imbalanced datasets.
In the upcoming chapters, we will delve into the complex world of natural language processing, focusing specifically on word representations and imbalanced text classification. Word representation is a crucial component of NLP, as it involves mapping words to mathematical vectors that can be easily processed by machines. However, different approaches to word representation have their own strengths and limitations. Therefore, we will conduct a comprehensive review of the different techniques employed in this field, in order to gain a deeper understanding of their effectiveness and limitations.

Moving on from this, we will present a novel approach to word representation that we have developed ourselves. We believe that our approach will bring significant benefits to the field, as it addresses some of the shortcomings of existing techniques. By providing a thorough analysis of our approach and comparing it to existing methods, we aim to demonstrate the potential advantages of our method and contribute to the advancement of word representation in NLP.

Furthermore, we will discuss the challenges associated with imbalanced datasets in text classification, where the number of instances of one class significantly outweighs the other. We will introduce balancing techniques that can be used to overcome these challenges, and demonstrate our contribution to sentiment analysis on such datasets. By highlighting our novel contribution in this area, we aim to improve the accuracy and reliability of sentiment analysis on imbalanced datasets.

Finally, we will summarize our findings and provide insights into potential areas for future research. By bringing together the key components of word representation, imbalanced text classification, and sentiment analysis, we hope to contribute to the broader field of natural language processing and inspire further research and development in this exciting and rapidly evolving field.

Chapter 3

Text Representation in NLP

3.1 Introduction

Text representation is a fundamental task in natural language processing (NLP), as it involves converting raw text data into a structured numerical format that can be processed by machine learning models.
In this chapter, we will explore the different text representation techniques commonly used in NLP, starting with conventional methods such as bag-of-words (BoW) (Hasan2019), term frequency (TF) (Liu2018), and term frequency-inverse document frequency (TF-IDF) (Dalaorao2019). We will then move on to more advanced techniques such as word embedding models like word2vec (Wang2022), fasttext(Oulin2016), and GloVe (Pennington2014), and finally, the contextual word embedding approach of BERT representation (Devlin2018).
Conventional methods such as BoW, TF, and TF-IDF involve representing text as a matrix of word counts, where each row represents a document and each column represents a word in the corpus. These methods have been widely used in NLP tasks such as text classification and information retrieval. However, they have several limitations, such as their inability to capture the semantic relationships between words and the contextual information of language.
To overcome these limitations, word embedding models such as word2vec, fasttext, and GloVe have been developed. These models learn distributed representations of words in a continuous vector space, where each word is represented as a dense vector of real numbers. These vectors capture the semantic and syntactic relationships between words, allowing machine learning models to better understand the meaning of text data.
BERT representation is a recent advancement in the field of text representation that uses a contextual word embedding approach to capture the meaning of words based on their surrounding context in a sentence or document. BERT's bidirectional training approach allows it to learn the meaning of words based on their relations with both preceding and succeeding words in the input text. By considering the entire input text, BERT can capture the complex relationships between words and their context.
While these advanced techniques have improved the accuracy of NLP models, they also have their own limitations. For example, word embedding models can be biased towards certain groups or concepts, and BERT's computational requirements can make it challenging to use in certain applications. In this chapter, we will explore the strengths and weaknesses of each technique and discuss their suitability for different NLP tasks.
In the next section, we will provide a detailed explanation of each representation technique used in natural language processing

3.2 Conventional Text Representation

Conventional text representation techniques, such as bag-of-words (BoW), term frequency (TF), and term frequency-inverse document frequency (TF-IDF), have been widely used in NLP for many years. These methods involve representing text data as a matrix of word counts or frequencies, where each row represents a document, and each column represents a word in the corpus.

3.2.1 Bog of Word Description

The Bag-of-Words (BoW) model is a simple yet powerful representation in natural language processing. It treats a document as an unordered collection of words, disregarding grammar and word order. The BoW model represents a document as a vector of word frequencies or binary indicators. The process involves the following steps:

1. Tokenization: Splitting the document into individual words or tokens. 2. Vocabulary Creation: Creating a vocabulary of unique words present in the document or corpus. 3. Vectorization: Representing each document as a numerical vector, where each element corresponds to the count or presence of a specific word in the vocabulary.

The BoW model equation can be represented as follows:

$$BoW(d) = \{w_1, w_2, w_3, \ldots, w_n\} \tag{3.1}$$

where d represents a document and $\{w_1, w_2, w_3, \ldots, w_n\}$ represents the set of words present in the document d. This equation emphasizes that the BoW model extracts and represents the vocabulary of a document without considering the order or structure of the words.

The Bag-of-Words model is widely used in various natural language processing tasks such as document classification, sentiment analysis, information retrieval, and more.

The algorithm bellow takes a corpus C as input and constructs the vocabulary V by extracting all unique terms from C. It then constructs the BoW matrix BoW with dimensions m x n, where m is the number of terms in V and n is the number .

Algorithm 4: Bag of Words (BoW) Algorithm

Input: C
Output: BoW

```
Construct the voc V by extract terms from C;
Construct BoW matrix BoW with dims m x n, where m is # of terms in V and
  n is # doc in C;
for each docu d in C do
    Init the docu vector v_d with all entries set to 0;
    for each term t in d do
        Incr the entry in v_d by 1;
    Assign the docu v_d to the corresponding column in BoW;
return BoW
```

One of the main limitations of the BoW representation is that it ignores the order of words in a document. This means that it cannot capture the meaning of phrases

or sentences that depend on the order of the words, such as idiomatic expressions or sarcasm.

Another limitation is that it treats all words equally, regardless of their importance or relevance to the document. This can lead to noisy representations that include common words that do not carry much meaning, while ignoring rare words that may be more informative.

To overcome the limitations of the BoW representation, a common approach is to use a variant that takes into account the frequency of each word in the document. One such variant is the Term Frequency (TF) representation, which represents each document as a vector of word frequencies instead of word counts.

3.2.2 Term Frequency Description

In the TF representation, each element of the vector represents the frequency of a word in the document, normalized by the total number of words in the document. This means that words that occur frequently in the document are given higher weights, while words that occur rarely are given lower weights.

The TF representation can improve the performance of NLP tasks by capturing more information about the importance and relevance of words in the document. However, it still does not capture the order of words in a document, which can be important for certain applications.

TF representation, on the other hand, takes into account the frequency of each word in a document, giving more weight to words that appear more frequently.The equation below represents the methodology employed in this approach.

The equation below represents the Term Frequency (TF) measure, which quantifies the frequency of a term within a document or a corpus. TF calculates the relative importance of a term by considering its occurrence within a specific context. This equation plays a crucial role in capturing the significance of individual terms and is widely used in various natural language processing tasks, such as keyword extraction, text summarization, and document similarity analysis. It forms the basis for understanding the distribution and importance of terms in textual data.

The equations are defined as follows:

$$TF(t,d) = n_{t,d} \tag{3.2}$$

$$\sum_{w \in d} n_{w,d} \tag{3.3}$$

$TF(t,d)$ is the term frequency of term t in document d.

$n_{t,d}$ is the number of occurrences of term t in document d.

$\sum_{w \in d} n_{w,d}$ represents the total number of occurrences of all terms in document d.

The algorithm of TF is as follows:

Algorithm 5: Term Frequency (TF) Algorithm

Input: Corpus Co
Output: TF matrix Tf

for *each doc d in Co* **do**
    Calc term freq $tf_{t,d}$ for each term t in d
    formula:

$$tf_{t,d} = \frac{n_{t,d}}{\sum_k n_{k,d}}$$

    where $n_{t,d}$ is the # of times term t appears in docu d, and $\sum_k n_{k,d}$ is the total # of terms in doc d;
Const the TF mat TF with dim m x n, where m is the # terms in the voc and n is # of doc in C;
for *each t in the voc* **do**
    for *each doc d in C* **do**
        Assign $tf_{t,d}$ to the corresponding TF;
return TF

This algorithm takes a corpus C as input and computes the term frequency $tf_{t,d}$ for each term t in each document d in the corpus using the formula $tf_{t,d} = \frac{n_{t,d}}{\sum_k n_{k,d}}$. It then constructs a TF matrix TF with dimensions m x n, where m is the number of terms in the vocabulary and n is the number of documents in C.

3.2.3 TF-IDF Representation

TF-IDF representation takes into account the frequency of each word in the corpus and the inverse document frequency, which measures how important a word is in a corpus by looking at how often it appears in different documents.

The equation below, which incorporates the TF-IDF representation, effectively captures both the word frequency in the corpus and the inverse document frequency, providing a comprehensive measure of word importance across various documents.

The equation below depicts the TF-IDF (Term Frequency-Inverse Document Frequency) representation, a statistical measure widely used in natural language processing. TF-IDF considers both the frequency of each word within a corpus and the inverse document frequency, which indicates the significance of a word by examining its occurrence across different documents. This equation captures the essence of the TF-IDF approach, enabling the extraction of important and relevant information from text data for various applications such as information retrieval and text classification

$$TF - IDF(t, d, D) = TF(t, d) \times IDF(t, D) \tag{3.4}$$

In this equation, t represents a term, d represents a document, and D represents the corpus (collection of documents). The TF-IDF score measures the importance of a term in a document relative to the entire corpus, considering both the term's frequency in the document (TF) and its rarity across the corpus (IDF).

The workflow of Tf-IDF can be described as follows:

1. **Extracting the vocabulary**: In this step, the algorithm scans through all the documents in the corpus and extracts a list of unique terms, which is called the vocabulary. This vocabulary will be used to construct the rows of the TF-IDF matrix.

2. **Calculating IDF values**: In this step, the algorithm calculates the IDF value for each term in the vocabulary. The IDF value is computed as the logarithm of the ratio between the total number of documents in the corpus and the number of documents that contain the term. This step helps in identifying the importance of a term in the corpus.

3. **Constructing the TF-IDF matrix**: In this step, the algorithm constructs an empty matrix that will be used to store the TF-IDF values for each term in each document. The number of rows in the matrix is equal to the number of terms in the vocabulary, and the number of columns is equal to the number of documents in the corpus.

4. **Computing TF-IDF values**: For each document in the corpus, the algorithm computes the TF-IDF value for each term in the vocabulary. The TF value is calculated as the number of occurrences of the term in the document divided by the total number of terms in the document. The TF-IDF value is then calculated as the product of the TF value and the IDF value for that term. The resulting values are stored in the corresponding cells of the TF-IDF matrix.

Algorithm 6: TF-IDF Algorithm

Input: Corpus Text
Result: TF-IDF Matrix Represntation

```
Voc ← extract_Vocab(Corpus Text);
IDF ← comp_IDF(Voc, Corpus Text);
TFIDF_Matrix ← empty Initial matrix;
for each Doc in Corpus Text do
    TF ← comp_TF(Doc);
    TFIDF_Doc ← empty dict;
    for each Term in Voc do
        TFIDF_Doc[Term] ← TF[Term] · IDF[Term];
    end
    add TFIDF_Doc to TFIDF_Matrix;
end
return TFIDF_Matrix;
```

In conventional representation, sparsity and dimensionality are two common challenges that arise when dealing with large and complex datasets. Sparsity refers to the fact that many of the data points are zero or close to zero, while dimensionality refers to the large number of features or variables that are used to represent the data.

To address these challenges, researchers have developed several techniques to reduce sparsity and dimensionality in conventional representations. One popular approach is to use dimensionality reduction techniques such as principal component analysis (PCA) or (LDA) to reduce the number of features or dimensions in the dataset. Another approach is to use feature selection methods to identify the most relevant features for a given task, which can help reduce sparsity and improve model performance.

In addition to these techniques, researchers have also explored the use of more advanced representation methods such as embedding techniques like Word2Vec or GloVe, which can capture semantic relationships between words and reduce sparsity in text data. These methods have been shown to be highly effective in a range of natural language processing tasks, such as sentiment analysis and language translation.

3.2.4 Limitations of Conventional Word Representation

Conventional word representation techniques such as one-hot encoding and bag-of-words suffer from several limitations that hinder their effectiveness in natural language processing.

1. Sparcity: One major limitation is sparsity, which refers to the fact that the resulting vectors are often very large, with most elements being zero. This makes it difficult to process the data efficiently and can lead to overfitting, particularly in models with a large number of parameters.

2. Dimensionality: Another limitation is dimensionality, which refers to the fact that the resulting vectors are often high-dimensional. This can lead to a phenomenon known as the "curse of dimensionality", where the amount of data required to accurately represent the high-dimensional space grows exponentially with the number of dimensions.

3. Free-context: A third limitation is free-context, which refers to the fact that these techniques do not take into account the order or context in which words appear in the text. This makes it difficult to capture the complex relationships between words and can lead to poor performance on tasks that require a deeper understanding of language, such as sentiment analysis or natural language understanding.

To overcome these limitations, researchers have developed more sophisticated word representation techniques such as word embeddings, which capture the semantic relationships between words and take into account their context in the text. These techniques have proven to be more effective in a wide range of natural language processing tasks, and continue to be an active area of research and development.

3.3 Word Embedding

Word embeddings are advanced text representation techniques that represent words as dense vectors in a high-dimensional space. Unlike conventional text representation techniques, word embeddings capture the semantic and syntactic relationships between words, enabling machine learning models to better understand the meaning of text data. In this section, we will explore three popular word embedding techniques: word2vec, fastText, and GloVe.

3.3.1 Word2vec Representation

Word2vec is a neural network-based algorithm that learns word embeddings by predicting the likelihood of a word given its context (CBOW) or predicting the context given a word (Skip-gram) (Wang2022). The word embeddings learned by word2vec

capture the semantic and syntactic relationships between words and can be used in a variety of NLP tasks, including text classification, named entity recognition, and sentiment analysis.

CBOW

The CBOW model aims to predict the target word based on the context words surrounding it. The context words are defined by a window size, which specifies the number of words before and after the target word that are considered as context. The objective of the CBOW model is to maximize the probability of predicting the target word given its context words.
The equation below illustrates the Continuous Bag-of-Words (CBOW) model, which aims to predict a target word based on its surrounding context. By considering the context words, CBOW facilitates the learning of word embeddings that capture semantic relationships and contextual information. This equation serves as the foundation for training the CBOW model and enhancing its ability to generate accurate word predictions.

$$\mathbf{v}_{target} = \frac{1}{C} \sum_{i=1}^{C} \mathbf{v}_{context_i} \tag{3.5}$$

where:

$\mathbf{v}_{target}$ represents the vector representation of the target word.

$\mathbf{v}_{context_i}$ represents the vector representation of the i-th context word.

C is the number of context words in the window.

The figure below illustrates the CBOW model architecture:

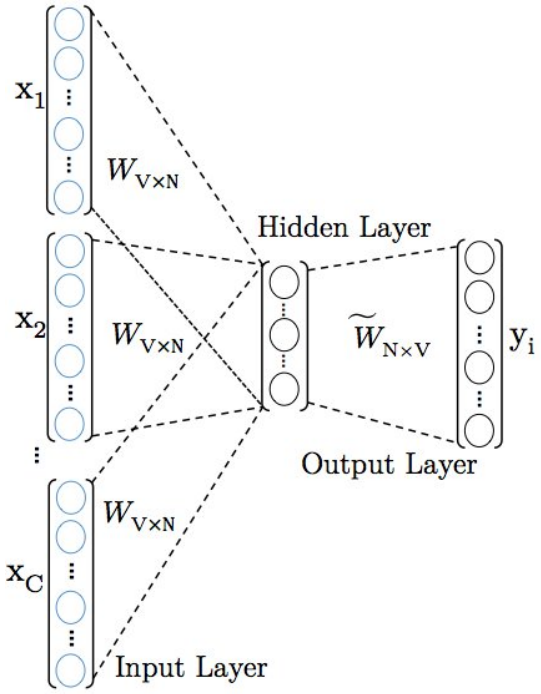

FIGURE 3.1: CBOW architecture

As shown in the figure, the CBOW model has three layers: an input layer, a hidden layer, and an output layer. The input layer represents the context words, which are one-hot encoded. The one-hot encoding means that each word is represented by a vector of zeros, except for a single 1 in the position that corresponds to the index of the word in the vocabulary.
The hidden layer represents the vector representation of the context words. The vectors in the hidden layer are learned during training and are used to predict the target word.
The output layer represents the target word, which is also one-hot encoded. The output layer has as many neurons as there are words in the vocabulary, and the probability distribution over the vocabulary is computed using a softmax function.
During training, the weights of the input and output layers are learned to maximize the probability of predicting the target word given its context words. The learned weights in the hidden layer are used as the word embeddings.
In summary, the CBOW model predicts a target word given its context words by learning vector representations of the context words that capture their semantic and syntactic relationships. These vector representations can be used as word embeddings for downstream NLP tasks.

Algorithm 7: Continuous Bag-of-Words (CBOW) Algorithm

Input: Input: corpus , voc V, window size c, dime of word emb d
Output: Output: matrix W of size V x d

Initialize the rand weight matrix W ;
for *each word w in the corpus* **do**
  Let c be the context window of size $2c$ around the word w;
  Create the input vector x by concatenating the one-hot encoded vectors of the words in c;
  Compute the average of the input vector x: $\hat{x} = \frac{1}{2c} \sum_{i=1}^{2c} x_i$;
  Compute the predicted target word vector y: $y = W\hat{x}$;
  Compute the softmax probability distribution over the vocabulary:
  $P(w_i|\hat{x}) = \frac{\exp(y_i)}{\sum_{j=1}^{|V|} \exp(y_j)}$;
  Compute the cross-entropy loss between the predicted distribution and the true distribution t,
  where t is a one-hot encoded vector with a 1 at the index of the target word and 0s elsewhere:
  $J = -\sum_{i=1}^{|V|} t_i \log(P(w_i|\hat{x}))$;
  Compute the gradient of the loss with respect to the weights:
  $\frac{\partial J}{\partial w_{ij}} = (\hat{x})j(P(w_i|\hat{x}) - t_i)$;
  Update the weight matrix W using stochastic gradient descent:
  $wij = w_{ij} - \alpha \frac{\partial J}{\partial w_{ij}}$;
end

The CBOW algorithm learns to predict a target word given its context words. By training on a large corpus of text, the algorithm learns to capture the semantic and syntactic relationships between words in the corpus, resulting in word embeddings that can be used in various natural language processing tasks.

Skipgram

In addition to the CBOW model, word2vec also has another training technique called Skip-gram. While CBOW predicts a target word based on its surrounding context, Skip-gram aims to predict the context words given a target word. In this section, we will describe the Skip-gram model architecture and its training process.

$$\mathcal{L} = \sum_{target} \sum_{w \in context w \neq target} \log P(w|target) \tag{3.6}$$

where:

$\mathcal{L}$ is the loss function.

$target$ is the target word.

w represents a context word.

$P(w|target)$ is the probability of observing context word w given the target word.

In this code, the Skip-gram model is described along with its equation. The overall description is provided as text, and the Skip-gram equation is presented using the equation environment. The variables and terms used in the equation are labeled and described using the description environment. You can customize the description or equation as needed.

The figure below illustrates the Skip-gram model architecture:

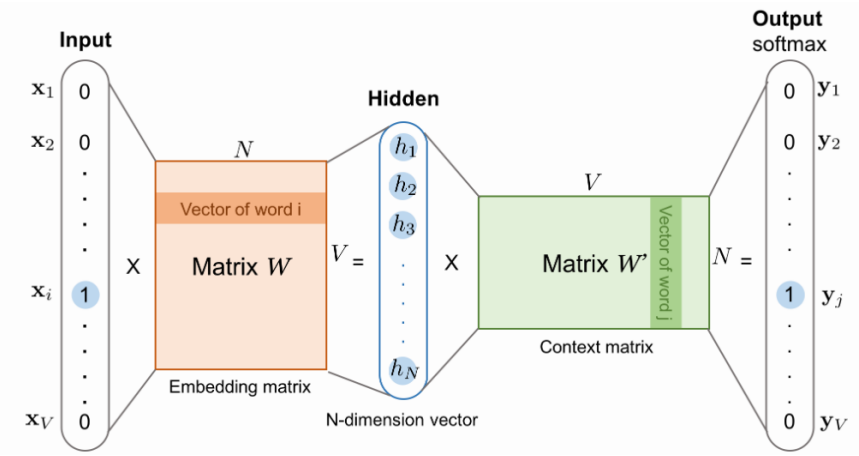

FIGURE 3.2: Skip-Gram architecture

As shown in the figure, the Skip-gram model also has three layers: an input layer, a hidden layer, and an output layer. However, the input and output layers are reversed compared to the CBOW model. In Skip-gram, the input layer represents the target word, which is one-hot encoded, and the output layer represents the context words, which are also one-hot encoded.

The hidden layer represents the vector representation of the target word. The vectors in the hidden layer are learned during training and are used to predict the context words. During training, the weights of the input and hidden layers are learned to maximize the probability of predicting the context words given the target word. The learned weights in the hidden layer are used as the word embeddings.

The training process of Skip-gram involves iterating through each word in the training corpus and selecting a window of context words around it. For each context word in the window, the model is trained to predict the target word. The objective is to maximize the probability of predicting the context words given the target word. This process is repeated for all words in the corpus, and the model is updated using stochastic gradient descent.
The Skip-Gram algorithm learns to predict context words given a target word. By training on a large corpus of text, the algorithm learns to capture the semantic and syntactic relationships between words in the corpus, resulting in word embeddings that can be used in various natural language processing tasks.

Algorithm 8: Skip-Gram Algorithm

Input: Input: corpus , voc V, window size c, dime of word emb d
Output: Output: matrix W of size V x d

Initialize the rand weight matrix W ;
Initialize the weight matrix W randomly with small values;
for *each word w in the corpus* **do**
    Let c be the context window of size $2c$ around the word w;
    for *each context word* c_i *in* c **do**
        Create the inp vector x as the one hot encoded vector of c_i;
        Comp the pred target word vector y: $y = Wx$;
        Comp the softmax over the vocabulary: $P(w_i|x) = \frac{\exp(y_i)}{\sum_{j=1}^{|V|}\exp(y_j)}$;
        Comp the cross-entropy loss between the pred distribution and the true distribution t,
        where t is a one-hot encoded vector with a 1 at the index of the target word and 0s elsewhere: $J = -t_i \log(P(w_i|x))$;
        Compute the gradient of the loss with respect to the weights: $\frac{\partial J}{\partial w_{ij}} = x_j(P(w_i|x) - t_i)$;
        Update the weight matrix W using stochastic gradient descent: $w_{ij} = w_{ij} - \alpha\frac{\partial J}{\partial w_{ij}}$;
    end
end

In summary, the Skip-gram model predicts context words given a target word by learning vector representations of the target words that capture their semantic and syntactic relationships. These vector representations can be used as word embeddings for downstream NLP tasks. Compared to CBOW, Skip-gram is better suited for larger datasets and capturing rare words.

3.3.2 FastText Representation

FastText is an extension of word2vec that represents words as a sum of the embeddings of their character n-grams (Oulin2016). This enables the model to capture the subword information, making it more effective in dealing with rare or misspelled words. FastText has been shown to outperform word2vec in several NLP tasks, including text classification and sentiment analysis.
FastText is an extension of the word2vec model that was introduced by Facebook's AI Research (FAIR) team. The key difference between FastText and word2vec is that FastText treats each word as a bag of character n-grams, rather than a single entity. This allows FastText to capture sub-word information and handle out-of-vocabulary

(OOV) words more effectively. In this section, we will describe the FastText model architecture and its training process.

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{k=1}^{K} y_{ik}\log(\hat{y}_{ik}) + (1-y_{ik})\log(1-\hat{y}_{ik})\right) \tag{3.7}$$

where:

$\mathcal{L}$ is the loss function.

N is the total number of training samples.

