



Faculty of Computer Science and Information Technology

**Malay-English Code-switched Social Media Sentiment Corpus and
Supervised Benchmark**

Afifah binti Mohd Shamsuddin

**Master of Science
2025**

Malay-English Code-switched Social Media Sentiment Corpus and Supervised Benchmark

Afifah binti Mohd Shamsuddin

A thesis submitted

In fulfillment of the requirements for the degree of Master of Science

(Computer Science)

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2025

DECLARATION

I declare that the work in this thesis was carried out in accordance with the regulations of Universiti Malaysia Sarawak. Except where due acknowledgements have been made, the work is that of the author alone. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.



.....

Signature

Name: Afifah binti Mohd Shamsuddin

Matric No.: 21020462

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

Date : 24 July 2025

ACKNOWLEDGEMENT

Here, I would like to thank everyone who has contributed to the completion of my thesis in any form. First, I want to give my highest gratitude and praise to God the Almighty for the wisdom and strength given by Him in doing this research.

Next, I would like to give my sincere gratitude to my supervisor, Ts. Dr. Sarah Flora Samson Juan, and my co-supervisor, Associate Professor Dr. Stephanie Chua Hui Li for their precious assistance, support, guidance, comments, and suggestions on this research. Their input truly helped me a lot to have a clearer understanding when doing this research project and to finish writing this thesis.

After that, here is my sincere gratitude also to the Centre for Graduate Studies, and the management of the Universiti Malaysia Sarawak for making it possible for me to complete my study in Universiti Malaysia Sarawak (UNIMAS).

Then, I want to give a big thank you to my mother, family, and friends for their support, physically and mentally throughout this project. Lastly, to Daddy, this is my answer to your wish, and my love is forever.

ABSTRACT

Code-switching between Malay and English is common on social media platforms like Twitter (currently called X) and YouTube, reflecting Malaysia's linguistically diverse society. However, this phenomenon presents a significant challenge for sentiment analysis, as the mixing of languages within sentences or phrases increases the likelihood of sentiment misclassification. To address this issue, this research proposed a methodology consisting of a corpus construction framework and a supervised benchmark for sentiment classification on Malay-English code-switched social media data, aimed to improve sentiment interpretation accuracy within such mixed-language content. The first part of the methodology focused on constructing a Malay-English code-switched sentiment corpus using social media data, referred to as MESocSentiment. This corpus consists of 78.23% neutral, 16.33% positive, and 5.44% negative tweets. Data from MESocSentiment were used in a supervised machine learning approach that included a bootstrapping technique for benchmarking sentiment models. The bootstrapping technique was implemented in four rounds, with training data from the corpus added incrementally to increase the dataset size at each round. Subsequently, 6,000 tweets with sentiment labels from bootstrapping were used to train six selected machine learning and deep learning models for benchmarking. The findings showed that SVM was the best model for classifying code-switched data in terms of accuracy and AUC-ROC metrics. It achieved 76.55% accuracy, and all its AUC values ranged between 0.90 and 0.83. Its AUC values for the micro-average and macro-average ROC curves were 0.90 and 0.84, respectively. Lastly, the MESocSentiment corpus and selected experiment outputs have been published on GitHub for public use.

Keywords: Sentiment analysis, machine learning, deep learning, artificial intelligence, natural language processing

***Korpus Sentimen Media Sosial Beralih Kod Bahasa Melayu - Bahasa Inggeris dan
Penanda Aras Terselia***

ABSTRAK

Penukaran kod antara Bahasa Melayu dan Bahasa Inggeris adalah lazim di platform media sosial seperti Twitter (kini dikenali sebagai X) dan YouTube, mencerminkan kepelbagaian linguistik masyarakat Malaysia. Walau bagaimanapun, fenomena ini memberikan cabaran besar bagi analisis sentimen kerana pencampuran bahasa dalam ayat atau frasa meningkatkan kemungkinan salah klasifikasi sentimen. Untuk menangani isu ini, kajian ini mencadangkan satu metodologi yang merangkumi rangka kerja pembinaan korpus dan penanda aras terselia untuk pengelasan sentimen terhadap data media sosial dwibahasa Bahasa Melayu-Bahasa Inggeris,, yang bertujuan meningkatkan ketepatan tafsiran sentimen dalam kandungan bercampur bahasa seperti ini. Bahagian pertama metodologi ini memberi tumpuan kepada pembinaan korpus sentimen Bahasa Melayu-Bahasa Inggeris menggunakan data media sosial, yang dinamakan MESocSentiment. Korpus ini mengandungi 78.23% ciapan neutral, 16.33% ciapan positif, dan 5.44% ciapan negatif. Data daripada MESocSentiment telah digunakan dalam pendekatan pembelajaran mesin terselia yang merangkumi teknik bootstrapping untuk penanda aras model sentimen. Teknik bootstrapping ini dilaksanakan dalam empat pusingan, di mana data latihan daripada korpus ditambah secara berperingkat untuk menambah saiz set data pada setiap pusingan. Seterusnya, sebanyak 6,000 ciapan yang dilabelkan dengan sentimen daripada proses bootstrapping telah digunakan untuk melatih enam model pembelajaran mesin dan pembelajaran mendalam terpilih bagi tujuan penanda aras. Dapatan menunjukkan bahawa SVM merupakan model terbaik untuk mengklasifikasikan data peralihan kod dari segi ketepatan dan AUC-ROC. Model ini mencapai ketepatan sebanyak 76.55%, serta semua

nilai AUC berada dalam julat antara 0.90 hingga 0.83. Nilai AUC untuk lengkung ROC purata mikro dan makro masing-masing ialah 0.90 dan 0.84. Akhir sekali, korpus MESocSentiment dan hasil eksperimen terpilih telah diterbitkan di Github untuk kegunaan umum.

Kata kunci: *Analisis sentimen, pembelajaran mesin, pembelajaran mendalam, kecerdasan buatan, pemprosesan bahasa semula jadi*

TABLE OF CONTENTS

	Page
DECLARATION	i
ACKNOWLEDGEMENT	ii
ABSTRACT	iii
<i>ABSTRAK</i>	iv
TABLE OF CONTENTS	vi
LIST OF TABLES	xi
LIST OF FIGURES	xiii
LIST OF ABBREVIATIONS	xv
CHAPTER 1: INTRODUCTION	1
1.1 Background of Study of Sentiment Analysis in Code-Switched Social Media Data	1
1.2 Problem Statement	3
1.3 Research Questions and Research Objectives	5
1.4 Scope of Research	6
1.5 Significance of the Study	6
1.6 Chapter Summary	7
CHAPTER 2: LITERATURE REVIEW	9

2.1	Overview	9
2.2	Sentiment Analysis	9
2.2.1	Definition and Objectives	9
2.2.2	Benefits of Sentiment Analysis	11
2.2.3	Challenges of Sentiment Analysis	13
2.3	Techniques Used in Sentiment Analysis	17
2.3.1	Machine Learning	18
2.3.2	Deep Learning	24
2.4	Code-Switching in Sentiment Analysis	31
2.5	Related Works to Malay-English Code-Switching	33
2.6	Related Works to Code-Switching Sentiment Analysis In Social Media	35
2.6.1	Sentiment Analysis Using Social Media Data	35
2.6.2	Sentiment Analysis Using Code-Switched Data	36
2.6.3	Malay-English Sentiment Analysis	37
2.7	Related Works to Corpus Construction for Sentiment Analysis	38
2.8	The Use of Voting Approach in Sentiment Analysis	41
2.9	Sentiment Analysis in Code-Switched Social Media Data: Literature-based Insights	43
2.9.1	Languages Used in Code-switched Sentiment Analysis	43
2.9.2	Social Media Platforms for Code-switched Data Collection	44

2.9.3	Techniques Used for Classifying Sentiments of Code-switched Data	45
2.10	Evaluation Metrics Used in Sentiment Analysis	49
2.11	Bootstrapping Technique in Sentiment Analysis	54
2.12	Use of Oversampling for Imbalanced Datasets	55
2.13	Background of Tools	59
2.14	Identified Gaps in the Literature	63
2.15	Chapter Summary	64
	CHAPTER 3: METHODOLOGY	65
3.1	Overview	65
3.2	Tools Used in Experiments	65
3.3	Research Methodology Overview	67
3.4	MESocSentiment Corpus Construction Framework	70
3.4.1	Data Collection	72
3.4.2	Pre-processing	72
3.4.3	Language Identification	75
3.4.4	Sentiment Identification	83
3.5	A Supervised Machine Learning Approach using MESocSentiment Corpus	87
3.5.1	Evaluation Method	89
3.5.2	Bootstrapping Technique for Building Training Data for Benchmarking Sentiment Models Using Support Vector Machine Model	90

3.5.3	Benchmarking Sentiment Models Built Using Machine Learning and Deep Learning Techniques	94
3.6	Chapter Summary	99
	CHAPTER 4: RESULTS AND DISCUSSION	100
4.1	Overview	100
4.2	MESocSentiment Construction Framework End Results	100
4.3	Sentiment Modelling Strategy Supervised Machine Learning Approach for Benchmarking Sentiment Models Results	101
4.3.1	Results of Bootstrapping Technique	101
4.3.2	Results of Benchmarking Sentiment Models	114
4.4	Analysis and Discussion of Results	117
4.4.1	Analysis of Findings from MESocSentiment	117
4.4.2	Advantages and Limitations of MESocSentiment Construction Framework	119
4.4.3	MESocSentiment Limitations	120
4.4.4	Analysis of Supervised Machine Learning Approach for Benchmarking Sentiment Models Results	120
4.4.5	Strength and Limitations of Bootstrapping Technique in the Supervised Machine Learning Approach for Benchmarking Sentiment Models	129
4.5	Chapter Summary	131
	CHAPTER 5: CONCLUSION AND FUTURE WORKS	132
5.1	Conclusion	132

5.2	Research Contribution	133
5.2.1	Framework for Building Malay-English Code-Switched Sentiment Corpus Using Semi-Automatic Approaches	133
5.2.2	The Malay-English Code-Switched Sentiment Corpus (MESocSentiment) from Social Media Data	134
5.2.3	A Supervised Approach with Bootstrapping Technique Implementation for Benchmarking Sentiment Models	134
5.3	Research Limitations	135
5.4	Future Works	136
	REFERENCES	138
	APPENDICES	159

LIST OF TABLES

	Page
Table 2.1 Summary of aspects of chosen machine learning and deep learning approaches	18
Table 2.2 Samples of Code-Switched Texts (Ismail & Abdullah, 2023)	32
Table 2.3 Techniques, Approaches, and Methods Used in the Selected Studies	45
Table 2.4 Comparative analysis of Malaya, Lingua, CLD3, langid.py and fastText	61
Table 3.1 Libraries, Packages and Modules Used in the Research	66
Table 3.2 Mapping table of research problems, objectives and contributions	68
Table 3.3 Samples of raw tweets and clean tweets after pre-processing	75
Table 3.4 Sample of Language Identification for Malaya	77
Table 3.5 Confidence Scores Groups and Respective Ranges	78
Table 3.6 Sample of Language Identification for Lingua	79
Table 3.7 Dominant Language Confidence Score Groups and Code-Switching Classification for Lingua	82
Table 3.8 The Result of the Sentiment Identification Process Using Voting Approach	85
Table 3.9 Sentiment Labels on Random Tweets from MESocSentiment Corpus	85
Table 3.10 Samples of True Tweets	86
Table 3.11 Samples of False Tweets	87
Table 3.12 Hyperparameter values and settings for machine learning techniques	95
Table 3.13 Hyperparameter values and settings for LSTM model	96
Table 3.14 Hyperparameter values and settings for BERT	97
Table 3.15 Hyperparameter values and settings for the CNN model	98
Table 4.1 The Statistics of the MESocSentiment Corpus In Terms of Sentiment	101
Table 4.2 Accuracy of Bootstrapping SVM Model in All Training Rounds to MESocSentiment	103

Table 4.3	Precision, Recall, and F1-Score Results for All Training Rounds	103
Table 4.4	Statistics for Tweets with Different Sentiment Labels in All Training Rounds	107
Table 4.5	Test Set Statistics for MESocSentiment, SVM Model Prediction and Post-Edited Tweets for All Training Rounds	110
Table 4.6	Accuracy of Machine Learning Models	115
Table 4.7	Accuracy of Deep Learning Models	115
Table 4.8	BERT models with hyperparameter values and settings	116

LIST OF FIGURES

	Page
Figure 2.1 Machine learning sentiment polarity classification (Dang et al., 2020)	19
Figure 2.2 Support Vector Machine (García-Gonzalo et al., 2016)	20
Figure 2.3 Random forest (Khan et al., 2021)	23
Figure 2.4 Deep learning sentiment polarity classification (Dang et al., 2020)	25
Figure 2.5 LSTM cell architecture (Shailendra et al., 2023)	26
Figure 2.6 BERT input representation (Devlin et al., 2018)	28
Figure 2.7 Pre-training and fine-tuning procedure of BERT (Devlin et al., 2018)	28
Figure 2.8 Schematic diagram of CNN (Phung & Rhee, 2018)	30
Figure 2.9 Languages used in the research papers	43
Figure 2.10 Social media sites used by the studies	44
Figure 2.11 Example of a confusion matrix	49
Figure 2.12 Example of ROC graph	53
Figure 3.1 Overview of the research methodology	69
Figure 3.2 The MESocSentiment corpus construction framework using a semi-automatic approach	71
Figure 3.3 The workflow of the sentiment identification stage	83
Figure 3.4 A supervised machine learning approach with bootstrapping technique for benchmarking sentiment models using MESocSentiment	89
Figure 3.5 The workflow of the bootstrapping technique	91
Figure 4.1 Confusion Matrices for All Training Rounds	104
Figure 4.2 Accuracy Comparison for Test Set MESocSentiment and SVM Model Prediction to Ground Truth	111
Figure 4.3 Composition of Training Set for Benchmarking Sentiment Models	113
Figure 4.4 Composition of Training Set for Benchmarking Sentiment Models after Oversampling	114

Figure 4.5	Accuracy Results of Code-Switched Sentiment Models	116
Figure 4.6	ROC graph for SVM Model	123
Figure 4.7	ROC graph for Naive Bayes Model	124
Figure 4.8	ROC graph for Random Forest Model	125
Figure 4.9	ROC graph for CNN Model	126
Figure 4.10	ROC graph for LSTM Model	127
Figure 4.11	ROC graphs for BERT Models	128

LIST OF ABBREVIATIONS

ANN	Artificial Neural Networks
BERT	Bidirectional Encoder Representations from Transformers
BiGRU	Bidirectional Gated Recurrent Unit
BiLSTM	Bidirectional Long Short-Term Memory
CGS	Centre for Graduate Studies
CNN	Convolutional Neural Networks
DA	Dialectal Arabic
DT	Decision Tree
GloVe	Global Vectors for Word Representation
LLM	Large Language Model
LSTM	Long Short-Term Memory
mBERT	Multilingual Bidirectional Encoder Representations from Transformers
MSA	Modern Standard Arabic
NB	Naïve Bayes
NER	Named Entity Recognition
NLTK	Natural Language Toolkit
NLP	Natural Language Processing
PTSD	Post-traumatic Stress Disorder
QA	Question Answering
QAS	Question Answering System
RF	Random Forest
RNN	Recurrent Neural Networks

RoBERTa	Robustly Optimized BERT Pre-training Approach
TF-IDF	Term Frequency - Inverse Document Frequency
SVM	Support Vector Machine
UNIMAS	Universiti Malaysia Sarawak
XLM	Cross-lingual Language Model

CHAPTER 1

INTRODUCTION

1.1 Background of Study of Sentiment Analysis in Code-Switched Social Media Data

Sentiment analysis is a Natural Language Processing (NLP) task used to get the public's perception of certain topics. There has been a steady interest in research about the sentiment analysis task, especially on code-switched social media data in recent years. Millions of users of social media platforms such as Twitter (currently known as X), Facebook, and YouTube are hailing from various countries, worldwide. A lot of social media users are bilingual or multilingual speakers. The ability of some social media users to communicate in two or more languages has resulted in code-switching, where they seamlessly incorporate or alternate between languages within a single post.

Code-switching is generally the situation of mixing different languages in communicating (Poplack, 2015). Code switching situation has given some challenges to sentiment analysis. These challenges include spelling variants, informal grammar forms, and scarcity of annotated data (Agüero-Torales et al., 2021). Multilingual speakers usually mix languages during informal communication, where the grammar rules loosen. This situation has led to some characteristics, including spelling variations for words and informal sentence structures (Srinivasan & Subalalitha, 2023). Hence, informal communication is a challenge for sentiment analysis models that are trained with formal language. This is because varieties of spellings and informal sentence structure will cause challenges for sentiment models to give accurate sentiments.

Another challenge for sentiment analysis on code-switched data is the scarcity of annotated data. There is still limited research on sentiment analysis of code-switched social media data, especially on Malaysian social media posts. Consequently, developing classifiers capable of handling code-switched data remains time-consuming and costly. Sentiment analysers built with models trained with monolingual data will cause misclassification due to out-of-vocabulary issues, thus making these sentiment analysers less efficient on code-switched data.

This research is focused on the sentiment analysis of code-switched Malay-English social media data. Malaysia is a multicultural society, and Malaysians generally use two or more languages in daily communications. Malay and English are two prominently used languages in Malaysia. Malay is Malaysia's official language, while English is widely spoken and used in Malaysia. There are several works regarding sentiment analysis on code-switched Malay-English data. One of the works is the study by Kong et al. (2023) that focused on sentiment analysis on a mixed Malay-English COVID-19 Twitter dataset. The dataset consists of Malay and English language and in three-class (positive, negative, or neutral) and two-class (positive and negative) sentiment labels. Nevertheless, the dataset is not fully available for public use because of restrictions on access. Another work on sentiment analysis on code-switched data involves Malay and English is the study by Kasmuri and Basiron (2019). The authors have developed the MY-EN-CS corpus for sentiment analysis using entries from personal blogs. However, the corpus is not publicly available and does not use data from any social media sites.

Therefore, this work proposes a framework to build a code-switched Malay-English social media data and a code-switched Malay-English sentiment modelling strategy with a

bootstrapping technique. The first framework applies a semi-automatic approach to construct a Malay-English code-switched corpus for sentiment analysis from Twitter data. Then, a sentiment modelling strategy is designed to develop initial sentiment classifiers using selected machine learning and deep learning models.

1.2 Problem Statement

Sentiment analysis on code-switched data, particularly in Malay-English social media contexts, presents several inherent challenges due to language variability, informal grammar, and limited availability of annotated datasets (Agüero-Torales et al., 2021). The flexibility with which users alternate between languages introduces unique linguistic variations and informal structures, which complicate annotation processes and increase the likelihood of misclassification in sentiment models. This complexity makes high-quality annotated data costly and time-consuming to produce (Ranjan et al., 2022), especially since user-generated content often includes non-standardized forms, leading to a scarcity of annotated resources (Khanuja et al., 2020).

As mentioned previously by Agüero-Torales et al. (2021), the lack of publicly available annotated Malay-English code-switched corpora from social media platforms specifically prevents the progress of sentiment analysis in this domain. While some prior studies have produced Malay-English code-switched datasets, they are often limited in scope, lack public accessibility, or are obtained from sources other than social media. Thus, these limitations reduce their relevance for analyzing informal code-switching contexts. Publicly available datasets often come with limited content and researchers cannot fully utilize them for training effective sentiment models.

On the other hand, traditional machine learning approaches for sentiment analysis face additional obstacles, such as limited ability to generalize across varied linguistic domains (Bahrin & Jantan, 2024). Although deep learning models offer powerful alternatives, they demand large, labeled data, computational resources, and hardware (e.g., GPUs) (Mao et al., 2024). Furthermore, due to the lack of sufficiently annotated data, both machine learning and deep learning approaches struggle to capture the intricate, non-linear patterns essential for accurate sentiment classification of code-switched texts. (Yadav & Vishwakarma, 2020).

Addressing these issues, the first aim of this research is to bridge the resource gap by developing a publicly accessible Malay-English code-switched sentiment corpus sourced from Twitter data. A significant challenge in sentiment analysis for Malay-English code-switched data lies in the process of manual data annotation and the lack of labeled data, which can limit the effectiveness of sentiment models. Existing methods are affected by the lack of scalable methods to annotate large or informal datasets efficiently (Iosifidis & Ntoutsi, 2020). This shortage hinders the development of sentiment classifiers for code-switched text, particularly in social media contexts where linguistic variability is high.

To address these issues, there is a need for an approach that can automate the annotation process without compromising accuracy. Prior work, such as that of Juan (2015), has demonstrated the potential of semi-supervised bootstrapping techniques to incrementally build annotated corpora with reduced reliance on manual annotation, suggesting a promising direction for addressing this challenge.

Additionally, despite advancements in sentiment analysis, identifying the most effective sentiment analysis models, whether traditional machine learning or advanced deep

learning frameworks like BERT, remains a critical challenge in optimizing classification accuracy in Malay-English code-switched contexts. By addressing these challenges, the second aim of this research is to establish a flexible framework for sentiment analysis on code-switched social media data for enhancing research capabilities and resources in this under-resourced area.

1.3 Research Questions and Research Objectives

Several research questions for this research have been created, which are:

- i) How to design a framework to build a Malay-English code-switched corpus using social media data for a sentiment analysis task?
- ii) How can a supervised machine learning approach be effectively designed to benchmark sentiment analysis models trained on the Malay-English code-switched corpus?

Hence, two objectives have been designed to answer the research questions of this research. The objectives are:

- i) To design a framework for building a Malay-English code-switched corpus from Twitter data using language identifiers and existing sentiment analysers.
- ii) To develop a supervised machine learning approach that includes a bootstrapping technique for benchmarking sentiment models using the constructed Malay-English code-switched corpus.

1.4 Scope of Research

There are several constraints used in this research. Targeted data source is from social media platforms because the presence of code-switching is prominent on those platforms. This is because most of these sites are widely used worldwide, hence, most of the users of these social media sites are bilingual or multilingual speakers. Thus, the use of multiple languages translated into their social media posts. This research is demonstrated using the social media platform Twitter, currently known as X. The targeted languages in this research are Malay and English, specifically Malay-English code-switched data. This is because Malay and English are widely spoken in Malaysia as Malaysia is a multilingual society where most of the people use two or more languages in their daily life. Malay language is the national language of Malaysia, while English is considered as second language in Malaysia after Malay. Hence, the geographical focus of collected tweets is on Malaysia-related tweets.