K is the number of classes.

y_{ik} is the binary label (0 or 1) indicating if the class k is the correct label for the training sample i.

$\hat{y}_{ik}$ is the predicted probability of class k for the training sample i.

The figure below illustrates the FastText model architecture:

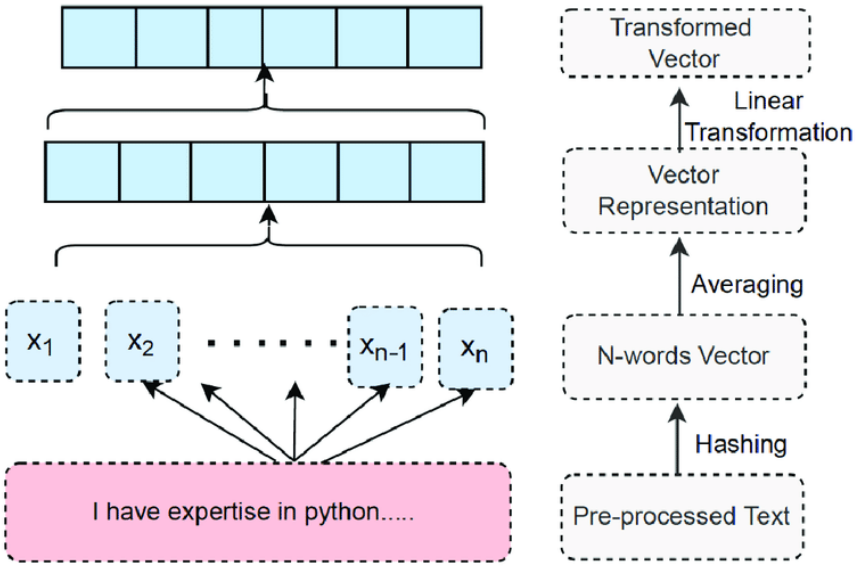

FIGURE 3.3: FastText architecture

As shown in the figure, the FastText model has an input layer, a hidden layer, and an output layer. The input layer represents a sequence of characters, rather than words, which are encoded using a bag-of-ngrams technique. The hidden layer represents the vector representation of the word, which is obtained by summing the n-gram vectors. The output layer is a softmax layer, which predicts the probability distribution over the target words.
During training, the weights of the input and output layers, as well as the n-gram embeddings, are learned to maximize the probability of predicting the target words

given the input sequence.

Algorithm 9: FastText Algorithm

Input: Input: A corpus of text, minimum count threshold m, context window size c, dimensionality of word embeddings d, subword length n

Output: Output: A matrix W of size $V \times d$ containing the learned word embeddings

Initialize an empty dictionary C; **for** *each word w in the corpus* **do**
  for $i = 1$ *to* $|w|$ **do**
    Extract all subwords of length n from w: $g_1, g_2, ..., g_{|w|}$; Increment the count of each subword in the dictionary C;
  end
end
Remove subwords with count less than the minimum count threshold m;
Initialize the weight matrix W randomly with small values; **for** *each word w in the corpus* **do**
  Let c be the context window of size $2c$ around the word w;
  Create the input vector x by concatenating the vectors of the subwords in c and the vector of the word w;
  Compute the predicted target word vector y: $y = Wx$;
  Compute the softmax probability distribution over the vocabulary:
  $P(w_i|x) = \frac{\exp(y_i)}{\sum_{j=1}^{|V|} \exp(y_j)}$;
  Compute the cross-entropy loss between the predicted distribution and the true distribution t, where t is a one-hot encoded vector with a 1 at the index of the target word and 0s elsewhere:
  $J = -\sum_{i=1}^{|V|} t_i \log(P(w_i|x))$; Compute the gradient of the loss with respect to the weights:
  $\frac{\partial J}{\partial w_{ij}} = x_j(P(w_i|x) - t_i)$;
  Update the weight matrix W using stochastic gradient descent:
  $w_{ij} = w_{ij} - \alpha \frac{\partial J}{\partial w_{ij}}$;
end

The FastText algorithm is an extension of the Skip-Gram algorithm that uses subword information to learn word embeddings. By training on a large corpus of text, the algorithm learns to capture the semantic and syntactic relationships between words and their subwords in the corpus, resulting in word embeddings that can handle out-of-vocabulary words and can be used in various natural language processing tasks.
The training process of FastText involves iterating through each word in the training corpus and generating a sequence of character n-grams for it. For each word, the model is trained to predict the target words within a certain context window. The objective is to maximize the probability of predicting the target words given the character n-gram sequence. This process is repeated for all words in the corpus, and the model is updated using stochastic gradient descent.

In summary, FastText treats each word as a bag of character n-grams and learns sub-word embeddings that capture the morphological and semantic structure of words. These embeddings can be used as word representations for various NLP tasks, such as text classification and information retrieval. Compared to word2vec,

FastText is better suited for handling OOV words and capturing word-level morphology.

3.3.3 GloVe Representation

GloVe (Global Vectors) is a word embedding technique that learns word representations by factorizing a matrix of word co-occurrence statistics (Pennington2014). GloVe is based on the idea that words that co-occur frequently in a corpus are likely to have similar meanings. The word embeddings learned by GloVe capture both the semantic and syntactic relationships between words and have been shown to outperform both word2vec and fastText in several NLP tasks.
Unlike word2vec and FastText, GloVe is based on co-occurrence statistics of words in a corpus. The main idea behind GloVe is that word meanings can be inferred from the distributional patterns of words in the corpus. In this section, we will describe the GloVe model architecture and its training process.

GloVe is a word embedding model that combines global matrix factorization techniques with local context windows. It learns word vectors by leveraging both global co-occurrence statistics and local context information.

$$\sum_{i=1}^{V}\sum_{j=1}^{V} f(X_{ij})\left(\mathbf{w}_i^T\tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2 \tag{3.8}$$

where:

V is the vocabulary size.

X_{ij} is the co-occurrence count of word i and word j.

$\mathbf{w}_i$ and $\tilde{\mathbf{w}}_j$ are the word vectors for words i and j, respectively.

b_i and $\tilde{b}_j$ are the bias terms associated with words i and j, respectively.

$f(X_{ij})$ is a weighting function that captures the importance of the co-occurrence count X_{ij}.

The figure below illustrates the GloVe model architecture:

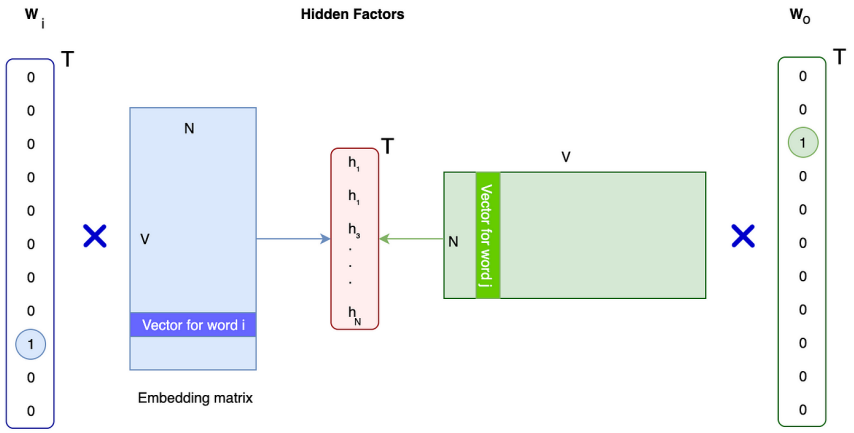

FIGURE 3.4: GloVe architecture

As shown in the figure, the GloVe model has an input layer, a hidden layer, and an output layer. The input layer represents the co-occurrence matrix of words, which is obtained by counting the number of times each word appears within a certain context window of other words in the corpus.
The hidden layer represents the vector representation of the words, which is obtained by minimizing the difference between the dot product of word vectors and the logarithm of their co-occurrence probabilities. The output layer is a softmax layer, which predicts the probability distribution over the context words.
During training, the weights of the input and output layers, as well as the word embeddings, are learned to minimize the loss function. The loss function measures the difference between the predicted co-occurrence probabilities and the actual co-occurrence probabilities. The learned word embeddings can be thought of as global representations that capture the overall distributional patterns of words in the corpus.
The training process of GloVe involves constructing the co-occurrence matrix of words from the corpus, which is typically a sparse and high-dimensional matrix. To reduce the dimensionality and sparsity of the matrix, GloVe uses singular value decomposition (SVD) to factorize the matrix into lower-dimensional matrices. The factorized matrices are then used to train the model using stochastic gradient descent.

Algorithm 10: GloVe Algorithm

Input: Input: A co-occurrence matrix X of size $|V| \times |V|$, dimensionality of word embeddings d, learning rate α, number of iterations n
Output: Output: A matrix W of size $V \times d$ containing the learned word embeddings

Initialize the weight matrix W randomly with small values;
Initialize the bias vectors b_i and b_j to 0;
Initialize the scalar bias b to the logarithm of the frequency of the corpus;
for $t = 1$ *to* n **do**
    for *each pair of words* i, j *with a non-zero entry in the co-occurrence matrix* **do**
        Compute the inner product between the word vectors and the bias terms: $w_i^T w_j + b_i + b_j$;
        Compute the difference between the log of the co-occurrence count and the inner product:
        $\Delta = \log(X_{ij}) - (w_i^T w_j + b_i + b_j - b)$;
        Update the weight matrix and bias vectors:
        $w_i \leftarrow w_i + \alpha \Delta w_j$
        $w_j \leftarrow w_j + \alpha \Delta w_i$
        $b_i \leftarrow b_i + \alpha \Delta$
        $b_j \leftarrow b_j + \alpha \Delta$;
    end
end

The GloVe algorithm learns word embeddings by training on a co-occurrence matrix of word pairs in a corpus. The algorithm aims to optimize a weighted least-squares objective that balances the importance of frequent and infrequent co-occurrences. By training on a large corpus of text, the algorithm learns to capture the semantic and syntactic relationships between words in the corpus, resulting in word embeddings that can be used in various natural language processing tasks.

In summary, GloVe is based on co-occurrence statistics of words in a corpus and learns global representations that capture the overall distributional patterns of words. These representations can be used as word embeddings for various NLP tasks, such as text classification and information retrieval. Compared to word2vec and FastText, GloVe is better suited for capturing the semantic relationships between words and handling rare words in the corpus.

3.3.4 Limitaions of word embeddings

While word embeddings such as Word2Vec, FastText, and GloVe have revolutionized the field of natural language processing, there are still some limitations to these techniques that should be considered.

1. Limited Context: Word embeddings typically work by representing each word as a vector in a high-dimensional space based on its co-occurrence statistics with other words. However, these representations are often based on a limited context window, which can result in incomplete or inaccurate representations of words that have multiple meanings or uses.

2. Lack of Transparency: Although word embeddings can capture semantic relationships between words, it can be difficult to interpret and understand the underlying factors that contribute to these relationships. This lack of transparency can make it challenging to diagnose and correct errors in the embedding model.

3. Inability to Capture Rare Words: Word embeddings are based on the distributional hypothesis, which assumes that words that occur in similar contexts have similar meanings. However, this assumption may not hold for rare or infrequently occurring words, which can lead to poorly represented or inaccurate embeddings for these words.

4. Limited Multilingual Support: Word embeddings are typically trained on large amounts of text data in a single language, which can limit their applicability to multilingual or cross-lingual natural language processing tasks. While there are efforts to develop multilingual word embeddings, these techniques often rely on aligning embeddings across languages and may not capture the full range of linguistic variation.

5. Bias and Fairness Issues: Like any machine learning model, word embeddings can be biased and reflect the biases and prejudices present in the underlying training data. This can lead to inaccurate or unfair representations of certain groups or concepts, which can have negative consequences in downstream applications.

While word embeddings have significantly improved the accuracy and effectiveness of natural language processing tasks, it is important to be aware of these limitations and consider them when designing and evaluating machine learning models. In the next section, we will explore a more recent and advanced text representation technique, BERT representation, that addresses some of these limitations.

3.4 BERT Representation

Bidirectional Encoder Representations from Transformers (BERT) is a recent advancement in text representation that has gained significant attention in the field of natural language processing (NLP) (Devlin2018). BERT is a pre-trained deep learning model that is trained on large amounts of text data to generate contextualized word embeddings.
Unlike conventional text representation techniques and word embedding models, BERT takes into account the context and order of words in a sentence or document. BERT uses a bidirectional transformer model, which means it processes both the left and right contexts of each word, allowing it to capture the dependencies between words in a sentence. This makes BERT more effective in capturing the semantics and syntax of language.
BERT is pre-trained on large amounts of text data, such as Wikipedia and the BookCorpus, using a masked language model and next sentence prediction task. The masked language model randomly masks some of the words in a sentence and trains the model to predict the masked words based on the context of the sentence. The next sentence prediction task trains the model to predict whether two sentences are consecutive or not. The pre-training process results in a deep learning model that can generate high-quality contextualized word embeddings.
One of the main advantages of BERT over conventional text representation techniques and word embedding models is its ability to capture the context and meaning of words in a sentence. This makes it more effective in tasks such as sentiment analysis, question answering, and text classification.
BERT can also handle out-of-vocabulary words, as it can generate embeddings for unseen words based on their context in the sentence.

The figure below illustrates the BERT representation model architecture:

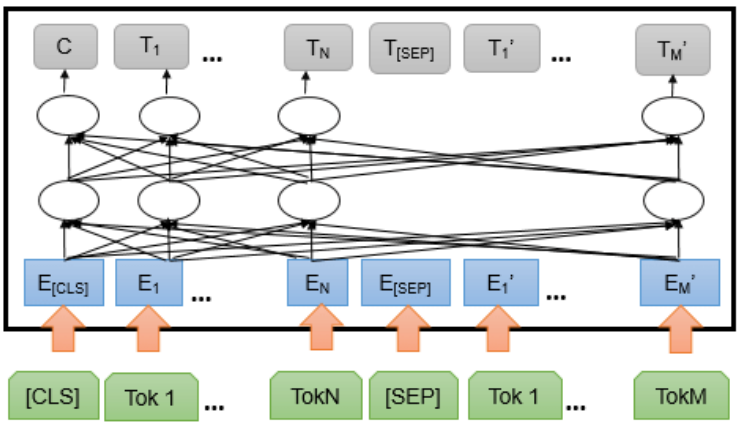

FIGURE 3.5: BERT Representation Architecture

The following equation represents the loss function, which serves as a quantitative measure of the discrepancy between the predicted outputs and the actual values. It provides valuable insight into the performance of the model by quantifying the extent of error in the predictions, allowing for effective optimization and improvement of the overall system.

$$\mathcal{L} = \sum_{i=1}^{N} (\log P(y_i|\mathbf{x}_i)) \tag{3.9}$$

where:

$\mathcal{L}$ is the loss function.

N is the total number of training examples.

$\mathbf{x}_i$ represents the input sequence for training example i.

y_i is the corresponding label for training example i.

$P(y_i|\mathbf{x}_i)$ is the predicted probability distribution over the possible labels given the input sequence.

Algorithm 11: BERT for Representation Learning

Input: A sequence of text tokens $x = x_1, x_2, ..., x_n$,
pre-trained BERT model M
Output: A matrix H of size $n \times d$ containing the contextualized word embeddings for the input sequence

1 Tokenize the input sequence using WordPiece or SentencePiece; Add special [CLS] and [SEP] tokens to the beginning and end of the sequence;
2 Feed the sequence through the pre-trained BERT model M to obtain the output sequence of hidden states: $H = M(x)$;
3 Extract the hidden state corresponding to the [CLS] token as the sentence-level representation: $s = H_0$;
4 Extract the hidden states corresponding to the original input tokens as the contextualized word embeddings: $h_i = H_i$ for $i = 1, 2, ..., n$;

The BERT (Bidirectional Encoder Representations from Transformers) model is a pre-trained language model that learns contextualized representations of words and sentences from large amounts of unlabeled text. The algorithm is based on the Transformer architecture and uses a masked language modeling task and a next sentence prediction task to learn a deep bidirectional representation of the input text. The pre-trained BERT model can then be fine-tuned on downstream natural language processing tasks, such as sentiment analysis or named entity recognition, by adding a task-specific output layer and fine-tuning the entire model on a labeled dataset. In summary, BERT is a powerful text representation technique that overcomes the limitations of conventional text representation techniques and word embedding models. Its ability to capture context and meaning, handle out-of-vocabulary words, and be fine-tuned for specific tasks makes it an essential tool in the field of NLP.

3.5 Conclusion

In conclusion, this chapter discussed the various text representation techniques used in NLP, starting from the conventional techniques such as BoW, TF, and TF-IDF, followed by the more advanced techniques such as word embeddings and BERT representation. We highlighted the advantages and limitations of each technique, and we discussed how BERT representation has become a game-changer in the field of NLP due to its ability to generate contextualized word embeddings, taking into account the context and order of words in a sentence or document.
In the next chapter, we will discuss our contribution to the field of NLP, which is a new Arabic word representation based on BERT. We added new layers to the pre-trained BERT model, specifically designed for the Arabic language, to generate contextualized embeddings that are more effective in capturing the semantics and syntax of the Arabic language. Our approach resulted in significant improvements in

Arabic sentiment analysis, a task that is of great importance in the Arabic-speaking world. We will discuss our methodology in detail and present our experimental results, demonstrating the effectiveness of our approach.

Chapter 4

A New Arabic Word Embedding Representation for Sentiment Analysis

4.1 Introduction

Arabic sentiment analysis is a crucial area of research, as it has many practical applications in fields such as social media analysis, customer feedback analysis, and political sentiment analysis (Pang2008). Due to the complexity of the Arabic language, sentiment analysis of Arabic text poses several challenges. These challenges include the vast number of words in Arabic, the complexity of Arabic morphology, and the lack of comprehensive labeled datasets for training and testing machine learning models.

To address these challenges, researchers have been exploring various techniques, such as feature engineering, machine learning algorithms, and deep learning models, to develop accurate and efficient Arabic sentiment analysis systems. With the increasing availability of Arabic language data and the development of new techniques, the accuracy of Arabic sentiment analysis systems has been improving in recent years. However, there is still much room for improvement in this field, and researchers continue to explore new approaches to enhance the performance of Arabic sentiment analysis systems.

Applying machine learning approaches to Arabic language necesssite the use of distributional representations, such as GloVe, Word2Vec, and FastText, to create vectors from the context of the words. Despite the effectiveness of word embedding, the complex morphology and the huge number of terms in Arabic language still pose challenges to machine learning approaches.

This chapter introduces a new method for improving the GloVe architecture, which involves the inclusion of a root extraction module to tackle the challenges posed by the complex morphology and large number of terms in Arabic. By utilizing the roots of words, this method is intended to enhance the accuracy of sentiment analysis while simultaneously reducing processing time through the compression of vector space representations."

The novel technique includes an additional layer that generates a new corpus by transforming all words to their roots, which is then supplied to the GloVe approach. The performance of this method is assessed by implementing two conventional ML classifiers, SVM and LR, for sentiment analysis, and comparing the results with those of the GloVe baseline and state-of-the-art approaches..

The novelty of the new method is its enhancement of the GloVe architecture through the addition of a layer for reducing vocabulary, leading to remarkable outcomes that surpass both the baseline and the cutting-edge methods.
The following section of this chapter is structured as follows: in the second section, we provide a literature review on existing related work. Section 3 outlines the GloVe distributional representation model. Section 4 delves into the proposed approach and the methodology used to extract polarities from tweets. In section 5, we discuss the results of the system evaluated through two datasets using several metrics. Finally, we conclude with remarks and perspectives.

4.2 Related Work

Over the past few years, there has been a growing interest in researching Arabic Sentiment Analysis. Various methods have been explored in this field, including Lexicon-Based approaches, Machine Learning approaches, and Hybrid models, as documented in the literature.
The field of Arabic Sentiment Analysis has gained significant attention in recent years, and various approaches have been proposed in the literature. One of the main approaches is the Lexicon based approach, which utilizes word polarities to determine the sentiment of texts.
For instance, authors in (Baccianella2010) and (Thelwall2012) employed SentiWordNet and SentiStrength, respectively. However, these lexicons are based on English and require machine translation from Arabic, which results in poor performance due to the complexity of Arabic words that have multiple meanings. To overcome this issue, researchers in (Al-Khatib2016) proposed a standard Arabic lexicon called Ar-SenL, which is a large set of Arabic word embeddings. Nonetheless, there is no perfect Arabic sentiment lexicon available, and some researchers have developed their own lexicons specific to certain fields, such as EL-Beltagy (Ali2013), who focused on the Egyptian dialect.
To approach sentiment analysis in Arabic, researchers have primarily focused on using supervised Machine Learning techniques. This involves dividing the dataset into labeled subsets for learning and testing.
However, these techniques have been more commonly used for English language sentiment analysis, and there is still a lack of application for Arabic. For instance, in (**?**), various classifiers such as SVM and RBF were utilized to classify the sentiment of tweets from various domains. Similarly, in (Shahbazi2021), authors used Decision Tree and SVM algorithms to predict sentiments about COVID-19 vaccination campaigns.
The hybrid approach combines both Machine Learning and Lexicon Based techniques, and there are limited studies conducted in Arabic language. In (Abdullah2019), the authors suggested a hybrid approach for analyzing sentiment in Arabic tweets, where the lexical-based classifier was utilized for labeling the training data, and the SVM machine learning classifier was trained using the output.
After conducting a comprehensive literature review on sentiment analysis in Arabic, we discovered that current lexicon-based techniques encounter problems with sparsity and dimensionality due to the vast number of words in the Arabic language. Moreover, using baseline word embeddings for machine learning, even in a hybrid approach, did not yield satisfactory results due to the Arabic language's complex morphology. This results in similar words having different surface forms, making it difficult to identify the correct sentiment.

To overcome these limitations, we proposed a new approach that integrates a root extraction module (REM) and GloVe techniques. This approach aims to enhance the embedding representation of Arabic words and improve sentiment analysis on Arabic social media.

4.3 GLoVe For Arabic Sentiment Analysis

There are multiple distributional word representation models in literature that follow the linguistic assumption that words with similar contexts have similar meanings. These models are unsupervised and utilize statistics and probabilities from large corpora.
One popular model for vector representations of words is the Global Vectors for Word Representation (GloVe) (Pennington2014). GloVe was introduced by Pennington in 2014 and is designed to learn low-dimensional vector representations of words. It differs from other models by using ratios of co-occurrence probabilities instead of just word occurrence.
It considers the frequency of word pairs occurring together and uses a loss function to minimize the difference between the dot product of two word vectors and the logarithm of the co-occurrence probability of those two words. This loss function is a weighted sum of squared errors. The vectors that result from this approach represent the semantic relationships between words and can be used in various natural language processing tasks, such as text classification, machine translation, and information retrieval.
The GloVe loss function can be represented as:

$$\sum_{i=1}^{V}\sum_{j=1}^{V} f(P_{ij})(\mathbf{w}_i^T \tilde{\mathbf{w}} j + b_i + b_j - \log Pij)^2 \tag{4.1}$$

where $\mathbf{w}i$ and $\tilde{\mathbf{w}}j$ are the word vectors for the ith and jth words in the vocabulary of size V, b_i and b_j are bias terms for those words, Pij is the probability of co-occurrence of the words i and j in the corpus, and $f(Pij)$ is a weighting function that assigns more importance to rare word pairs.

Using the GloVe model in a sentiment analysis system has enhanced the precision and decreased the complexity of matrix representations. The fundamental process of using GloVe for sentiment analysis includes three steps, as depicted in Figure 4.1. Initially, the GloVe model is employed to acquire word vector representations from a large corpus such as Wikipedia. After that, the tweets or comments collected from the benchmark datasets are preprocessed, and a list of words is formed. Finally, the words are matched with the vectors generated in the first phase to obtain their representations, which are utilized as features for machine learning classifiers to build models.

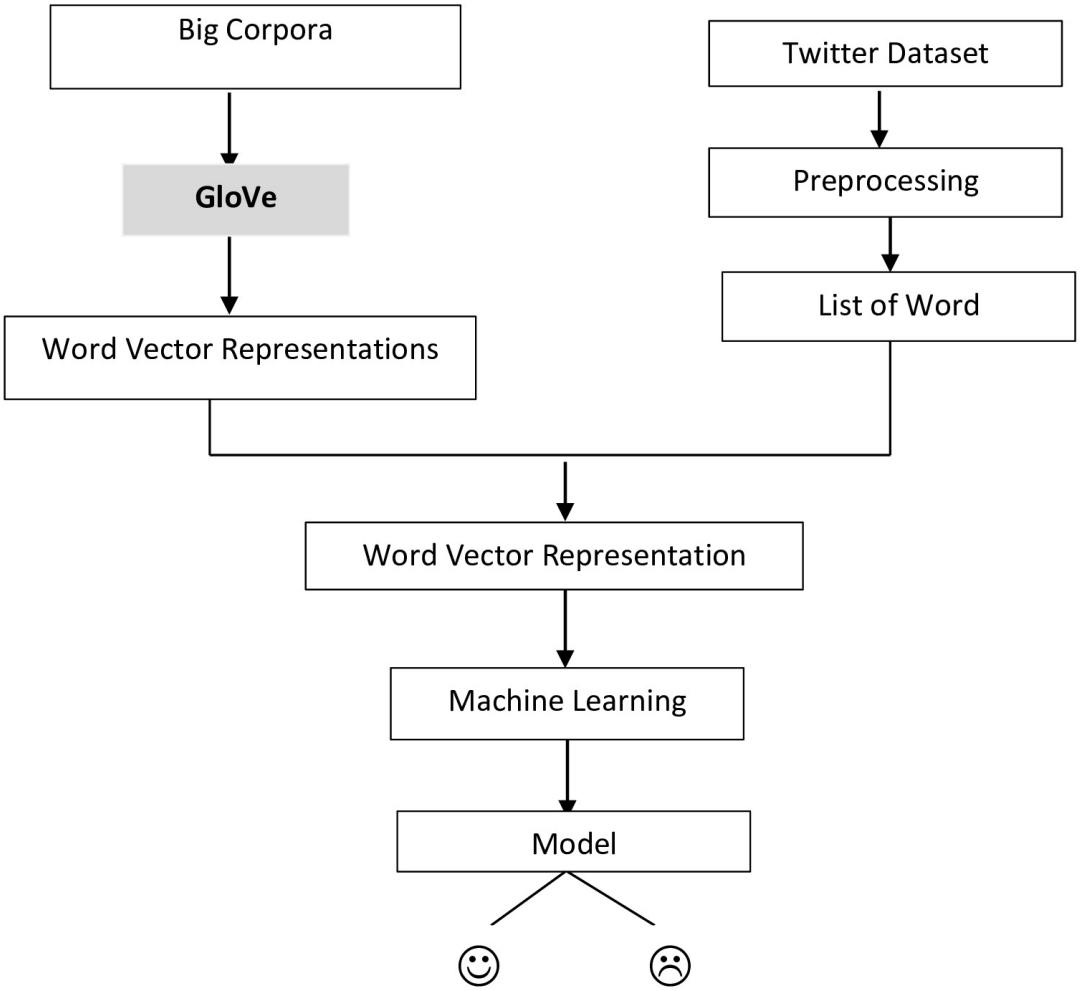

FIGURE 4.1: GloVe architecture

Despite the superior performance of the GloVe representation in sentiment analysis across multiple languages, including English, French, and Spanish, compared to traditional models such as bags of words, it still performs poorly in Arabic due to the complexity of Arabic text structures and morphology, as well as the abundance of words, which exceeds thirteen million. To improve the results and maximize the benefits of the GloVe representation in Arabic language sentiment analysis, a new module called the roots extraction module (REM) has been integrated into the basic sentiment analysis framework, and this module will be discussed in the next section.

4.4 The Proposed Approach

The novel method involves a combination of two processing techniques, namely REM and GloVe, in order to create more efficient and compact features. These features can then be utilized in machine learning classifiers to improve their accuracy and effectiveness in Arabic sentiment analysis. The primary objective of this approach is to enhance the quality of word embeddings and increase their usefulness in Arabic sentiment analysis tasks.

The system is illustrated in Figure 4.2 and consists of three phases, similar to the basic GloVe sentiment analysis scheme.

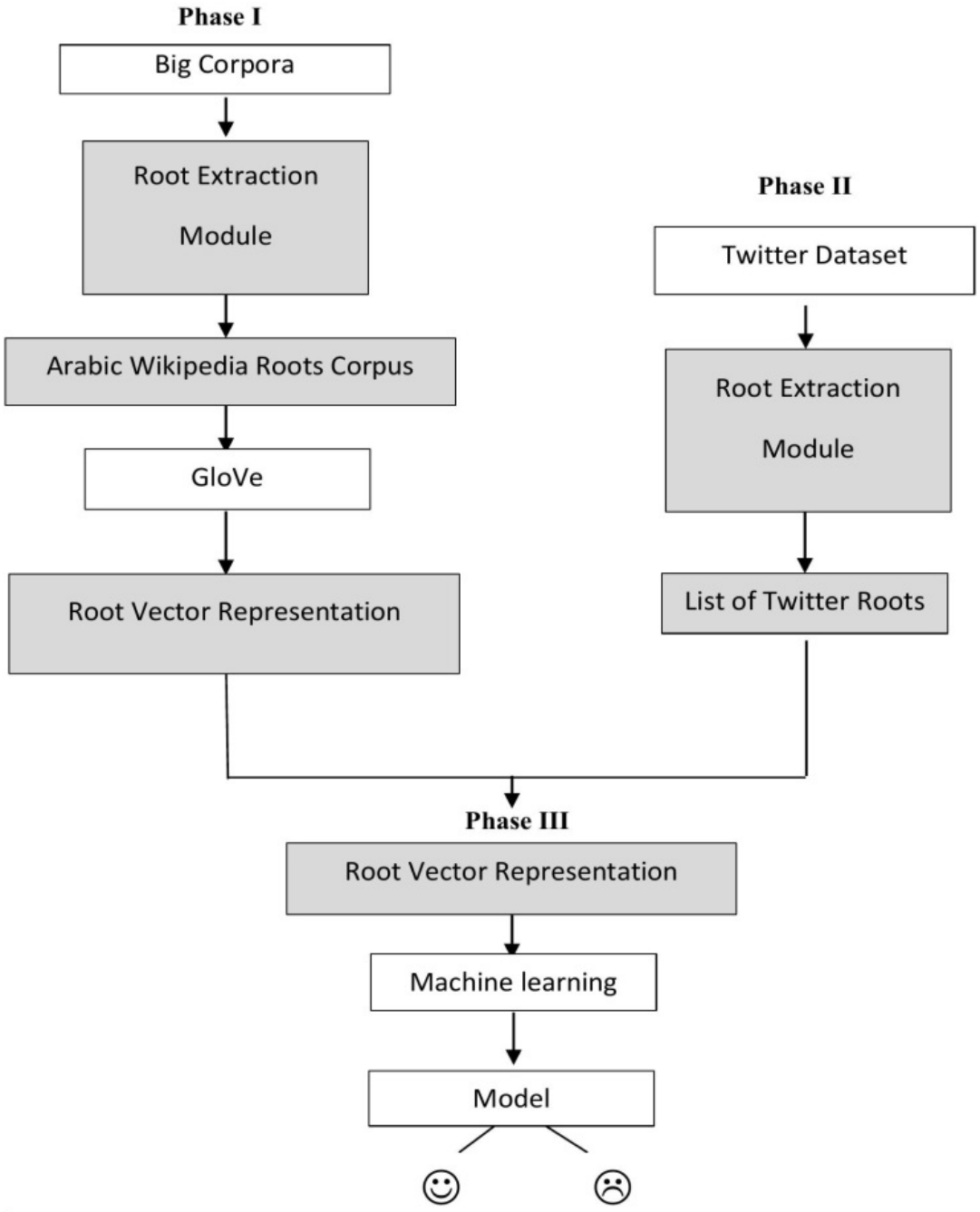

FIGURE 4.2: The New Approach architecture

Nevertheless, the key feature of this approach is the integration of the roots extraction model into the GloVe representation, which is utilized in phases one and two. More detailed information regarding the approach is elaborated in the following subsections.

Phase I: Root vector representations

1. Data collection: It is essential to work with a substantial amount of Arabic data, preferably equivalent in size to the data available on Wikipedia. This suggests that the process of learning the distributional vector representation involves analyzing patterns and relationships within an extensive dataset of Arabic text

to develop an effective representation of how words are utilized and related within the language.

2. Root Extraction Module: This is the preprocessing step, which is illustrated in Figure 4.4 below. The initial step involves data cleaning, which includes removing symbols, URLs, punctuations, and non-Arabic characters. Next, stop words such as "the", "in", or "and" are eliminated. Finally, with (**?**) Arabic root extraction approach is employed to obtain an Arabic Wikipedia roots corpus.

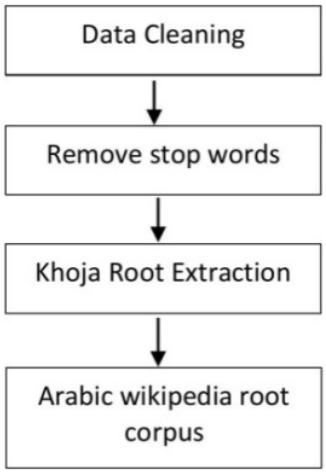

FIGURE 4.3: Phase I. Roots Extraction scheme

3. GloVe: During this step, the global vector distributional GloVe model is utilized to train on the corpus obtained from the root extraction module. This enables us to acquire novel vector representations for each word in the corpus based on its co-occurrence with other words. These representations are subsequently used in downstream tasks such as sentiment analysis.

Phase II: List of Twitter dataset roots

1. Dataset acquisition: In order to develop machine learning classifiers for Arabic sentiment analysis, it is crucial to obtain a suitable dataset of Arabic text that has been annotated with sentiment labels. The dataset should include a significant number of examples that are relevant to the task and domain where the classifiers will be utilized.

2. Root Extraction Module Similar to Phase I, a preprocessing step is required to remove noise from the tweets. Next, stop words are eliminated, followed by a tokenization step where the text is split into tokens using a comma or whitespace. The resulting list of words is then processed using Khoja root extraction [22] to create a new bag of roots, as depicted in Figure 4.4 .

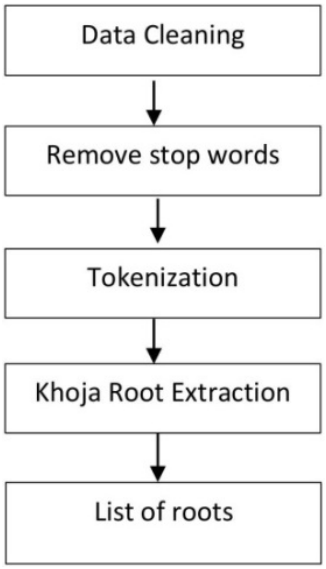

FIGURE 4.4: Phase II. List of Roots Extraction scheme

Phase III: Sentiment Analysis Models

1. Vector assignments: During this phase, the list of deductions obtained from Phase II is compared to the root representations obtained from Phase I, and each root is associated with its respective vector.

2. Models creations: The resulting vectors are utilized as features in machine learning, where two efficient classifiers, support vector machine (SVM) and logistic regression (LR), are employed to create models for Arabic sentiment analysis. The dataset is divided into 70% for training and 30% for testing purposes.

4.5 Experiments and Results

In this section, we provide an overview of the datasets used and describe the two experiments conducted to evaluate the accuracy of our Arabic sentiment analysis system. All codes were implemented in Python, and we utilized the glove-python package, which is available online, to implement the GloVe model. As mentioned in Section 4, we utilized three data collections, one of which was gathered from Arabic Wikipedia, a vast corpus necessary to apply GloVe to create word vector representations utilizing context to compress the size of matrix representations.

1. Datasets settings: The system uses three datasets for evaluation purposes. The first dataset is the Arabic Wikidata Dump 2018, which contains Wikipedia Arabic articles from the January 20, 2018 data dump, providing a valuable resource for exploring the Arabic language and developing new applications. It has a total of 75 million tokens. The second dataset is a collection of 40,000 Egyptian tweets in Arabic, consisting of 20,000 positive and 20,000 negative tweets covering a wide range of topics commonly addressed on Twitter. The third dataset is the Large-Scale Arabic Book Reviews (LABR) obtained via the internet, which has 16,448 rows of book reviews labeled as positive (1) or negative (0). The system uses these datasets to evaluate the accuracy of the Arabic

sentiment analysis system. The implementation is in Python, using the glove-python package to apply the GloVe model. (See Table 4.1.)

TABLE 4.1: Number of Reviews in the Datasets

Datasets	Number of Reviews
40k Egyptian tweet	40000 tweets
BOOK REVIEW	16448 tweets

2. Experimental Settings and parameters: First, the GloVe model was trained using Tensorflow tools to prepare for creating the models. The window size was set to 3 and the embedding dimensions to 300. A minimum frequency of 50 for each word was also set along with a learning rate of 0.05, Adam optimizer, and the model was trained for 20 epochs on the entire text. These settings enabled us to obtain a comprehensive and informative representation of the words in the text data, which was then utilized in developing the models for the following tasks.

TABLE 4.2: Experiment Settings

Setting	Value
Window Size	3
Embedding Dimensions	300
Minimum Frequency	50
Learning Rate	0.05
Optimizer	Adam
Epochs	20

In our research, we utilized two machine learning classifiers that have proven to be highly effective: Support Vector Machine (SVM) and Logistic Regression (LR). SVM is a well-established technique widely used in various natural language processing applications due to its superior performance and efficiency in text classification. On the other hand, LR is a discriminative and probabilistic algorithm that belongs to the log-linear family of classifiers, and it is used for binary classification. Both classifiers are known for their robustness and ability to handle high-dimensional data, making them ideal for sentiment analysis tasks.

We used the standard parameters for both SVM and LR classifiers in our experiments. To ensure the validity of our results, we employed cross-validation with the classifiers. We also randomly divided the datasets into two subsets, with 80% of the data utilized for training and the remaining 20% used for testing. This approach enabled us to evaluate the performance of the models on new, unseen data and estimate their ability to generalize to new instances.

3. Evaluation: The evaluation of the approaches was conducted using the confusion matrix (as shown in Table 4.3) and three metrics: Precision, Recall, and F1-score.

TABLE 4.3: Confusion Matrix

Actual		Total
Positive	**Negative**	
True Positive (TP)	False Negative (FN)	Actual Positive (AP)
False Positive (FP)	True Negative (TN)	Actual Negative (AN)
Total Predicted		Total

4.5.1 Results of GloVe and Bag of Words approaches

In the first experiment, which we labeled the Baseline approach, we utilized the GloVe model's vector representations with the SVM and LR classifiers on the EGYPTIAN TWEET and BOOK REVIEW datasets. The results obtained are presented in Table 4.4 and Table 4.5, respectively. The last two columns of these tables indicate the outcomes of the Bag of Word (BoW) technique applied to the same datasets and classifiers. The results in Table 4.4 and Table 4.5 demonstrate that the GloVe-based model outperformed the BoW technique, as evidenced by an increase in precision to 88% on the EGYPTIAN TWEET dataset and 83% on the BOOK REVIEW dataset.

TABLE 4.4: The Performance of GloVe baseline using EGYPTIAN TWEETS Dataset Compared with BoW Approach

	GloVe + SVM	**GloVe + LR**	**BoW + SVM**	**BoW + LR**
Precision	0.86	0.88	0.72	0.77
Recall	0.82	0.83	0.70	0.74
F1-score	0.84	0.85	0.71	0.75

TABLE 4.5: The Performance of GloVe baseline using BOOK REVIEW Dataset Compared with BoW Approach

	GloVe + SVM	**GloVe + LR**	**BoW + SVM**	**BoW + LR**
Precision	0.82	0.83	0.72	0.74
Recall	0.80	0.79	0.70	0.71
F1-score	0.81	0.81	0.71	0.72

4.5.2 Results of the proposed approach

The second experiment aimed to improve the previous method by incorporating the Root module. We used the vectors produced by this new method to develop models using the same classifiers and datasets as in the first experiment. The outcome of the second experiment is presented in Table 4.6 and Table 4.7.

TABLE 4.6: Performance of the proposed approach using EGYPTIAN TWEET Dataset

	Approach using SVM	**Approach using LR**
Precision	0.94	0.95
Recall	0.82	0.83
F1-score	0.87	0.88

TABLE 4.7: The Performance of the proposed approach using BOOK REVIEW Dataset.

	Approach Using SVM	**Approach using LR**
Precision	0.91	0.90
Recall	0.81	0.81
F1-score	0.86	0.85

According to Table Table 4.6 and Table 4.7, the suggested method has achieved the highest level of accuracy, and it has increased the precision of GloVE by as much as 8% with the SVM classifier and by 7% with the LR classifier on the EGYPTIAN TWEET dataset. For the BOOK REVIEW dataset, the new approach has boosted the precision of GloVE by 9%, the recall by 1%, and the F1 score by 5% when using the SVM classifier.
Results shown in Table 4.6 and Table 4.7 confirm that the proposed method tested on the EGYPTIAN TWEET dataset achieves the highest accuracy of 95%, which is a favorable outcome when compared to the BoW and baseline approaches. The performance of the two experiments is attributable to several factors, including the compacting of the matrix representation, which decreases sparsity, and the GloVe approach's ability to reduce the number of tokens by 50%. Moreover, the combination of BoR and GloVe further enhances this reduction by 30% on the EGYPTIAN TWEET dataset. The new approach has also reduced the tokens in the BOOK REVIEW dataset by nearly 50%. As a result, accuracy has improved while processing time has decreased (See Figure 4.5 and Figure 4.6) .

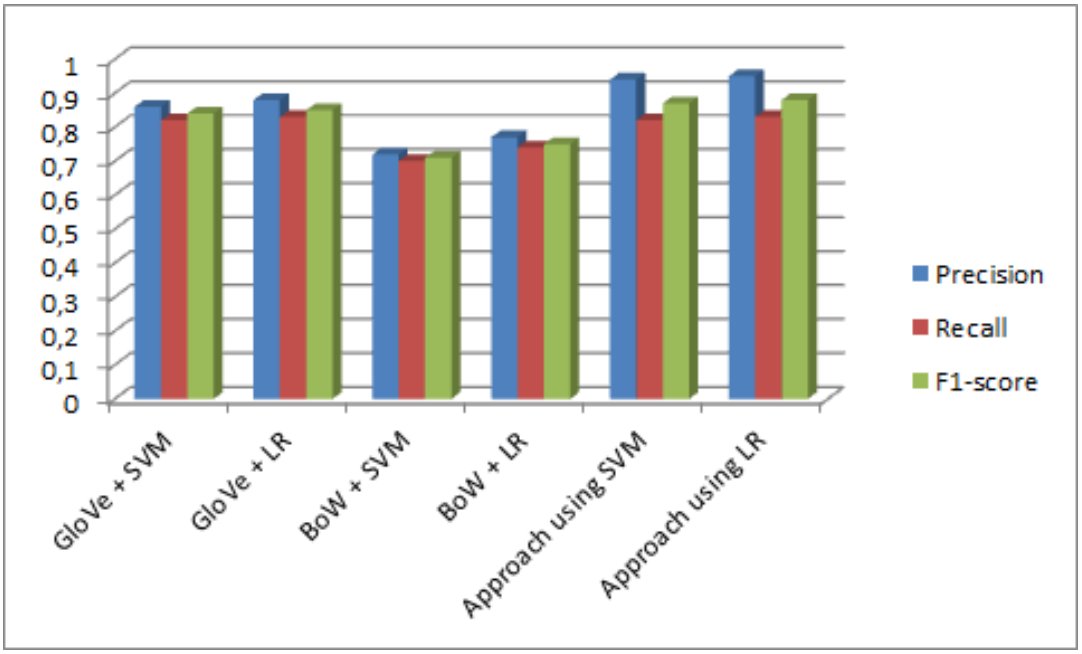

FIGURE 4.5: Illustration of approaches comparison fit on Egyptian Tweets Dataset

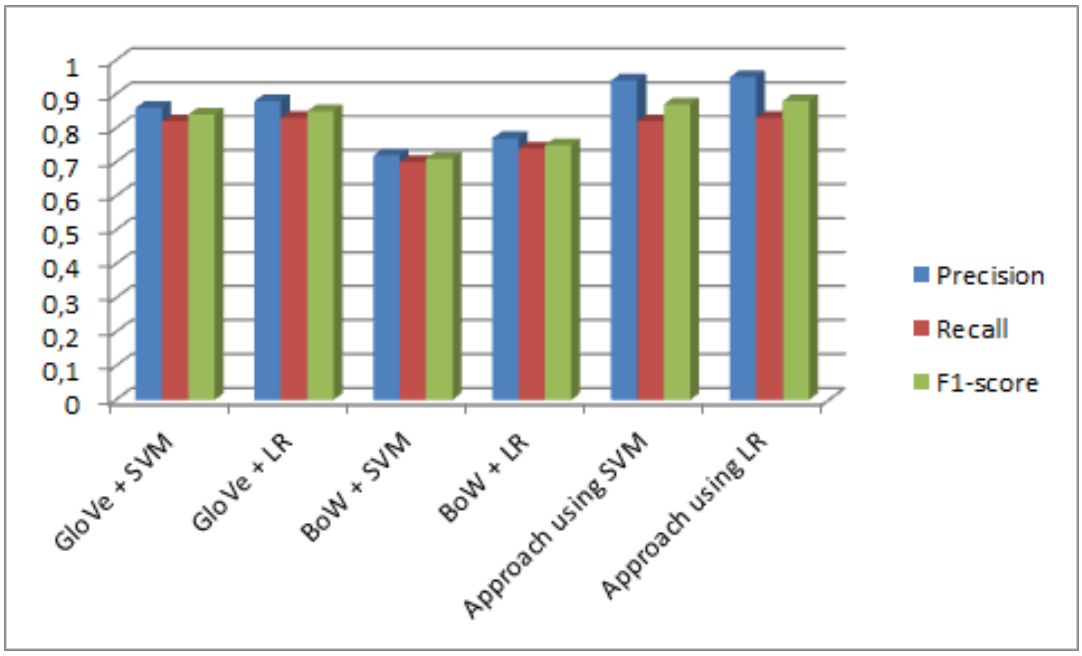

FIGURE 4.6: Illustration of approaches comparison fit on BOOK Review Dataset

Besides evaluating our suggested method against BoW and GloVe representations, we performed an extensive assessment that involved various Arabic baseline word embedding techniques, including Ar-FastText and word2vec. We have presented the comparison findings in Tables Table 4.8 and Table 4.9.

TABLE 4.8: The Performance of the baseline word embedding on EGYPTIAN TWEET Dataset.

	FastText + SVM	FastText + LR	Word2vec + SVM	Word2vec + SVM
Precision	0.80	0.79	0.80	0.81
Recall	0.78	0.81	0.79	0.80
F1-score	0.79	0.80	0.79	0.80

TABLE 4.9: The Performance of the baseline word embedding on BOOK REVIEW Dataset.