1.5 Significance of the Study

Generally, this research aims to contribute to the advancement of sentiment analysis, especially for code-switched Malay-English social media data. This research addresses gaps in sentiment analysis for low-resourced languages specifically Malay, which is considered a low-resource language (Fu et al., 2021). Then, this study also contributes valuable insights to the NLP community especially related to sentiment analysis. It also contributes to diversifying sentiment analysis methods in handling code-switched data more effectively.

Firstly, we propose a framework for the construction of a Malay-English sentiment corpus using semi-automatic approaches. This corpus is designed for the sentiment analysis task and is planned to be made publicly available on GitHub. By making this corpus

accessible, especially to the research community, we aim to support future studies in sentiment analysis, particularly in terms of low-resource languages and social media content. Moreover, the corpus adds a new valuable resource for NLP research on code-switched texts, especially the Malay-English language pair.

The second contribution of this research is the development of a supervised machine learning approach to benchmark sentiment models on Malay-English code-switched data using the newly acquired corpus. The approach implements bootstrapping to build initial Malay-English sentiment models, which are built from several machine learning techniques. The performance of each model is evaluated and compared to determine the best-performing one. Finally, the potential of this research is it can be used to enhance current sentiment analysis tools and applications, hence offering benefits to society, for example better understanding of public opinion or sentiment about certain issues and assisting in more effective communication on social media platforms in multilingual societies, especially in Malaysia.

1.6 Thesis Outlines

This thesis consists of five chapters. Chapter 1 briefly introduces the background of the research work. The problem statement, research questions and objectives, and the research's scope and significance are also presented in this chapter. Chapter 2 discusses literature reviews related to sentiment analysis, especially ones related to code-switched social media data and previous works. It also discusses other related topics to the research. This chapter aims to explore the current progress in this research area and show the motivation of this research based on the literature reviews.

Chapter 3 introduces the methodology used in this research and explains the framework for building the Malay-English code-switched corpus for sentiment analysis using social media data. The chapter also explains the sentiment modelling strategy using the built corpus. In Chapter 4, results from experiments in Chapter 3 are presented. The results are also analysed and discussed in this chapter. Lastly, in Chapter 5, the conclusion of this research thesis is presented with a summary of research, research contributions, research limitations, and future works for this research project.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

The background of this research is explained in this chapter. Firstly, Section 2.2 explains the background overview of the sentiment analysis task. The next section shows the techniques used in sentiment analysis, especially on machine learning techniques and deep learning models. After that, an overview of the code-switching phenomenon and its impact on the sentiment analysis task is also discussed in this chapter. Related works to sentiment analysis, especially on using social media data and code-switched data, especially on Malay-English code-switched data, are also discussed in this chapter. Lastly, this chapter also shows the motivation of this research based on the literature review analysis.

2.2 Sentiment Analysis

The definition, objectives, benefits, and challenges of sentiment analysis are discussed in this section.

2.2.1 Definition and Objectives

Sentiment analysis or opinion mining is one of the Natural Language Processing (NLP) tasks generally used to get the public's perception about particular use and specific-focused topics. According to Liu (2015), sentiment analysis (or opinion mining) is a research area that deals with people's opinions, sentiments, emotions, appraisals, and attitudes toward entities and their characteristics in text. Entities include products, services, organisations, events, issues, and topics (Liu, 2015).

An example of sentiment analysis on certain events is Malaysia's public opinions on Movement Control Order (MCO) during the COVID-19 pandemic, as portrayed in the work by Alamoodi et al. (2022). In the study, the authors investigated Malaysian public sentiment during three series of lockdowns (MCO 1.0, MCO 2.0, and MCO 3.0) using collected MCO-related tweets. The authors collected MCO-related tweets posted across three periods of lockdowns from March 2020 to November 2021. The findings showed that each lockdown reported different sentiment levels, where in all three MCOs, most tweets were positive tweets, followed by neutral and negative tweets. These results showed that in general, Malaysia's public had positive sentiment regarding lockdown measures taken by the government.

Then, an example of sentiment analysis on services is sentiment analysis on food delivery services. Samah et al. (2024) conducted sentiment analysis on users' feedback on three online food delivery (OFD) services, Foodpanda, GrabFood, and ShopeeFood in Malaysia, using Twitter data. Five main factors in reviews were considered in classifying the best online food delivery services: affordable price, promotion and discount, review rating, delivery time, and food delivery conditions. The results showed that most of the tweets in the final dataset were neutral, amounting to 72.5% of the dataset. Then, 18.4% of the dataset was positive tweets, and 9.15% of the dataset was negative tweets. Naive Bayes classification models produced an accuracy of 71.67% for the English model and 76.29% accuracy for the Malay model (Samah et al., 2024).

Sentiment analysis has also been used to gauge people's opinions on political matters, such as elections. Firdaus et al. (2024) collected tweets discussing candidates for the 2024 Indonesian Presidential Election. The final dataset contained 23,446 positive and

negative tweets. Support Vector Machine and Naive Bayes methods were used for the sentiment analysis task in this paper. The dataset was split into different ratios for training and testing, and both methods achieved an average accuracy above 75%. Support Vector Machine models showed a higher average accuracy than Naive Bayes (Firdaus et al., 2024).

Additionally, sentiment analysis is used to determine the public's opinions about environmental issues such as climate change, as shown in n in Taufek et al. (2021). The authors utilized editorial news about climate change from The Sun Daily newspaper in the Malaysian Diachronic Climate Change Corpus (MyDCCC). The findings showed that 53 articles were labelled as negative, and 6 articles were labelled positive by Azure Machine Learning. There was no article labelled as neutral in the study. The results also showed that the top three words associated with strong negative sentiments were "long," "critical," and "serious," while the top three words with strong positive sentiments were "better," "best," and "hope". From the analysis, the public demonstrated a fair degree of awareness regarding climate change, even if their sentiments were largely negative. This negativity was largely attributed to public frustration with how decision-makers handle the climate change issue (Taufek et al., 2021).

2.2.2 Benefits of Sentiment Analysis

The application of sentiment analysis in different domains has benefited various parties. The work by Gallestegui et al. (2024) applied several machine learning techniques on Banco Bilbao Vizcaya Argentaria (BBVA) bank tweets for strategic marketing insight. The results showed that eXtreme Gradient Boosting (XGBoost) algorithm achieved the highest precision of 91.2%. Findings also showed that the sentiments towards the bank were affected by marketing activities and global events, thus benefitted as a tool for marketers to

get real-time assessment in the effectiveness of campaigns and brand perceptions. The good precision results from all models used showed that the bank could leverage these techniques in customer relationship improvement and business strategy enhancement due to understanding and engagement with its digital audience (Gallestegui et al., 2024). Hence, in general, sentiment analysis enabled business to always keep aware of customers' sentiments and market trend to improve their businesses, making them remain competitive in the market.

Sentiment analysis is also applied in crisis and disaster management benefit government agencies and humanitarian aid organizations in estimating relief supply demand needed by victims (Lin et al., 2020). In the study by Ragini et al. (2018), tweets related to several disasters were collected and categorized into water, food, shelter, medical emergency and electricity before underwent sentiment analysis. The findings could benefit responders and emergency personnel to get sentiments for aids needed by victims and the location of help needed in the disaster affected areas, thus helped in mobilization of aids and rescue operation by government agencies more effectively. The analysis also could help related government agencies to create sound strategies for information management systems for the upcoming disasters (Ragini et al., 2018). The study by Behl et al. (2021) proposed a multilayer perceptron (MLP) model on sentiment analysis regarding earthquake disaster and COVID-19. Two earthquakes' datasets were used for model training while COVID-19 tweets collected by the author were used in model testing. These models were used to explore the reusability of the model trained on previous disasters. The proposed model achieved 83% classification accuracy on the original COVID-19 dataset. One drawback of the proposed method was its dependence on the resource type being requested in the disaster under examination, which can substantially affect its classification accuracy.

Furthermore, sentiment analysis is utilised in the healthcare sector for various reasons. In the work of Aattouchi et al. (2021), medical sentiments consisted of several concerns: diagnosis or medical conditions, patient's health, and treatment. The patient's health is impacted by diagnosis and treatment. Ismail et al. (2020) utilised a deep learning approach to identify cancer survivors living with PTSD from tweets. The authors gave several suggestions from the findings, such as a prediction system that acts as an alarm system to detect the presence of PTSD based on users' postings. This suggestion may benefit social media users who are cancer survivors with PTSD. The authors also anticipated that the data collection approach in the study may be used to assist current trauma screening questionnaire-based, thus helping more patients to get proper medical treatment. The results of sentiment analysis also benefit governments in improving their healthcare system based on the public's opinion, such as the work by Wang et al. (2020). In the paper, the authors performed sentiment analysis and semantic network analysis on collected HPV-vaccine-related Sina Weibo posts from females and retrieved users' locations. From the findings, the authors concluded the study with several suggestions, including that the government should create awareness programmes to educate the female population about HPV vaccine use. The government should also provide domestic bivalent HPV vaccination/screening for low-income, high-risk women, which would help to prevent cervical cancer.

2.2.3 Challenges of Sentiment Analysis

The code-switching phenomenon and low-resource languages have imposed some challenges to the sentiment analysis task. Low-resource language or under-resourced language means the language has a linguistic resource shortage (Birjali et al., 2021). Examples of linguistic resources for sentiment analysis include sentiment lexicon and annotated sentiment corpus or dataset. A challenge of sentiment analysis in low-resource

languages is the lack of language tools (Aliyu et al., 2024). An example of this situation was in the study by Hegde et al. (2022), where the authors needed to manually filter out the collected YouTube comments from Tulu language-related videos due to the absence of tools to identify Tulu language texts.

A significant disadvantage of supervised learning approaches widely used in sentiment analysis is the need to get massive training data for sentiment classifiers to achieve high accuracy (Ren et al., 2014). Hence, the scarcity of data and lack of pre-trained models have become major challenges for languages with linguistic resource shortages (Girija et al., 2023). Several techniques have been used to solve this issue, for example constructing the data resource from scratch as well as using unsupervised, semi-supervised, and transfer learning approaches (Birjali et al., 2021).

Due to a lack of resources for sentiment analysis in the Albanian language, Kastrati et al. (2021) created a large-scale dataset of 10,742 manually classified comments from Facebook in Albanian and proposed the ALBANA analyser for sentiment analysis of the Albanian texts, especially social media contents. The effect of the attention mechanism on capturing long-range dependencies in the texts was also studied in this paper. The findings showed that the BiLSTM model with attention mechanism and word embeddings generated from the dataset achieved the highest F1-score compared to other models, with 72.09% (Kastrati et al., 2021).

Meanwhile, the code-switching situation has presented challenges for sentiment analysis tasks, including variation of spelling, informal grammar forms, and scarcity of annotated data (Agüero-Torales et al., 2021). Informal grammar forms happen because multilingual speakers usually use a mix of languages in informal communication with

relaxed grammar rules. The leniency leads to several characteristics, including multiple variations of spellings of words and non-standard sentence structures (Srinivasan & Subalalitha, 2023). This variability in code-switching based on individual preferences makes it difficult to uncover the underlying deep compositional semantics within code-switched text (Poria et al., 2020).

Code-switched text often appears in informal communication, leading to various characteristics that complicate analysis. For instance, users may choose different word spellings due to a lack of consistent spelling rules, making normalisation of the text difficult. Additionally, code-switched texts have no fixed word order, as users structure their sentences freely, further complicating sentiment analysis (Khanuja et al., 2020). Ambiguous words also pose a challenge, as they cannot be definitively classified into a specific language within a language pair without considering the context (Srivastava & Singh, 2020). Therefore, informal communication, especially involving code-switching, is a challenge for sentiment analysis models that are trained using formal language. It is challenging for sentiment analysis models to give accurate sentiment labels to the data due to spelling variations and non-standard sentence structure.

Another challenge for sentiment analysis of code-switched texts is the limited availability of annotated resources for code-switched data. Currently, standardized datasets in code-switched languages are scarce, with only datasets of a few language pairs available for shared tasks. This scarcity is due to a lack of data and annotated resources for code-switched languages, even when one or both languages in the pair are well-resourced (Khanuja et al., 2020). Additionally, the varied prevalence of different languages complicates and increases the cost of the annotation process (Ranjan et al., 2022). Moreover, the evaluation of annotations and the proficiency of human annotators is hard due to the

characteristics of code-switched texts (Srivastava & Singh, 2021). Hence, it is challenging to build classifiers that can handle code-switched data in terms of time and cost because of the unavailability of labelled code-switched corpus. Sentiment analysis models trained exclusively on monolingual data may lead to misclassification problems stemming from out-of-vocabulary issues, thereby restricting the effectiveness of these models when handling code-switched data (Shamsuddin et al., 2024).

The presence of code-switching in short texts such as tweets and comments on social media also posed challenges in NLP tasks including sentiment analysis due to the nature of the texts. One of the challenges in detecting code-switched social media short texts is code-switching can occur at the sentence or clause level, the word level, and the sub-word level (Mave et al., 2018). This code-switching complexity especially for intra-word code-switching resulted in a need for the models to be capable of segmenting and tagging each part of the word with the correct language identifiers (Sabty et al., 2021).

Moreover, the lexical relatedness between languages in a language pair also complicates language detection of code-switched social media texts. The more the lexical relatedness between languages in a language pair, the more challenging it is to differentiate both languages. As shown in the study by Mave et al., (2018), it is more challenging to determine the language of a word in Spanish-English tweets than in Hindi-English social media texts due to Latin influence in Spanish and English. Meanwhile, the findings from Molina et al. (2019) showed that the language identification task was harder for MSA-DA texts than Spanish-English texts because both languages are closely related.

Besides, some users use transliteration for words in certain languages such as Hindi in their social media posts. However, these transliterated words are not standardised, hence a single word in a script word can have various transliterated surface forms in another script. As an example, instances of the transliterated forms of the Telugu word తిన్నావా (winnAvA) are tinnaavaa (repeating the long vowel twice) or just tinnava (not indicating the long vowel at all) or tinnavaa, tinnaava (mixture of double and single vowels) (Kusampudi et al., 2021).

2.3 Techniques Used in Sentiment Analysis

According to Birjali et al. (2021), techniques used in sentiment classification can be divided into several major categories which are machine learning approaches, lexicon-based approaches, hybrid approaches, and other approaches that consist of transfer learning and aspect-based approaches. This research focuses on machine learning approaches especially traditional machine learning algorithms and deep learning models. Table 2.1 shows the summary table of traditional machine learning algorithms and deep learning models used in this research. The techniques are Naïve Bayes (NB), Support Vector Machine (SVM), Random Forest (RF), LSTM, BERT, and CNN. These techniques are summarised in terms of model type, use cases, and previous works. Previous works shown in this table were code-switching sentiment analysis works in various language pairs. More background details about these techniques are shown in Section 2.3.1 and Section 2.3.2 for machine learning algorithms and deep learning models, respectively.

Table 2.1: Summary of aspects of chosen machine learning and deep learning approaches

Aspect	Naive Bayes	Support Vector Machine	Random Forest	LSTM	BERT	CNN
Model Type	Probabilistic classifier	Linear classifier (with kernel trick for non-linear data)	Ensemble learning	Recurrent Neural Network (RNN)	Transformer-based model	Artificial Neural Network (ANN)
Use Cases	Classification	Regression, Classification	Regression, Classification	Time series prediction, Natural language processing tasks	Natural language processing (NLP) tasks	Computer Vision
Used In	Yadav et al. (2020), Chakravarthi et al. (2022)	Singh et al. (2020), Raveendrarasa and Amalraj (2020)	Sabri et al. (2021), Chakravarthi et al. (2022)	Thara and Poornachandran (2022), Shanmugavadivel et al. (2022)	Younas et al. (2020), Bhagya and Nadera (2022)	Thara and Poornachandran (2022), Shanmugavadivel et al. (2022)

2.3.1 Machine Learning

Machine learning techniques are one of the categories of techniques used in the sentiment analysis task. Some of the widely used machine learning techniques used in the sentiment analysis task are Support Vector Machine (SVM), Naïve Bayes (NB), and Random Forest (RF). Figure 2.1 shows the sentiment polarity classification using a machine learning technique. The diagram is adapted from a figure in Dang et al. (2020).

From Figure 2.1, the sentiment classification process is categorized into three stages: pre-processing, feature extraction, and sentiment modelling. The texts first undergo pre-processing steps such as tokenization and removing stop words. Then, the pre-processed texts undergo the feature extraction stage. The feature extraction output then undergoes the modelling stage, where the texts are used to train machine learning models such as Naive Bayes, Support Vector Machine and Logistic Regression. The machine learning models then predict the sentiment labels of texts into either positive or negative (two-class sentiment classification) or positive, negative or neutral (three-class sentiment classification).

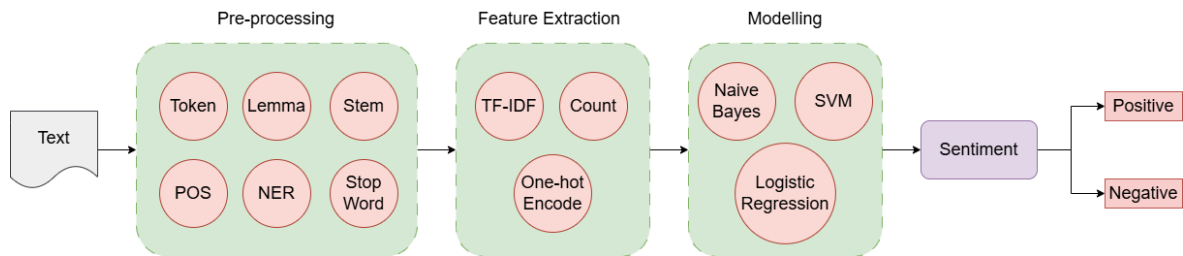


Figure 2.1: Machine learning sentiment polarity classification (Dang et al., 2020)

Support Vector Machine is a machine learning algorithm used in classification and regression problems. Vladimir Vapnik and colleagues developed the algorithm in several publications. The papers included Boser, Guyon and Vapnik (1992) and Cortes and Vapnik (1995) work. Boser, Guyon and Vapnik (1992) proposed a training algorithm for optimal margin classifiers. Meanwhile, Cortes and Vapnik (1995) proposed support vector networks for two-group classification problems.

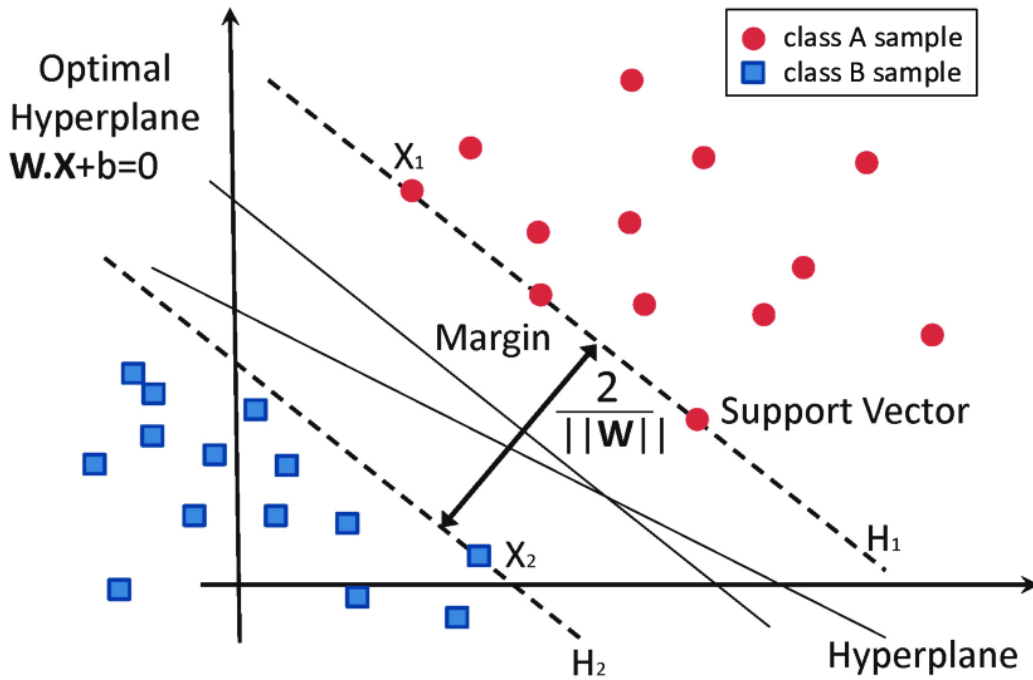


Figure 2.2: Support Vector Machine (García-Gonzalo et al., 2016)

Support Vector Machine aims to find the hyperplane that can separate support vectors of two classes with maximum margin. Figure 2.2 shows two essential components of this technique: the hyperplane and support vector. Support vectors are data points from different classes nearest to the hyperplane. However, to separate non-linear data, the kernel trick is used. Examples of kernel functions include polynomial and Gaussian Radial Basis Function kernel functions. Depending on the issue or application, an appropriate kernel is needed to improve the technique (Jakkula, 2006).

Singh et al. (2020) used the Support Vector Machine model as the only traditional machine learning technique as one of the baseline models for sentiment polarity classification of the Nepali Sentiment Analysis (NepSA) dataset. The dataset contained some code-switched Nepali-English comments from YouTube. The findings showed that SVM model achieved an accuracy of 0.684 for not concatenated embeddings. The model achieved an accuracy of 0.714 for concatenated embeddings. Support Vector Machine (SVM) was

also used in the work by Raveendrarasa and Amalraj (2020) for sentiment analysis of code-switched Tamil-English texts. The findings showed that SVM achieved accuracy values of 0.5232 and 0.5262 for unigram and unigram+bigram scenarios respectively. However, the F1-score for SVM (unigram) was higher than SVM (unigram+bigram) with 63.53% compared to 57.6%.

Naïve Bayes classifier is one of the traditional machine learning techniques that are usually used in text classification tasks such as sentiment analysis. Based on Thomas Bayes's Bayes theorem, published after his death in 1763, the classifier is widely used in various fields due to its robustness, easy implementation, and high speed and accuracy (Raschka, 2014; Routledge, 2018). The equation of Bayes Theorem is in Equation 2.1.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \quad \text{Equation 2.1}$$

Where,

$P(A|B)$: probability of hypothesis A (such as a specific sentiment) given evidence B is true (the words in the tweet)

$P(B|A)$: probability of evidence B (the specific words in the tweet) given hypothesis A (a particular sentiment) is true

$P(A)$: prior probability of hypothesis A which is the initial probability of the sentiment class (positive, negative, or neutral) before considering the tweet's content.

$P(B)$: probability of evidence B (the words in the tweet), regardless of the sentiment

The situation of predicting or classifying sentiment may also be explained using the theorem. The example scenario is to predict the sentiment of a tweet as positive, negative, or neutral. The Bayes theorem for the situation is shown in Equation 2.2.

$$P(C_i | tweet) = \frac{P(tweet|C_i) \cdot P(C_i)}{P(tweet)} \quad \text{Equation 2.2}$$

Where,

$P(C_i | tweet)$: Probability that the tweet is of sentiment class C_i (Positive, Negative, Neutral) given the words in the tweet.

$P(tweet|C_i)$: Probability of the tweet appearing with its specific words if the sentiment is C_i

$P(C_i)$: Prior probability of sentiment class C_i (i.e., how common the sentiment is in your dataset)

$P(tweet)$: probability of evidence B

There are several types of Naïve Bayes classifiers, such as Multinomial Naïve Bayes, Gaussian Naïve Bayes, and Bernoulli Naïve Bayes. Multinomial Naïve Bayes is usually used in document classification by checking and classifying them into a particular category. The frequency of words present in the document is used as the predictor for this classifier. For Gaussian Naïve Bayes, the predictors are treated as having continuous values, assuming that these values are drawn from a Gaussian distribution or normal distribution. Lastly, Bernoulli Naïve Bayes' predictors are Boolean values where only yes or no values are accepted. An example is whether a word occurs in a text or not (Sarkar, 2024).

Yadav et al. (2020) utilized Multinomial Naïve Bayes as one of the baseline models for sentiment analysis of code-mixed Hindi-English texts. The model achieved an accuracy

of 0.71, a little bit lower to the accuracy of the ensemble classifier (0.74) proposed in the study. Chakravarthi et al. (2022) utilized Multinomial Naïve Bayes for sentiment analysis task of DravidianCodeMix dataset that contained code-mixed comments from Tamil, Kannada, and Malayalam languages. The model achieved weighted F1-score of 0.47 for Tamil, 0.46 for Malayalam, and 0.48 for Kannada. Based on these results, the model performed best for code-mixed Kannada sentiment analysis.

Random Forest (RF) is another machine learning algorithm that is commonly used in sentiment analysis. It is a supervised learning technique and a type of ensemble learning method which was introduced by Leo Breiman in 2001. Random Forest is widely used in classification and regression methods (Biau & Scornet, 2015; Breiman, 2001). The diagram of the random forest is shown in Figure 2.3.

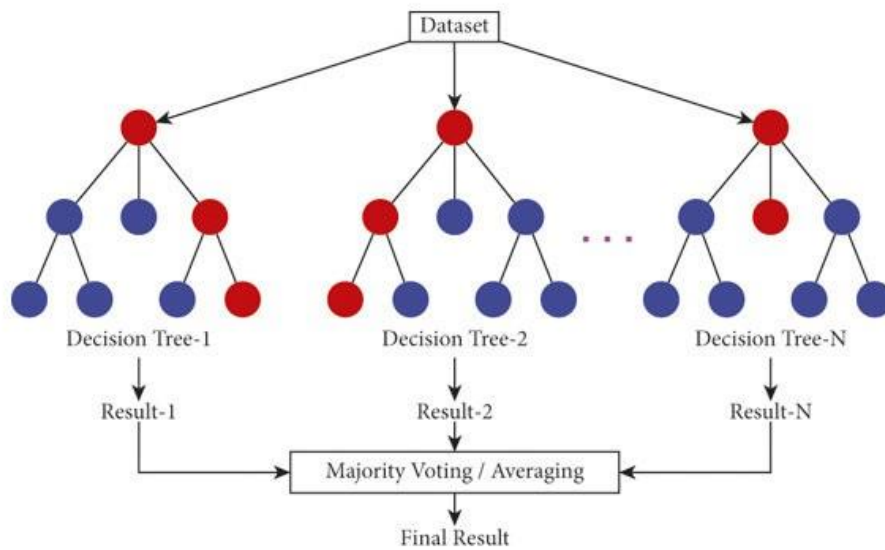


Figure 2.3: Random forest (Khan et al., 2021)

Random Forest consists of a group of decision trees built based on bootstrap aggregating and random subspace methods (Ren et al., 2017). Three main parameters need to be set before the experiment which are the node size, number of trees, and the number of feature sizes. Bootstrapping in this algorithm happens where a random subset of training

data is selected with replacement. This process occurs at each tree in the forest. Data points are into trees and each tree will make its prediction. For classification cases, the predicted class will be based on the majority vote. After that, the test data (out-of-bag) sample will be used for cross-validation to finish the prediction (IBM, 2024). Sentiment analysis falls into the classification case.

Random Forest was utilised as one of the baseline models in the paper by Sabri et al. (2021). The results of sentiment analysis on code-mixed Persian-English texts showed that the Random Forest model achieved an accuracy of 62.12, lower than the accuracy of the proposed model (66.17). Random Forest also had been used by Chakravarthi et al. (2022) for the sentiment analysis task of the DravidianCodeMix dataset. The dataset consisted of code-mixed comments from Tamil, Kannada, and Malayalam languages. The findings showed that the weighted F1-score of Random Forest was 0.53 for Tamil, 0.56 for Malayalam, and 0.55 for Kannada. These scores showed that Random Forest performed best for code-mixed Malayalam sentiment analysis.

2.3.2 Deep Learning

Deep learning models such as Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN) are currently widely used in sentiment analysis research area. Examples of deep learning models are Bidirectional Encoder Representations from Transformers (BERT), Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN). Deep learning approaches is a subset of machine learning approaches (Birjali et al., 2021).

Figure 2.4 shows the sentiment classification process using deep learning models based on the work of Dang et al. (2020). The pre-processing step consists of several common

tasks in this step such as tokenization, lemmatization, part-of-speech tagging, and stop words removal. Instead of using feature extraction, the pre-processed text will pass through a dense embedding (dense layer or embedding layer) and hidden layer. After passing the hidden layer, the sentiment of the text will be classified into positive or negative polarity.

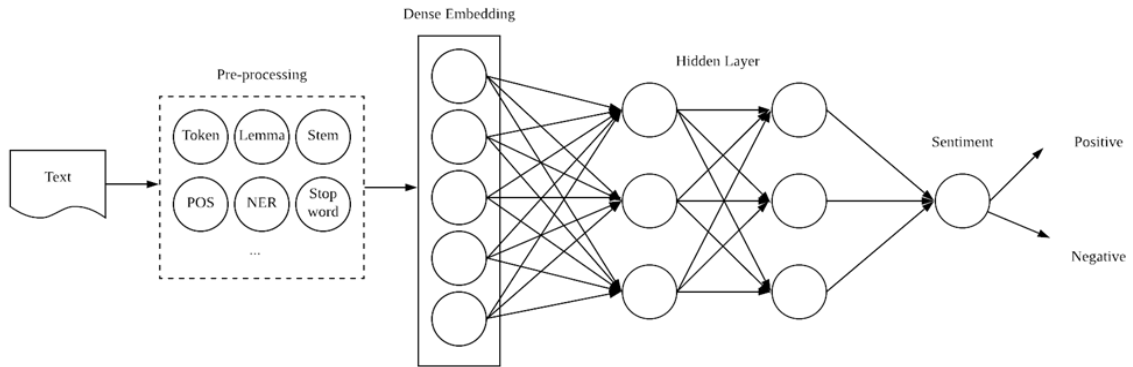


Figure 2.4: Deep learning sentiment polarity classification (Dang et al., 2020)

The long short-term memory (LSTM) model has been introduced by Hochreiter and Schmidhuber (1997) to address error backflow issues in the recurrent neural network using a gradient-based learning algorithm. Some popular applications of LSTM include time series prediction and natural language processing tasks (Van Houdt et al., 2020). Shailendra et al. (2023) have shown the architecture of an LSTM cell in their paper, as in Figure 2.5. As shown in Figure 2.5, in an LSTM layer, there are memory blocks, each of which has one or more recurrently connected memory cells and three multiplicative units, which are input, output, and forget gates where these gates are the only communication points between cells and net (Graves & Schmidhuber, 2005). LSTM suits tasks that involve sequences of input data due to their ability to select either to remember or forget information from preceding time steps (Shailendra et al., 2023).

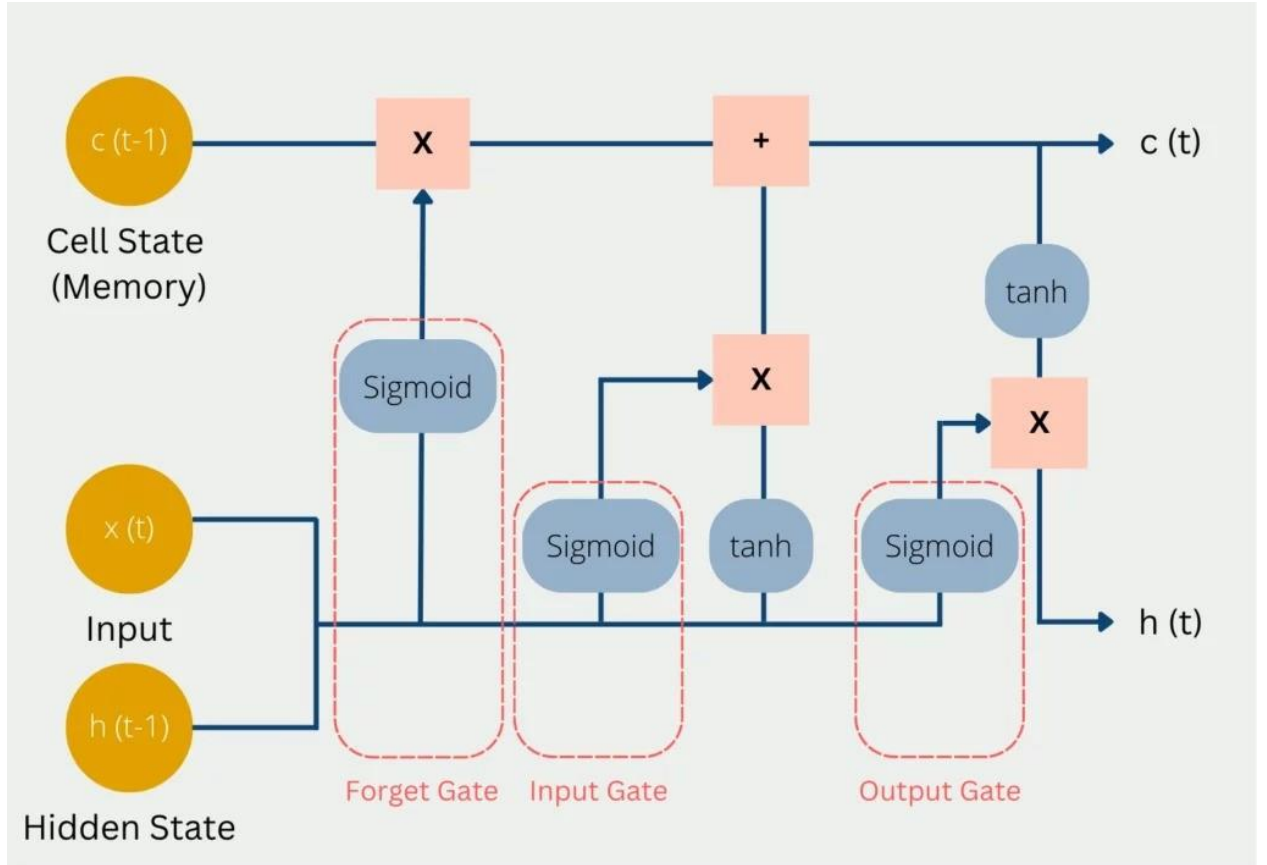


Figure 2.5: LSTM cell architecture (Shailendra et al., 2023)

In LSTM architecture, there are two states of cells which are cell state and hidden state. Cell state and hidden state are used to carry information from the previous and current time steps. The cell state is considered as the “long-term” memory and the hidden state is considered as the “short-term” memory (Lawrence, 2023). There are three gates in the LSTM architecture which are the forget gate, input gate, and output gate (Shailendra et al., 2023). The role of the forget gate is to decide which information should be removed from the cell state by taking the hidden state of the previous time step and the current input. The information is passed to a sigmoid activation function that produces outputs with a value between 0 and 1. The meaning of 0 is to forget and 1 is to keep (Lawrence, 2023).

The value of information stored in the cell state is determined by the input gate by taking the current input and hidden state of the previous time step. Divided into two parts,

the first part is using the sigmoid activation function to decide the percentage of the required information. Then, the two values are passed to a Tanh activation function in the second part. The function is used to map the data between -1 to 1 . The output of tanh is multiplied with the output of sigmoid activation function to get the output of the input gate. This updates the cell state (Lawrence, 2023). The hidden state for the next time stamp is returned by the output gate. The first part of the gate is the sigmoid function which decides the information to be kept. Then, the updated cell state is passed through a tanh function and then is multiplied with output of sigma function to get the new hidden state. The previous cell state is multiplied by the output of the forget gate. The result is summed with the output of the input gate to get the updated cell state (Lawrence, 2023).

Thara and Poornachandran (2022) utilised LSTM for sentiment analysis task of code-mixed Malayalam-English dataset with two word embedding methods: Word2Vec and FastText. The findings showed that in terms of accuracy, LSTM with FastText achieved higher accuracy (0.7381) than CNN with Word2Vec (0.6995). An LSTM model was used in the work by Shanmugavadivel et al. (2022) for sentiment analysis of code-mixed Tamil-English data as one of the deep learning models utilised in this research. The LSTM model achieved an accuracy of 0.63, comparable to all models used in the research.

Introduced by Google in 2018, this transformer-based model is considered as one of the state-of-art techniques in Natural Language Processing (NLP) tasks (Devlin et al., 2018). Its framework is divided into pretraining and fine-tuning steps. The BERT model works by using masked language models when pre-training deep bidirectional language representations. In the pretraining steps, authors have used two unsupervised tasks which are masked language models and next sentence prediction and in the original paper, the author pretrained BERT using document-level corpus which are BooksCorpus and English

Wikipedia text passages only (Devlin et al., 2018). BERT input representation is shown in Figure 2.6 based on the paper by Devlin et al. (2018). The input representation is the result of combinations of token embedding, segment embedding, and position embedding. The [CLS] token is used to indicate special classification on the first token of the sequence. The [SEP] token or separate token is used to differentiate sentences in a sequence.

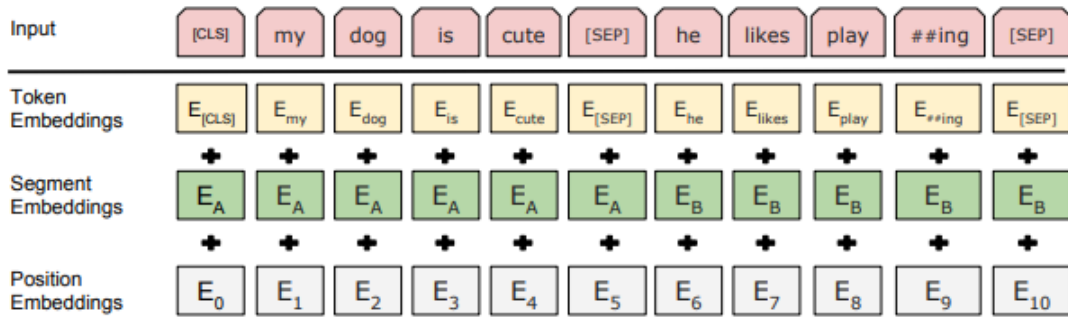


Figure 2.6: BERT input representation (Devlin et al., 2018)

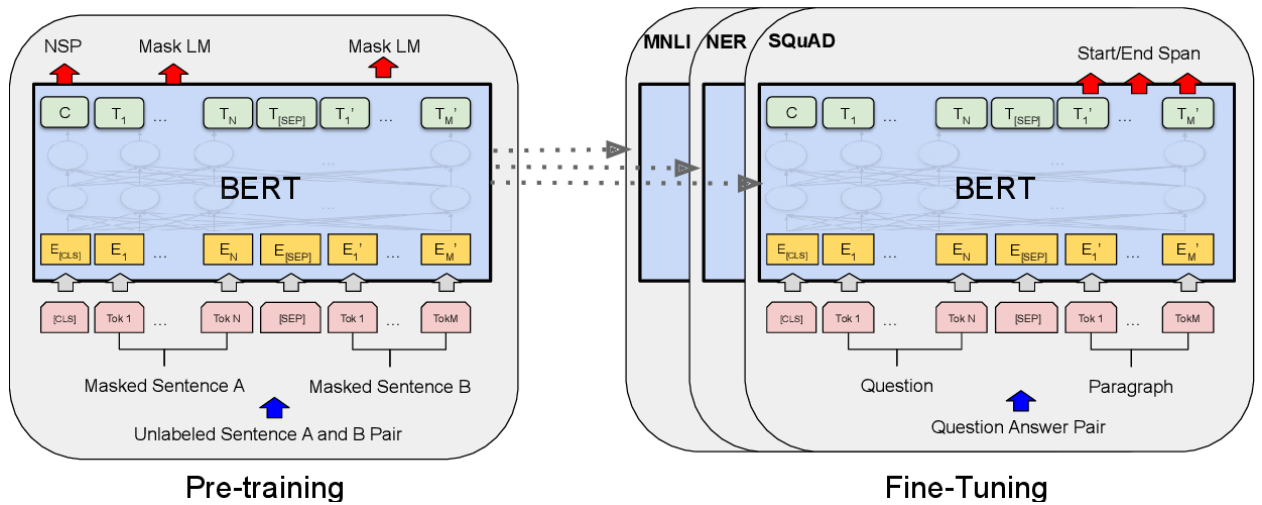


Figure 2.7: Pre-training and fine-tuning procedure of BERT (Devlin et al., 2018)

Figure 2.7 describes pre-training and fine-tuning procedures for BERT based on Devlin et al. (2018). The same architectures are used in both pre-training and fine-tuning procedures except for output layers. The same pre-trained model parameters are used to initialize models for different downstream tasks. During fine-tuning, all parameters are fine-

tuned. [CLS] token is in front of every input example, and the [SEP] token is used to separate sentences in the input (Devlin et al., 2018).

From Figure 2.7, the pre-training of BERT utilizes two unsupervised tasks which are masked language modeling (masked LM) and next sentence prediction (NSP). In masked language modeling, a percentage of input tokens are randomly masked and then predicted. The task is used for BERT to capture contextualized word embedding by conditioning on the left and right context. It allows BERT to capture dependencies from left and right directions. The aim of Next Sentence Prediction (NSP) is to predict whether two sentences are consecutive or not. The task is also for BERT to understand the relationship between sentences and to capture the semantics at the sentence level. This to make BERT understand the information flow between sentences better (Devlin et al., 2018).

Fine-tuning of BERT involves taking the pre-trained BERT model and adapting it to specific downstream tasks by adding a task-specific output layer and training it on labeled data for that task. Fine-tuning aims to capitalize on the pre-trained BERT capability on a specific task such as sentiment analysis and question answering. This procedure usually involves training an additional task-specific layer while keeping the pre-trained BERT model weights or updating them with a smaller learning rate to preserve the learned representations (Devlin et al., 2018).

Younas et al. (2020) had utilized multilingual BERT (mBERT) for sentiment analysis of code-mixed Roman Urdu and English text. The dataset used for training was the MultiSenti dataset, which included social media text from the 2018 general elections in Pakistan. The mBERT model achieved 0.65 for both accuracy and F1-score without any hyperparameters fine-tuning. Meanwhile, the model achieved 0.69 for both accuracy and F1-score with fine-tuning of hyperparameters (Younas et al., 2020). Bhagya and Nadera

(2022) also used the pre-trained mBERT model in their paper for sentiment analysis of Malayalam-English (Manglish) code-mixed social media texts. The findings showed that the model achieved the highest accuracy of 0.71, same as the DistilBERT model.

CNN is a deep learning model that is a type of artificial neural network (ANN) that has achieved successful performance in computer vision disciplines. The difference between CNN and traditional ANNs is that CNN utilises information about the input type instead of focusing on the problem domain (O'Shea & Nash, 2015). Some contributions in the paper by Fukushima (1980) have become the groundwork for modern CNN. After that, in the paper by LeCun et al. (1998), the authors have introduced LeNet-5 architecture which has influenced the development of modern CNNs. In the paper by LeCun et al. (1998), the authors have introduced LeNet-5 architecture which has influenced the development of modern CNNs. Then, the work by Krizhevsky et al. (2012) has shown the capability of CNN in the image classification task. The model consists of three main layers: a convolution layer, a pooling layer, and a fully connected layer. The layers involved in the CNN model are shown in Figure 2.8.

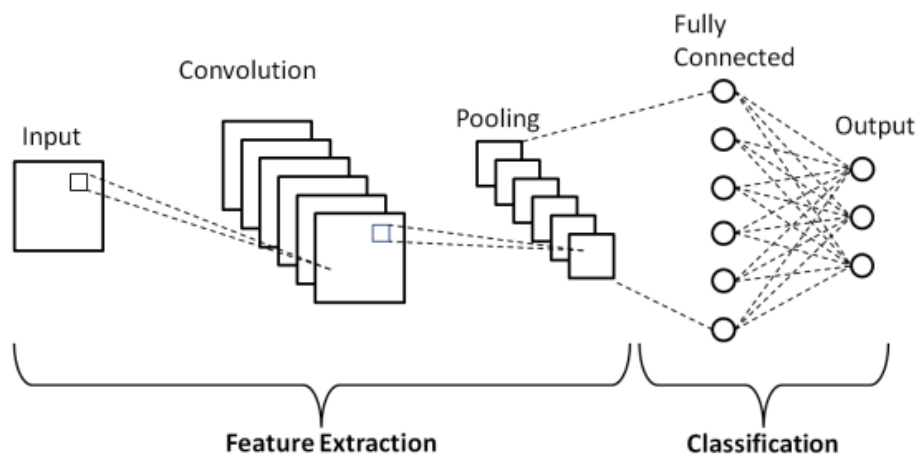


Figure 2.8: Schematic diagram of CNN (Phung & Rhee, 2018)

Firstly, the input will be passed to the convolution layer. The most vital layer, the convolution layer is used for feature extraction which usually has both linear and non-linear operations that is convolution and activation operations (Yamashita et al., 2018). Then, the output will be passed to the pooling layer. The pooling layer is used to decrease the dimensionality of feature maps and produce new feature maps that have condensed resolution thus, the layer is used to reduce the number of parameters by extracting only vital information and to manage overfitting (Gholamalinezhad & Khosravi., 2020). Then, the output will be passed to the fully connected layer. The fully connected layer is where the output feature maps from the convolution or pooling layer are flattened and then connected to one or more fully connected layers (dense layers) to the final outputs of the network (Yamashita et al., 2018).