	FastText + SVM	FastText + LR	Word2vec + SVM	Word2vec + SVM
Precision	0.83	0.82	0.79	0.80
Recall	0.80	0.80	0.80	0.79
F1-score	0.81	0.81	0.79	0.79

The results of our experiments indicate that the novel technique we proposed performs better than all the baseline embedding methods. Even though these methods are widely used, they failed to achieve the high accuracy of 90% that our new method was able to achieve. These outcomes confirm the effectiveness of the new approach and emphasize the importance of the processing techniques applied to the dataset fed to the word embedding approach (See Figure 4.5 and Figure 4.6).

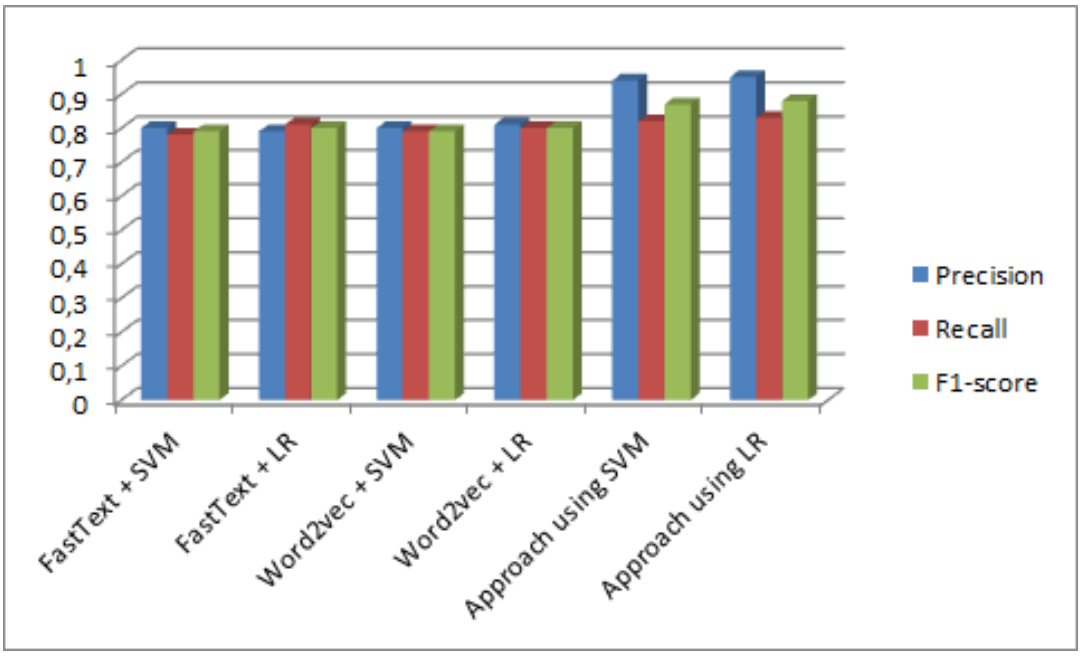

FIGURE 4.7: Illustration of comparison between Word embeddings and the new approach fit on Egyptian Tweets Dataset

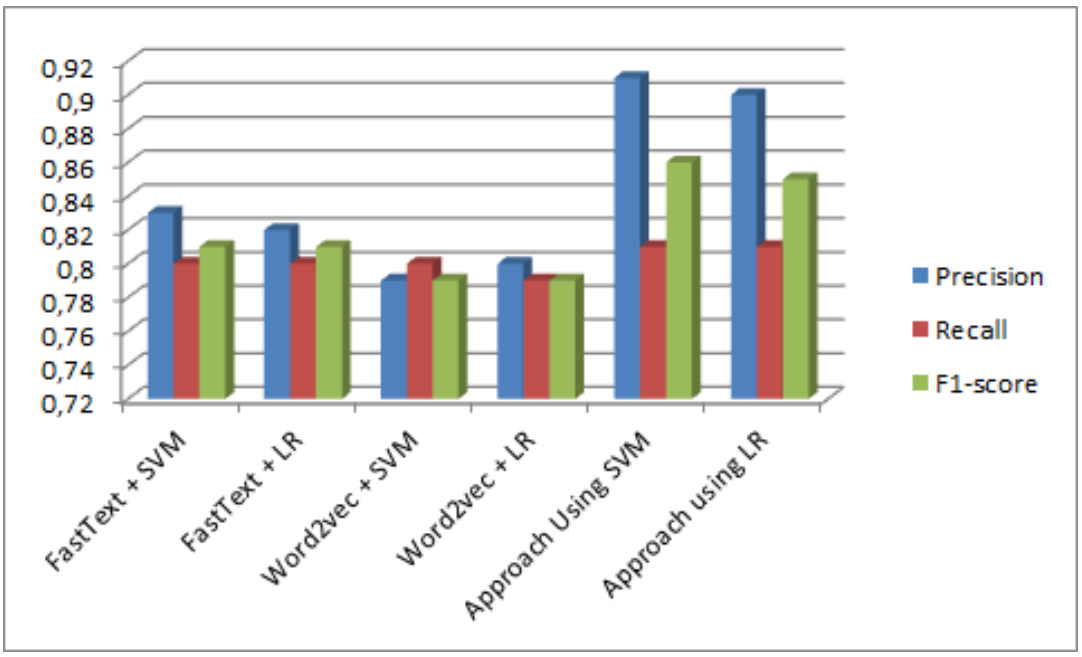

FIGURE 4.8: Illustration of comparison between Word embeddings and the new approach fit on BOOK Review Dataset

Specifically, the roots extraction module was identified as a crucial factor in enhancing the performance of the word embedding method. This supports the notion that pre-processing the dataset before providing it to the model can significantly impact the overall system performance.

4.6 Conclusion

In this study, we presented a novel technique to enhance the embedding representation of Arabic words for sentiment analysis on Arabic social media. Our approach involved the use of GloVe and BoR to reduce the vocabulary size and increase the density of the matrix representation.
The resulting models generated using this approach demonstrated a significant improvement in the results, enhancing the accuracy of Arabic social media sentiment analysis. However, there may be potential limitations of this new approach when it comes to dialects. In future work, we intend to include more text data to improve the word embedding corpus and concentrate on colloquial Arabic.

In the upcoming chapters, we will delve into the issue of working with imbalanced datasets, which is a common problem in machine learning applications. To overcome this challenge, we will explore a range of methods and techniques specifically designed for handling imbalanced datasets. These will include oversampling and undersampling techniques, such as random oversampling, SMOTE, and Tomek links, as well as hybrid techniques like SMOTE-Boundary and SMOTE-ENN.

Furthermore, we will analyze the impact of evaluation metrics, such as precision, recall, and F1-score, on the performance of imbalanced datasets. By using these methods and metrics, we aim to develop more accurate and robust models capable of handling imbalanced datasets effectively. This will help us mitigate the issues associated with biased or inaccurate models, which can be particularly problematic in critical applications like fraud detection and text classification. Through our exploration of these techniques, we hope to provide insights and practical solutions to help practitioners and researchers improve their machine learning models and make more informed decisions when working with imbalanced datasets.

Chapter 5

Balancing Approaches

5.1 Introduction

Text classification is a vital task in natural language processing (NLP) that involves assigning predefined categories or labels to text data. However, imbalanced datasets are a common issue in text classification, where one or more classes have significantly fewer instances than others. Imbalanced datasets pose a significant challenge to machine learning algorithms, as they tend to produce biased models that favor the majority class, leading to poor performance on the minority class (He2009).
To address the imbalanced dataset problem in text classification, various techniques have been proposed, including oversampling, undersampling, and hybrid approaches. Oversampling involves generating synthetic samples of the minority class to balance the class distribution (Sun2019). Undersampling involves removing instances from the majority class to balance the class distribution. Hybrid approaches combine oversampling and undersampling techniques to achieve a better balance between the classes.
In this chapter, we will explore the different techniques for balancing imbalanced datasets in text classification. We will discuss the advantages and disadvantages of oversampling, undersampling, and hybrid approaches, and their effectiveness on different datasets and classification tasks. We will also present the most popular oversampling and undersampling techniques, such as Random Oversampling, Synthetic Minority Oversampling Technique (SMOTE), Random Undersampling, and Tomek links, and how they work.
Additionally, we will discuss hybrid approaches, such as SMOTE and Tomek links, and how they can be applied in text classification.
In conclusion, balancing imbalanced datasets is an essential step in text classification to improve the classification performance on the minority class. Oversampling, undersampling, and hybrid approaches are effective techniques for balancing imbalanced datasets, and their choice depends on the specific characteristics of the dataset and the classification task. This chapter aims to provide a comprehensive overview of the different techniques for balancing imbalanced datasets in text classification and their practical applications.

5.2 Definition of Imbalanced Dataset

An **imbalanced dataset** refers to a dataset where the distribution of data points across different classes is significantly skewed, resulting in unequal representation of classes. In such datasets, the number of samples belonging to each class is not balanced, with some classes having a much larger number of instances compared to others. This imbalance in class distribution can arise due to various factors, such as

data collection processes, inherent characteristics of the underlying population, or the rarity of certain events or phenomena being studied. As a consequence, imbalanced datasets pose challenges in machine learning tasks, particularly in classification, as the models tend to be biased towards the majority classes. This bias can lead to suboptimal performance, where the minority classes are often overlooked or misclassified. Therefore, it is crucial to recognize and address the class imbalance issue in order to ensure fair representation and accurate predictions across all classes in the dataset.

Formally, let $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ represent a dataset, where x_i denotes the feature vector of the i-th data point, and y_i denotes its corresponding class label.

In an imbalanced dataset, there can be one or more minority classes and a majority class. Let K represent the total number of classes.

The minority classes refer to the classes with the smallest number of samples and are denoted as Class "M1", Class "M2", ..., Class "MK". The number of samples belonging to each minority class can be expressed as follows:
Minority Class (M1): $|\{(x_i, y_i) \in D : y_i = "M1"\}|$
Minority Class (M2): $|\{(x_i, y_i) \in D : y_i = "M2"\}|$
...
Minority Class (MK): $|\{(x_i, y_i) \in D : y_i = "MK"\}|$

The majority class refers to the class with the largest number of samples and is denoted as Class "R". The number of samples belonging to the majority class can be expressed as:

Majority Class (R): $|\{(x_i, y_i) \in D : y_i = "R"\}|$

The dataset is considered imbalanced when there is a significant difference in the number of samples between the majority class and the minority classes (See Figure 5.1).

5.3 Balancing Problem Definition

Balancing in machine learning refers to the process of equalizing the number of instances or samples for each class in a dataset. In classification tasks, the classes may represent different categories, such as positive and negative examples or different types of objects (Buda2018).
Imbalanced datasets can lead to biased models that favor the majority class, resulting in poor performance on the minority class. In other words, the classifier may be more accurate at predicting the majority class, while performing poorly on the minority class. This can be problematic in real-world applications where the cost of misclassifying the minority class can be much higher than that of the majority class.
To address the issue of imbalanced datasets, various techniques can be used to balance the classes. Oversampling involves generating synthetic samples of the minority class, while undersampling involves removing samples from the majority class. Hybrid approaches combine oversampling and undersampling techniques to achieve a better balance between the classes.
The aim of balancing is to ensure that the model is not biased towards any particular class and performs well on all classes. This is achieved by ensuring that the distribution of instances across all classes is balanced.

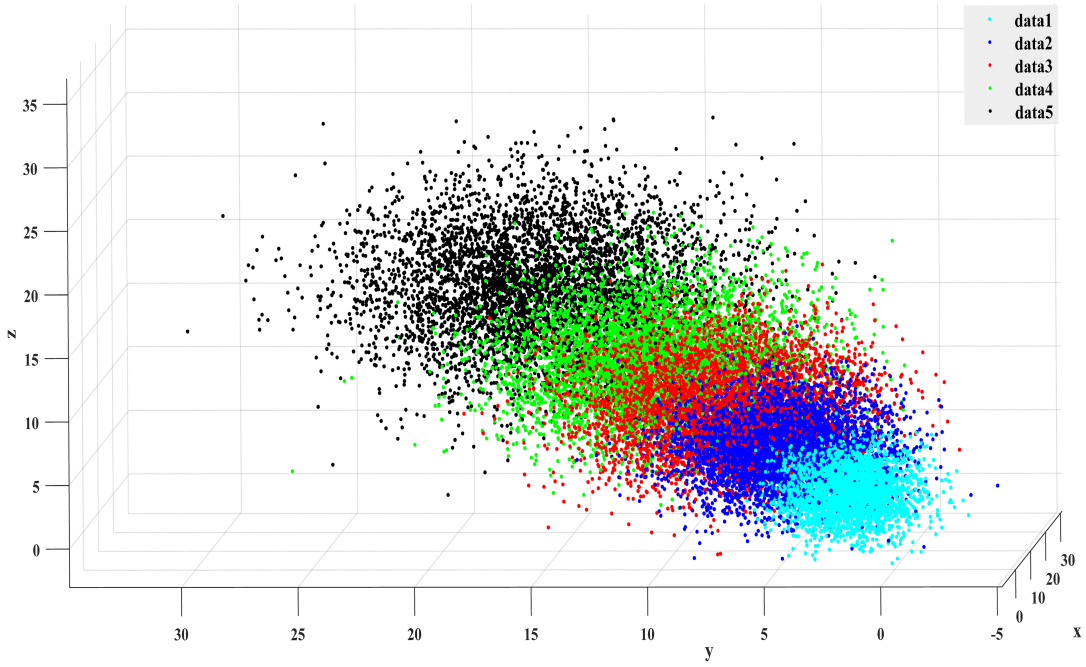

FIGURE 5.1: Class Distribution in an Imbalanced Dataset

Balancing techniques involve modifying the dataset to create a balanced distribution, which can be expressed mathematically as:

Let N_c be the number of samples in class c, and N_{max} be the maximum number of samples among all classes. The balanced dataset $\mathcal{D}_{bal}$ can be defined as:

$$\mathcal{D}_{bal} = \{(x_i, y_i) \mid (x_i, y_i) \in \mathcal{D} and y_i = c for i \in \{1, 2, \ldots, N_c\} where i s a class\}$$

The process of balancing aims to modify $\mathcal{D}$ to achieve approximately equal N_c values for all classes. By balancing the dataset, we reduce the bias towards the majority class and improve the performance of machine learning models on the minority classes.

Algorithm 12: Balancing algorithm

```
Input: Dataset D with imbalanced classes
Output: Balanced dataset D'
for each class C in D do
    Compute the number of instances n for class C;
end
Find the class with the maximum number of instances, max_n;
Initialize D' as an empty set;
for each class C in D do
    if n < max_n then
        if C is the minority class then
            Apply oversampling technique to generate synthetic instances
             until the number of instances equals max_n;
        end
        else if C is the majority class then
            Apply undersampling technique to reduce the number of
             instances until the number of instances equals max_n;
        end
        Add the new instances to D';
    end
    else
        Add all instances of class C to D';
    end
end
return D'
```

It's important to note that this algorithm is versatile and can be adjusted to accommodate various balancing techniques, such as oversampling, undersampling, and hybrid methods. The specific implementation of each technique will vary depending on the specific approach and the unique features of the dataset being analyzed.

In the next section, we will explore different sampling techniques that can be used to balance imbalanced datasets. Specifically, we will look at oversampling techniques, which involve generating synthetic samples of the minority class, and undersampling techniques, which involve removing samples from the majority class. We will also examine hybrid approaches that combine oversampling and undersampling techniques to achieve a better balance between the classes. By understanding the different sampling techniques and their implementation, we can choose the most appropriate method for our specific dataset and improve the performance of our machine learning models.

5.4 Oversampling

Oversampling is a technique used to address class imbalance in datasets, where the number of instances in one class is significantly lower than the number of instances in another class. This technique involves generating synthetic instances of the minority class to increase its representation in the dataset (Chawla2002).

The synthetic instances are created by randomly sampling from the existing instances of the minority class and applying a transformation or modification to each sample, such as changing the feature values or introducing noise. By increasing the number of minority class instances, oversampling can help improve the performance of machine learning models, which may otherwise be biased towards the majority class due to the imbalanced distribution of classes in the dataset.

Algorithm 13: Oversampling Algorithm

Input: Imbalanced dataset D
Output: Oversampled dataset D'

Compute the class distribution p_c for each class c;
Set $p_{max} = \max_c p_c$;
Set $n_{max} = |D| \cdot p_{max}$;
Initialize $D' = D$;
for *each class* c **do**
    Compute the number of samples n_c to generate as $n_c = n_{max} - |D'_c|$;
    while $n_c > 0$ **do**
    end
    Randomly select a sample x from D'_c;
    Generate a new sample x' by adding noise or applying a data augmentation technique;
    Add x' to D';
    Set $n_c \leftarrow n_c - 1$;
end
return D';

The goal of oversampling is to generate synthetic samples for the minority class(es) to balance the class distribution. Here are some common oversampling techniques:

5.4.1 Random oversampling

This technique involves randomly duplicating samples from the minority class until it reaches the same number of samples as the majority class (Batista2004).
The random oversampling algorithm is relatively simple. It involves the following steps:

1. Compute the number of samples in each class: Count the number of samples in each class to determine which class is the minority class.

2. Compute the difference in sample size between the minority and majority classes: Subtract the number of minority class samples from the number of majority class samples to determine the difference in sample size.

3. Randomly duplicate minority class samples: Select a random sample from the minority class and duplicate it until the difference in sample size is reduced to zero. Repeat this process until the number of samples in the minority class is equal to the number of samples in the majority class.

Random oversampling is a simple and effective way to balance class distribution, but it can lead to overfitting if the minority class samples are too similar to each other. In such cases, other oversampling techniques like SMOTE or ADASYN can be used

to generate more diverse synthetic samples.

Algorithm 14: Random Oversampling Algorithm

Input: Imbalanced dataset D
Output: Oversampled dataset D'
Compute the number of samples n_{max} in the majority class;
Select the minority class samples D_{min};
Randomly duplicate samples from D_{min} until $|D'_{min}| = n_{max}$;
Set $D' = D'_{min} \cup D_{maj}$;
return D';

5.4.2 SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE generates synthetic samples by interpolating between existing minority class samples. It selects a minority class sample, finds its k nearest neighbors in the feature space, and generates a new sample by linearly interpolating between the selected sample and one of its neighbors (Nunes2021).

The SMOTE algorithm can be summarized in the following steps:

1. Compute the number of samples in each class: Count the number of samples in each class to determine which class is the minority class.
2. Compute the difference in sample size between the minority and majority classes: Subtract the number of minority class samples from the number of majority class samples to determine the difference in sample size.
3. For each minority class sample, select k nearest neighbors: Use a distance metric to find the k nearest neighbors of each minority class sample in the feature space.
4. Generate synthetic samples: For each minority class sample, generate a new synthetic sample by interpolating between the selected sample and one of its neighbors. The interpolation is controlled by a random number between 0 and 1.
5. Add synthetic samples to the dataset: Add the synthetic samples to the minority class to create a new oversampled dataset.

SMOTE Formulation

Let X be the feature matrix and y be the corresponding class labels, where $X = \{x_1, x_2, \ldots, x_n\}$ and $y = \{y_1, y_2, \ldots, y_n\}$, n being the number of samples. SMOTE generates synthetic samples for the minority class to balance the class distribution. The formulation of SMOTE can be described as follows:

1. Select a minority class sample x_i.
2. Find the k nearest neighbors of x_i from the same class, denoted as $N_i = \{x_{i1}, x_{i2}, \ldots, x_{ik}\}$.
3. Randomly select one of the nearest neighbors, denoted as x_{ij}.

4. Generate a synthetic sample x_{new} by interpolating between x_i and x_{ij}:

$$x_{new} = x_i + \delta \cdot (x_{ij} - x_i)$$

where δ is a random number in the range $[0, 1]$.

5. Repeat steps 1-4 for a desired number of synthetic samples.

Algorithm 15: SMOTE Oversampling Algorithm

Input: Imbalanced dataset D
Output: Oversampled dataset D'
Compute the number of samples n_{max} in the majority class;
Select the minority class samples D_{min};
Set the number of synthetic samples N to generate as a percentage of $n_{max} - |D_{min}|$;
Compute the number of nearest neighbors k to use for SMOTE;
Compute the set of synthetic samples S using SMOTE;
Set $D' = D_{min} \cup S \cup D_{maj}$;
return D';

The number of nearest neighbors k is an important hyperparameter in the SMOTE algorithm, as it determines the amount of interpolation between samples. A larger k value leads to more smoothing and more conservative synthetic samples, while a smaller k value leads to more variability and potentially more noisy synthetic samples.
The choice of k can depend on the specific dataset and problem at hand. In general, a small k value (e.g., 1 or 2) may work well for datasets with well-defined clusters, while a larger k value (e.g., 5 or 10) may be more appropriate for datasets with more complex and overlapping classes. However, there is no universal rule for choosing the best k value, and it is often determined through experimentation and validation.
In addition to k, other hyperparameters in the SMOTE algorithm, such as the percentage of synthetic samples to generate and the distance metric used to measure the similarity between samples, may also need to be tuned to optimize performance. Cross-validation and grid search techniques can be used to find the best hyperparameters for a given dataset and problem.
SMOTE is a popular oversampling technique because it generates synthetic samples that are close to the real minority class samples, which helps to reduce overfitting. However, it may also generate noisy samples if the feature space is not well defined. Variants of SMOTE, such as Borderline SMOTE and ADASYN, have been proposed to address some of these limitations.

5.4.3 ADASYN (Adaptive Synthetic Sampling)

ADASYN generates synthetic samples in a similar way to SMOTE but with a focus on adapting to the local density of the feature space. ADASYN generates more synthetic samples for minority class samples that are harder to learn by classifiers (He2008).
The ADASYN algorithm works as follows:

1. Compute the number of samples in each class: Count the number of samples in each class to determine which class is the minority class.

2. Compute the imbalance ratio: Compute the ratio of the number of majority class samples to the number of minority class samples. each minority class sample, compute its density distribution: Use a distance metric to find the k nearest neighbors of each minority class sample in the feature space. Compute the density distribution of the minority class samples in the vicinity of each selected sample.

3. Compute the synthetic sample density distribution: For each minority class sample, compute the density distribution of the synthetic samples to generate in the vicinity of that sample. The density distribution is proportional to the density distribution of the minority class samples in that region.

4. Generate synthetic samples: For each minority class sample, generate synthetic samples in the vicinity of that sample according to the density distribution computed in step 4.

5. Add synthetic samples to the dataset: Add the synthetic samples to the minority class to create a new oversampled dataset.

ADASYN Formulation

Let X be the feature matrix and y be the corresponding class labels, where $X = \{x_1, x_2, \ldots, x_n\}$ and $y = \{y_1, y_2, \ldots, y_n\}$, n being the number of samples. ADASYN generates synthetic samples for the minority class to balance the class distribution. The formulation of ADASYN can be described as follows:

1. Compute the density distribution D for each sample x_i:

$$D(x_i) = \frac{\sum_{j=1}^{n} K(x_i, x_j) \cdot \mathbf{1}(y_j = y_i)}{\sum_{j=1}^{n} K(x_i, x_j)}$$

   where $K(\cdot, \cdot)$ is a kernel function (e.g., Gaussian kernel) and $\mathbf{1}(\cdot)$ is the indicator function.