CNN also has been used in the sentiment analysis task such as in the work by Thara and Poornachandran (2022). The authors utilised a CNN model for the sentiment analysis task of a code-mixed Malayalam-English dataset with two word embedding methods: Word2Vec and FastText. The findings showed that CNN with Word2Vec achieved higher accuracy (0.7455) than CNN with FastText (0.7329). Shanmugavadivel et al. (2022) also used a CNN model for sentiment analysis of code-mixed Tamil-English dataset as one of the deep learning models used in the paper. The CNN model achieved an accuracy of 0.65, the highest among deep learning models in the paper, along with the BiGRU model.

2.4 Code-Switching in Sentiment Analysis

Code-switching is a situation of mixing different languages in a communication (Poplack, 2015). Code-switching or code-mixing situation, in general, is the use of two or more languages in the same sentence. This situation occurs typically in communications within bilingual and multilingual societies due to their fluency in two or more languages.

Poplack (1980) has proposed several code-switching types which are inter-sentential, intra-sentential, and tag-switching. Inter-sentential code-switching is when a speaker uses two distinct languages across separate sentences within an utterance, where the first sentence is fully expressed in the speaker's primary language, followed by a complete sentence in the second language (Khalil & Firda, 2018).

Meanwhile, intra-sentential code-switching is where a speaker changes languages in the same sentence, usually with smooth articulation or delivery without noticeable pause, interruption, or hesitation (Zirker, 2007). After that, tag-switching refers to the insertion of certain words or short phrases from one language into an utterance that is otherwise structured in another language (Ismail & Abdullah, 2023). Table 2.2 below shows samples of code-switched texts that were categorized into either of these three types of code-switching. These samples were referenced from Instagram comments collected in the work by Ismail and Abdullah (2023).

Table 2.2: Samples of Code-Switched Texts (Ismail & Abdullah, 2023)

No.	Code-Switched Samples	Code-switching Types
1.	What was the answer? tak sempat dengar	Inter-sentential
2.	Selamat hari raya 😂nanti macam biasa video call yer 😂😂😂no need to buy baju raya	Intra-sentential
3.	Alamak 4.15 Am im gonna sleep	Tag-switching

Yadav et al. (2020) have stated three reasons why code-mixing happens in social media posts and other informal communications. First, bilingual and multilingual speakers use words from other languages they know when the translation is not enough, the word is hard to pronounce, or they are unsure how to use it in sentences. Secondly, parents want to

train their children to communicate in other languages and opt for language mixing to ensure they learn words in both languages fairly. Third, trends and dominant languages such as English may influence the words used to communicate.

Meanwhile, according to Roslan et al. (2021), the top three reasons undergraduate students use Malay-English code-switched text on WhatsApp are habitual expressions, emphasis in a point, and lack of vocabulary. The first reason is that students often use discourse particles like “kan” and “lah” in their communication. Secondly, they believe that switching languages during an argument can strengthen the message. Additionally, they use code-switching when they have difficulty explaining a matter in their first or second language or when they cannot find equivalent words in their first language.

Similarly, Lubis et al. (2017) identified several reasons why late adolescents post code-switched messages on Facebook. One reason is repetition, which is used to clarify or emphasise the message. For example, the post “Siap-siap kerja..prepare to work” suggests that the user is ready to start working, with the emphasis on Bahasa Indonesia message by repeating it in English. Although code-switching enhances expressiveness in multilingual communication, it also introduces various complexities to the sentiment analysis task. These challenges are elaborated in Section 2.2.3.

2.5 Related Works to Malay-English Code-Switching

There are several prior studies have been done concerning code-switching, specifically for Malay-English in various NLP tasks or domains, including sentiment analysis. In the study by Lim et al. (2021), the authors proposed a conceptual framework for a Malay-English question answering (QA) system using the combination of translation models with CNN as a deep learning question classification model to improve Malay-English mixed-language QAS. The framework was developed to tackle the classification

issue of Malay-English code-switched questions for answer generation. The data used for the study was collected from DoctorOnCall, a COVID-19 health forum platform. The first phase of the system (question processing) consisted of language identification of questions, text normalisation, and translation of the questions. In the second phase, the CNN model was utilised for the question classification of the translated questions. The last phase was the development of a retrieval model via NER, check and retrieve, and answer ranking (Lim et al., 2021).

After that, the study by Maskat et al. (2024) developed a bi-annotated Malay-English code-switching dataset from collected X (formerly known as Twitter) posts. Each post was annotated with the gender of the author and the author's identifiers for anonymisation. The main purpose of the developed dataset was to facilitate the development of NLP tasks related to biological gender identification and authorship attribution. Examples of these tasks include content and recommendation personalisation and plagiarism detection. The dataset also served as a valuable training resource for novel and pre-existing Malay-English (Manglish) language transformers (Maskat et al., 2024).

Furthermore, Suhaimin et al. (2024) constructed an annotated dataset for sarcasm detection and sentiment analysis for the public security domain using English-Malay data from TikTok and Twitter. The comments and tweets in the dataset were labelled with three-classes sentiment labels as well as either sarcastic or vice versa. The dataset could be utilised for various purposes including sentiment analysis and sarcasm detection in code-switched texts. It also could serve as training data and references for tasks related to multilingual data and the public security domain (Suhaimin et al., 2024).

Meanwhile, the primary goal of the study by Suresh et al. (2025) was to introduce CS-Sum, a benchmark to gauge the ability of LLMs to comprehend code-switching

phenomenon via code-switching dialogue summarisation to English. This study covered three language pairs: Mandarin-English, Tamil-English, and Malay-English, with 900-1,300 human-annotated dialogues for each language pair. Nine open-source and one proprietary LLMs were evaluated in this research for several tasks including few-shot learning and translate-summarise. From the findings, three major failures of LLM were code-switching loss, meaning shift from poor translation, and speaker misattribution (Suresh et al., 2025).

2.6 Related Works to Code-Switching Sentiment Analysis in Social Media

Related works in this section are divided into three categories: sentiment analysis using social media data, sentiment analysis using code-switched data, and works on Malay-English sentiment analysis.

2.6.1 Sentiment Analysis Using Social Media Data

Sentiment analysis is one of the key tasks in social media analytics (Fan & Gordon, 2014). The task falls into the ‘understand’ stage in social media analytics, where the captured data are analysed for their meanings and to generate metrics for decision-making (Fan & Gordon, 2014). There are some examples of sentiment analysis using data from social media sites.

In the paper by Altawaier and Tiun (2016), the authors utilised several machine learning techniques for the performance comparison of sentiment analysis using Arabic Twitter data. Machine learning techniques used in this study were Naïve Bayes (NB), Decision Tree (DT), and Support Vector Machine (SVM). The DT technique performed best among all three techniques regarding precision, recall and f-measure with values of 80.9%, 79.4%, and 78.9% respectively. In Singh et al. (2019), the authors extracted agriculture-

related code-mixed English-Punjabi data from Facebook, Twitter, and YouTube. Sentiments of pre-processed data were classified using the Support Vector Machine and Naive Bayes techniques. SVM (trigram) has achieved the highest accuracy, with nearly 86%.

Singh et al. (2020) collected Nepali data from YouTube for aspect-based abusive sentiment detection. The collected data also contained Nepali-English code-switched data. SVM, BERT, CNN and BiLSTM models were used for the sentiment classification task. From the result, the BiLSTM model achieved the highest accuracy in concatenated category with 0.815 accuracy. Meanwhile, Maity et al. (2020) built a benchmark Hindi-English code-mixed corpus (BullySentEmo) annotated with bully, emotion, and sentiment labels. The authors have developed the MT-MM-BERT+VecMap framework based on BERT and VecMap with emoji modality. For the sentiment analysis task (single task), the framework achieved 75.53(+/-1.76) accuracy in the unimodal (text) category.

2.6.2 Sentiment Analysis Using Code-Switched Data

The sentiment analysis task is also used on code-switched data in various language pairs. For example, Utomo and Romadhony (2023) utilised the mBERT model on code-mixed Indonesian-English tweets for sentiment analysis. The highest accuracy is 76% using epoch 7 and batch size 16. Then, Yadav et al. (2020) used code-mixed Hindi-English texts on two proposed approaches: an ensembled classifier and BiLSTM model, along with five baseline models: Multinomial Naive Bayes, Support Vector Machine, Stochastics Gradient Descent, Extreme Gradient Boosting, and LSTM. The ensembled classifier achieved the highest accuracy among all models, with 0.74 accuracy.

Jamatia et al. (2020) utilised several machine learning and deep learning models for sentiment analysis of code-switched English-Bengali and two English-Hindi datasets. In the

10-fold Cross-Validation experiment on the English-Bengali dataset, the GLoVe DoubleBiLSTM model achieved the highest accuracy of 65.8 among all models. Meanwhile, the BERT (base, uncased) model achieved the highest accuracy for both English-Hindi datasets among all models for the 10-fold Cross-Validation experiment.

2.6.3 Malay-English Sentiment Analysis

Several works on sentiment analysis have used code-switched data involving Malay and English languages. Kasmuri and Basiron (2019) developed the MY-EN-CS corpus for sentiment analysis using entries from personal blogs. Using lexicons, code-switched sentences using Malay and English words were identified from chosen blogs' entries. Sentences were annotated as either factual or opinion sentences. However, the corpus was not publicly available and did not use data from any social media sites.

Kong et al. (2023) recently focused on sentiment analysis on a mixed Malay-English COVID-19 Twitter dataset. The authors collected 108,246 tweets from September to December 2021, with 67% in Malay, 27% in English, and the rest in Chinese and other languages. After that, 11,568 tweets were randomly chosen to be manually annotated to create the MyCovid-Senti dataset consisting of Malay and English. The dataset was in three-class (positive, negative, or neutral) and two-class (positive and negative) sentiment labels. However, currently, the dataset is not fully available for public use due to restrictions of access.

The work by Yann et al. (2024) focused on using artificial neural network in sentiment analysis of Malay-English mixed language text. The data used in this study were comments collected from Malaysia's Ministry of Health news portal and The Star social media pages related to COVID-19 daily cases. The dataset contained 7,907 Malay-English comments that pre-processed and manually annotated into either positive, negative or neutral

sentiments. The author proposed a framework that utilized both text and emoji analysis on the mixed dataset using BiLSTM. The proposed method with dropout and batch normalization on dataset with emoji achieved the highest accuracy among all comparison models with 76.60%. However, the dataset is not publicly available.

However, instead of focusing on the Malay-English code-switched dataset, Romadhona et al. (2022) focused on Bahasa Rojak, a famous dialect in Malaysia that consists of Malay, English, and Chinese. The authors constructed the Bahasa Rojak Crawled Corpus (BRCC) using data augmentation techniques to pre-train language models for code-mixing phenomena. They also developed a pre-trained model named "Mixed XLM" that could automatically tag the language of input tokens to process code-mixing input. The author compiled the SentiBahasaRojak sentiment analysis dataset to evaluate the effectiveness of the Mixed XLM model. However, one of the SentiBahasaRojak dataset data sources was from financial and stock websites instead of entirely using social media data.

2.7 Related Works to Corpus Construction for Sentiment Analysis

Authors of various previous sentiment analysis research for different code-switching language pairs have decided to construct their corpora for their research using various data sources such as social media sites and blogs. In the study by Vilares et al., (2016), the authors utilised the Spanish-English training set from a workshop that originally contained 11, 400 tweets in the first step of the study (Solorio et al., 2014). The dataset was filtered out to remove monolingual tweets resulting in 3,062 filtered tweets. After that, the filtered tweets were annotated by three annotators fluent in Spanish and English based on two labelling strategies: SentiStrength criteria and standard polarity classes (positive, neutral, negative). In terms of standard polarity classes, the corpus comprised 42.88% neutral tweets, 31.45% positive tweets, and 25.67% negative tweets.

Meanwhile, the focus of the study by Mandal et al. (2018) was the corpus construction for Bengali-English code-mixed texts. The study started with data collection which the authors collected code-mixed data from Twitter and filtered and cleaned the data. Then, the post-processed tweets underwent the data annotation phase. This study utilised both system-based annotations that consisted of language and sentiment tagging systems as well as manual annotators. Manual annotators' efforts were drastically reduced with the use of system-based annotations. However, manual annotators were still needed to build a gold-standard corpus. The released data distribution for the corpus showed the combination of training and test datasets was 4, 960 tweets (Mandal et al., 2018).

Besides, the Malay-English subjectivity corpus (MY-EN-CS) was developed by Kasmuri and Basiron (2019) for sentiment analysis using blog entries from 23 chosen blogs. The Malay-English code-switched entries were identified using lexicons. Then, two annotators who were proficient in written and spoken Malay and English were chosen to annotate identified code-switched entries for the corpus. At the end of the study, the MY-EN-CS corpus contains selected 2, 316 subjective and objective Malay-English code-switched sentences (Kasmuri & Basiron, 2019).

After that, the study by Chakravarthi et al. (2020a) focused on the corpus creation for Malayalam-English code-mixed texts for the sentiment analysis task. Firstly, the authors collected 116, 711 sentences from YouTube post comments under 2019 Malayalam movie trailers. Any monolingual sentences and sentences with Malayalam scripts were removed and the remaining texts were further pre-processed. The cleaned Malayalam-English code mixed comments amounted to 6, 739 sentences. The sentences were annotated into either positive, negative, mixed feelings, neutral, or non-Malayalam by six annotators that

proficient in Malayalam and English. Many of the sentences in the corpus were positive sentences, amounting to 41.71% of the corpus (Chakravarthi et al., 2020a).

Meanwhile, Chakravarthi et al. (2020b) developed on Tamil-English code-mixed sentiment corpus creation. In this study, the authors collected 184, 573 YouTube comments under 2019 movie trailers videos. Any monolingual Tamil and English sentences were removed and sentences in other languages besides Tamil and English were identified. After pre-processing, the remaining sentences amounted to 15, 744 Tamil-English (Tanglish) sentences. The code-mixed sentences were annotated by 11 annotators into either positive, negative, mixed feelings, neutral, or other language. The corpus contained 15, 744 sentences, with 67.07% of the corpus being positive sentences (Chakravarthi et al., 2020b).

On the other hand, the focus of the study by Kusampudi et al. (2021) was on code-mixed Telugu-English texts (CMTET) dataset construction for sentiment analysis. In the data collection phase, the authors collected CMTET data from Twitter and movie reviews on YouTube. The collected data were pre-processed before undergoing the data annotation phase. In the data annotation phase, the annotation was carried out by five Telugu native speakers who are proficient in English. Any monolingual Telugu and English sentences were removed, and the remaining sentences were annotated with either positive, neutral, or negative sentiments. At the end of the phase, the dataset contained 19, 857 sentences of which 7, 925 were positive, 7, 713 were negative and 4, 219 were neutral sentences (Kusampudi et al., 2021).

Furthermore, the study by Hegde et al. (2022) focused on the corpus construction for Tulu code-mixed texts. The first step was the collection of YouTube comments under Tulu language-related videos. It was found that the comments were written in English, Kannada, Tulu, or a combination of these languages in Kannada/Latin or Kannada and Latin

characters. The comments written fully in English or Kannada script were manually filtered out, and after other pre-processing steps, the result was 7, 171 Tulu code-mixed comments. In the data annotation phase, each comment was annotated as either positive, neutral, negative, mixed feelings, or not Tulu. Not Tulu comments were the comments that do not contain Tulu material written in Kannada or Latin script. This process involved 15 native Tulu speakers as human annotators. The Tulu code-mixed corpus contained 7, 171 sentences, where 44.12% of the corpus were positive sentences (Hegde et al., 2022).

2.8 The Use of Voting Approach in Sentiment Analysis

The voting approach has been utilised by some prior sentiment analysis studies. In general, it works by combining outputs from multiple classifiers to predict the label of an instance (Zhou, 2012). Examples of the voting approach include majority voting and soft voting. The mechanism of hard voting, also known as majority voting, is where each classifier predicts one class label for an instance, these votes are collected and the class label with the most votes becomes the final label of the instance (Mohammed & Kora, 2023). Majority voting has been applied in the study by Suandi et al. (2024) with SVM, RF, and KNN for sentiment classification on collected online learning-related tweets. The findings showed that the majority voting with SMOTE achieved the highest accuracy over other models with 97% accuracy (Suandi et al., 2024).

Differing from majority voting, weighted voting includes the weight of the classifier in the calculation. One type of weighted voting is the weighted average voting. The mechanism of this method is different weights are given to each model based on their importance in prediction. The mechanism is for a particular instance; each classifier's predictions for each class label are multiplied by the weight of the models to produce the

weighted sum. The result then is divided by the sum of the weights of the classifier. These weights may be used to calculate the weighted average of each class label. Hence, the class label with the highest weighted average becomes the final class label of the instance (Mohammed & Kora, 2023).

The weighted voting has been applied in the study by Matrane et al. (2024) for automatic sentiment annotation of Moroccan dialect comments. In this study, models utilised in the voting were BiGRU, CNN-BiGRU, LSTM, CNN-LSTM, BiLSTM, and CNN-BiLSTM. Predictions from all these models were aggregated to derive a single polarity of a comment by the calculation of the weighted arithmetic mean of predicted labels. These predictions were inputted into a meta-classifier based on logistic regression algorithms that derived coefficients based on the inputs. These coefficients were the weights for the models that were utilised in the weighted arithmetic mean calculation along with the initial prediction of all models. The findings showed that the proposed weighted voting approach outperformed other approaches in 41.7% of cases, making it the best-performing approach (Matrane et al. 2024).

The voting approach specifically majority voting is chosen in this research over other ensemble methods because it is to be used in the sentiment identification task. This method is chosen due to the intention of imitating the manual annotation process by several fluent or native speakers of language(s) in a dataset to create a high-quality sentiment corpus using selected existing sentiment analysers. Majority voting is also chosen because of its more straightforward mechanism of directly using the class label as output compared to other voting methods, making it easier to understand.

2.9 Sentiment Analysis in Code-Switched Social Media Data: Literature-based Insights

A literature review was conducted to discover recent trends in sentiment analysis, especially on code-switched social media data. The purpose of this review was to examine recent research efforts in this area and to understand the development and progression of the field. This section presents a synthesis of existing literature on sentiment analysis involving code-switched social media data. A total of 22 articles were reviewed, focusing on studies published between 2017 and 2022. The complete list of reviewed studies is provided in Appendix A. The reviewed works are analysed in terms of languages used, social media platforms used for data collection, and techniques used for sentiment classification.

2.9.1 Languages Used in Code-switched Sentiment Analysis

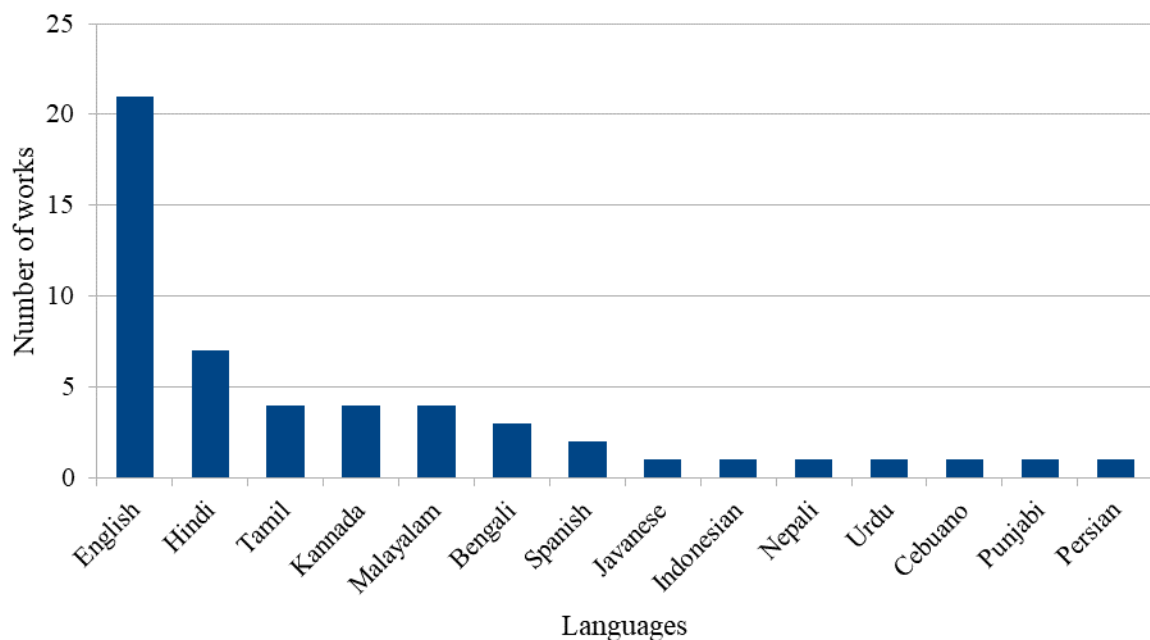


Figure 2.9: Languages used in the research papers

From 22 selected works, there were 14 languages that have been studied for code-switching sentiment analysis. Based on the result, the Malay language was not yet widely

researched in code-switching sentiment analysis, especially in social media data. Figure 2.9 showed languages that were investigated in the selected articles. There were 14 languages studied and English is the most used language as it was used in 21 studies. Most of the other languages were used in one study only besides Hindi, Tamil, Spanish, Bengali, Kannada, and Malayalam languages which were used in two or more studies. Based on the figure, besides English, which was used in most of the chosen studies, most of the studied languages are mainly spoken in the Asia continent only except for Spanish. This means that much of the research on code-switching sentiment analysis involves languages that are widely spoken in this geographical area.

2.9.2 Social Media Platforms for Code-switched Data Collection

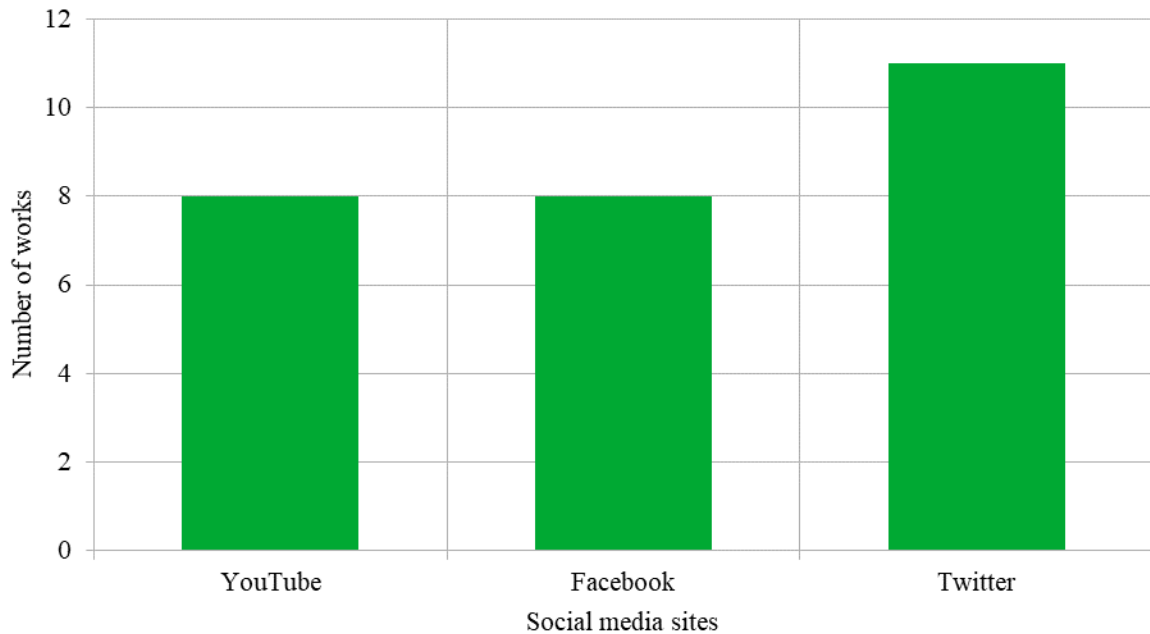


Figure 2.10: Social media sites used by the studies

From all the collected works, most of them created their own datasets or used other works' datasets collected from social media sites such as Facebook, Twitter, and YouTube regarded various categories of topics. The social media sites used by the selected studies

showed in Figure 2.10. The graph showed that 11 studies used Twitter data in their works. Then, 8 studies used Facebook data, and 8 studies used YouTube data in their works. It was seen that most of the selected works used Twitter as their main data source. According to the article by Drus and Khalid (2019), Twitter was the most used social media site to extract data. This social media platform became popular because its data is available for people to access and copy depending on their desired topics by using keywords and hashtags via API.

2.9.3 Techniques Used for Classifying Sentiments of Code-switched Data

In general, techniques used for sentiment classification are categorized into several major categories which are machine learning approaches, lexicon-based approaches, hybrid approaches and other approaches, which are transfer learning and aspect-based approaches (Birjali et al., 2021). Most of the approaches used in sentiment analysis can be categorized into the lexicon-based approach and machine learning approach, while the deep learning approach is one of the subsets of the machine learning approach (Birjali et al., 2021). Examples of machine learning techniques are Support Vector Machine (SVM) and Random Forest (RF). Deep learning models such as CNN and LSTM are also used in sentiment analysis of code-switched data. Based on the findings, most of the chosen articles used two or more approaches as comparison models in their studies. The techniques, approaches, and methods used are shown in Table 2.3.

Table 2.3: Techniques, Approaches, and Methods Used in the Selected Studies

Techniques/Approaches/Methods	2017	2018	2019	2020	2021	2022	Total
Lexicon-based							
VADER					1		1
Sentiwordnet					1		1
Lexicon Lookup			1	1			2

Table 2.3: continued

Machine Learning							
Multinomial Naïve Bayes	1		1	2		2	6
Support Vector Machine (SVM)	1		2	2		1	6
Naïve Bayes	1		1	2	1		5
Random Tree	1						1
Decision Tree	1					1	2
Sequential Minimal Optimization (SMO)				1			1
Random Forest (RF)				1	1	2	4
Stochastic Gradient Descent (SGD)				1			1
Extreme Gradient Boosting (XGBoost)				1			1
K-Nearest Neighbor (K-NN)						1	1
Logistic Regression						2	2
Linear Support Vector Classifier (LSVC)						1	1
Deep Learning							
Multilayer Perceptron	1						1
Convolutional Neural Networks (CNN)		2		2		2	6
BiLSTM		1		4	1	2	8
Double BiLSTM				1			1
Char-LSTM			1				1
Subword LSTM			1	2			3
LSTM				1		2	3
ELMo				1			1

Table 2.3: continued

BERT				5	2	3	10
Modified mBERT				1			1
DistilBERT						2	2
CharBERT						1	1
ALBERT						1	1
Attention-based Bidirectional RNN				1			1
XLM						2	2
XLNet						2	2
RoBERTa						2	2
XLM-R				1		1	2
BiGRU						2	2
GRU						1	1
Hybrid Approach							
CMSA (Lal et al., 2019)			1				1
NBSVM			1	1			2
SACMT			1				1
Ensembled Classifier (Yadav et al., 2020)				1			1
BERT+VecMap						1	1
CNN + LSTM						1	1
LSTM + CNN						1	1
CNN + BiLSTM						1	1
BiLSTM + CNN						1	1

Table 2.3: continued

Feature Extractor							
Count Vectorizer	1						1
TF-IDF				1			1
Embedding							
GLoVE				1			1
MUSE				1			1
BiSkip				1			1
BiCVM				1			1
Synthetic Data (GCM)				1			1
VecMap						1	1
Doc2Vec		1					1
FastText		1	1			1	3
Word2Vec						1	1
Transfer Learning							
IndicBERT						1	1
Extreme Learning Machine (ELM) Kernels							
RBF						1	1
Polynomial						1	1
Linear						1	1

From Table 2.3, 59 techniques, approaches, or tools were used by the chosen studies that come from either lexicon-based, machine-learning, or deep-learning approaches. Other techniques also included hybrid models, feature extractors, and embedding. From the table, the most used traditional machine learning approaches were Support Vector Machine

(SVM), Naïve-Bayes (NB), and Random Forest (RF). Meanwhile, the most used deep learning approaches were Bidirectional Encoder Representation from Transformer (BERT), Long Short-Term Memory (LSTM), and Convolution Neural Network (CNN). These included variations of LSTM such as Bidirectional LSTM (BiLSTM) and Subword LSTM.

2.10 Evaluation Metrics Used in Sentiment Analysis

Several evaluation metrics have been used to analyse results in sentiment analysis works. Some evaluation metrics include confusion matrix, accuracy, precision, recall, F1-score, and AUC-ROC. The first metric is the confusion matrix. A confusion matrix is a cross table that records a number of occurrences between the actual (true) classification and the predicted classification (Grandini et al., 2020). Figure 2.11 shows an example of a confusion matrix. The columns show the predicted label for all classes, while the rows show the actual (true) labels.

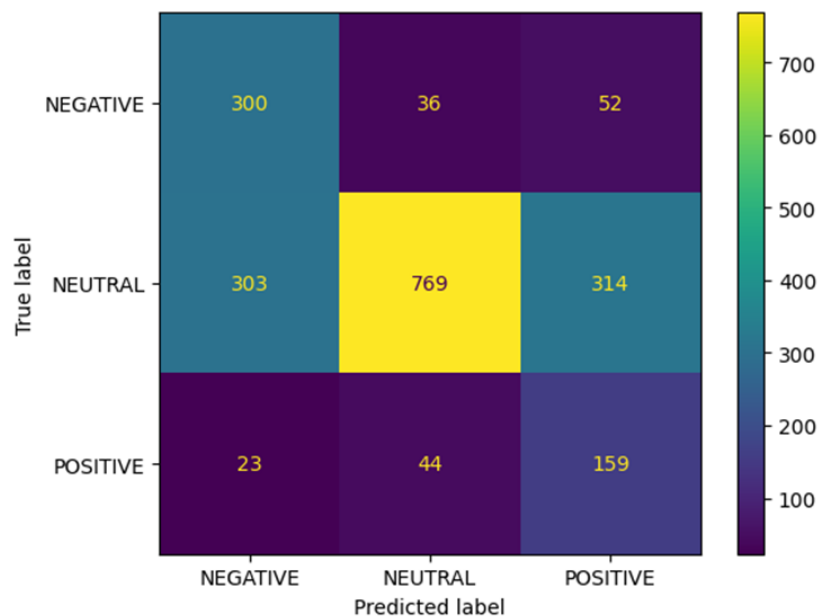


Figure 2.11: Example of a confusion matrix

From the figure above, the columns show the model prediction while the rows display the true classification. The classes are listed in the same order in the rows and columns. Hence, the correctly classified elements are located on the main diagonal from top left to bottom right, and they correspond to the number of times the two raters agree (Grandini et al., 2020). Rows represent the actual classes of the data, and columns represent the predicted classes by the model. Four important metrics can be interpreted from the confusion metrics: true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN). Firstly, true positives (TP) are when the model correctly predicts the actual class of an instance. The true positive values for all classes in the dataset are shown in diagonal elements of the confusion matrix.

Meanwhile, off-diagonal elements of the confusion matrix correspond to false positive (FP) and false negative (FN). False positives (FP) are when the model incorrectly predicts an instance to belong to a particular class when it belongs to other classes. These are values in the columns outside the diagonal for a particular predicted class. For example, the value 303 in the "True label: NEUTRAL" row and "Predicted label: NEGATIVE" column is a false positive for class NEGATIVE because 303 instances of class NEUTRAL were incorrectly predicted as NEGATIVE.

False negatives (FN) are when the model failed to predict the instances of a particular class. These are values in the rows outside the diagonal for a particular actual class. For example, the value 36 in the "True label: NEGATIVE" row and "Predicted label: NEUTRAL" column is a false negative for class NEGATIVE because 36 instances of class NEGATIVE were incorrectly predicted as NEUTRAL. True negatives (TN) are when the model correctly predicts that an instance is not belonging to a particular class. Elements of TN are ones outside the row and column of a particular class. For example, all instances that

are not in the “Actual label: NEGATIVE” row or “Predicted label: NEGATIVE” column are added to get true negative (TN) for the NEGATIVE class. These metrics are also important in the following explained evaluation metrics.

The second metric is accuracy which is used to measure the performance of the model in predicting the correct class of each data in a dataset. The metric is suitable if the analysis is more towards individual data instead of multiple classes. It provides an average measure that is well-suited to balanced datasets because it disregards the class distribution (Grandini et al., 2020). The formula for accuracy is shown in Equation 2.3.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{Equation 2.3}$$

The next metric is precision. The metric is used to show the percentage of correctly predicted sentiments in a specific sentiment category (FasterCapital, 2023). For example, if the class analysed is ‘Positive’ class, it measures the elements that are true positives divided by elements that are predicted as positives by the model. The metric show how can the model be trusted in predicting an individual data as positive (Grandini et al., 2020). The formula for precision is shown in Equation 2.4.

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{Equation 2.4}$$

Recall is a performance metric used to find a model’s performance in getting all the data of a class in the dataset. For example, if the focus is on the positive class, recall is the true positive elements divided by the row sum of actual positive elements. Recall is used to measure the ability of the model to identify the positive data from all positive samples present in the dataset (Kundu, 2023). The formula for recall is shown in Equation 2.5.

$$\text{Recall} = \frac{TP}{TP + FN} \quad \text{Equation 2.5}$$

The F1 score is a performance metric that uses the harmonic mean of precision and recall. It combines precision and recall into a single metric, making it particularly useful in scenarios with class imbalance, as it helps maintain a balance between precision and recall (Christen et al., 2023). The equation for F1-score is shown in Equation 2.6.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad \text{Equation 2.6}$$

AUC-ROC metrics are also used to evaluate the performance of classifiers or classification models in distinguishing between classes. The receiver operating characteristics (ROC) graph is a technique used to visualize, organize and selecting classifiers according to their performances. Meanwhile, the area under an ROC curve (AUC) is used to represent the expected performance of ROC to a single scalar value. The classifier's AUC is equivalent to the probability that a classifier will place a randomly picked positive instance above a randomly picked negative instance in terms of ranking (Fawcett, 2006). Yet, AUC is a measure of the discriminability of a pair of classes in multi-class classification. Figure 2.12 shows an example of a ROC graph.

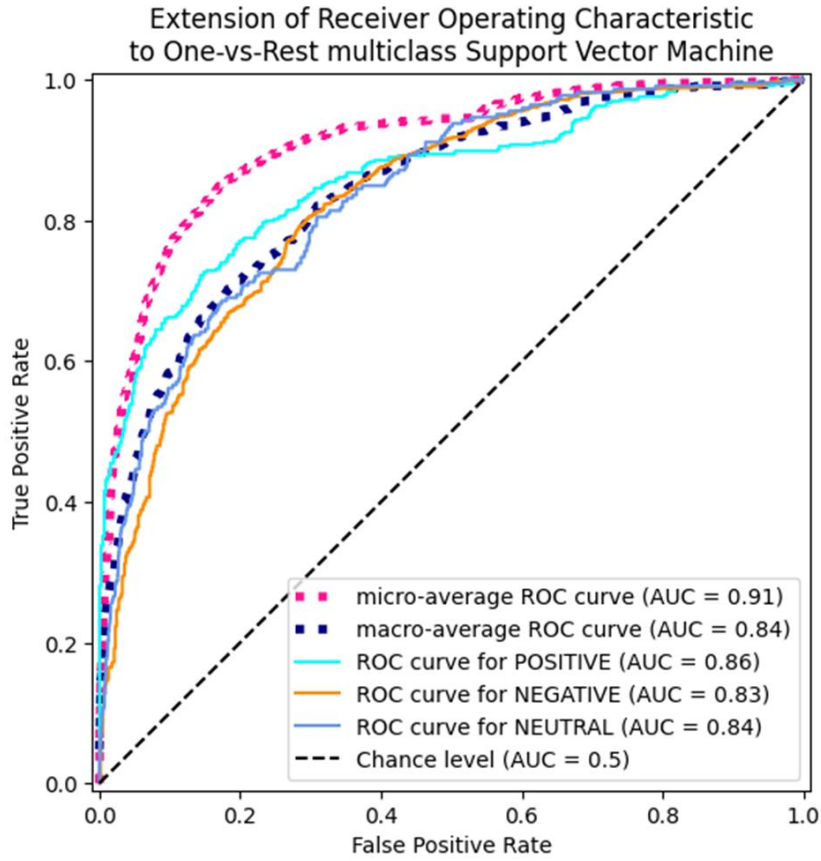


Figure 2.12: Example of ROC graph

Figure 2.12 shows the ROC graph consisting of the false positive rate (FPR) on the x -axis and the true positive rate (TPR) on the y -axis. According to Fawcett (2006), a ROC graph portrays the relative trade-off between true positives (benefit) and false positives (cost). The equations for both TPR and FPR are shown in Equation 2.7 and Equation 2.8, as in the paper by Wang and Carvalho (2024).

$$\text{TPR} = \frac{\text{Total True Positive}}{\text{Total Positive}} \quad \text{Equation 2.7}$$

$$\text{FPR} = \frac{\text{Total False Positive}}{\text{Total Negative}} \quad \text{Equation 2.8}$$

In Figure 2.12, five ROC curves are plotted into the graph, and each ROC curve has its own AUC values. Three ROC curves correspond to each class, while another two ROC curves are for average. Because the ROC graph is for multi-class classification, there are two ROC curves for average: micro-average and macro-average. The micro-averaging approach combines the contributions from every class to determine the average metrics. Meanwhile, the macro-average is determined by independently computing the metric for each class and then averaging these values, thereby ensuring equal treatment of all classes a priori. Hence, the initial step is to aggregate each class's true and false positive rates before averaging them (Pedregosa et al., 2011).

2.11 Bootstrapping Technique in Sentiment Analysis

The bootstrapping technique is a resampling technique that uses data from a sample to create a sampling distribution. This is done by repeatedly drawing random samples with replacements from the known sample (The Pennsylvania State University, 2024). This technique has been used in several sentiment analysis tasks such as in the work by Hassan et al. (2013). In the paper, the author proposed a bootstrapping ensemble framework (BPEF) to improve sentiment classification performance. The framework consisted of two phases: the expansion stage and the contraction stage. The expansion stage was about bootstrap parametric ensemble framework, and the contraction stage was about Step-wise Iterative Model Selection (SIMS). The results showed that BPEF could attain higher product recall and average recall across classes compared to other methods (Hassan et al., 2013).

The bootstrapping technique also has been used in the work by Xie and Wang (2013). The bootstrapping technique in this paper referred to a semi-supervised framework that combined a general classifier and a context classifier. The bootstrap technique enhanced the performance of the sentiment classification without a need for extensive labelled data. The

experiment results showed that the framework achieved better performance than state-of-art supervised classifiers (Xie & Wang, 2013). Another work that has used bootstrap for sentiment analysis is by Mudinas et al. (2018). The authors applied two-phase bootstrapping in the paper to create domain-specific sentiment classifiers using unlabelled data. The findings showed that the bootstrapping method achieved higher accuracy than unsupervised approaches and achieved comparable accuracy with supervised approaches. Lastly, in the work by Deng et al. (2023), bootstrapping has been integrated with a large language model (LLM) for market sentiment analysis using Reddit posts. The authors utilised an LLM to generate weak sentiment labels for financial Reddit posts. Then, the data was utilised to train a small student model for production. The findings showed that the produced classifier performed on par with the existing supervised model even with minimal manually annotated data.

2.12 Use of Oversampling for Imbalanced Datasets

Oversampling is one of the techniques used to handle imbalanced datasets to improve sentiment classification performance. The aim of oversampling is to increase the sample of minority classes deliberately to balance the dataset. This is because imbalance datasets negatively impact prediction capabilities of classifiers (Mujahid et al., 2024). This leads models to favour the majority class thus lead to skewed accuracy metrics. This metrics do not show true performance on minority classes. An advantage of oversampling is there is no loss of original information from the initial dataset because all samples from majority and minority classes are kept in the resampled dataset (Mujahid et al., 2024).

There are several oversampling techniques such as SMOTE and ADASYN. Synthetic minority oversampling technique (SMOTE) was introduced by Chawla et al. (2002) by generating fresh synthetic samples for minority class using K-Nearest Neighbor (KNN) technique. The oversampling process for minority class in this technique is to generate synthetic samples along the line segments joining any or all the k minority class nearest neighbours. Synthetic samples are produced multiplying the difference between feature vector or sample under consideration and its nearest neighbour by a random value between 0 to 1. The result is added to the feature vector under consideration. This results in a random point along the line segments between two features, which generalize the decision region for the minority class (Chawla et al., 2002).

There are also some variations of SMOTE that have been introduced to improve the original SMOTE approach. One of the variations is Borderline-SMOTE which is introduced by Han et al. (2005). The focus of this technique is to generate synthetic samples for minority classes from borderline examples. Borderline examples are samples of minority class that are located near the decision boundary between minority and majority classes. This is because these examples more likely to be misclassified compared to examples that are further away from the decision boundary. The paper proposed two types of Borderline-SMOTE which are borderline-SMOTE1 and borderline-SMOTE2 (Han et al., 2005).

Borderline-SMOTE identifies borderline examples by examining the neighbourhood of each minority class sample. It considers a minority sample to be "borderline" if most of its nearest neighbours are from the majority class. In Borderline-SMOTE 1, synthetic samples are generated only from borderline examples of the minority class. This is to improve the representation of important samples near the decision boundary (Han et al.,

2005). Meanwhile, Borderline-SMOTE2 also generates synthetic samples from borderline examples but, also from their nearest positive minority neighbours and nearest negative majority neighbours. This is to create samples that closer to minority class but also still considering the effect of the majority class (Han et al., 2005).

The next oversampling technique is Adaptive Synthetic Oversampling (ADASYN). Introduced by He et al. (2008), the technique uses samples density distribution as a basis to automatically decide the number of synthetic samples that need to be created from each minority class samples. In this technique, the algorithm adaptively creates synthetic samples based on learning difficulty for different minority class examples to effectively address class imbalance. More synthetic samples are created for harder-to-learn minority class samples compared to the easier-to-learn minority class samples. This technique able to reduce bias caused by class imbalance and to adaptively shift the decision boundary towards harder-to-learn samples. This technique can be modified for incremental learning and extended for multiple-class imbalance learning issue (He et al., 2008).

Another variation of SMOTE is SVM-SMOTE or borderline-SMOTE SVM (Mujahid et al., 2024). This technique is introduced in the paper by Nguyen et al. (2011). This technique has a foundation in both Borderline-SMOTE and SVM techniques. In this technique, the Support Vector Machine (SVM) classifier is trained on the original dataset to find support vectors to get the decision boundary. Support vectors is used to the borderline area where the minority class samples are located. This area is vital for establishing the decision boundary. Then, k nearest neighbours for each minority class support vector to generate instances or samples. New samples are randomly generated along the lines connecting each minority class support vectors to its nearest neighbours either via

extrapolation or interpolation. This depends on the density of majority class samples around each minority class support vector. When the majority class density is high, new samples are created by interpolating between existing samples. Boundary areas are consolidated in this approach. However, if the majority class samples are sparse, new samples are created by extrapolating towards the majority class area. Boundary areas for the minority class are expanded in this approach. After that, new samples are created in order from the first to the k -th nearest neighbours rather than randomly choose the nearest neighbour (Nguyen et al., 2011).

Another SMOTE variation for oversampling is K-Means SMOTE approach introduced by the Douzas et al. (2018). The technique is based on SMOTE oversampling and k-means clustering. This technique consists of three steps: clustering, filtering, and oversampling. Firstly, the input data is clustered into k groups using the k-means clustering. Naturally occurring groups within data especially on the minority class samples are identified in this step. Then, during the filtering step, clusters with a high proportion of minority class samples are chosen for oversampling. Then, the number of synthetic samples to generate is distributed among the selected clusters, with more samples assigned to clusters that are sparsely populated with minority samples. Lastly, SMOTE is used on each selected cluster to reach ratio targeted for minority and majority samples (Douzas et al., 2018).

The paper by Mujahid et al. (2024) explored performance of several oversampling techniques with two feature extraction methods, TF-IDF and bag of words (BoW) on two highly imbalanced, multiclass datasets containing Twitter data. Oversampling techniques used in this paper are SMOTE, Borderline-SMOTE, K-means SMOTE, SVM-SMOTE, and ADASYN. The findings showed that SMOTE, ADASYN and Borderline-SMOTE achieved

better results than other techniques for most section. The study found that the SVM models performed best in terms of accuracy on SMOTE (99.57%), ADASYN (99.67%) and Borderline-SMOTE (99.59%) with TF-IDF features. (Mujahid et al., 2024).

2.13 Background of Tools

This subsection describes tools, libraries, packages and modules used in the methodology section based on their use. Firstly, the Tweepy library was used in the data collection stage of the framework for building the code-switched Malay-English corpus using social media data. This library was used to access Twitter API to extract data from Twitter (Roesslein, 2023). Then, the CSV module in Python was used to handle CSV files in the data collection stage (Python Software Foundation, 2024). After that, the contractions library was used during the pre-processing stage to expand contractions in the collected tweets (Van Kooten, 2022). NLTK library was also used in the pre-processing stage to remove stop words.