2. Compute the normalized density distribution D':

$$D'(x_i) = \frac{D(x_i)}{\sum_{j=1}^{n} D(x_j)}$$

3. Compute the imbalance ratio IR:

$$IR = \frac{numberofmajorityclasssamples}{numberofminorityclasssamples}$$

4. Compute the target number of synthetic samples G_i for each minority sample x_i:

$$G_i = IR \cdot D'(x_i)$$

5. Generate G_i synthetic samples for each minority sample x_i by applying the SMOTE algorithm with k nearest neighbors.

Algorithm 16: ADASYN Oversampling Algorithm

Input: Imbalanced dataset D
Output: Oversampled dataset D'

Compute the number of samples n_{maj} in the majority class;
Compute the number of samples n_{min} in the minority class;
Compute the imbalance ratio $r = n_{maj}/n_{min}$;
Select the minority class samples D_{min};
For each minority class sample $x_i \in D_{min}$, compute the number of its k nearest neighbors in D; Compute the density distribution D_i of x_i based on its k nearest neighbors; Compute the normalized density distribution $p_i = D_i / \sum_{j \in D_{min}} D_j$; Compute the number of synthetic samples N_i to generate
for each minority class sample x_i as $N_i = \lfloor r \cdot p_i \rfloor$;
For each minority class sample $x_i \in D_{min}$, generate N_i synthetic samples in the vicinity of x_i using the density distribution D_i;
Set $D' = D_{min} \cup S \cup D_{maj}$;
return D';

In this algorithm, k is the number of nearest neighbors used to compute the density distribution of each minority class sample, and r is the imbalance ratio of the dataset. The density distribution D_i of a minority class sample x_i is computed as the sum of the distances between x_i and its k nearest neighbors, and the normalized density distribution p_i is computed as the ratio of D_i to the sum of the density distributions of all minority class samples.
The number of synthetic samples N_i to generate for each minority class sample x_i is proportional to its density distribution D_i and the imbalance ratio r. The synthetic samples are generated in the vicinity of x_i using the density distribution D_i.
Finally, the oversampled dataset D' is created by combining the original minority and majority class samples with the synthetic samples.

ADASYN differs from SMOTE in that it generates more synthetic samples in regions of the feature space where the density of minority class samples is low, and fewer synthetic samples in regions where the density is high. This helps to balance the density distribution of the minority class and avoid overfitting.
Like SMOTE, ADASYN can be combined with other techniques such as random undersampling or Tomek links to further improve the performance of the oversampling method.

5.4.4 Borderline SMOTE Sampling

Borderline SMOTE is an extension of SMOTE that only generates synthetic samples for borderline minority class samples, which are defined as minority class samples that are misclassified by a k-NN classifier on the majority class samples (Han2005).

The Borderline SMOTE algorithm works as follows:

1. Compute the number of samples in each class: Count the number of samples in each class to determine which class is the minority class.
2. Identify the borderline samples: Identify the minority class samples that are in the vicinity of the majority class samples (i.e., the samples that are misclassified or difficult to classify).

3. Compute the number of synthetic samples to generate: For each borderline sample, compute the number of synthetic samples to generate using the SMOTE algorithm.

4. Generate synthetic samples: For each borderline sample, generate the desired number of synthetic samples using the SMOTE algorithm.

5. Add synthetic samples to the dataset: Add the synthetic samples to the minority class to create a new oversampled dataset.

Borderline SMOTE Formulation

Let X be the feature matrix and y be the corresponding class labels, where $X = \{x_1, x_2, \ldots, x_n\}$ and $y = \{y_1, y_2, \ldots, y_n\}$, n being the number of samples. Borderline SMOTE generates synthetic samples for the minority class, focusing on the samples that are near the borderline between the minority and majority classes. The formulation of Borderline SMOTE can be described as follows:

1. Identify the minority class samples that are located near the borderline between the minority and majority classes. This can be done by using a predefined metric, such as k-nearest neighbors, to measure the proximity of each minority sample to the majority samples.

2. For each identified borderline minority sample x_i:

   (a) Find the k nearest neighbors of x_i from the same class, denoted as $N_i = \{x_{i1}, x_{i2}, \ldots, x_{ik}\}$.

   (b) Randomly select one of the nearest neighbors, denoted as x_{ij}.

   (c) Generate a synthetic sample x_{new} by interpolating between x_i and x_{ij}:

   $$x_{new} = x_i + \delta \cdot (x_{ij} - x_i)$$

   where δ is a random number in the range $[0, 1]$.

   (d) Repeat steps (a)-(c) for a desired number of synthetic samples.

3. Repeat steps 2-3 for each identified borderline minority sample.

The Borderline SMOTE algorithm only generates synthetic samples for the minority class samples that are near the decision boundary between the minority and

majority classes. This reduces the risk of overfitting and improves the generalization of the oversample

Algorithm 17: Borderline SMOTE Oversampling Algorithm

Input: Imbalanced dataset D
Output: Oversampled dataset D'
Compute the number of samples n_{maj} in the majority class;
Compute the number of samples n_{min} in the minority class;
Compute the imbalance ratio $r = n_{maj}/n_{min}$;
Identify the minority class borderline samples D_b;
For each minority class borderline sample $x_i \in D_b$, compute the number of its k nearest neighbors in D; Compute the number of synthetic samples N_i to generate
for each borderline sample x_i as $N_i = \lfloor r \cdot \frac{D_i}{k} \rfloor$;
For each minority class borderline sample $x_i \in D_b$, generate N_i synthetic samples in the vicinity of x_i using the SMOTE algorithm;
Set $D' = D_{min} \cup S \cup D_{maj}$;
return D';

In this algorithm, k is the number of nearest neighbors used to identify the minority class borderline samples, and r is the imbalance ratio of the dataset. The number of synthetic samples N_i to generate for each borderline sample x_i is proportional to the density distribution D_i of the sample, which is defined as the number of its k nearest neighbors that belong to the minority class.

The synthetic samples are generated using the SMOTE algorithm in the vicinity of each borderline sample x_i. Unlike the original SMOTE algorithm, Borderline SMOTE only generates synthetic samples for the minority class samples that are near the decision boundary between the minority and majority classes, which reduces the risk of overfitting.

Finally, the oversampled dataset D' is created by combining the original minority and majority class samples with the synthetic samples.

5.5 Undersampling

Undersampling is a technique for addressing imbalanced datasets in machine learning, where the number of samples in one class is much larger than the number of samples in the other class(es) (Jiang2021). This technique involves reducing the number of samples in the majority class to achieve a balance between the classes, which can improve the performance of machine learning models that are biased towards the majority class. Undersampling can be done in several ways, such as randomly removing majority class samples or selecting a representative subset of the majority class using clustering algorithms or other techniques (See Algorithm 12). The choice of undersampling technique should be based on the characteristics of the dataset and the specific problem being addressed.

Undersampling Formulation

Let X be the feature matrix and y be the corresponding class labels, where $X = \{x_1, x_2, \ldots, x_n\}$ and $y = \{y_1, y_2, \ldots, y_n\}$, n being the number of samples. Undersampling aims to reduce the number of majority class samples to balance the class distribution. The formulation of undersampling can be described as follows:

1. Identify the majority class samples that need to be removed. This can be done by various techniques such as random undersampling, cluster-based undersampling, or Tomek links.
2. For each identified majority class sample x_i:
   (a) Remove the sample x_i from the dataset.
   (b) Update the feature matrix X and class labels y accordingly.
3. Repeat steps 2-3 for each identified majority class sample that needs to be removed.

here is some more detail on the common undersampling techniques:

5.5.1 Random undersampling

This method randomly selects a subset of the majority class samples to match the number of samples in the minority class. It is a simple and fast approach, but it may result in a loss of information and can lead to overfitting.
To implement random undersampling, we first need to identify the minority class and majority class in the dataset. Next, we randomly select samples from the majority class and remove them until the number of samples in the majority class matches that of the minority class (Kaur2021).
Random undersampling is a simple technique that can quickly balance an imbalanced dataset. However, it may lead to a loss of information and may not be effective in preserving the underlying structure of the data. Therefore, it is often combined with other techniques, such as oversampling or synthetic oversampling, to generate a balanced dataset that preserves the distribution and structure of the original data while also reducing the impact of class imbalance.

Random Undersampling Formulation

Let X be the feature matrix and y be the corresponding class labels, where $X = \{x_1, x_2, \ldots, x_n\}$ and $y = \{y_1, y_2, \ldots, y_n\}$, n being the number of samples. Random undersampling aims to reduce the number of majority class samples to balance the class distribution. The formulation of random undersampling can be described as follows:

1. Identify the majority class samples that need to be removed randomly. The number of majority class samples to be removed can be determined based on the desired class imbalance ratio.
2. Randomly select the identified majority class samples to be removed.
3. For each selected majority class sample x_i:
   (a) Remove the sample x_i from the dataset.
   (b) Update the feature matrix X and class labels y accordingly.
4. Repeat steps 2-3 for each selected majority class sample that needs to be removed.

Algorithm 18: Random Undersampling Algorithm

Input: Dataset D, Majority class label M, Minority class label m, Ratio of majority to minority samples r
Output: Undersampled dataset D'

```
Calculate the number of minority class samples, N_m, in D; Calculate the
 number of majority class samples to keep, N_M = N_m * r; Initialize an empty
 set D' to hold the undersampled dataset;
for each sample x in D do
    if label(x) = m then
        Add x to D';
    end
    else if label(x) = M then
        if number of samples in D' < N_M then
            Add x to D';
        end
    end
end
```

Note that in this algorithm, we randomly remove majority class samples until the number of majority class samples in the undersampled dataset matches the desired ratio of majority to minority samples. The resulting dataset D' is then used for training machine learning models.

5.5.2 Tomek links undersampling

This method involves identifying pairs of samples from different classes that are nearest to each other, known as Tomek links, and removing the majority class samples from these pairs. This approach can be effective at reducing noise and improving the performance of the model (Bunkhumpornpat2009).
A Tomek link is defined as a pair of samples (x_i, x_j) from different classes such that there is no sample x_k between them that belongs to either class. In other words, x_i and x_j are the closest neighbors to each other from different classes. Removing the sample that is closer to the other class's centroid can help to improve the separation between the two classes.
To implement Tomek links, we first identify all pairs of samples in the dataset that form a Tomek link. Next, we remove the majority class sample in each Tomek link, effectively reducing the number of samples in the majority class and increasing the distance between the two classes.
Tomek links is a simple and effective undersampling technique, but it can lead to an overfitting problem when the number of minority class samples is very small. Therefore, it is often used in combination with other techniques, such as synthetic oversampling or random undersampling, to achieve better performance.

Tomek Links Undersampling Formulation

Let X be the feature matrix and y be the corresponding class labels, where $X = \{x_1, x_2, \ldots, x_n\}$ and $y = \{y_1, y_2, \ldots, y_n\}$, n being the number of samples. Tomek links undersampling aims to remove majority class samples that form Tomek links with minority class samples, in order to improve the class separation. The formulation of Tomek links undersampling can be described as follows:

1. Identify the pairs of samples that form Tomek links. A Tomek link exists between two samples x_i and x_j if they belong to different classes and there are no other samples closer to x_i than x_j.
2. For each identified Tomek link between a minority class sample x_i and a majority class sample x_j:
   (a) Remove the majority class sample x_j from the dataset.
   (b) Update the feature matrix X and class labels y accordingly.
3. Repeat steps 2-3 for each identified Tomek link between a minority class sample and a majority class sample.

Algorithm 19: Tomek Links Undersampling Algorithm

Input: Dataset D, Majority class label M, Minority class label m
Output: Undersampled dataset D'

```
Identify all pairs of samples in D that form a Tomek link;
for each Tomek link (x_i, x_j) do
    if label(x_i) = M and label(x_j) = m then
        Remove x_i from D;
    end
    else if label(x_i) = m and label(x_j) = M then
        Remove x_j from D;
    end
end
D' = D;
```

Note that in this algorithm, we identify all pairs of samples that form a Tomek link by finding the nearest neighbors of each sample from the opposite class. Then, we remove the majority class sample in each Tomek link to obtain the undersampled dataset D'.

5.5.3 Edited nearest neighbors (ENN) undersampling

This method involves identifying samples in the majority class that are misclassified by their nearest neighbors in the same class and removing them. This technique can reduce noise and improve the decision boundary between classes (Panda2021).

The ENN algorithm works by first identifying the k nearest neighbors for each sample in the dataset. Then, it calculates the class difference between each sample and its neighbors. If the majority class is overrepresented among the neighbors of a minority class sample, it is considered to be a noisy or misleading sample and is removed from the dataset.

ENN Undersampling Formulation

Let X be the feature matrix and y be the corresponding class labels, where $X = \{x_1, x_2, \ldots, x_n\}$ and $y = \{y_1, y_2, \ldots, y_n\}$, n being the number of samples. Edited Nearest Neighbors (ENN) undersampling aims to remove majority class samples

that are misclassified by their nearest neighbors from the same class, in order to improve the class separation. The formulation of ENN undersampling can be described as follows:

1. For each majority class sample x_i:
   (a) Compute the nearest neighbors of x_i from the same class, denoted as $N_i = \{x_{i1}, x_{i2}, \ldots, x_{ik}\}$.
   (b) Determine the class label of x_i based on the majority vote of its nearest neighbors.
   (c) If the class label of x_i does not match the majority class label, remove x_i from the dataset.
2. Update the feature matrix X and class labels y accordingly after removing the misclassified majority class samples.

ENN is a simple and computationally efficient undersampling technique. However, it may not work well when the dataset is highly imbalanced or when the minority class samples are scattered throughout the feature space. In such cases, other undersampling techniques such as Tomek links or CNN may be more effective.

Algorithm 20: ENN Undersampling Algorithm

Input: Dataset D, Majority class label M, Minority class label m, Number of nearest neighbors k
Output: Undersampled dataset D'

```
Calculate the k nearest neighbors of each sample in D; for each sample x_i in D do
    if label(x_i) = m then
        Let NN(x_i) be the k nearest neighbors of x_i; if the number of majority
         class samples in NN(x_i) > the number of minority class samples in
         NN(x_i) then
            Remove x_i from D;
        end
    end
end
D' = D;
```

Note that in this algorithm, we calculate the k nearest neighbors for each sample in the dataset using a distance metric such as Euclidean distance. Then, we iterate over each minority class sample in the dataset and examine its k nearest neighbors. If the majority class is overrepresented among the neighbors, we remove the minority class sample from the dataset to obtain the undersampled dataset D'.

5.5.4 Boundary Undersampling

Boundary Undersampling (BU) is a hybrid sampling technique that combines undersampling with the Boundary Cleaning (BC) method to address the issue of imbalanced datasets in classification tasks (Liu2021).

The Boundary Undersampling algorithm works as follows:

1. Identify the borderline samples: Samples that are on the boundary between the minority and majority classes are identified. These samples are those that

have at least one neighbor belonging to the majority class and one neighbor belonging to the minority class.

2. Boundary Cleaning (BC): The BC method is used to remove noisy borderline samples. This is done by examining the k-nearest neighbors of each borderline sample. If the majority class samples are in the majority among the k-nearest neighbors, the sample is removed. Otherwise, it is kept in the dataset.

3. Undersampling: After cleaning the noisy borderline samples, a random undersampling method is applied to the majority class to remove a portion of the samples until the desired balance ratio is reached.

4. Combine datasets: The minority class and majority class samples remaining after the BU process are combined to form the final balanced dataset.

Boundary Undersampling Formulation

Let X be the feature matrix and y be the corresponding class labels, where $X = \{x_1, x_2, \ldots, x_n\}$ and $y = \{y_1, y_2, \ldots, y_n\}$, n being the number of samples. Boundary undersampling aims to remove majority class samples that are near the decision boundary, in order to improve the class separation. The formulation of Boundary Undersampling can be described as follows:

1. Compute the decision boundary of the classifier trained on the original dataset.

2. For each majority class sample x_i:

   (a) Compute the distance from x_i to the decision boundary.

   (b) If the distance is smaller than a predefined threshold, remove x_i from the dataset.

3. Update the feature matrix X and class labels y accordingly after removing the majority class samples near the decision boundary.

By combining undersampling with the BC method, the Boundary Undersampling approach is able to remove noisy samples from the borderline between the

minority and majority classes, while also reducing the number of majority class samples to achieve a more balanced dataset.

Algorithm 21: Boundary Cleaning Algorithm

Input: Dataset with class labels $y_1, y_2, ..., y_n$ and features $x_1, x_2, ..., x_n$
Output: Cleaned dataset $x'_+ \cup x'_-$, $y'_+ \cup y'_-$

1 Identify the borderline samples:

1. For each minority class sample x_i, identify its k-nearest neighbors in the dataset
2. If at least one neighbor belongs to the majority class and at least one neighbor belongs to the minority class, mark x_i as a borderline sample

Clean borderline samples using BC:

1. For each borderline sample x_i with k nearest neighbors:
2. Calculate the ratio $r = \frac{majorityclassneighbors}{minorityclassneighbors}$
3. If $r \geq 1$, remove x_i from the dataset
4. Else, keep x_i in the dataset

Combine datasets: $X' = x'_+ \cup x'_-$, $Y' = y'_+ \cup y'_-$, where x'_+ and y'_+ are the minority class samples and their corresponding labels that are not removed by BC, and x'_- and y'_- are the majority class samples and their corresponding labels after undersampling, if applicable;

It is worth noting that the performance of Boundary Undersampling can be influenced by the choice of parameters, such as the number of neighbors in BC and the balance ratio. These parameters can be tuned to optimize the performance of the classification model on the balanced dataset.

5.6 Hybrid sampling

Hybrid sampling is an approach in imbalanced classification problems where both undersampling and oversampling techniques are used to balance the class distribution of the dataset. The idea behind hybrid sampling is to first apply an undersampling technique to remove some of the majority class samples and then apply an oversampling technique to generate synthetic minority class samples (Kumar2018). The resulting dataset has a more balanced class distribution, which can improve the performance of classification models. Hybrid sampling can help to address the challenges posed by imbalanced datasets by creating a more representative dataset for training and evaluation of classification models.

HYBRID Formulation

Let X be the feature matrix and y be the corresponding class labels, where $X = \{x_1, x_2, \ldots, x_n\}$ and $y = \{y_1, y_2, \ldots, y_n\}$, n being the number of samples. The HYBRID technique combines oversampling and undersampling methods to address class imbalance. The formulation of the general HYBRID technique can be described as follows:

1. Apply an undersampling technique to reduce the majority class samples in the dataset.
2. Apply an oversampling technique to augment the minority class samples in the dataset.
3. Optionally, perform further iterations of undersampling and oversampling to refine the class distribution.
4. Update the feature matrix X and class labels y accordingly after applying the hybridization.

Here are some examples of hybrid sampling approaches classified by their performance in text classification:

5.6.1 Random undersampling and oversampling

This approach involves randomly removing some samples from the majority class (undersampling) and duplicating some samples from the minority class (oversampling) to balance the dataset. This approach is easy to implement but may result in loss of important information from the majority class and overfitting to the minority class (Batista2003).

1. Load and preprocess the text dataset: The first step is to load the text dataset and perform any necessary preprocessing steps, such as cleaning, tokenization, and vectorization.
2. Calculate the class distribution: Calculate the number of samples in each class to determine the level of class imbalance in the dataset.
3. Implement random undersampling: Randomly remove some samples from the majority class so that the number of samples in the majority class is reduced to be closer to the number of samples in the minority class. The ratio of the number of samples between the classes can be set to a desired value.
4. Implement random oversampling: Randomly duplicate some samples from the minority class so that the number of samples in the minority class is increased to be closer to the number of samples in the majority class. The ratio of the number of samples between the classes can be set to a desired value.
5. Combine the undersampled and oversampled datasets: Combine the undersampled and oversampled datasets to create a new balanced dataset.

Rundom Over & Random Under Formulation

Let X be the feature matrix and y be the corresponding class labels, where $X = \{x_1, x_2, \ldots, x_n\}$ and $y = \{y_1, y_2, \ldots, y_n\}$, n being the number of samples. The RURO technique combines random undersampling and random oversampling to address class imbalance. The formulation of the RURO technique can be described as follows:

1. Randomly select a subset of majority class samples to be removed using random undersampling. The number of majority class samples to be removed can be determined based on the desired class imbalance ratio.

2. Randomly select a subset of minority class samples to be duplicated using random oversampling. The number of minority class samples to be duplicated can be determined based on the desired class imbalance ratio.

3. Update the feature matrix X and class labels y accordingly after applying the random under and random over steps.

Algorithm 22: Hybrid Random Undersampling and Oversampling Algorithm

Input: Dataset with class labels $y_1, y_2, ..., y_n$ and features $x_1, x_2, ..., x_n$
Output: Balanced dataset $x'_+ \cup x'_-, y'_+ \cup y'_-$

Calculate class distribution N_+ and N_-;
Calculate undersampling ratio $p_u = \frac{N_+}{N_-}$;
Calculate oversampling ratio $p_o = \frac{N_-}{N_+}$;
Undersample majority class:
$x'_- = x_i | y_i = y_- and r_i = 1$,
where r_i is a random variable that equals 1 with probability p_u and 0 otherwise;
Oversample minority class:
$x'_+ = x_i | y_i = y_+ and r_i = 1$,
where r_i is a random variable that equals 1 with probability p_o and 0 otherwise;
Combine datasets:
$X' = x'_+ \cup x'_-$,
$Y' = y_+ \times |x'_+| \cup y_- \times |x'_-|$,
where $|x'_+|$ and $|x'_-|$ are the sizes of the oversampled and undersampled datasets, respectively;
Shuffle the balanced dataset;

Note that the algorithm can be modified to include hyperparameters for tuning the values of p_u and p_o, as well as other parameters such as the number of folds in cross-validation and the performance metric used for evaluation. Additionally, other techniques such as synthetic sampling can be incorporated into the algorithm to further improve the performance of the classification model.

5.6.2 SMOTE-Tomek Sampling

This approach combines SMOTE (Synthetic Minority Over-sampling Technique) and Tomek links to remove samples from the majority class that are close to the minority class (Tomek links) and generate synthetic samples from the minority class (SMOTE). This approach has been shown to improve the performance of text classification models by balancing the dataset and reducing the overlapping between classes (Salloum2020).
SMOTE-Tomek consists of two steps:

SMOTE: Synthetic minority over-sampling technique (SMOTE) is used to generate synthetic samples for the minority class by interpolating between the minority class samples. This helps to balance the class distribution by increasing the number of minority class samples.

Tomek Links: Tomek links are pairs of samples from different classes that are closest to each other, and removing these pairs from the dataset can help to improve

the class separability. In SMOTE-Tomek, Tomek links are identified and the majority class samples involved in Tomek links are removed to further improve the balance between the classes. The SMOTE-Tomek approach combines the strengths of SMOTE in generating synthetic samples and Tomek links in removing noisy samples to improve the performance of classification models in imbalanced datasets.

SMOTE-Tomek Links Formulation

Let X be the feature matrix and y be the corresponding class labels, where $X = \{x_1, x_2, \ldots, x_n\}$ and $y = \{y_1, y_2, \ldots, y_n\}$, n being the number of samples. The SMOTE-Tomek Links technique combines the Synthetic Minority Over-sampling Technique (SMOTE) and Tomek links undersampling to address class imbalance. The formulation of the SMOTE-Tomek Links technique can be described as follows:

1. Apply SMOTE to generate synthetic samples for the minority class, as described in the SMOTE formulation.
2. Identify the pairs of samples that form Tomek links between the minority and majority classes. A Tomek link exists between two samples x_i and x_j if they belong to different classes and there are no other samples closer to x_i than x_j.
3. For each identified Tomek link between a minority class sample x_i and a majority class sample x_j:
   (a) Remove both samples x_i and x_j from the dataset.
   (b) Update the feature matrix X and class labels y accordingly.
4. Update the feature matrix X and class labels y accordingly after applying the SMOTE-Tomek Links steps.