Malaya is one of the selected language identification tools in this research. Malaya is a natural language toolkit library for Bahasa Malaysia that is available for various tasks (Zolkepli, 2018). This research uses the first Malaya's fastText model to identify a tweet's language. Currently, the first version of the model is limited to identifying six languages (categories): Malay, English, Indonesian, Rojak, Manglish, and Other. The model identifies the data's language by providing a probability value to one of the languages mentioned. Another selected language identification tool for this research is Lingua. This tool is a new natural language processing library in Python that provides language detection functionality using rule-based and statistical methods. It can be set to detect specified language(s) supported by the library (Stahl, 2023).

Other popular language identification tools include CLD3, `langid.py`, and `fastText`. Table 2.4 shows the language identification tools comparison between Malaya, Lingua, CLD3, `langid.py`, and `fastText`. CLD3 is a neural network model developed by Google that is used as a language identification tool (Google, 2016). It supports 102 languages, and a few languages have the support of two scripts. Hence the total of the supported languages is 107 languages. The output returns the probability of the detected language and the language code in BCP-47-style.

Another well-known language identification tool is `langid.py` which is a stand-alone and off-the-shelf language identification tool (Lui & Baldwin, 2012). The tool was pre-trained on 97 languages using data from JRC-Acquis, ClueWeb 09, Wikipedia, Reuters RCV2, and Debian i18n. It was trained over a Naïve Bayes classifier with a multinomial event model over a mixture of byte n-grams (Lui & Baldwin, 2012). It returns the log probability of the detected language, but the probability normalisation can be enabled.

Meanwhile, `fastText` is a library for text classification and representation learning that can be used as a language identification tool developed by Facebook AI Research (Joulin et al., 2016a; 2016b). There are two models released for language identification which are the original model and the compressed version. The models were trained with data from Wikipedia, Tatoeba and SETimes and can recognise 176 languages. `FastText` detects language by representing text as a bag of word and character n-gram features, embedding and averaging these embeddings to form the hidden variable, and classifying the result using a linear classifier optimized with hierarchical softmax for efficiency (Joulin et al., 2016a). For language detection, it returns the probability of the detected language.

Table 2.4: Comparative analysis of Malaya, Lingua, CLD3, langid.py and fastText

Aspect	Malaya	Lingua	CLD3	langid.py	fastText
Detection Method	fastText model	Rule-based and statistical methods	Neural network model	Naive Bayes classifier trained on a large corpus	Linear classifier and character n-grams and word embeddings
Language Coverage	Malay, English, Indonesian, Rojak, Manglish, and Other	75 languages supported including Malay and English. Users can also specify detection targets	102 languages supported. Some languages have two supported scripts.	97 languages including Malay and English	176 languages including Malay and English.
Output Format	Probability values for each supported language	Confidence score of specified languages	Probability value of the detected language	Log - probability but probability normalisation can be enabled	Probability value of the detected language
Specialisation	Focused on Malay and other local languages variants	General purpose and multilingual detection	General purpose and multilingual detection	General purpose detection	General purpose detection

This research selected three tools or libraries for sentiment identification: TextBlob, pysentimiento, and VADER. TextBlob is a Python library that processes textual data in several natural language processing tasks (NLP), including sentiment analysis (Loria, 2024). This tool was chosen because it was one of the prominent tools for sentiment identification and was easy to use. The next chosen sentiment analyser in this research was pysentimiento. This toolkit is currently available for several tasks, including sentiment analysis. The pysentimiento toolkit was trained for sentiment analysis in English, Spanish, Italian and Portuguese (Pérez et al., 2021). This library was chosen because it is also trained in English tweets for the sentiment analysis task using the dataset from Rosenthal et al. (2019). The last sentiment analyser is VADER, a rule-based model used for sentiment analysis (Hutto & Gilbert, 2014). The library was chosen because VADER was made to focus on social media texts and is currently widely used for sentiment analysis.

Another Python library used was NumPy to handle any mathematical operations in the methodology (NumPy Developers, 2024). The Scikit-learn module in Python by Pedregosa et al. (2011) was used in the sentiment modelling strategy both in the bootstrapping method and in comparing sentiment modelling between machine learning techniques and deep learning models. The imbalance-learn package was used in oversampling (Lemaître et al., 2017). The Pandas package is used to handle data in this research due to its capability to work with data structures (Pandas, 2024). TensorFlow is the library used for machine learning and artificial intelligence. Meanwhile, Keras is an API that acts as the high-level API for the TensorFlow library focused on machine learning (TensorFlow, 2023). Keras is also available for use in other machine learning frameworks, including PyTorch and Jax (Keras Team, n.d.).

2.14 Identified Gaps in the Literature

Nevertheless, works on sentiment analysis for code-switched Malay-English data showed shows underexplored aspects that motivated the development of a computational framework to construct the MESocSentiment corpus. We are motivated to obtain this corpus because there is still a lack of publicly available sentiment analysis corpus containing social media data in the Malay-English language pair. Plus, the creation of this corpus addresses one of the challenges for sentiment analysis of code-switched texts, which is the lack of annotated data and resources.

From the trends of recent sentiment analysis research, the techniques to be used in benchmarking code-switched sentiment models in Chapter 3 have been chosen. The chosen machine learning techniques are Support Vector Machine, Naive Bayes, and Random Forest. Then, the chosen deep learning models are Bidirectional Encoder Representation from Transformer (BERT), Long Short-Term Memory (LSTM), and Convolution Neural Network (CNN). These techniques and models are chosen to compare the performance of machine learning techniques and deep learning models in sentiment modelling.

Based on the data source of the analysed works, most works use Twitter data for their works with 11 works. This reason makes the chosen data source for this work is Twitter data. From the trends, most of the researched languages are used in Asian countries, mainly in South Asia. However, there is still a lack of research on languages spoken in Southeast Asian countries such as Malay, Bahasa Indonesia, Filipino and Thai, especially research on sentiment analysis code-switched social media data. This becomes a motivation to do research on sentiment analysis in code-switched Malay-English social media data. One aim is to publicly publish a Malay-English code-switched corpus on GitHub or Kaggle.

2.15 Chapter Summary

In this chapter, the background of the research is explained. Firstly, the background of sentiment analysis is explained in terms of definition, advantages, disadvantages, and challenges. Then, the techniques used in sentiment analysis are explained, especially on machine learning techniques and deep learning models. The background of the code-switching situation in the sentiment analysis task is also presented in this chapter. After that, some of the related works related to sentiment analysis in social media and code-switched texts are also discussed in this chapter. Evaluation metrics that are typically used in sentiment analysis works are also discussed in this chapter. Lastly, the research motivation based on the literature review is also stated in this chapter.

CHAPTER 3

METHODOLOGY

3.1 Overview

The methodology used in this research was explained in this chapter. The tools used in the methodology were described in Section 3.2. Section 3.3 described the overview of the overall research methodology for this study. Section 3.4 described the framework for constructing the code-switched Malay-English corpus for this research, which was named MESocSentiment. In Section 3.5, the strategy for sentiment modelling was explained. The strategy consisted of a bootstrapping technique followed by a comparison of chosen machine learning techniques and deep learning models. Lastly, Section 3.6 provided a summary of the methodology used in this research.

3.2 Tools Used in Experiments

The programming language used in the codes was Python. Most codes were run in Google Colaboratory (Google Colab), excluding data collection. The Python code was edited using Sublime Text 3 and run in the command prompt for data collection. After that, Table 3.1 shows the libraries, modules, and packages used in the research and their uses.

There were also several other Python libraries, modules, and packages used in this research. The pandas package was used from the data preprocessing stage until the final stage of the sentiment modelling strategy. The csv library was used to handle csv files in the data collection stage. The contractions library was used in expansion of English contractions in data preprocessing stage (Van Kooten, 2022). Another Python library used was NumPy, which was used for any mathematical operations in the methodology (NumPy Developers, 2024).

Table 3.1: Libraries, Packages and Modules Used in the Research

Library/Package/Module	Uses
Pandas	Data handling
csv	Handling csv files
Numpy	Mathematical operations
Tweepy	Used in data collection
Twitter API	Used in data collection
contractions	Expansion of English contractions
Malaya	Used in language identification
Lingua	Used in language identification
VADER	Used in sentiment identification
TextBlob	Used in sentiment identification
pysentimiento	Used in sentiment identification
Scikit-learn	Used for bootstrapping, and comparison of models results
Imbalance-learn	Used for oversampling
TensorFlow	Used for LSTM, CNN, and BERT
Keras	Used for LSTM, CNN, and BERT
Joblib	Used to save SVM, Naïve Bayes, and Random Forest models

The Scikit-learn module in Python by Pedregosa et al. (2011) was used in sentiment modelling strategy both in bootstrapping technique and comparison of sentiment modelling between machine learning techniques and deep learning models. It was used for Support Vector Machine, Naive Bayes, and Random Forest. In the sentiment modelling strategy, TensorFlow library and Keras API were used in building, fine-tuning, training, and testing

chosen deep learning models which were LSTM, CNN, and BERT. Meanwhile, the Joblib tool was used to save SVM, Naïve Bayes, and Random Forest models after training (Joblib Developers, 2021).

3.3 Research Methodology Overview

In this research, the methodology was divided into two parts which are the framework of construction of Malay-English sentiment corpus and the supervised benchmark of code-switched sentiment models based on chosen classical machine learning algorithms and deep learning models. Table 3.2 shows the mapping table that included research problems along with related research objectives to address the problems. Summaries of corresponding contributions in this research to each research objective are also shown in the table. Thus, the research methodology proposed in this study was based on the research objectives of this study. Figure 3.1 shows the overview of the research methodology proposed and utilised in this research.

Table 3.2: Mapping table of research problems, objectives and contributions

Research Problem	Research Objective	Contribution
Lack of publicly available code-switched sentiment corpus especially on Malay-English social media data.	To design a framework for building a Malay-English code-switched corpus from Twitter data using language identifiers and existing sentiment analysers.	A framework to construct a Malay-English code-switched sentiment corpus using language identifiers and existing sentiment analysers. The constructed code-switched corpus was made publicly available.
There is a need to utilise a quicker approach for sentiment modelling for benchmarking sentiment models trained on code-switched data.	To develop a supervised machine learning approach that includes a bootstrapping technique for benchmarking sentiment models using the constructed Malay-English code-switched corpus.	A bootstrapping technique is applied in a supervised machine learning approach to benchmark sentiment models, which are based on several classical machine learning algorithms and deep learning models, using Malay-English code-switched data from the constructed corpus.

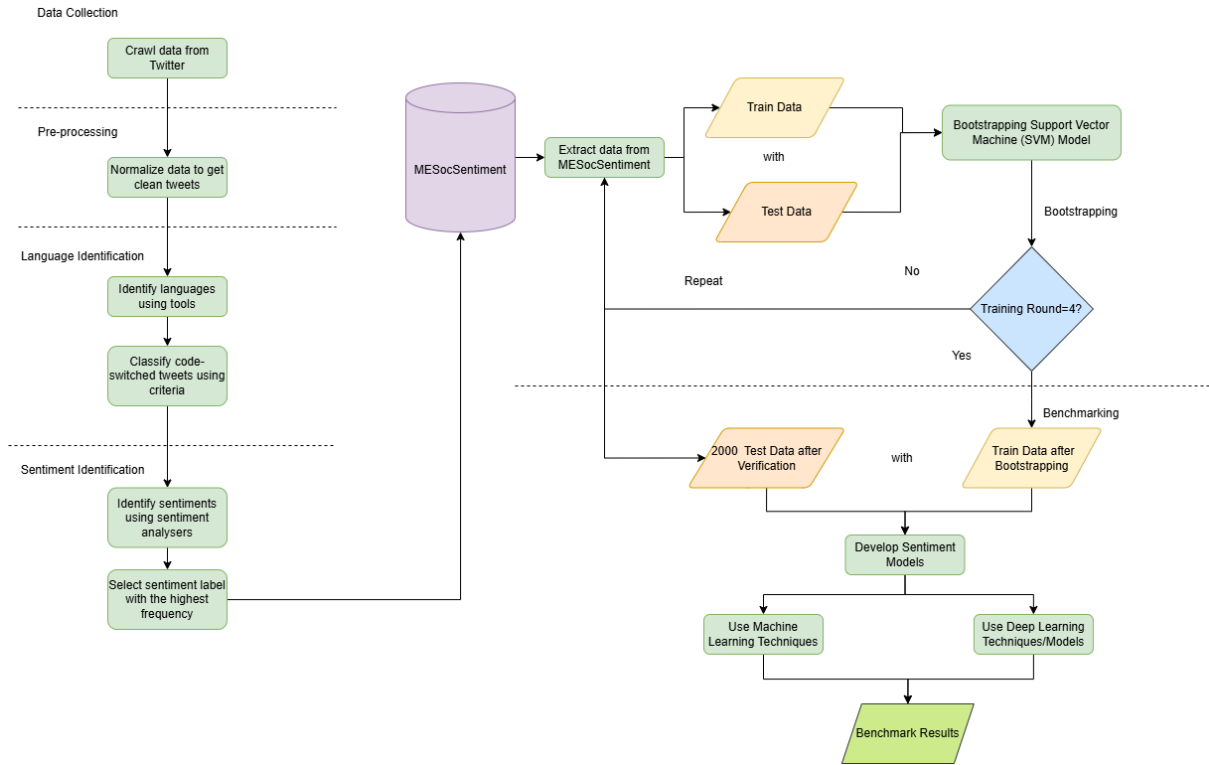


Figure 3.1: Overview of the research methodology

Based on Figure 3.1, the diagram can be divided into two sections. The left side of the diagram shows the general steps used in the construction framework for Malay-English code-switched sentiment corpus from social media data. The first stage of this framework was data collection where the data were collected from Twitter. The next stage was data pre-processing where the collected data were normalised to get clean tweets. After that, as the first step in the language identification stage, language identification tools were used to identify languages in the clean tweets. After the identification, code-switched tweets were classified using certain criteria. Next, as the first step of the sentiment identification stage, sentiments of code-switched tweets were identified using existing sentiment analysers. After the results from all utilised sentiment analysers were compiled, the final label of each code-switched tweet was identified. The labelling for each code-switched tweet was done by selecting the sentiment label with the highest frequency from the results as the final

sentiment label. The final version of the code-switched dataset was named as the MESocSentiment corpus.

The MESocSentiment corpus constructed in the first framework played a key role in the next part of the methodology: a supervised machine learning approach that includes a bootstrapping technique for benchmarking sentiment models. The first stage of this approach was the use of bootstrapping technique. Train data and test data were extracted from MESocSentiment to be used in the training round with a bootstrapping SVM model. The training rounds were repeated until the fourth training round. The next stage of this approach was the benchmarking of sentiment models. In this stage, train data after bootstrapping and another 2000 test data after verification were utilised in developing several sentiment models. Sentiment models used in benchmarking were from several traditional machine learning techniques and deep learning models. The results from all code-switched sentiment models were benchmarked to assess the models' performance.

3.4 MESocSentiment Corpus Construction Framework

The framework for constructing the Malay-English code-switched corpus for sentiment analysis from social media data named the MESocSentiment corpus using a semi-automatic approach is shown in Figure 3.2. The framework consisted of four stages which were data collection, pre-processing, language identification, and sentiment identification. This framework was called a semi-automatic approach due to some human intervention in some stages, specifically in data collection and language identification stages as shown with brown human icons in Figure 3.2. The semi-automatic approach has been applied in prior sentiment analysis studies such as in Shirazi et al. (2023) the authors proposed a semi-automatic approach for labelled data generation for sentiment analysis of the Persian language. Human intervention in this study was during correcting false labels for the data

labelled by a zero-shot learning-based transformer. The following subsections provided comprehensive descriptions of each stage as they were executed throughout the research.

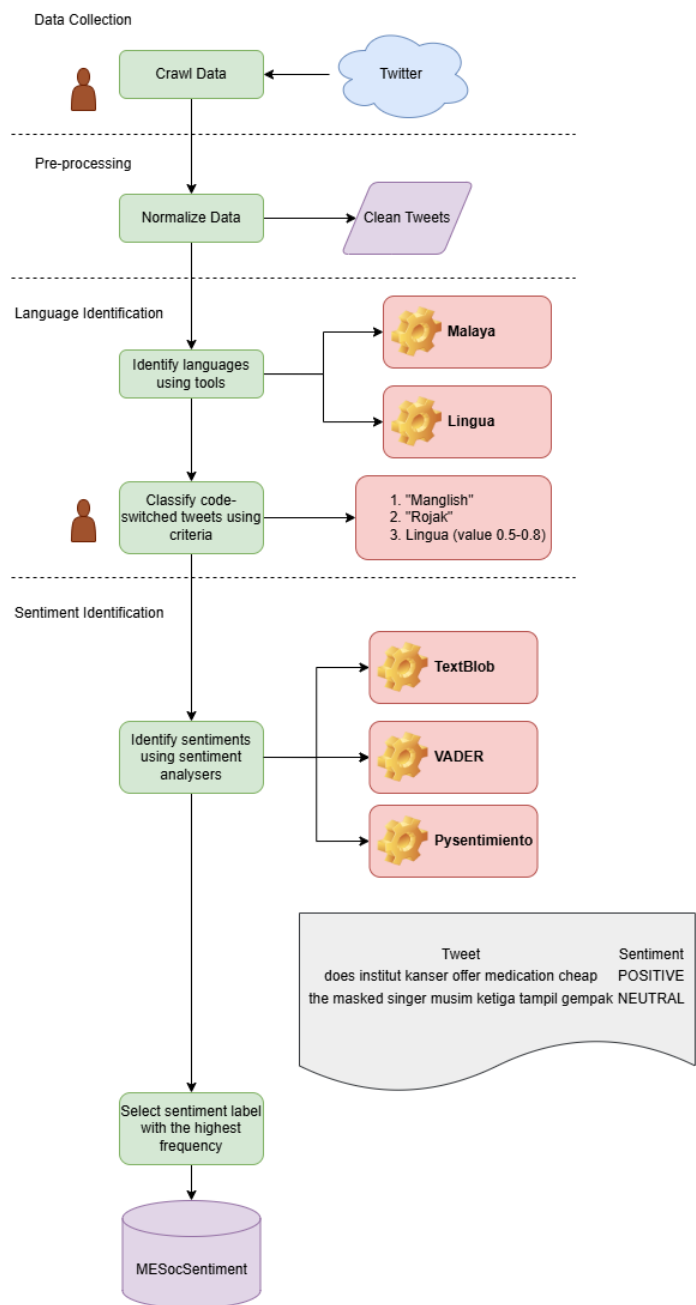


Figure 3.2: The MESocSentiment corpus construction framework using a semi-automatic approach

3.4.1 Data Collection

Data collection was the first stage in the framework, as shown in Figure 3.2. In this research, Twitter was used to demonstrate the framework. During this stage, data were collected from Twitter between 12 August 2022 and 15 May 2023. The Tweepy library and Twitter API were employed to crawl the data, which were collected weekly.

Firstly, several variables were used to store credentials for authentication with Twitter API. The credentials were consumer key, consumer secret, access token, access token secret, and bearer token. Then, the Tweepy library was used to create a client object. The bearer token was needed when creating the client object. The search query was defined to retrieve tweets with the keyword #Malaysia excluding retweets. The strategy was similar to the one used by Zabha et al. (2019). This specific keyword was chosen to solely accumulate tweets originating from Malaysia.

The length of the DataFrame and the names of the columns were printed for checking. When the Tweepy library was used for extracting tweets, it returned 12 columns, that contained timestamps, tweets, and unnamed columns containing additional context information for the tweets. The brown human icon in Figure 3.2 indicates human intervention. Hence, as shown in the figure's data collection stage, the weekly output CSV files were manually compiled into a huge CSV file that contained all collected weekly data. The compiled raw data was pre-processed in the next stage. In this stage, the number of collected tweets by using the keyword #Malaysia was 229, 566 tweets.

3.4.2 Pre-processing

Based on Figure 3.2, the data collected from the previous stage underwent data preprocessing. The step was taken because the newly acquired social media data required

normalization to get clean tweets before proceeding to the next stage, which was language identification. For cleaning tweets or social media posts, only the 'Tweets' column was pre-processed, and the other columns were dropped. The reason was the focus of the data preprocessing stage was to produce clean tweets from the collected raw tweets. Then, tags and other invalid characters were removed followed by converting texts into lowercase. Hashtags, mentions, URLs, numbers, returns, newline, and tabs along with underscores were also removed in the process.

The next pre-processing step involved expanding English contractions using the Python-based contractions library to standardize English words in the collected tweets (Van Kooten, 2022). The keyword used for data collection was #Malaysia, which is the name of the country and can appear in tweets written in various languages, particularly Malay and English. Therefore, it was assumed that tweets containing this keyword were likely to include English words. Although language identification was performed in the next stage of the framework, this step was justified as the contractions library only expands known English contractions, minimizing the likelihood of affecting non-English words in the data.

Stop words were also removed from the data in this stage. The stop word lists used were from NLTK English stop words list and Malay stop words list from Stopword Malay (MS) (Bird et al., 2009; Stopwords-ISO., 2016). There was no custom stop word list used in this study due to limited linguistic expertise to create a comprehensive stop word list. The Stopword Malay list was used in this study, although it was created nearly a decade ago, and could be considered a formal list via observation because it was the most comprehensive Malay stop word list that was readily available. There were prior studies on Malay stop words such as the studies by Chekima and Alfred (2016) as well as Abdul Rahman et al.

(2021). However, stop word lists created in these studies were not either in the social media domain or available for public use.

Punctuations and non-alphanumeric sentences were also removed from the data. Next, excess whitespace was also reduced to make the tweets look cleaner. As shown in Appendix C, the tokenisation process in this study adopted a rule-based approach using Python's `.split(function)`, which segmented texts based on whitespace. It was supported by regular expressions (regex) rules for data cleaning. No model-based tokeniser was used in this study. After that, sentences with only one word or token were removed because the code-switching situation could not occur in that type of sentence. Lastly, duplicated sentences and empty rows were dropped to make the dataset cleaner.

At the end of the data preprocessing stage, the number of cleaned tweets was 138,646 tweets. The cleaned tweets then underwent the language identification stage using Malaya and Lingua libraries. Table 3.3 shows samples of raw tweets along with their post-processed tweets. Firstly, the post-processed Tweet 1 showed the removal of tags, invalid characters, hashtags, a URL, punctuations, and a Malay stop word as well as changing all letters to lowercase to raw Tweet 1. Next, the raw Tweet 2 was pre-processed with the removal of tags, newlines, some Malay stop words, hashtags, punctuations, and a URL along with changing all letters into lowercase. Lastly, the raw Tweet 3 was pre-processed with the removal of tags, punctuations, hashtags, a URL, and some English stop words along with changing all letters into lowercase.

Table 3.3: Samples of raw tweets and clean tweets after pre-processing

No.	Raw Tweet	Clean Tweet
1	b'Does Institut Kanser Negara offer FREE medication + CHEAP chemotherapy?! \xf0\x9f\xa4\x94\nhttps://t.co/oWiG0fuf62\n T echARP #Malaysia #KKM #Hospital #KKMMalaysia #Facts #FactsMatter #FactCheck #Hoax #Fake #FakeNews #Health #Healthcare #Medicine #Chemotherapy #Cancer'	does institut kanser offer free medication cheap chemotherapy
2	b'The Masked Singer Malaysia musim ketiga tampil dengan peserta yang lebih gempak. #khaarlifstar #themaskedsingermalaysia #artisindonesia #artismalaysia #viral #beritaterkini #motifviral #malaysia #makcikbawang #trending #news https://t.co/dewurWFsI9	the masked singer musim ketiga tampil gempak
3	b'Authentic Malay cuisine at Kampung Melayu Subang. Delicious masak lemak, smoke duck, fried cat fish, ulam and more... #foodie #Malaysia https://t.co/oDMI87qirg	authentic malay cuisine kampung melayu subang delicious masak lemak smoke duck fried cat fish ulam more

3.4.3 Language Identification

From Figure 3.2, clean tweets underwent the language identification stage. This stage aimed to identify the type of language used in the tweets. Language identification tools were utilized in this stage because the number of tweets collected for building the code-switched corpus was large. Hence, it would be costly to classify whether each tweet was in Malay, English, or code-switched between the two languages manually. The selected language

identification tools were Malaya and Lingua libraries because both tools supported Malay and English language identification (Stahl, 2023; Zolkepli, 2018).

Malaya and Lingua were selected over other language identification tools mentioned in Section 2.13 which were CLD3, langid.py, and original fastText models for several reasons. CLD3 was not chosen because the tool was written in C++ meanwhile, this study used Python as the programming language. Even though there was Python binding for CLD3, the tool did not run natively on Python. The langid.py was not chosen because it only returned the most dominant language and its probability, making it unsuitable enough for identifying code-switched texts. Lastly, original fastText models were not chosen in this stage because both models only returned one detected language and its probability, similar to langid.py. Moreover, Malaya's fastText model was better for detecting Malay-English code-switched texts due to its hyper-localisation for Malaysia. It meant that the model returned probabilities for all its supported languages which include Malay, English, and other regional languages and dialects.

The first part of this stage identified tweets' languages using Malaya's fastText model and Lingua separately. The Malaya model assigned each tweet a probability value to one of the supported languages (categories) which were Malay, English, Indonesian, Rojak, Manglish, and Other. The output was saved into a CSV file. For this research, the Lingua model was set to detect two languages in this experiment, which are Malay and English. This model generated two confidence values, one for Malay and one for English, resulting in a total of 1.0. The output of this model was saved into a CSV file.

Language identification stage results for both Malaya and Lingua libraries are shown in this section. Malaya's fastText model was used in the language identification stage. For this model, the result only showed the probability of detected languages to only one of the

supported languages. The sample of tweets after applying the Malaya tool in the identification task is shown in Table 3.4.

Table 3.4: Sample of Language Identification Results Using Malaya

No.	Tweets	Malay	English	Bahasa Indonesia	Manglish	Rojak
1.	pusat hiasan ikan near teluk kemang nice small aquarium lots colorful fish free get tickets popular things klook	0.534	0.0	0.0	0.0	0.0
2.	does institut kanser offer free medication cheap chemotherapy	0.0	0.0	0.0	0.596	0.0
3.	gurkhali machas bars rap titled suara rakyat	0.0	0.785	0.0	0.0	0.0
4.	jangan lupa ubah akta perumahan perolehi rumah dibina menghuni kaya buat untungan impose empty house tax like canada	0.0	0.0	0.774	0.0	0.0
5.	the masked singer musim ketiga tampil gempak	0.0	0.0	0.0	0.0	0.849

In Table 3.4, all sample tweets were code-switched Malay-English tweets but were classified into different categories by Malaya. Tweet 1 was categorized as a Malay tweet, but it consisted of Malay and English words. Then, Tweet 2 was categorized as a Manglish tweet. The structure of the tweet was in English, but the phrase "institut kanser" was used rather than "cancer institute." Meanwhile, Tweet 3 was labeled as an English tweet although

there were two Malay words in the sentence. Then, Tweet 4 was labeled as a Bahasa Indonesia or Indonesian tweet by the model. Lastly, Tweet 5 was categorized as a Rojak tweet. The sentence started with the first three words in English, and the following words were in Malay.

On the other hand, each tweet identified by Lingua received two confidence scores: one for Malay and another for English. Because there were two specified languages (Malay and English) the confidence score for each language was between 0 to 0.5. For displaying the results, confidence scores were rounded down to three decimal places for ease of analysis, and the tweets were grouped into six groups based on the confidence scores obtained, specifically for the dominant language of each tweet. The groups were 1.0 group, 0.9 group, 0.8 group, 0.7 group, 0.6 group, and 0.5 group. The confidence score ranges for each group are shown in Table 3.5.

Table 3.5: Confidence Scores Groups and Respective Ranges

Confidence Score Groups	Confidence Score Range
0.5	0.500-0.599
0.6	0.600-0.699
0.7	0.700-0.799
0.8	0.800-0.899
0.9	0.900-0.999
1.0	1.0

Some samples from language identification using Lingua are shown in Table 3.6. In this table, all samples were code-switched Malay-English sentences with different confidence scores for both languages. For example, Tweet 7 was in the 1.0 group with a

confidence value of 1.0 for the Malay language. However, it contained one English word and 18 Malay words. Additionally, Tweets 1, 2, and 8 were the same as the samples in Table 3.4, and all of them were in the 0.9 group. Tweet 1 had a confidence value of 0.945 for the English language. However, it was classified as a Malay tweet by Malaya. For Tweet 2, its higher confidence value was 0.994 for English, but it was also classified as a Manglish tweet by Malaya. Tweet 8 had a Malay confidence value of 0.908 and was classified as a Rojak tweet by Malaya. Next, the example of the 0.8 group was Tweet 4, with a confidence value of 0.831 for English. It contained 10 English and 4 Malay words. Meanwhile, Tweet 3 was in the 0.7 group because it had a confidence value of 0.730 for Malay. Tweet 5 was in the 0.6 group because it had a confidence value of 0.686 for English. The tweet's structure was two English words followed by the Malay word 'merdeka'. The word 'merdeka' was repeated three times consecutively. Lastly, Tweet 6 was in the 0.5 group because it had a confidence value of 0.571 for English. In this tweet, the Malay words referred to a place and cuisines.

Table 3.6: Sample of Language Identification for Lingua

No.	Tweets	English	Malay
1.	pusat hiasan ikan near teluk kemang nice small aquarium lots colorful fish free get tickets popular things klook	0.945	0.054
2.	does institut kanser offer free medication cheap chemotherapy	0.994	0.05
3.	gurkhali machas bars rap titled suara rakyat	0.269	0.730
4.	start new life circle baru vibe baru let flow meski kadang sad fell lonely	0.831	0.168
5.	happy independence merdeka merdeka merdeka	0.686	0.313

Table 3.6: continued

6.	authentic malay cuisine kampung melayu subang delicious masak lemak smoke duck fried cat fish ulam more	0.571	0.428
7.	inisiatif bagus tapi sepatutnya declare tegas saman tamat diskaun jangan ajar tunggu diskaun selesaikan saman hormat undang melindungi nyawa	0	1.0
8.	the masked singer musim ketiga tampil gempak	0.091	0.908

In the second part of this stage, code-switched tweets were selected using the outputs from both language tools. As shown in the first part of this stage, this study used Malaya and Lingua, which identified languages on the sentence level by giving probabilities and confidence scores of detected languages of a sentence. These outputs were utilised for the selection of code-switched tweets. Hence, no segmentation technique was applied to further separate or identify code-switched segments within a sentence. The brown human icons in Figure 3.2 indicated human intervention in the language identification stage, specifically in the classification of code-switched tweets based on results from Malaya and Lingua. The criteria for the selection of code-switched tweets were determined based on the manual observations of tweets' conditions and results.

Based on the observation, for Malaya outputs, tweets labelled as either 'Rojak' or 'Manglish' were considered code-switched tweets. These tweets were chosen because there was a higher probability of Malay-English code-switching under these two categories. Rojak is a famous dialect in Malaysia that consists of Malay, English, and Chinese words (Romadhona et al., 2022). Additionally, Manglish is a colloquial form of English widely spoken in Malaysia that adopts linguistic expressions from ethnic languages in Malaysia including Malay (Serip Mohamad & Shafie, 2023).

Meanwhile, for Lingua outputs, tweets with confidence values ranging from 0.5 to 0.8 in either Malay or English were considered code-switched tweets. Specifically, since Lingua generated output with a lot of decimal places, hence the confident score value for the dominant language rounded down to 3 d.p. for ease of analysis. Thus, the chosen Malay-English code-switched tweets were the tweets with the dominant languages' confidence score value of 0.500 to 0.799. These tweets fell under the 0.5, 0.6, and 0.7 groups based on Table 3.5. These tweets were chosen because based on the manual observation, there was a higher presence of Malay-English code-switching within this range of confidence values.

Although there were prior studies about finding or handling the confidence threshold as well as finding code-switched or code-mixed points in the code-switched texts, none of the studies utilised the Lingua tool in their studies. Some examples of the studies were the language threshold mechanism for multilingual sentiment analysis system by Kit and Mokji (2024), and the multi-step natural language processing (NLP) algorithm that identifies points of code-switching by Aryal et al. (2022).

Hence, the threshold value has been defined in this study based on the manual observations of Lingua's output. The criteria focused on Lingua output was the confidence score of the dominant language in the tweet, meaning that the range of confidence score was 0.5 to 1.0. Table 3.7 shows the confidence score ranges and primary classification of each range. From the table, based on the manual observation, many code-switched tweets fell within the 0.500 – 0.799 (rounded up to 0.8) dominant language confidence range, where the presence of both Malay and English elements caused the model to assign a lower confidence score to a single language. In contrast, monolingual tweets were predominantly observed in the 0.800-1.000 confidence range, where the clear dominance of a single language results in higher confidence scores assigned by the classifier.

However, several distinct exceptions were observed in the results. Within the confidence score range of 0.800 and above, some noise, and Malay-English code-switched tweets were still present. This may be due to minimal language mixing that did not significantly impact the dominant language's confidence score. Meanwhile, within the confidence score range of 0.500 to 0.799 (rounded to 0.8), some noise and monolingual tweets were present. This may be attributed to the limitations of the tool. Hence, while the confidence score range 0.50 – 0.80 threshold did not perfectly distinguish all Malay-English code-switched tweets from monolingual tweets, it provided a practical operational boundary that captured most code-switched tweets with acceptable precisions.

Table 3.7: Dominant Language Confidence Score Groups and Code-Switching Classification for Lingua

Confidence Score Range	Primary Classification	Observation
0.500 – 0.799	Code-switched (likely)	Contained many code-switched tweets, but also some monolingual tweets and noise.
≥ 0.800	Monolingual (likely)	Mostly monolingual, but there was some presence of code-switched tweets and noise.

In this stage, Malay-English code-switched tweets were selected based on the selected criteria. However, due to the limitation of tools used, certain code-switched tweets were misclassified as monolingual code-switched tweets and vice versa by the tools. These misclassified instances were not manually corrected or reclassified to not compromise the current condition of tweets. Instead, these tweets were only filtered out from the dataset if they fell outside the selected criteria, as in the objective and intended workflow of this stage.

Code-switched tweets from the outputs of both models were combined into a new CSV file. After removing any duplicates, the remaining code-switched tweets were used in the next stage of the framework, which was sentiment identification. From the results of both language identification tools, tweets under ‘Rojak’ and ‘Manglish’ for Malaya and tweets with confidence scores of 0.5 to 0.8 (specifically 0.5 until 0.799) for Lingua were determined as code-switched data. These tweets were combined, and after removing any duplicates, this dataset contained 20, 226 tweets.

3.4.4 Sentiment Identification

Based on Figure 3.2, chosen code-switched tweets chosen at the end of the last stage were utilised in this stage. The number of code-switched tweets from the previous stage were 20, 226 tweets. The purpose of sentiment identification stage was to get sentiment for each code-switched tweet to build the Malay-English code-switched corpus from social media data (MESocSentiment). Figure 3.3 shows the detailed workflow of the sentiment identification stage.

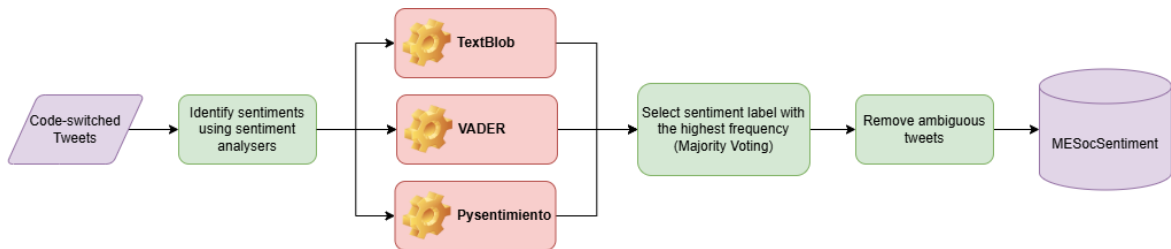


Figure 3.3: The workflow of the sentiment identification stage

In the first part of sentiment identification, sentiments of the tweets were identified using three existing sentiment analysers. Sentiment analysers used in this part acted like human annotators which were leveraged to give sentiment labels for code-switched tweets identified in the previous stage. The chosen sentiment analysers were TextBlob, pysentimiento, and VADER. Each sentiment analyser labelled each tweet with either

positive, negative, or neutral sentiment. The sentiment identification output from each sentiment analyser was saved into their own CSV files.

The second part of this stage was the implementation of a voting approach, specifically majority voting. The goal of this approach was to get the final sentiment labels for each code-switched tweet to build the MESocSentiment. In this approach, generated sentiment labels for code-switched tweets from all sentiment analysers were combined for review. The generated results were compared to get the final sentiment output for the tweets. The sentiment label with the highest frequency was selected as the final sentiment output for each tweet. The voting approach gave an early level of confidence in the final labels of the dataset.

However, ambiguous results were expected because of the number of sentiment analysers used in this stage. The ambiguous case occurred when all sentiment analysers gave different sentiment labels for a tweet, resulting in no definite sentiment label for the tweet. Tweets that fell under this category were removed from the dataset. The final dataset was named the MESocSentiment corpus. The statistics of the voting approach is shown in Table 3.8. The table contained four categories: NEUTRAL, POSITIVE, and NEGATIVE. There was also the presence of an ambiguous category in the result, which was POSITIVE/NEGATIVE/NEUTRAL. Each tweet fell under one of these categories.

Table 3.8: The Result of the Sentiment Identification Process Using Voting Approach

Sentiment	Number of Tweets	Percentage (%)
NEUTRAL	15422	76.23
POSITIVE	3219	15.92
NEGATIVE	1073	5.31
POSITIVE/NEGATIVE/NEUTRAL	512	2.53

From the result, most tweets were labelled neutral, followed by positive and negative sentiments. However, there was also a minor presence of ambiguous tweets, accounting for 2.53% of the dataset. The ambiguous case happened when each sentiment analyser gave different sentiment labels for a particular tweet. These ambiguous tweets were removed to ensure that the corpus only contained tweets from absolute categories, which are positive, negative, and neutral.

Table 3.9: Sentiment Labels on Random Tweets from MESocSentiment Corpus

No.	Tweets	Sentiment (All)
1.	does institut kanser offer free medication cheap chemotherapy	POSITIVE
2.	telur supply tergugat ikan bilis too travesty injustice done holy dish nasi lemak	NEGATIVE
3.	the masked singer musim ketiga tampil gempak	NEUTRAL

Table 3.9 shows a selection of tweets from the MESocSentiment corpus, which consists of two columns: 'Tweets' and 'Sentiment (All)'. The 'Sentiment (All)' column displays the final sentiment label assigned to each tweet, which is determined using a voting approach that utilises the outputs of three existing sentiment analysis analysers:

pysentimiento, TextBlob, and VADER. As shown in the table, Tweet 1 was labelled as POSITIVE. Meanwhile, Tweet 2 and Tweet 3 were labelled as NEGATIVE and NEUTRAL, respectively.

Some samples of true tweets from the MESocSentiment are shown in Table 3.10. True tweets had the same result from the final sentiment label via the voting approach and from a manual check. To illustrate, Tweet 1 was considered a negative tweet because there was anger in the tweet where the tweet was about a useless oath made by a group due to bribery. Meanwhile, Tweet 2 was an example of a positive tweet because it showed happiness in the sentence. The tweet was about wishing a happy Independence Day to a neighbouring country. Lastly, Tweet 3 was an example of a neutral tweet because the tweet was a statement about a group of leaders entering the palace to meet the king.

Table 3.10: Samples of True Tweets

No.	Tweets	Sentiment (All)	Sentiment (True)
1.	their called sumpah worthless dna makan suap rasuah	NEGATIVE	NEGATIVE
2.	happy birthday neighbour across causeway selamat merdeka	POSITIVE	POSITIVE
3.	gabungan leaders enter istana audience agong	NEUTRAL	NEUTRAL

Meantime, there were also some false tweets found in the MESocSentiment. False tweets were tweets that had false sentiment labels given through the voting approach when manually checked. Some samples of false tweets from the MESocSentiment are shown in Table 3.11. For instance, Tweet 1 should be POSITIVE instead of NEUTRAL because it

was about a beautiful village that had become a tourist destination for photography and camping. Tweet 2 was also the same as Tweet 1 because the tweet was about wishing followers of the account happy Deepavali. Lastly, Tweet 3 was a positive tweet instead of a neutral tweet because the context was about an athlete who has become a champion in a championship.

Table 3.11: Samples of False Tweets

No.	Tweets	Sentiment (All)	Sentiment (True)
1.	antara tarikan penang kampung ni memang cantik sangat sampai tarikan bergambar camping kampung agong campsite penaga pulau pinang sumber kampung agong	NEUTRAL	POSITIVE
2.	selamat deepavali buat followers bikers ranger	NEUTRAL	POSITIVE
3.	syabda raih gelar juara international series	NEUTRAL	POSITIVE

3.5 A Supervised Machine Learning Approach using MESocSentiment Corpus

The MESocSentiment corpus obtained after completing the stages in the previous framework was utilised in this sentiment modelling strategy. Although the previous approach helped in building the corpus with sentiment labels, it was acknowledged that there existed false labels when the voting approach was used. This was due to the limitations of the chosen sentiment analysers to provide true labels. However, these labels were assumed to be the starting point to get the ground truths.

Hence, a supervised machine learning approach that included a bootstrapping technique was developed for benchmarking sentiment models using the MESocSentiment corpus. The approach was structured into two consecutive stages to address which were done

consecutively, to address a need for a quicker approach for sentiment modelling, specifically in building training data for sentiment models used for benchmarking. The first stage involved a bootstrapping technique which was used to build the training data for sentiment models using data from the MESocSentiment corpus. This step was crucial because the manual annotation process was challenging and time-consuming, especially with code-switched texts.

The second stage was to train six sentiment models from several machine learning techniques and deep learning models with train data from bootstrapping techniques and benchmark the results of the code-switched sentiment models for performance analysis. This separation into two stages was necessary to ensure that the sentiment models for benchmarking could be trained with a dataset with reliable sentiment labels, given that the original data source (MESocSentiment) contained inaccurate labels due to the tools used in the sentiment corpus construction framework. Figure 3.4 shows the flow of the proposed supervised machine learning approach.

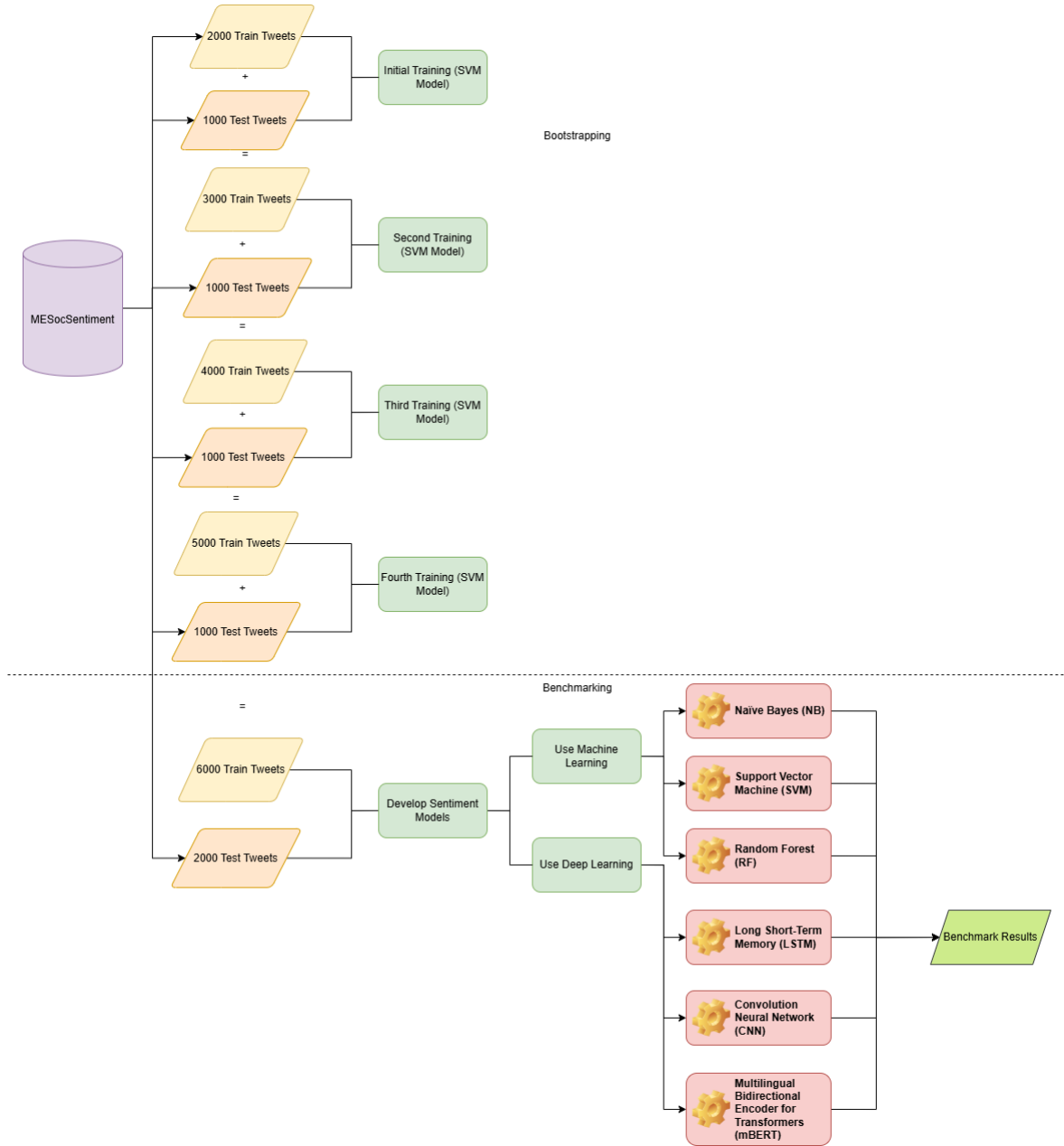


Figure 3.4: A supervised machine learning approach with bootstrapping technique for benchmarking sentiment models using MESocSentiment

3.5.1 Evaluation Method

The evaluation or performance metrics were used to determine the performance of the model. The first evaluation metric used in this sentiment modelling strategy was accuracy. Accuracy was chosen to show the overall correctness of the model's results by

measuring the percentage of tweets that were classified correctly in relation to the total number of tweets analysed (FasterCapital, 2024).