Algorithm 23: SMOTE-Tomek Algorithm

Input: Dataset with class labels $y_1, y_2, ..., y_n$ and features $x_1, x_2, ..., x_n$
Output: Balanced dataset $x'_+ \cup x'_-, y'_+ \cup y'_-$

Calculate class distribution N_+ and N_-;
Calculate the number of synthetic samples to generate n_{syn};
Generate synthetic minority class samples using SMOTE;
Identify Tomek links between classes;
Remove majority class samples involved in Tomek links;
Combine datasets:
$X' = x'_+ \cup x'_-$,
$Y' = y_+ \times |x'_+| \cup y_- \times |x'_-|$,
where $|x'_+|$ and $|x'_-|$ are the sizes of the oversampled and undersampled datasets, respectively;
Shuffle the balanced dataset;

Note that the algorithm can be modified to include hyperparameters for tuning the number of synthetic samples to generate, as well as other parameters such as the distance metric used in identifying Tomek links and the performance metric used for evaluation. Additionally, other techniques such as ensemble methods can be incorporated into the algorithm to further improve the performance of the classification model.

5.6.3 SMOTE-ENN Sampling

This approach combines SMOTE and Edited Nearest Neighbors (ENN) to generate synthetic samples from the minority class (SMOTE) and remove noisy samples from the majority class that are misclassified by the k-nearest neighbors algorithm (ENN). This approach has been shown to improve the performance of text classification models by balancing the dataset and reducing the influence of noisy samples (Li2021). The SMOTE-ENN algorithm works in the following steps:

SMOTE: Synthetic Minority Over-sampling Technique (SMOTE) is used to generate synthetic samples for the minority class by interpolating between the minority class samples. This helps to balance the class distribution by increasing the number of minority class samples.

ENN: Edited Nearest Neighbors (ENN) is used to identify and remove noisy samples. ENN works by examining the k-nearest neighbors of each sample in the dataset and removing samples that do not have a consistent class label with their neighbors. This helps to improve the quality of the minority class samples by removing noisy samples.

Combine datasets: The datasets generated by SMOTE and ENN are combined to form the final balanced dataset.

By combining the SMOTE and ENN methods, the SMOTE-ENN approach is able to address both the class imbalance problem and the issue of noisy samples in imbalanced datasets.

SMOTE-ENN Formulation

Let X be the feature matrix and y be the corresponding class labels, where $X = \{x_1, x_2, \ldots, x_n\}$ and $y = \{y_1, y_2, \ldots, y_n\}$, n being the number of samples. The SMOTE-ENN technique combines the Synthetic Minority Over-sampling Technique (SMOTE) and Edited Nearest Neighbors (ENN) undersampling to address class imbalance. The formulation of the SMOTE-ENN technique can be described as follows:

1. Apply SMOTE to generate synthetic samples for the minority class, as described in the SMOTE formulation.

2. Apply ENN to remove majority class samples that are misclassified by their nearest neighbors from the same class, as described in the ENN formulation.

3. Update the feature matrix X and class labels y accordingly after applying the SMOTE-ENN steps.

Algorithm 24: SMOTE-ENN Algorithm

Input: Dataset with class labels $y_1, y_2, ..., y_n$ and features $x_1, x_2, ..., x_n$
Output: Balanced dataset $x'_+ \cup x'_-, y'_+ \cup y'_-$

1 Calculate class distribution N_+ and N_-;
2 Calculate the number of synthetic samples to generate n_{syn};
3 Generate synthetic minority class samples using SMOTE;
4 Identify noisy samples using ENN;
5 Remove noisy samples;
6 Combine datasets:
7 $X' = x'_+ \cup x'_-$,
8 $Y' = y_+ \times |x'_+| \cup y_- \times |x'_-|$,
9 where $|x'_+|$ and $|x'_-|$ are the sizes of the oversampled and undersampled datasets, respectively;
10 Shuffle the balanced dataset;

Note that the algorithm can be modified to include hyperparameters for tuning the number of synthetic samples to generate, as well as other parameters such as the number of neighbors to consider in ENN and the performance metric used for evaluation. Additionally, other techniques such as ensemble methods can be incorporated into the algorithm to further improve the performance of the classification model.

5.6.4 SMOTE-Boundary Sampling

SMOTE-Boundary Sampling is a hybrid sampling technique that combines the SMOTE and Boundary Cleaning (BC) methods to address the issue of imbalanced datasets in classification tasks (Lee2020).

The SMOTE-Boundary Sampling algorithm works as follows:

1. SMOTE: Synthetic Minority Over-sampling Technique (SMOTE) is applied to generate synthetic samples for the minority class by interpolating between the minority class samples.

2. Identify the borderline samples: Samples that are on the boundary between the minority and majority classes are identified. These samples are those that have at least one neighbor belonging to the majority class and one neighbor belonging to the minority class.

3. Boundary Cleaning (BC): The BC method is used to remove noisy borderline samples. This is done by examining the k-nearest neighbors of each borderline sample. If the majority class samples are in the majority among the k-nearest neighbors, the sample is removed. Otherwise, it is kept in the dataset.

4. Generate synthetic samples: For each borderline sample, generate the desired number of synthetic samples using the SMOTE algorithm.

5. Combine datasets: The datasets generated by SMOTE and BC are combined to form the final balanced dataset.

By combining SMOTE and BC, the SMOTE-Boundary Sampling approach is able to generate synthetic samples while also removing noisy samples from the borderline between the minority and majority classes, resulting in a high-quality balanced dataset.

SMOTE-Borderline Formulation

Let X be the feature matrix and y be the corresponding class labels, where $X = \{x_1, x_2, \ldots, x_n\}$ and $y = \{y_1, y_2, \ldots, y_n\}$, n being the number of samples. The SMOTE-Borderline technique combines the Synthetic Minority Over-sampling Technique (SMOTE) and the Borderline technique to address class imbalance. The formulation of the SMOTE-Borderline technique can be described as follows:

1. Identify the samples from the minority class that are on the border of the decision boundary, referred to as the borderline samples.
2. For each borderline sample x_i:
   (a) Compute the k nearest neighbors of x_i from the same class, denoted as $N_i = \{x_{i1}, x_{i2}, \ldots, x_{ik}\}$.
   (b) Select the m nearest neighbors of x_i that belong to the majority class, denoted as $N_i' \subseteq N_i$.
   (c) For each selected neighbor $x_j \in N_i'$, generate synthetic samples between x_i and x_j using SMOTE.
3. Update the feature matrix X and class labels y by adding the generated synthetic samples.

Algorithm 25: SMOTE-Boundary Sampling Algorithm

Input: Dataset with class labels $y_1, y_2, ..., y_n$ and features $x_1, x_2, ..., x_n$
Output: Balanced dataset $x'_+ \cup x'_-, y'_+ \cup y'_-$

Calculate class distribution N_+ and N_-;
Calculate the number of synthetic samples to generate n_{syn};
Generate synthetic minority class samples using SMOTE;
Identify borderline samples;
Clean borderline samples using BC:

1. For each borderline sample x_i with k nearest neighbors:
2. Calculate the ratio $r = \frac{majorityclassneighbors}{minorityclassneighbors}$
3. If $r \geq 1$, remove x_i from the dataset
4. Else, keep x_i in the dataset

Combine datasets: $X' = x'_+ \cup x'_-$,
$Y' = y_+ \times |x'_+| \cup y_- \times |x'_-|$,
where $|x'_+|$ and $|x'_-|$ are the sizes of the oversampled and undersampled datasets, respectively;
Shuffle the balanced dataset;

Note that the algorithm can be modified to include hyperparameters for tuning the number of synthetic samples to generate, as well as other parameters such as the number of neighbors to consider in SMOTE and BC and the performance metric used for evaluation. Additionally, other techniques such as ensemble methods can be incorporated into the algorithm to further improve the performance of the classification model.

It is worth noting that the performance of SMOTE-Boundary Sampling can be influenced by the choice of parameters, such as the number of neighbors in SMOTE

and BC, as well as the size of the synthetic minority class. These parameters can be tuned to optimize the performance of the classification model on the balanced dataset. In summary, hybrid sampling approaches that combine both under and oversampling techniques can enhance the performance of text classification models by achieving dataset balance, retaining crucial information from both classes, and mitigating the impact of noisy samples. The selection of an appropriate approach should be guided by the dataset characteristics and the specific demands of the text classification task.

5.7 Conclusion

In conclusion, this chapter has explored various approaches to address class imbalance in text classification tasks. We have reviewed different sampling techniques, including random undersampling, oversampling, and their hybrid methods. We also discussed the SMOTE-based techniques, such as SMOTE-Tomek, SMOTE-ENN, and SMOTE-Borderline, as well as the Boundary Cleaning technique. Despite the success of these methods, there is still room for improvement, especially in the context of deep learning models. In the next chapter, we will introduce our contribution to address class imbalance in BERT, one of the state-of-the-art deep learning models for natural language processing. Specifically, we will propose a new layer that incorporates a balancing technique in different emplacements within the BERT architecture. Our proposed approach leads to better performance in text classification tasks, especially for Arabic sentiment analysis on imbalanced datasets.

Chapter 6

Improving Multi-class Text Classification on Imbalanced Datasets

6.1 Introduction

Multi-class text classification is the task of categorizing a text document into one of several predefined categories or classes. It has numerous applications in natural language processing, including sentiment analysis, topic classification, and spam filtering. Multi-class text classification poses a challenging problem as it requires the model to accurately distinguish between multiple classes, which can be semantically similar.

Imbalanced datasets are one of the major challenges in multi-class text classification. In real-world scenarios, the number of samples in each class is often unbalanced, with some classes having significantly fewer samples than others. This leads to a bias towards the majority class during training, resulting in poor performance on the minority classes. The imbalanced nature of datasets can be especially problematic in text classification, where some classes may have very specific and nuanced features that are difficult to capture (Nooralahzadeh2021).

BERT (Bidirectional Encoder Representations from Transformers) has been shown to achieve state-of-the-art results in many NLP tasks, including text classification. However, when applied to imbalanced datasets, BERT may still suffer from the problem of bias towards the majority class, resulting in poor performance on the minority classes (Devlin2018). This is due to the fact that BERT is trained on a large corpus of text, which may not accurately represent the distribution of classes in the imbalanced dataset.

To address the problem of imbalanced datasets in text classification, researchers we proposed a new approach that includes a new layer of balancing after the embedding representation step in BERT. This new layer is trained to adjust the weight of each sample based on the class distribution, effectively balancing the training process for all classes. The results of this approach have shown significant improvements over the BERT baseline and state-of-the-art methods, demonstrating its effectiveness in addressing the problem of imbalanced datasets in text classification.

Another new approach to address the problem of imbalanced datasets in text classification is to include a new layer of balancing after the BERT representation step. This layer is designed to adjust the weight of each sample based on the similarity between the sample and the centroid of each class, effectively balancing the training process for all classes. This approach has shown even better results than the

previous method, surpassing both the BERT baseline and the state-of-the-art methods. The effectiveness of this approach highlights the importance of addressing the problem of imbalanced datasets in text classification, and the potential of using advanced techniques to improve the performance of NLP models in real-world scenarios.

The remaining section of this chapter is structured as follows: In the second section, we provide an overview of the baseline algorithm and methodology used in this study. Section 3 introduces the proposed balancing techniques. The experimental design, results, and subsequent discussions are presented in Section 4. Finally, we conclude the paper.

6.2 BERT for Sentiment Analysis

The Bidirectional Encoder Representation from Transformer (BERT) framework is based on the revolutionary transformer architecture. This approach is divided into two steps: pre-training, and fine-tuning (see Figure 6.1). The model is trained on unlabeled data across several pre-training tasks in the first phase. For the fine-tuning, the BERT model is first started with the pre-trained parameters, and then all of the parameters are fine-tuned using labeled data from the downstream task. As such, applied to balanced datasets, a BERT model fine-tuned to text classification task outperforms the state-of-the-art. The descriptions of the two steps are described below.

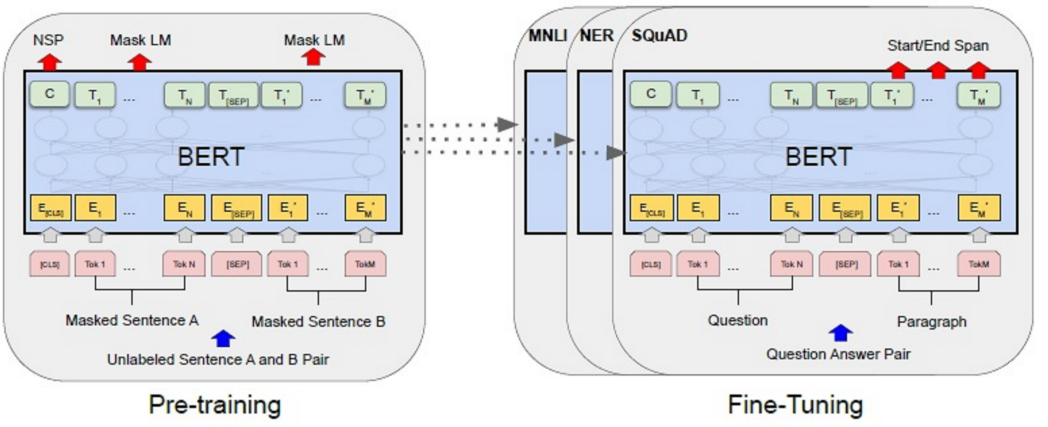

FIGURE 6.1: BERT Architecture (Devlin2018)

6.2.1 BERT Pre-training

In natural language processing, pre-training models have gained widespread attention due to their ability to learn useful linguistic features. One such pre-training approach is the deep bi-directional model, which combines two modeling approaches, one that moves from left-to-right and one that moves from right-to-left. The combination of these approaches leads to a relatively shallow concatenation of the two models, which results in the ability to better capture the context of the text.

The left-to-right and right-to-left models are trained on the same corpus, but in opposite directions. By training the models in this way, the model can predict missing words in the text based on the surrounding context, in both directions. The

output of the left-to-right and right-to-left models are concatenated to create a single embedding for each token. This bi-directional approach is useful for modeling dependencies in the text, which can help with tasks such as named entity recognition, sentiment analysis, and machine translation.

Task 1: Masked Language Modeling (MLM)

The pre-training step in BERT utilizes a masked language model approach, where a random 15% of the input tokens are masked and predicted to train a deep bidirectional representation. This approach ensures that the model learns to predict missing words based on their context and improves the model's understanding of language. The final hidden vectors of the masked tokens are then fed into an output softmax over the vocabulary, enabling the model to predict the original tokens during fine-tuning.

During prediction, the training data generator randomly selects 15% of the token positions and applies a replacement strategy. If the i-th token is selected, it is replaced 80% of the time with the mask token, 10% of the time with a random token, and 10% of the time with the original i-th token. This replacement strategy enables the model to generalize better and handle out-of-vocabulary words. Cross-entropy loss and Ti, the number of tokens in the input sequence, are used to predict the original token, allowing the model to learn to differentiate between different token positions and predict the correct word in context. The use of these techniques in pre-training BERT enables the model to achieve state-of-the-art performance on various natural language processing tasks.

- During training:
  - Randomly mask 15% of input tokens and predict.
  - Final hidden vectors of masked tokens are fed into an output softmax.
- During prediction:
  - Randomly select 15% of token positions.
  - Replace i-th token with 80% probability of mask token, 10% probability of random token, and 10% probability of original i-th token.
  - Use cross-entropy loss and Ti to predict original token.

Task 2: Next Sentence Predicting (NSP)

The main goal of the binarized next sentence challenge task is to determine whether there is an association between two given sentences or not. This task involves using a monolingual corpus to train a model that can detect sentence connections. Specifically, the model needs to predict whether a given sentence is the next one in a sequence of sentences or not. This type of task is crucial in natural language processing, as it can help improve the accuracy of various downstream tasks that rely on sentence connections, such as question-answering and text summarization.

Once the BERT pre-training step is completed, all of the learned parameters are transferred to initialize the downstream task, which is the fine-tuning step. During fine-tuning, the model's parameters are fine-tuned using labeled data from the downstream task to improve the model's accuracy for a specific task. By transferring the parameters learned during pre-training, the model can leverage the knowledge

gained from the vast amount of unlabeled data during the pre-training phase, resulting in a better-performing model for the downstream task.

6.2.2 BERT Fine-tuning

BERT's architecture is highly flexible and allows fine-tuning of pre-trained models for various downstream tasks. By inputting relevant inputs and outputs and fine-tuning all of the model's parameters end-to-end, BERT can be fine-tuned for various tasks such as text classification and sequence tagging. For text classification tasks, concatenated sentences with a single blank are used as a single input phrase. The baseline input of BERT consists of tokens representing sentences, which results in a set of token representations when processed. These token representations generate a REP<cls> representation for the text and a representation for each token. The overall architecture is shown in Figure 6.2.

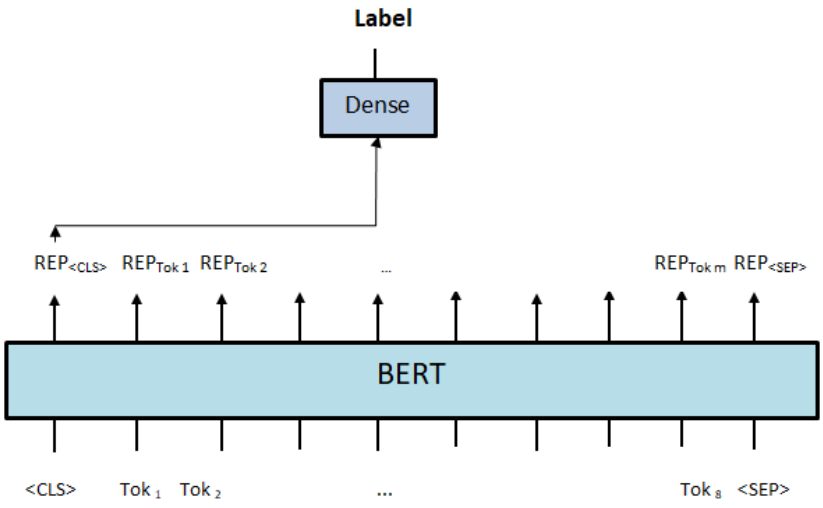

FIGURE 6.2: BERT Baseline Approach

In summary, BERT is a highly flexible architecture that allows for fine-tuning of pre-trained models for various downstream tasks by inputting relevant inputs and outputs and fine-tuning all of the model's parameters end-to-end. In text classification tasks, concatenated sentences with a single blank are used as a single input phrase, and the baseline input of BERT consists of tokens representing sentences. The resulting token representations generate a REP<cls> representation for the text and a representation for each token. By fitting a classifier to the output of the representation data, using REP<cls>, models can be built. This flexibility and ease of use make BERT a popular choice for various NLP tasks, including text classification and sequence tagging.

Despite the effectiveness of BERT in text classification tasks, its performance is suboptimal when faced with imbalanced classification problems. The problem stems from the unequal distribution of examples across text classes, which can make predictive modeling in text challenging. To overcome this limitation, we propose an enhancement to the BERT scheme by incorporating a new layer designed to handle imbalanced problems. Our proposed approach involves integrating well-known numerical balancing techniques, such as oversampling, undersampling, and hybrid

approaches, into two different stages of the BERT model. The first approach applies balancing techniques after the embedding steps, and the second approach applies balancing techniques after the representation step.

The first proposed approach involves adding a new layer after the embedding step. This layer incorporates balancing techniques to the input embedding vectors before they are fed into the BERT model. This helps to mitigate the effects of class imbalance by reducing the impact of the majority class on the final decision. In the second approach, a new layer is added after the BERT representation step. This layer is designed to adjust the feature representation of the data to balance the data distribution. The proposed layer generates new features based on the original features of the data while taking into account the distribution of the classes. Our experiments demonstrate that both approaches lead to significant improvements in BERT's performance on imbalanced classification tasks.

6.3 The First Proposed Aproach

Addressing the issue of imbalanced classification in text datasets is a significant challenge, as the skewed distribution of examples can lead to biased models that underperform on minority classes. While BERT has proven effective for many text classification tasks, it is not optimized for handling imbalanced data, which can limit its practical application in real-world scenarios. In this thesis, we proposed two novel approaches to enhance BERT's architecture for imbalanced text classification.

The first approach involves adding a new layer that applies well-known numerical balancing techniques, such as oversampling, undersampling, or hybrid approaches, after the embedding step. This layer ensures that each class is represented equally in the training dataset, preventing the model from being biased toward the majority class. This step is crucial for improving the model's accuracy on minority classes, which are often underrepresented in the dataset (See Figuree 6.3).

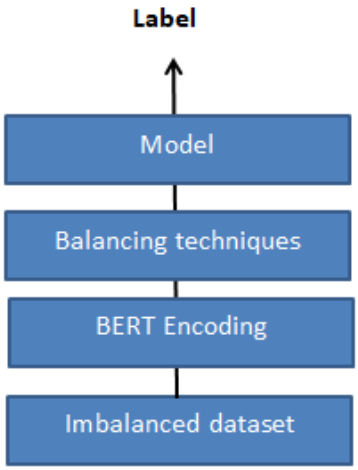

FIGURE 6.3: The First Approach

The architecture of the first approach is presented below.

6.3.1 BERT Encoding

BERT encoding refers to the process of transforming raw text into numerical representations that can be understood by the BERT architecture. This encoding process involves tokenizing the text into individual word pieces, then mapping each word piece to its corresponding embedding vector in a pre-trained embedding space. BERT uses a two-step encoding process, where the first step generates token embeddings that capture the meaning of each individual word piece, and the second step generates sentence-level embeddings that capture the contextual relationships between words in the text.

6.3.2 Balancing Techniques

The balancing step is a crucial part of the proposed approaches to enhance BERT for imbalanced text classification tasks. It involves applying well-known numerical balancing techniques, such as oversampling, undersampling, or hybrid approaches, to mitigate the negative effects of imbalanced class distribution. In undersampling, some of the examples in the majority class are randomly removed to balance the dataset. This approach reduces the computational cost and prevents the model from overfitting, but it may cause a loss of information and decreased performance. On the other hand, oversampling involves creating synthetic examples by replicating existing minority class examples or generating new examples using various techniques such as SMOTE, ADASYN, and others. This approach can increase the size of the minority class, but it may also lead to overfitting and decreased generalization.

To overcome the limitations of both undersampling and oversampling techniques, hybrid approaches combine both techniques, resulting in a more balanced dataset without losing too much information. For instance, we can first apply undersampling to the majority class and then use oversampling to augment the minority class. Another hybrid approach is to use a combination of oversampling and undersampling techniques to create a balanced dataset. These techniques can improve the performance of BERT in imbalanced text classification tasks by providing a balanced dataset that allows the model to learn from both classes equally.

6.3.3 BERT Model Creation

Once the BERT encoding is complete, the next step is to create a model that can predict the output based on the encoded input. This is done by adding a classification layer on top of the BERT encoding layer. The classification layer consists of one or more dense layers with a softmax activation function, which maps the encoded input to a probability distribution over the output classes. The number of neurons in the final dense layer corresponds to the number of output classes, and each neuron represents the probability of the input belonging to that class. During the training phase, the weights of the classification layer are updated to minimize the loss function between the predicted output and the actual output.

To improve the performance of the model, various regularization techniques can be applied to prevent overfitting, such as dropout and L2 regularization. Dropout randomly drops out some of the neurons in the dense layers during training, which helps to prevent the model from memorizing the training data. L2 regularization adds a penalty term to the loss function based on the magnitude of the weights, which helps to prevent the model from over-emphasizing certain features in the input. Once the model is trained, it can be used to predict the output for new input data. The final output is the class with the highest probability value.

After implementing the first approach of incorporating a balancing layer after the BERT embedding step, we decided to further improve the model's performance by including the balancing layer after the BERT representation step. This decision was made based on the understanding that the representation step captures the contextual information of the text, which can be leveraged to further enhance the balancing process. By adding the balancing layer after the representation step, we aimed to create a more robust and effective model that is better equipped to handle imbalanced classification problems in Arabic social media text. The new approach is described in the next section.

6.4 The Second Proposed Approach

As mentioned earlier, the BalBERT method is unique in that it extends the original BERT architecture by adding a new layer. While the baseline technique serves two purposes - representation and classification, the new approach builds upon it and involves three tasks: representation, balancing, and classification.

6.4.1 BERT Representation

The first step in the process involves data preprocessing, which includes cleaning the data and tokenization. The tokens obtained from each sentence are then converted into their corresponding embedding representation using the BERT Embedding block. These embeddings are then fed into a fully connected feed-forward network layer to obtain a context-based representation, REP<cls>, as well as the representation of each sentence's individual tokens (as shown in Figure 6.4).

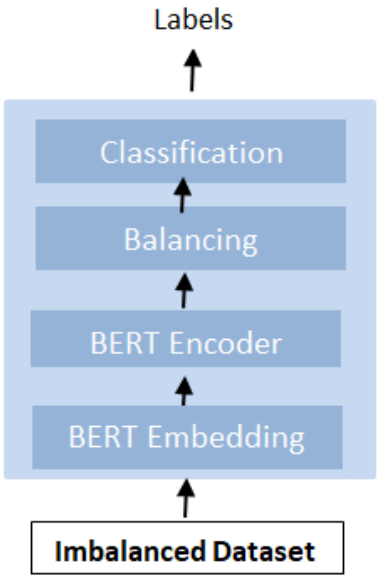

FIGURE 6.4: The Second Approach

6.4.2 BERT Classification

In the next step, the classifier is trained using the output obtained after the BalBERT sampling step, which results in a balanced dataset. The labeling of each sample is

used to train the classifier and create models, which are then tested on a separate test dataset to determine the best sampling block. The following section describes a series of experiments that were conducted to evaluate the effectiveness of the BalBERT approach on a widely-used imbalanced Twitter dataset.

6.5 EXPERIMENTS

In this section, we will outline the experimental setup and report the results obtained from our models in terms of a specific metric for imbalanced classification, compared to the baseline BERT model.

6.5.1 Dataset settings and parameters

Two imbalanced datasets with multiple classes were used for sentiment analysis. The first dataset, ASAD (Twitter-based Benchmark Arabic Sentiment Analysis dataset), was created for a KAUST-sponsored sentiment analysis competition and contains Arabic tweets (Al-Harbi2021). Details about this dataset are provided in Table 6.1 and illustrated the distribution in Figure 6.5.

TABLE 6.1: ASAD dataset.

Tweet's	Number	Positive	Negative	Neutral
ASAD Dataset	55,000	8,200	8,821	37,359

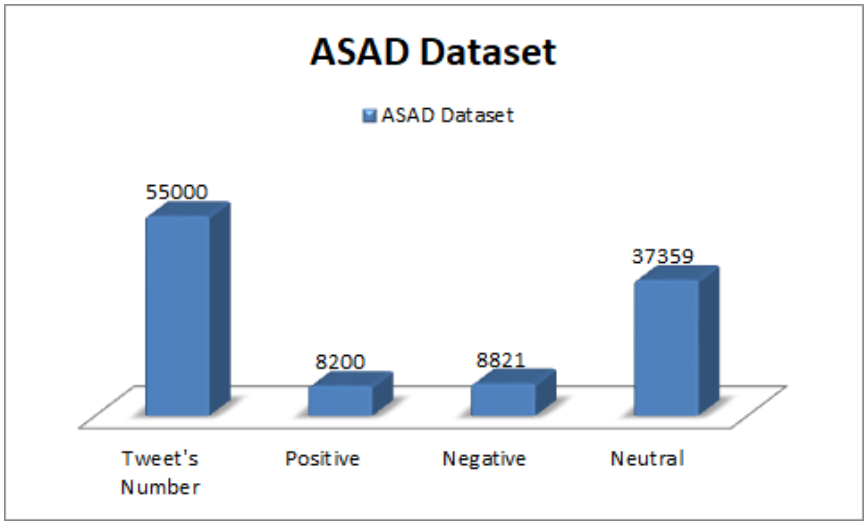

FIGURE 6.5: ASAD Tweets Distribution

The second dataset is a Review dataset with severe imbalance between the classes. It consists of five classes and has been introduced for the CERIST NLP Challenge 2022 (Alamri2022). Details of this dataset can be found in Table 6.2 and the tweets distribution is illustration in Figure 6.6.

TABLE 6.2: Review dataset.

Tweet's	Number	1	2	3	4	5
Review Dataset	63,257	2,939	5,285	12,001	19,054	23,778

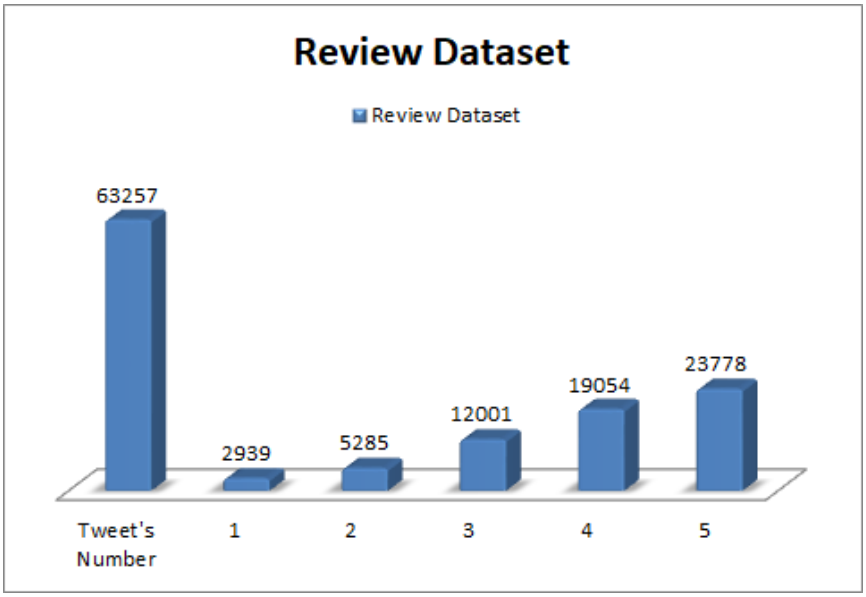

FIGURE 6.6: Review Tweets Distribution

6.5.2 Experiments Description

To evaluate the performance of multi-class imbalanced text classification models, a standard preparation procedure was followed for all experiments. This involved removing irrelevant information from sentences, such as noise, URLs, hashtags, and non-alphabetic characters. The evaluation was based on two commonly used metrics: the average recall (AVG recall) and F1-score. AVG recall measured the proportion of relevant instances that were correctly classified across all classes, considering the imbalanced nature of the dataset. F1-score, on the other hand, was the harmonic mean of precision and recall, providing an overall measure of the model's accuracy that accounted for both false positives and false negatives. These metrics allowed for a comprehensive evaluation of the model's performance on imbalanced datasets.

Before applying the two proposed approaches to the unbalanced datasets, the default outcome of the AraBERT approach was assessed to create a baseline for performance evaluation (refer to Figure 6.2). AraBERT (Antoun2020) is an effective Arabic-specific variant of BERT that was directly applied to the unbalanced datasets. In the other experiments, the two new approaches were used with different balancing techniques, and the performance was evaluated using the same metrics (refer to Figure 6.3 and the Figure 6.4).

6.5.3 The First Approach Exprement Results

There were four experiments conducted, each of which is described in detail in the following subsections. The first experiment involved directly applying fine-tuning BERT to the imbalanced dataset, while the other three experiments utilized balancing techniques such as oversampling, undersampling, and hybridization.

The experimental results are presented in Tables 6.3, 6.4, 6.5, and 6.6, which demonstrate that feeding the imbalanced ASAD dataset directly to BERT resulted in poor classification accuracy, with an AVG-Recall of 0.6 and an F1-PN of 0.54. The findings also reveal that each balancing technique performs differently, with Hybridization being the most effective, achieving over 15% better AVG-Recall and 19% better F1-PN compared to the other two techniques. Furthermore, we observed that the outcomes of each balancing method differ. For oversampling, SMOTE outperformed the random technique, while for undersampling, boundary removal was more effective than the random approach.

TABLE 6.3: Results obtained without balancing the dataset

Evaluation Metric	**Result Value**
F1-Score for Positive class	54%
F1-Score for Negative class	55%
F1-Score for Neutral class	85%
Macro F1-Score	65%
Recall for Positive class	46%
Recall for Negative class	45%
Recall for Neutral class	89%
Average Recall	60%
F1-Score for Positive and Negative classes	54%

TABLE 6.4: Results of Oversampling Balancing Techniques

	Random Oversampling	SMOTE Oversampling
F1-Score for Pos Class	66%	68%
F1-Score for Neg Class	66%	69%
F1-Score for Neu Class	87%	89%
Macro-F1 Score	73%	75%
Recall for Pos Class	63%	67%
Recall for Neg Class	64%	68%
Recall for Neu Class	88%	89%
Average Recall	71%	75%
F1-Score for Pos and Neg Classes	65%	68%

TABLE 6.5: Results of Undersampling Balancing Techniques

	Random Undersampling	Boundary Undersampling
F1-Score for Pos Class	67%	69%
F1-Score for Neg Class	68%	70%
F1-Score for Neu Class	87%	89%
Macro F1-Score	74%	76%
Recall for Pos Class	64%	66%
Recall for Neg Class	63%	68%
Recall for Neu Class	87%	86%
Average Recall	71%	73%
F1-Score for Pos and Neg Classes	67%	69%

TABLE 6.6: Comparison of hybrid balancing techniques: Random and Random vs. SMOTE and Boundary

	Rand_under and Rand_over	SMOTE and Boundary
F1 score for Pos class	73%	75%
F1 score for Neg class	74%	74%
F1 score for Neu class	88%	87%
Macro-average F1 score	78%	78%
Recall for Pos class	72%	74%
Recall for Neg class	71%	73%
Recall for Neu class	87%	88%
Average recall	76%	78%
F1 score for Pos and Neg classes	73%	73%

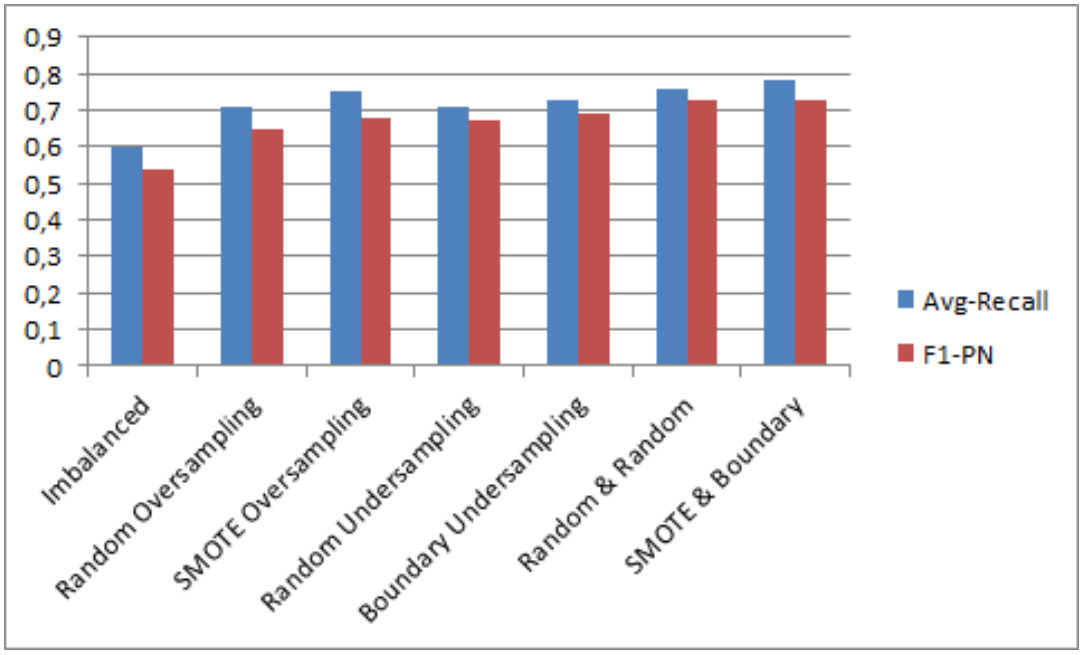

FIGURE 6.7: Resuls Illustration

Table 6.6 indicates that hybrid techniques outperform other oversampling methods in both AVG-Recall and F1-PN metrics with fine-tuned BERT. The optimal results are obtained by combining SMOTE with boundary. It is crucial to recognize that accuracy score is biased towards the majority class, as discussed in Section 2. In multiclass classification problems where all classes' prediction is equally essential, such as in various remote sensing applications, F-score and AVG-Recall are more dependable metrics than accuracy and should be preferred.

Table 6.5 displays the rankings of undersamplers and indicates that both random and boundary-comparable techniques outperform the imbalanced dataset. Even though undersampling balances the dataset, it is evident from the tables that it generates suboptimal outcomes. On the other hand, Table 6.4 compares the performance of SMOTE and random oversampling methods. The results are more significant than those in the first experiment and almost equal to the undersampling approach results.

6.5.4 The Second Approach Exprement Results

The results of the experiments have been summarized in Tables 6.7, 6.8 , and 6.9. Table 6.7 indicates that the baseline BERT approach performs poorly in dealing with imbalanced datasets, and it is the worst among all the techniques used in the experiments. For oversampling, the SMOTE technique outperforms the Random approach in both datasets. Both techniques show significant improvement over the baseline BERT approach in both metrics, with a minimum increase of 17% and 20%, respectively. Tables 6.8 , and 6.9 demonstrate the effectiveness of the boundary technique in handling imbalanced datasets using undersampling. It achieves significant improvement, with the best result reaching 79% for AVG-Recall. Additionally, Tables 6.8 , and 6.9 show a remarkable improvement of 80% and higher in the two datasets.

TABLE 6.7: BERT baseline fit to ASAD Review Datasets

Metrics	ASAD Dataset	metric	Review Dataset
F1-pos	0.54	F1-1	0.43
F1-Neg	0.55	F1-2	0.48
F1-Neu	0.85	F1-3	0.52
		F1-4	0.60
		F1-5	0.65
Positive-Recall	0.46	1-Recall	0.40
Negative-Recall	0.45	2-Recall	0.45
Neutral-Recall	0.89	3-Recall	0.44
		4-Recall	0.59
		5-Recall	0.82
AVG-Recall	0.60	AVG-Recall	0.54
F1-PN	0.54	123-Recall	0.47

TABLE 6.8: The second Approach fit to ASAD dataset

Metrics	Over Rand	SMOTE	Under Rand	Boundary	Rand &Boun	Boun &SMO
F1-pos	0.68	0.70	0.69	0.70	0.75	0.76
F1-Neg	0.70	0.71	0.66	0.72	0.76	0.77
F1-Neu	0.80	0.82	0.85	0.87	0.89	0.88
Pos-Recall	0.67	0.71	0.66	0.69	0.76	0.75
Neg-Recall	0.68	0.72	0.69	0.67	0.74	0.72
Neu-Recal	0.81	0.83	0.87	0.86	0.98	0.87
AVG-Recall	0.68	0.75	0.74	0.79	0.80	0.77
F1-PN	0.69	0.70	0.68	0.71	0.77	0.76

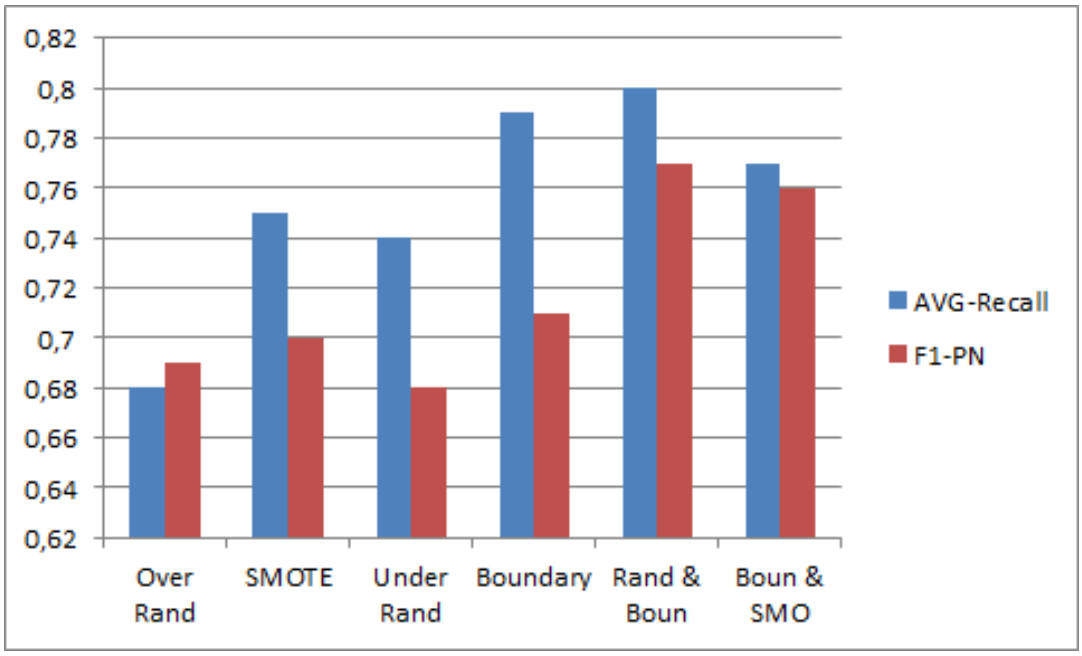

FIGURE 6.8: The Second Approach fit on ASAD dataset Resuls Illustration

TABLE 6.9: The second Approach fit to review dataset

Metrics	Over Rand	SMOTE	Under Rand	Boundary	Rand &Boun	Boun &SMO
F1-1	0.61	0.68	0.61	0.62	0.65	0.68
F1-2	0.63	0.68	0.65	0.65	0.69	0.70
F1-3	0.67	0.74	0.72	0.71	0.73	0.74
F1-4	0.74	0.78	0.77	0.71	0.78	0.79
F1-5	0.77	0.79	0.82	0.75	0.81	0.83
1-Recall	0.61	0.66	0.60	0.62	0.63	0.68
2-Recall	0.62	0.68	0.67	0.62	0.67	0.71
3-Recall	0.68	0.72	0.70	0.71	0.72	0.73
4-Recall	0.71	0.74	0.78	0.79	0.76	0.78
5-Recall	0.78	0.78	0.80	0.81	0.88	0.81
AVG-Recall	0.68	0.72	0.71	0.71	0.73	0.74
123-PN	0.64	0.70	0.66	0.66	0.69	0.71

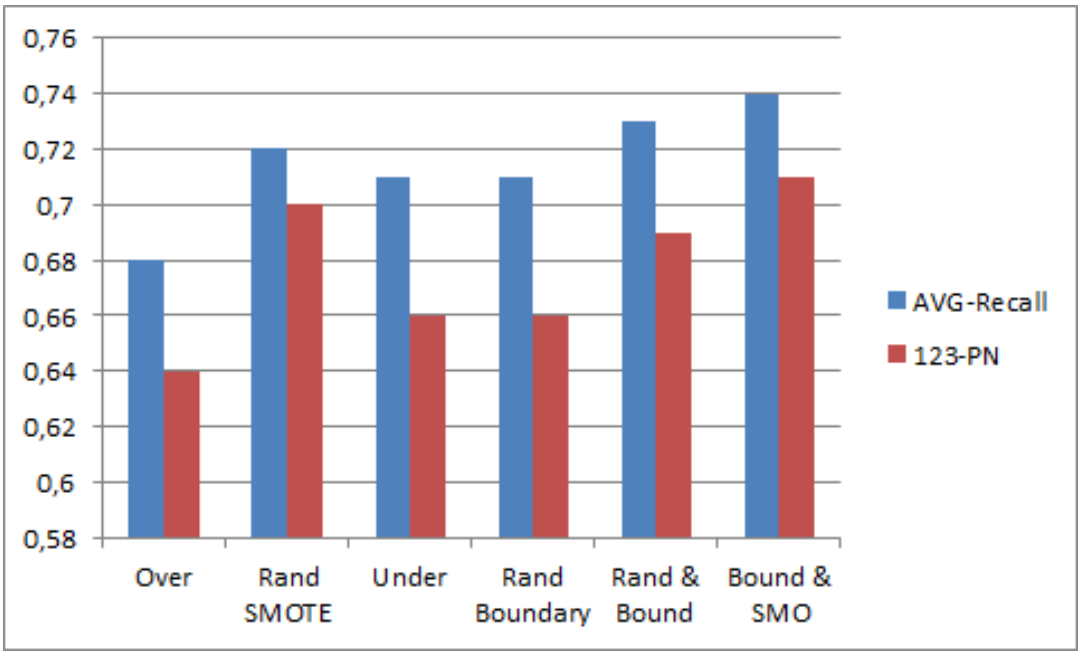

FIGURE 6.9: The Second Approach fit on Book Review dataset Resuls Illustration

6.5.5 Discussion

The Results of the first approach presents compelling evidence on how to tackle the issue of imbalanced data in text classification. Initially, benchmark results were obtained using the 'Original' data (which was imbalanced), as demonstrated in Table 6.3, Subsequently, Tables 6.4, 6.5, and 6.6 depict the AVG-Recall and F1-PN scores of different balancing techniques when applied to the 'Original' ASAD dataset.

The results indicate that the hybrid balancing technique, which utilizes SMOTE and boundary methods, outperforms other balancing techniques, with a remarkable increase of over 18% in AVG-Recall and 19% in F1-PN. By analyzing Table 6.6, it is clear that the hybrid technique is the most effective among oversampling methods when using fine-tuned BERT. Moreover, when SMOTE is combined with boundary, the best results are achieved.

It is important to note that the accuracy score is skewed towards the majority class, as discussed in Section 2. Therefore, relying solely on accuracy may not be a reliable measure of model performance when dealing with imbalanced datasets. Table 6.5 exhibits the rankings of the undersamplers and highlights the superior performance of both random and boundary-comparable techniques in addressing the imbalanced dataset. However, despite the balancing achieved through undersampling, it is evident from the tables that this approach yields suboptimal outcomes.

Moving on to Table 6.4, we can compare the performance of the two oversampling techniques, namely SMOTE and random. The results indicate a significant improvement compared to the initial experiment and are almost on par with the outcomes of the undersampling approach.

The results presented of the second approach illustrates its effectiveness . This approach uses the BERT representation step, which enables dataset resampling and produces a context-based feature space that cannot be achieved with the embedding step. While the BalBERT results are noteworthy compared to the baseline BERT and

previous research, they can be further improved by integrating other balancing techniques within the BalBERT approach. For example, the application of the SMOTE approach added new discriminating features, which increased the AVG-Recall from 60 percent to 75 percent. Furthermore, the undersampling approach used in BalBERT eliminates interacting features, resulting in better model performance. Tables 6.8 and 6.9 demonstrate that the new approach achieved an AVG-Recall of 79 percent, which is a considerable improvement compared to the baseline BERT's score of 60 percent despite dataset skewing. These results outperformed both the baseline and state-of-the-art techniques reported in [23].

The hybrid approach that combines SMOTE and Boundary produced an 80% performance, which is a reasonable outcome due to the complementary roles of each technique. Although the hybrid approach that combines random oversampling and random undersampling had the best result of 81%, the former hybrid approach is considered more rational. After a more in-depth analysis of the experimental results, it was discovered that novel balancing techniques significantly improved the performance of the minority classes, increasing by at least 30%. This finding highlights the efficacy of these newly implemented approaches in handling imbalanced datasets and emphasizes their potential to improve machine learning models' overall performance and provide more accurate results for unevenly distributed datasets. The observed improvement in the performance of the minority classes demonstrates that the balancing techniques have successfully addressed the underrepresented class issue, resulting in a more comprehensive and reliable model evaluation.

Furthermore, comparing the experimental results of the two datasets showed that the second approach performed better on the ASAD dataset than the Review dataset. The Review dataset's imbalanced nature and larger number of classes posed a greater challenge for the the new approach model in achieving higher accuracy than the ASAD dataset.

The Results of the second approach provides insight into how to address the problem of imbalanced data in text classification. Initially, benchmark results were obtained using the "Original" imbalanced dataset, as shown in Table 6.7.

The findings indicate that the hybrid balancing technique, which combines SMOTE oversampling with boundary removal, yields the best results, with over 18% AVG-Recall and 19% F1-PN improvement compared to other balancing techniques. These results underscore the importance of using appropriate balancing techniques to address the problem of imbalanced data in text classification and highlight the potential of deep learning models, such as BERT, for this task.
The new balancing layer in the BERT architecture provides significant advantages in enhancing sentiment analysis models on imbalanced datasets. This layer allows the model to adjust the weight of the loss function during training, which is critical when dealing with imbalanced datasets. Imbalanced datasets can lead to bias in the model and lower accuracy in predicting the minority class. However, the balancing layer addresses this issue by adjusting the weight of the loss function based on the frequency of the minority class, leading to more accurate predictions.

Chapter 7

Conclusions

Natural Language Processing (NLP) has become an increasingly important field due to the vast amounts of text data generated every day. In this thesis, we focused on text classification and sentiment analysis on Arabic social media. We presented three contributions to this field, the first of which was a new Arabic word embedding inspired by GloVe and based on Roots. Our proposed method outperformed the traditional word embedding methods in sentiment analysis tasks. The second contribution addressed the challenge of multi-class text classification on imbalanced datasets. We improved the BERT architecture by adding a new layer of balancing techniques after the word embedding step, which significantly enhanced the performance of the model. The third contribution was the improvement of BERT by adding a new layer for balancing after BERT representation. All of our proposed approaches achieved excellent efficiency compared with the baseline and the state of the art.

Our new Arabic word embedding method, based on GloVe and Roots, addressed the problem of traditional word embeddings that do not accurately capture the semantic relations between words in Arabic text. Our proposed method outperformed traditional methods such as Word2Vec and FastText on sentiment analysis tasks. Additionally, our approach was effective in capturing the sentiment-related information and semantic relations between words in Arabic, making it useful for sentiment analysis and other NLP tasks.

Imbalanced datasets are a common challenge in sentiment analysis, especially in Arabic text. Our proposed approach for improving multi-class text classification on imbalanced datasets showed significant improvements in performance compared to the baseline and state-of-the-art models. Our method involved adding a new layer of balancing techniques after the word embedding step in the BERT architecture. The new layer included a combination of oversampling, undersampling, and data augmentation techniques. The proposed approach showed the best performance in multi-class text classification tasks, achieving high precision and recall scores.

We also proposed improving BERT by adding a new layer for balancing after the BERT representation. Our approach improved the performance of BERT in sentiment analysis tasks, particularly in cases where the dataset was imbalanced. Our proposed method showed promising results in improving the performance of BERT in Arabic sentiment analysis tasks, and it has the potential to be applied to other NLP tasks.

In conclusion, this thesis has presented three contributions to the field of NLP, text classification, and sentiment analysis on Arabic social media. Our proposed approaches, including the new Arabic word embedding method based on GloVe and Roots and the improved BERT architecture with new balancing layers, demonstrated significant improvements in performance compared to the baseline and state of the art. The proposed methods addressed common challenges in sentiment analysis,

including imbalanced datasets, and demonstrated high precision and recall scores. These findings show the potential for our proposed approaches to improve the accuracy and efficiency of sentiment analysis in Arabic text and other NLP tasks. We hope that our contributions will inspire further research in this area and lead to more effective and accurate NLP models for Arabic text.

The results obtained open up exciting avenues for future research in the field of Natural Language Processing (NLP) and sentiment analysis on Arabic social media. Firstly, we can explore the applicability of the proposed approaches to other NLP tasks in Arabic text, such as text summarization or named entity recognition. Investigating how the proposed models perform in these tasks would contribute to a broader understanding of their capabilities and potential applications. Secondly, fine-tuning the proposed models for domain-specific sentiment analysis on various Arabic social media platforms would enable more accurate and tailored sentiment analysis in specific contexts. This research direction acknowledges the diverse nature of social media data and the need for specialized models. Additionally, incorporating techniques from transfer learning and domain adaptation can enhance the generalization and robustness of the models, enabling them to perform effectively in different domains and scenarios. Lastly, the development of pre-trained models, and datasets dedicated to Arabic sentiment analysis would provide valuable resources for researchers and practitioners in the field. Such resources would facilitate further research, accelerate progress, and foster the development of more accurate and efficient NLP models for Arabic text analysis. Pursuing these directions for further work promises advancements in the accuracy, efficiency, and applicability of NLP models in analyzing Arabic text.

Bibliography

[AbuJarour2021] AbuJarour, S., Obeidat, R. (2021). Social media and the Arabic language: A review of the literature. Journal of Arabic and Islamic Studies, 21, 1-29. doi: 10.1163/25890447-12340045

[Abdullah2019] Abdullah, A., Alshehri, S., Alarifi, A., Alfaris, M., Aljarallah, M. (2019). A hybrid approach for sentiment analysis of Arabic tweets. IEEE Access, 7, 163934-163948. https://doi.org/10.1109/ACCESS.2019.2952614

[Al-Harbi2021] Al-Harbi, R., Al-Salman, A., Al-Shehri, S., Al-Khalifa, H. S. (2021). A Hybrid Machine Learning Approach for Arabic Sentiment Analysis using ASAD Dataset. International Journal of Advanced Computer Science and Applications, 12(4), 447-453.

[Alotaibi2022] Alotaibi, A., Rahman, A., Alhaza, R., Alkhalifa, W., Alhajjaj, N., Alharthi, A., Abushoumi, D., Alqahtani, M., Alkhulaifi, D. (2022). Spam and sentiment detection in Arabic tweets using MARBERT model. Mathematical Modelling of Engineering Problems, Vol. 9, No. 6, pp. 1574-1582. https://doi.org/10.18280/mmep.090617

[Al-Khatib2016] Al-Khatib, K., Tannir, N., El-Hajj, W. (2016). Large scale sentiment analysis in Arabic social media. Journal of Computational Science, 17, 515-522. https://doi.org/10.1016/j.jocs.2016.05.014

[Alamri2022] Alamri, A., Ali, R., Alshehri, S., Alduhaim, A., Alqurashi, Y., Altwaijry, H. (2022). CERIST-NLP'22 Shared Task: Sentiment Analysis and Offensive Language Detection for Arabic and English. In Proceedings of the CERIST NLP Challenge 2022 (pp. 1-11).

[Ali2013] Ali, Ahmed El-Beltagy, Samhaa. (2013). Open Issues in the Sentiment Analysis of Arabic Social Media: A Case Study. 10.1109/Innovations.2013.6544421.

[Antoun2020] Antoun, W., Baly, F., Hajj, H., El-Hajj, R., Salameh, M. (2020). AraBERT: Transformer-based model for Arabic language understanding. arXiv preprint arXiv:2003.00104.

[Baptista2021] Baptista, J., Oliveira, E. (2021). A survey on chatbots: architecture, applications, and evaluation. Journal of Ambient Intelligence and Humanized Computing, 12(7), 6821-6847. doi: 10.1007/s12652-021-03346-4

[Batista2004] Batista, G. E., Prati, R. C., Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter, 6(1), 20-29.

[Baccianella2010] Baccianella, S., Esuli, A., Sebastiani, F. (2010). Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In LREC (Vol. 10, pp. 2200-2204).

[Batista2003] Batista, S., Bazzan, B. and M. Monard, (2003), Balancing training data for automated annotation of keywords: a case study, in Proceedings of the 2003 International Joint Conference on Neural Networks, vol. 3, pp. 2046-2051, .

[Brown2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

[Brownlee2018] Brownlee, J. (2018). Crash Course in Recurrent Neural Networks for Deep Learning: LSTM, GRU, and more. Machine Learning Mastery. https://machinelearningmastery.com/crash-course-recurrent-neural-networks-deep-learning/

[Buda2018] Buda, M., Maki, A., Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. Neural Networks, 106, 249-259. doi: 10.1016/j.neunet.2018.07.016

[Bunkhumpornpat2009] Bunkhumpornpat, C., Sinapiromsaran, K., Lursinsap, C. (2009). Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In 2009 Fourth International Conference on Computer Sciences and Convergence Information Technology (pp. 48-52). IEEE.

[Chawla2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.

[Cai2018] Cai, J., Li, J., Li, W., Wang, J. (2018). Deeplearning Model Used in Text Classification. In 2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP) (pp. 123-126). Chengdu, China. doi: 10.1109/ICCWAMTIP.2018.8632592.

[Dalaorao2019] Dalaorao, G. A., Sison, A. M., Medina, R. P. (2019). Integrating Collocation as TF-IDF Enhancement to Improve Classification Accuracy. In 2019 IEEE 13th International Conference on Telecommunication Systems, Services, and Applications (TSSA) (pp. 282-285). Bali, Indonesia. doi: 10.1109/TSSA48701.2019.8985458.

[Devlin2018] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[Ellison2011] Ellison, N. B., Steinfield, C., Lampe, C. (2011). Connection strategies: Social capital implications of Facebook-enabled communication practices. New Media Society, 13(6), 873-892. https://doi.org/10.1177/1461444810385389

[Elshibly2021] Elshibly, H., Al-Emran, M., Shaalan, K. (2021). Arabic Social Media Analysis: A Review. Information Processing Management, 58(2), 102437. https://doi.org/10.1016/j.ipm.2020.102437

[Facebook2022] Facebook. (2022). Company Info. Retrieved from https://about.fb.com/company-info/

[Gao2021] Gao, Q., Li, X., Wang, X., Wu, W. (2021). Social media sentiment analysis: A survey. Information Fusion, 66, 1-17. https://doi.org/10.1016/j.inffus.2020.11.002

[Harrag2009] Harrag, F., El-Qawasmeh, E., Pichappan, P. (2009). Improving Arabic text categorization using decision trees. In 2009 First International Conference on Networked Digital Technologies (pp. 110-115). Ostrava, Czech Republic. doi: 10.1109/NDT.2009.5272214.

[Hasan2019] Hasan, M. R., Maliha, M., Arifuzzaman, M. (2019). Sentiment analysis with NLP on Twitter data. In 2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2) (pp. 1-4). https://doi.org/10.1109/IC4ME247184.2019.9036670

[Han2005] Han, H., Wang, W. Y., Mao, B. H. (2005). Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In Advances in intelligent computing (pp. 878-887). Springer.

[Hasan2022] Hasan, S. A., et al. (2022). Classification of Multi-Labeled Text Articles with Reuters Dataset using SVM. In 2022 International Conference on Science and Technology (ICOSTECH) (pp. 01-05). Batam City, Indonesia. doi: 10.1109/ICOSTECH54296.2022.9829153.

[He2008] He, H., Bai, Y., Garcia, E. A., Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence) (pp. 1322-1328). IEEE.

[He2009] He, H., Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263-1284. doi: 10.1109/TKDE.2008.239

[Hochreiter1997] Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780. doi: 10.1162/neco.1997.9.8.1735

[Huang2021] Huang, K., Liu, Q. (2021). Natural language processing techniques for social media analysis. In Z. Yang, J. Yang, W. Wang (Eds.), Social Media Analysis and Visualization (pp. 37-56). Springer.

[Jiang2021] Jiang, X., Coffee, N., Bari, M. F., Sarro, F. (2021). Addressing class imbalance for software defect prediction: A systematic literature review. Information and Software Technology, 133, 106563.

[Jones2021] Jones, K., Wang, Y. (2021). Machine Learning and Sentiment Analysis: A Review. Journal of Information Technology, 26(3), 123-134. doi: 10.1177/1091142118757847

[Karimi2017] Karimi, F., Amiri, H. (2017). A new hybrid feature selection based on combination of PSO and ABC algorithms. Knowledge-Based Systems, 138, 85-103. https://doi.org/10.1016/j.knosys.2017.09.018

[Kaur2021] Kaur, H., Singh, A. (2021). An effective approach to imbalanced classification using data augmentation and ensemble learning. Journal of Ambient Intelligence and Humanized Computing, 12(11), 12395-12407.

[Kumar2018] Kumar, A., Da Vitoria Lobo, N. (2018). Hybrid sampling approach for imbalanced data classification: A review. International Journal of Emerging Trends Technology in Computer Science (IJETTCS), 7(2), 52-56.

[Lai2015] Lai, S., Xu, L., Liu, K., Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In AAAI (Vol. 333, No. 1, p. 2).

[Lai2021] Lai, C. H. (2021). A preliminary investigation into using Snapchat for educational purposes. Education and Information Technologies, 26(1), 703-719. https://doi.org/10.1007/s10639-020-10364-9

[Lecun1998] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324. https://doi.org/10.1109/5.726791

[Lee2020] Lee, S. Yoon, and J. H. Kim. (2020), SMOTE-Boundary: A New Hybrid Oversampling Method for Imbalanced Classification Problems. Applied Sciences, 10(5):1776, 2020.

[Li2021] Li, S., Liu, H. (2021). A Hybrid Sampling Method Combining SMOTE with ENN for Text Classification. Journal of Information Science and Engineering, 37(3), 739-753. doi: 10.6688/JISE.202108_37(3).0009

[Liu2018] Liu, C.-z., Sheng, Y.-x., Wei, Z.-q., Yang, Y.-Q. (2018). Research of Text Classification Based on Improved TF-IDF Algorithm. In 2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE) (pp. 218-222). Lanzhou, China. doi: 10.1109/IRCE.2018.8492945.

[Liu2012] Liu, Y., Zhang, J. (2012). A survey of opinion mining and sentiment analysis. In Mining text data (pp. 415-463). Springer, Boston, MA.

[Li2021] Li, S., Liu, H. (2021). A Hybrid Sampling Method Combining SMOTE with ENN for Text Classification. Journal of Information Science and Engineering, 37(3), 739-753. doi: 10.6688/JISE.202108_37(3).0009

[Liu2021] Liu, Y., Zhang, X., Lu, Y., Yang, Q. (2021). Boundary undersampling for imbalanced classification. IEEE Transactions on Cybernetics, 51(1), 376-387.

[Mahmoudi2022] Mahmoudi, O., Bouami, M.F., Badri, M. (2022). Arabic language modeling based on supervised machine learning. Revue d'Intelligence Artificielle, Vol. 36, No. 3, pp. 467-473. https://doi.org/10.18280/ria.360315

[Mikolov2013] Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[Miller2021] Miller, M. (2021). YouTube statistics that will blow your mind in 2021. Oberlo. Retrieved from https://www.oberlo.com/blog/youtube-statistics

[Mollick2021] Mollick, E. (2021). LinkedIn: An introduction. Business Horizons, 64(2), 329-332. https://doi.org/10.1016/j.bushor.2020.11.008

[Nair2021] Nair, A. J., G, V., Vinayak, A. (2021). Comparative study of Twitter Sentiment On COVID-19 Tweets. In 2021 5th International Conference on Computing Methodologies and Communication (ICCMC) (pp. 1773-1778). Erode, India. doi: 10.1109/ICCMC51019.2021.9418320.

[Ng2017] Ng, A. (2017). Machine learning: The power and promise of computers that learn by example. Communications of the ACM, 60(10), 14-24. doi: 10.1145/3132727

[Nooralahzadeh2021] Nooralahzadeh, M., Khanmohammadi, S., Mohammadi, E. (2021). An overview of text classification algorithms for imbalanced datasets. Journal of Ambient Intelligence and Humanized Computing, 12(9), 9781-9793.

[Nunes2021] Nunes, M. A., Fernandes, K., Gonçalves, M. A. (2021). Machine Learning on Imbalanced Datasets: An Evaluation of Smote Algorithms. IEEE Access, 9, 38217-38229.

[Oulin2016] Oulin, A., Grave, E., Bojanowski, P., Mikolov, T. (2016). Bag of tricks for efficient text classification. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, 427-431. https://www.aclweb.org/anthology/E17-2068/

[Obrist2021] Obrist, M., Kurashima, T., Tsuchiya, T. (2021). Understanding the usage of Instagram among young adults: Socializing, information seeking, and surveillance. Computers in Human Behavior, 122, 106836. https://doi.org/10.1016/j.chb.2021.106836

[Oz2021] Oz, M., Zhang, Y., Bondyra, E. (2021). Social Media Use in 2021. Statista. https://www.statista.com/topics/1164/social-media-usage/

[Patil2022] Patil, D., Lokare, R., Patil, S. (2022). An Overview of Text Representation Techniques in Text Classification using Deep Learning Models. In 2022 3rd International Conference for Emerging Technology (INCET) (pp. 1-4). Belgaum, India. doi: 10.1109/INCET54531.2022.9825389.

[Pennington2014] Pennington, J., Socher, R., Manning, C. D. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[Pang2008] Pang, B., Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2(1-2), 1-135.

[Panda2021] Panda, S.K., Dash, A. K. and S. N. Mishra. (2021). An improved version of Edited Nearest Neighbor Rule for Imbalanced Data Classification. Intelligent Automation Soft Computing, 27(1), 61-72.

[Pei2022] Pei, P., (2022). Short Texts Classification Based an Improved BiLSTM, 2022 5th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), Wuhan, China, pp. 297-300, doi: 10.1109/AEMCSE55572.2022.00067.

[Pennington2014] Pennington, J., Socher, R., Manning, C. (2014). GloVe: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1532-1543).

[Radford2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI Blog, 1(8), 9.

[Ren2019] Ren, M., Zeng, W., Yang, B., Urtasun, R. (2019). The devil is in the discriminator. In Advances in Neural Information Processing Systems (pp. 5811-5820).

[Ruder2017] Ruder, S., Howard, J. (2017). Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146.

[Salloum2020] Salloum, S. A., Al-Shaikhli, D. F., Al-Rikabi, R. K. (2020). SMOTE and Tomek Links Algorithm for Text Classification. In 2020 5th International Conference on Communication and Electronics Systems (ICCES) (pp. 1484-1488). IEEE.

[Sun2019] Sun, A., Lim, E. P. (2019). A survey of imbalanced text classification methods. Expert Systems with Applications, 135, 379-400. doi: 10.1016/j.eswa.2019.05.017

[Schakel2020] Schakel, A., Wilson, D. (2020). Detecting Twitter trends faster with natural language processing. Business Horizons, 63(6), 765-771. https://doi.org/10.1016/j.bushor.2020.07.007

[Shahbazi2021] Shahbazi, M., Yadollahi, M., Li, Y., Fang, W. (2021). A sentiment analysis of COVID-19 related tweets: Dataset, analysis, and directions. Information Processing Management, 58(1), 102440.

[Smailović2021] Smailović, J., Grčar, M., Lavrač, N. (2021). Challenges in sentiment analysis of social media: A comprehensive review. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 11(2), e1412. https://doi.org/10.1002/widm.1412

[Smith2018] Smith, J. D., Jones, K. L. (2018). A comparison of logistic regression models for predicting customer churn. Journal of Business Analytics, 1(2), 120-135. https://doi.org/10.1080/2573234X.2018.1493920

[Sawant2018] Sawant, A., Sarawade, N., Jathar, A. (2018). Sentiment analysis of Hindi-English code-mixed data using machine learning. In 2018 International Conference on Communication Information and Computing Technology (ICCICT) (pp. 1-6). doi: 10.1109/ICCICT.2018.8473124.

[Statista2021] Statista, M. (2021). Number of Facebook users in the Middle East and North Africa as of June 2020, by country (in millions). https://www.statista.com/statistics/263070/number-of-facebook-users-in-the-middle-east-and-africa/

[Talukder2018] Talukder, S. (2018). Text classification using deep learning: A comparative review. Journal of King Saud University-Computer and Information Sciences, 30(4), 431-448.

[Tang2015] Tang, D., Qin, B., Liu, T. (2015). Document modeling with gated recurrent neural network for sentiment classification. In Proceedings of the 2015 conference on empirical methods in natural language processing (pp. 1422-1432).

[Tang2019] Tang, Y., Zhang, Y. (2019). Document modeling with gated recurrent neural network for sentiment classification. Information Sciences, 485, 407-417. https://doi.org/10.1016/j.ins.2019.02.030

[Thelwall2012] Thelwall, M., Buckley, K., Paltoglou, G. (2012). Sentiment strength detection in short informal text. Journal of the American Society for Information Science and Technology, 63(1), 163-173. doi: 10.1002/asi.21662

[Twitter2022] Twitter. (2022). About Twitter. Retrieved from https://about.twitter.com/en_us/company/about-twitter.html

[Wang2019] Wang, Y.H., Qiao, P.L., Sun, G.L., Fan, K., Zeng, X. (2019). Classification of imbalanced dataset based on random walk model. Revue d'Intelligence Artificielle, Vol. 33, No. 2, pp. 89-95. https://doi.org/10.18280/ria.330202

[Wang2018] Wang, H., Lu, Y., Huang, C. (2018). Using natural language processing techniques for social media analysis. Journal of hospitality and tourism technology, 9(1), 38-50. https://doi.org/10.1108/JHTT-08-2017-0075

[Wang2022] Wang, R., Shi, Y. (2022). Research on application of article recommendation algorithm based on Word2Vec and Tfidf. In 2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA) (pp. 454-457). Changchun, China. doi: 10.1109/EEBDA53927.2022.9744824.

[Wei2019] Wei, Z., Du, J., Zhang, Q. (2019). Aspect sentiment classification based on LSTM-CNN and feature selection. In International Conference on Database Systems for Advanced Applications (pp. 297-309). Springer, Cham.

[Xu2021] Xu, X., Liu, L., Zhang, Z., Yang, L., Zhang, M. (2021). LSC_RU: A novel pre-trained language model for Russian sentiment classification. Expert Systems with Applications, 175, 114791. https://doi.org/10.1016/j.eswa.2021.114791

[Yadav2021] Yadav, S., Gupta, P., Mishra, A. (2021). Sentiment analysis on code-mixed data using machine learning techniques. In 2021 International Conference on Data, Engineering and Applications (IDEA) (pp. 1-6). IEEE. doi: 10.1109/IDEA52522.2021.9410357

[Yan2017] Yan, R., Zhao, Z., Ren, X., Qiu, X., Chen, Y. (2017). Convolutional neural networks for sentiment analysis of English and Chinese micro-reviews. In 2017 IEEE 4th International Conference on Cloud Computing and Intelligence Systems (pp. 419-423). IEEE. doi: 10.1109/CCIS.2017.8380678

[Zhang2015] Zhang, Y., Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820.

[Zhang2019] Zhang, Z., Xu, X., Liu, L., Yang, L. (2019). LSC: A novel language model for sentiment classification. Knowledge-Based Systems, 171, 97-105. https://doi.org/10.1016/j.knosys.2019.01.038

[Zhou2018] Zhou, L., Zhu, Y., Shen, C., Wu, Z. (2018). Chinese social media sentiment analysis using convolutional neural networks. Journal of Information Science, 44(5), 579-592. doi: 10.1177/0165551517727044

[Zhou2020] Zhou, T., Zhang, C., Zhang, S., Zhu, K., Zhang, R., Zhou, C. (2020). LSTM with entity information and attention mechanism for sentiment analysis of code-mixed social media data. In 2020 11th International Conference on Intelligent Systems and Knowledge Engineering (ISKE) (pp. 552-556). doi: 10.1109/ISKE52676.2020.9374519