Another evaluation metric was ROC (Receiver Operating Characteristic) and AUC (Area Under the Curve) or also known as AUC-ROC. ROC curve was an evaluation metric used in classification problems. AUC-ROC visualised the True Positive Rate and False Positive Rate at different thresholds through the graph. These metrics showed the ability of the model to discriminate classes. Because this research dealt with multiclass classification, the One-vs-Rest approach was used for these evaluation metrics. Hence, two metrics that were focused on for AUC-ROC were macro-average and micro-average.

3.5.2 Bootstrapping Technique for Building Training Data for Benchmarking Sentiment Models Using Support Vector Machine Model

From the previous framework, sentiment labels for the MESocSentiment corpus were decided through a voting approach using existing sentiment analysers. Yet, the generated sentiment labels might not have been correct due to the limitations of existing sentiment analysers used in the previous framework. However, manual checking could have been time-consuming. Hence, a bootstrapping technique was used to solve this issue. The bootstrapping technique was adapted from the study by Juan (2015), where it was used to build training data using the initial grapheme-to-phoneme labels on the Iban pronunciation dictionary. Because the bootstrapping technique in Juan et al. (2015) was for the speech recognition task, this technique has been adapted in this study for the sentiment analysis task by using data from the MESocSentiment corpus. The objective of the bootstrapping technique was to build training data with reliable sentiment labels for benchmarking

sentiment models. Figure 3.5 shows the diagram for the bootstrapping technique used in the proposed supervised machine learning approach in this study.

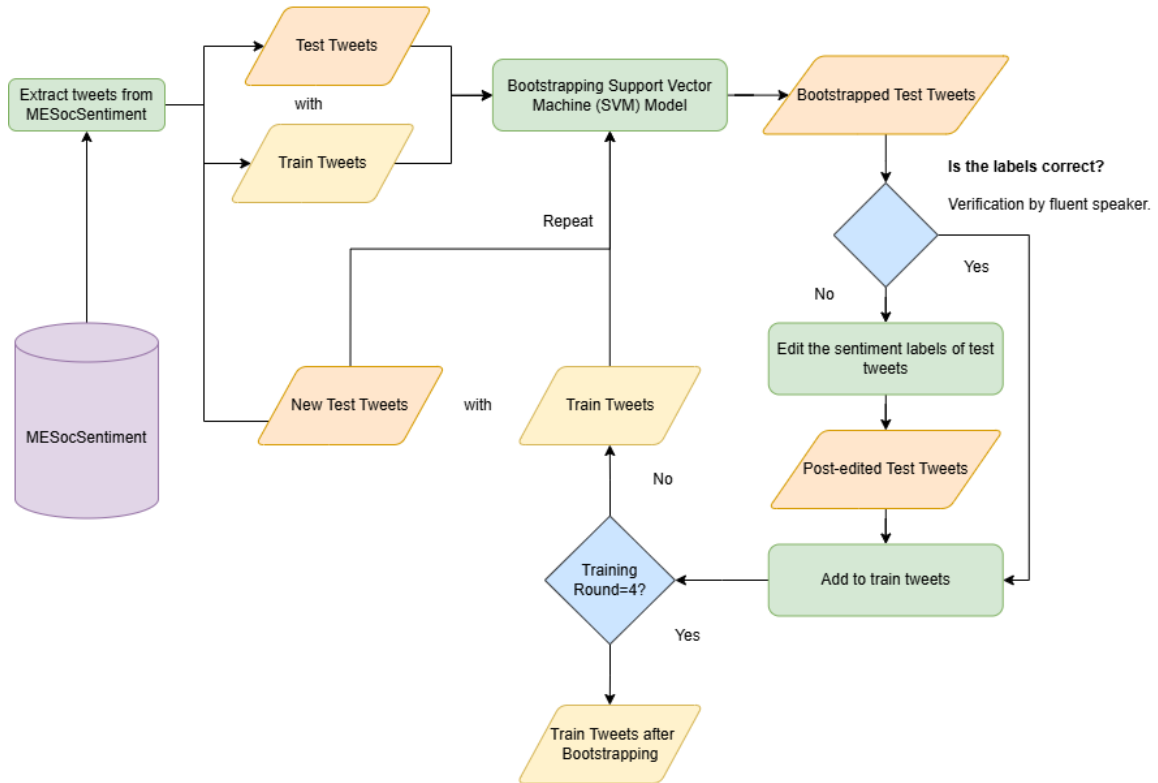


Figure 3.5: The workflow of the bootstrapping technique

Based on Figure 3.4 and Figure 3.5, in this research, in the first set, the initial training data (2000 Training Tweets) was carefully selected via manual observation based on the accuracy of sentiment labels from the MESocSentiment corpus. The data selected for the initial training data was the data categorised as true tweets. It meant that the data were labelled correctly via the voting approach in the corpus construction framework and had the same result when manually checked. The initial training dataset contained a nearly balanced amount of data from positive, negative, and neutral classes.

Instead of being chosen randomly, the initial training data were selected after a manual check to ensure that the dataset contained reliable sentiment labels. If the initial training data had been chosen randomly, there would have been a risk of noise and data with incorrect sentiment labels in the dataset. Such issues could negatively impact the performance of the bootstrapping model to generate accurate predictions of sentiment labels. Hence, this step was crucial to reduce any noise and any errors to be propagated throughout training rounds. As a result, the initial training data was suitable to train the bootstrapping sentiment model.

This dataset was then used for training the first sentiment model and the model was later used to classify another portion of tweets from the corpus (1000 Test Tweets) to check the model's performance. As shown in Figure 3.5, no confidence threshold was applied to those predictions, instead, the outputs were manually verified and any incorrect sentiment labels in the output were updated. This step was utilised to refine the quality of sentiment labels of the test set, thus also improving the quality of the training data used in benchmarking sentiment models. After corrections, the 1000 Test Tweets were added to the first training data to become the new training data for the next round (3000 Train Tweets). The steps in each round were repeated until the final round of this bootstrapping technique (until achieving 6000 Train Tweets).

Based on Figure 3.4, the bootstrapping technique was applied in four training rounds, and the SVM algorithm was utilised as the bootstrapping sentiment model in each training round. All parameters in each round were the same regarding feature extraction method, kernel, and approach type for multiclass classification in all training rounds. Term Frequency - Inverse Document Frequency (TF-IDF) was used for feature extraction to assess the importance of words in a document (Ahuja et al., 2019). Since the sentiment classifiers had

positive, negative, and neutral, a multiclass classification was employed, and the approach used in this research was called the One vs Rest strategy (Pedregosa et al., 2011). On top of that, a linear kernel was used for SVM because the option was widely used for text classification.

Since the bootstrapping technique in this study was utilised to build reliable training data for benchmarking rather than to develop a final sentiment classification model, a traditional performance comparison before and after bootstrapping was not applicable. Instead, the performance of the bootstrapping SVM model in each training round was monitored, with accuracy used as the primary evaluation metric. It was chosen because it reflected the proportion of clearly predicted sentiment labels, ensuring the reliability of the outputs. This step acted as internal quality control to ensure that the bootstrapping model progressively improved in each training round as more verified data was incorporated, hence also progressively refining the training data for benchmarking.

The initial sentiment labels in the MESocSentiment corpus were generated via the majority voting approach using three existing sentiment analysers in the previous framework. The limitation of the pseudo-labels generated from the voting was recognised, thus, manual verification in each training round ensured the reliability of the labels. Specifically, only tweets with accurate sentiment labels upon manual observation were selected as training data in the initial training round. Additionally, in each round of the bootstrapping process, model-predicted labels of the test data were manually reviewed and corrected before being added to the training data for the subsequent rounds. This iterative human-in-the-loop verification considerably mitigated the risk of error propagation and improved confidence in the quality of sentiment labels used in the built training data for benchmarking sentiment models.

3.5.3 Benchmarking Sentiment Models Built Using Machine Learning and Deep Learning Techniques

The objective of benchmarking was to evaluate the performance of code-switched sentiment models from selected machine learning and deep learning techniques on Malay-English code-switched data from the MESocSentiment corpus. After applying the bootstrapping technique, both fourth training data (5000 tweets) and the true labels from the fourth training round's test set were combined to create a new training set (6000 Train Tweets). This dataset with a size of 6000 Malay-English tweets was used in the next stage to train machine learning models for benchmarking sentiment models.

Thus, the train set for benchmarking contained 6000 Malay-English tweets initially taken from the MESocSentiment corpus but was refined via the bootstrapping process to build the training dataset for benchmarking of sentiment models. The bootstrapping process involved re-evaluating and updating the sentiment labels based on bootstrapping model predictions iteratively. This process helped improve the quality of the sentiment label of the train set. Hence, while the content was sourced from the MESocSentiment corpus, the bootstrapped 6000 train data was a modified version optimised for benchmarking sentiment models. Another randomly handpicked 2000 tweets from the MESocSentiment corpus were used to test the models. The labels of these tweets were manually checked to get the ground truth of the test set.

The train set and the ground truth of the test set were used for benchmarking sentiment models with chosen machine learning and deep learning models. Conventional machine learning techniques selected for experiments were Support Vector Machine (SVM), Naïve Bayes, and Random Forest (RF). Deep learning models that were used were

Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and Bidirectional Encoder Representations from Transformers (BERT). These techniques were selected because they were the most used machine learning and deep learning approaches in prior code-switching sentiment analysis studies, as identified in the conducted literature review presented in Section 2.9.3. In this stage, TF-IDF was still used in feature extraction for Support Vector Machine, Naïve Bayes, and Random Forest.

Table 3.12: Hyperparameter values and settings for machine learning techniques

Hyperparameter	Value	Techniques
Decision_function_shape	ovr	Support Vector Machine
kernel	linear	Support Vector Machine
Random State	1	Random Forest
The number of trees	default (100)	Random Forest
Maximum depth	None	Random Forest

Table 3.12 showed other hyperparameter values and settings used for the chosen conventional machine learning techniques. The hyperparameters for the SVM model were the same as those used in the bootstrapping technique. The decision_function_shape was set to “ovr”. This value set the SVM model to use the One vs Rest strategy for multiclass classification. The linear kernel was used for the SVM model because it was widely used for text classification. For Naïve Bayes, Multinomial Naïve Bayes was used for this stage because it was widely used for text classification.

For the Random Forest algorithm, the random state was set to 1 to control the randomness of feature sampling that was considered when looking for the best split at each node. The number of trees in the forest was set to the default, which was 100. The maximum

depth of the tree was set to the default, which was None. This meant that nodes were expanded until all leaves were pure. It also meant nodes were expanded until all leaves contained fewer than `min_samples_split` samples (Pedregosa et al., 2011). For LSTM, some structures and hyperparameter values and settings used in this model for this stage were adopted from the paper by Thara and Poornachandran (2022).

Table 3.13: Hyperparameter values and settings for LSTM model

Hyperparameter	Value
Maximum sequence length	16
Embedding size	100
Dropout rate	0.5
Activation function	tanh, softmax
Optimizer	Adam
Learning rate	0.001
Loss function	Categorical cross-entropy

Table 3.13 showed the hyperparameter values and settings used for the LSTM model. The value for maximum sequence length was 16 which was the highest value used in the paper. Meanwhile, word embedding size used was 100 which was the lowest value used in the paper. Word embeddings were passed through 1-D Spatial Dropout layer with a rate of 0.5. The next layer is the LSTM layer which uses hidden units of 130 and tanh activation function following Thara and Poornachandran (2022). Then, a dropout layer is added with a rate of 0.5 before the feedforward neural network layer. The feedforward layer also acts as a classification layer in this model. The value for units in this layer is 3 which is the number

of sentiment classes used in the dataset and using softmax activation from Thara and Poornachandran (2022). Softmax activation is used for multi-class classification.

For model compilation, the optimizer used was Adam which was one of the optimizers used in the paper with a learning rate of 0.001. The rate used was the default rate because the learning rate value was not stated in the paper by Thara and Poornachandran (2022). The performance metric used was accuracy and categorical cross-entropy loss function. The author only stated using cross-entropy as a loss function, thus a specific cross-entropy function type was chosen for this model, which was categorical cross-entropy (Thara & Poornachandran, 2022).

In experiments for Bidirectional Encoder Representations from Transformers (BERT), the pre-trained BERT model used in this experiment was the mBERT-Base model released by the developer. The model was trained in 104 languages and had 12 layers of transformer blocks, a hidden layer size of 768, 12 attention heads, and 110M parameters. The hyperparameter values and settings used for fine-tuning the model were from the suggestion from the BERT's original paper, as shown in Table 3.14 (Devlin et al., 2018).

Table 3.14: Hyperparameter values and settings for BERT

Hyperparameter	Value
Learning rate	2e-5, 3e-5, 5e-5
Batch size	16, 32
Epoch	2, 3, 4
Optimizer	Adam
Loss function	Categorical cross-entropy

Some structures and hyperparameter values and settings used in the CNN model were also adopted from the work of Thara and Poornachandran (2022). The hyperparameter values and settings used in this model are shown in Table 3.15.

Table 3.15: Hyperparameter values and settings for the CNN model

Hyperparameter	Value
Maximum sequence length	16
Embedding size	100
Filter size	256
Convolutional kernel size	1
Strides	1
Dropout rate	0.5
Activation function	selu, relu, softmax
Learning rate	0.001
Loss	Categorical cross-entropy

The maximum sequence length used was 16 which was the highest value from the paper. Word embedding value was 100 which was the lowest value used by Thara and Poornachandran (2022). After the embedding, the next layer was a 1-D convolutional layer with a filter size of 256, convolutional kernel size of 1, and strides value of 1. All values of hyperparameters in this layer were from Thara and Poornachandran (2022). Then, the output of this layer was passed through the 1-D global max pooling layer like the structure used in the paper.

Thara and Poornachandran (2022) used two feedforward neural networks in their model. The first feedforward neural network used the units value of 256 with a relu activation function. A dropout layer was added between two feedforward layers to prevent overfitting

with a rate of 0.5. The second feedforward layer was the classification layer for the model. In this layer, the units value was the number of sentiment categories and used the softmax activation function. In compiling the model, Adam was the chosen optimizer with the learning rate set to default of 0.001. The loss function used was categorical cross-entropy (Thara & Poornachandran, 2022).

3.6 Chapter Summary

In this chapter, the methodology used in this research was shown and explained. The summary of each following section was described in the chapter's introduction. Section 3.2 explained the stages in the framework of getting the MESocSentiment corpus. The corpus contained code-switched Malay-English social media data. The framework comprised four stages: data collection, data preprocessing, language identification, and sentiment identification.

Section 3.3 described the sentiment modelling strategy using data from the MESocSentiment corpus. The strategy contained two stages which were bootstrapping method and comparison of sentiment modelling results between chosen machine learning techniques and deep learning models. Lastly, Section 3.4 described the tools used in the methodology including Python libraries, modules, and packages used in the research.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Overview

In this chapter, the findings of stages in the methodology of this research were shown and discussed. Section 4.2 showed the results of the MESocSentiment corpus construction framework. Then, Section 4.3 showed the results of the bootstrapping technique and benchmarking results from chosen machine learning techniques and deep learning models in the supervised machine learning approach for benchmarking sentiment models. After that, Section 4.4 discussed results from both corpus construction framework and supervised approach for benchmarking. Lastly, Section 4.4 shows the summary of this chapter.

4.2 MESocSentiment Construction Framework End Results

The MESocSentiment construction framework aimed to construct a Malay-English code-switched sentiment corpus from social media data, demonstrated with tweets in this study. At the end of the framework, a Malay-English code-switched sentiment corpus has been constructed and named the MESocSentiment corpus. This corpus was created after removing ambiguous tweets from the result of the voting approach (majority voting) which involved three existing sentiment analysers: VADER, TextBlob, and pysentimiento. This voting approach was used to get final sentiment labels for code-switched tweets. This corpus currently contains 19, 714 tweets and its final statistics are shown in Table 4.1. From the table, most tweets had been labelled as NEUTRAL with 15, 422 tweets. Meanwhile, the number of negative tweets was the lowest with 1, 073 tweets.

Table 4.1: The Statistics of the MESocSentiment Corpus In Terms of Sentiment

Sentiment	Number of Tweets	Percentage (%)
NEUTRAL	15422	78.23
POSITIVE	3219	16.33
NEGATIVE	1073	5.44

4.3 Supervised Machine Learning Approach for Benchmarking Sentiment Models Results

The results for stages in sentiment modelling strategy were shown and discussed in this section. Section 4.3.1 showed the results for the bootstrapping technique. Meanwhile, sentiment modelling results of machine learning and deep learning approach was shown and discussed in Section 4.3.2.

4.3.1 Results of Bootstrapping Technique

The first stage in the sentiment modelling strategy was the bootstrapping technique. In the bootstrapping technique, some tweets from the MESocSentiment corpus were used in this stage. There were four training rounds in this stage and post-editing was done in each round. Each round used a Support Vector Machine model as the sentiment model. For post-editing, ground truth of the test set was obtained via manual editing and compared to the bootstrapping result in each round.

In the first round, the training set (Training Set 1) contained 2000 selected tweets and 1000 different random tweets from the MESocSentiment for the test set (Test Set 1) in this round. The tweets' sentiment in the training set was checked. After training the Support Vector Machine model with the training set, the model was applied to the test set for prediction.

In the second round, the training set (Training Set 2) contained 3000 tweets and 1000 test tweets from the MESocSentiment. The training set was comprised of Training Set 1 and ground truth of Test Set 1. The test set (Test Set 2) contained another randomly chosen 1000 unused tweets from the MESocSentiment. The Support Vector Machine model was trained with Training Set 2. Then, the model was used to predict sentiments of tweets Test Set 2.

In the third round, the training set (Training Set 3) contained 4000 tweets and 1000 different random tweets from the MESocSentiment for test set (Test Set 3). The training set was a combination of Training Set 2 and ground truth of Test Set 2. The test set (Test Set 2) tweets were from the left, unused tweets from the MESocSentiment corpus. The Support Vector Machine model was trained with Training Set 3. Then, the model was used to predict sentiments of tweets in Test Set 3.

In the fourth round, Training Set 4 contained 5000 tweets and Test Set 4 has 1000 tweets from the MESocSentiment. The training set was a combination of Training Set 3 and ground truth of Test Set 3. Tweets in Test Set 4 were different from previously used tweets. The tweets were taken from the left, unused tweets from the MESocSentiment corpus. After the Support Vector Machine model was trained with the training set, it was used to predict sentiments of tweets in Test Set 4.

Table 4.2 showed the accuracy results from all bootstrapping rounds. From the table, overall accuracy steadily increased with each training round. However, the increase could be minimal or quite high. From the result, the increase was apparent in Test Set 1 to Test Set 2 and from Test Set 3 to Test Set 4. But the increase from Test Set 2 to Test Set 3 was barely noticeable with an increase of 0.01 only. This occurrence could be attributed to the incremental increase in the number of tweets in each training set as training rounds progressed. As the training rounds progressed, the Support Vector Machine model in each

round learned from more data. Consequently, the models were more proficient at predicting sentiment labels for the test set in each round.

Table 4.2: Accuracy of Bootstrapping SVM Model in All Training Rounds to MESocSentiment

Test	Test Set 1	Test Set 2	Test Set 3	Test Set 4
Accuracy	0.61	0.72	0.73	0.81

Table 4.3 showed all training rounds' precision, recall, and F1-score results. From the table, the neutral class performed best in precision, recall and F1-score for all training rounds. This table showed that each round's SVM model was most reliable in identifying and correctly classifying neutral tweets.

Table 4.3: Precision, Recall, and F1-Score Results for All Training Rounds

Training Round	Sentiment	Precision	Recall	F1-Score	Support
1	NEGATIVE	0.06	0.48	0.11	33
	NEUTRAL	0.93	0.62	0.74	843
	POSITIVE	0.40	0.59	0.47	124
2	NEGATIVE	0.01	0.33	0.01	3
	NEUTRAL	0.95	0.76	0.85	895
	POSITIVE	0.41	0.32	0.36	102
3	NEGATIVE	0.01	0.22	0.02	9
	NEUTRAL	0.92	0.82	0.86	862
	POSITIVE	0.63	0.22	0.33	129
4	NEGATIVE	0.08	0.60	0.14	15
	NEUTRAL	0.91	0.90	0.90	831
	POSITIVE	0.85	0.36	0.50	154

As in Table 4.3, in the first round, the worst-performing class in terms of precision, recall, and F1-Score was the negative class. In the second round, the negative class performed worst in general by having the worst values for precision and F1-score. Same as in the second round, the negative class performed worst in terms of precision and F1-Score in the third round. Surprisingly, the neutral and negative classes had the same recall value of 0.22. However, both classes had different precision and F1-score values. Then, as in the second and third training rounds, the negative class performed worst in terms of precision and F1-Score the fourth round.

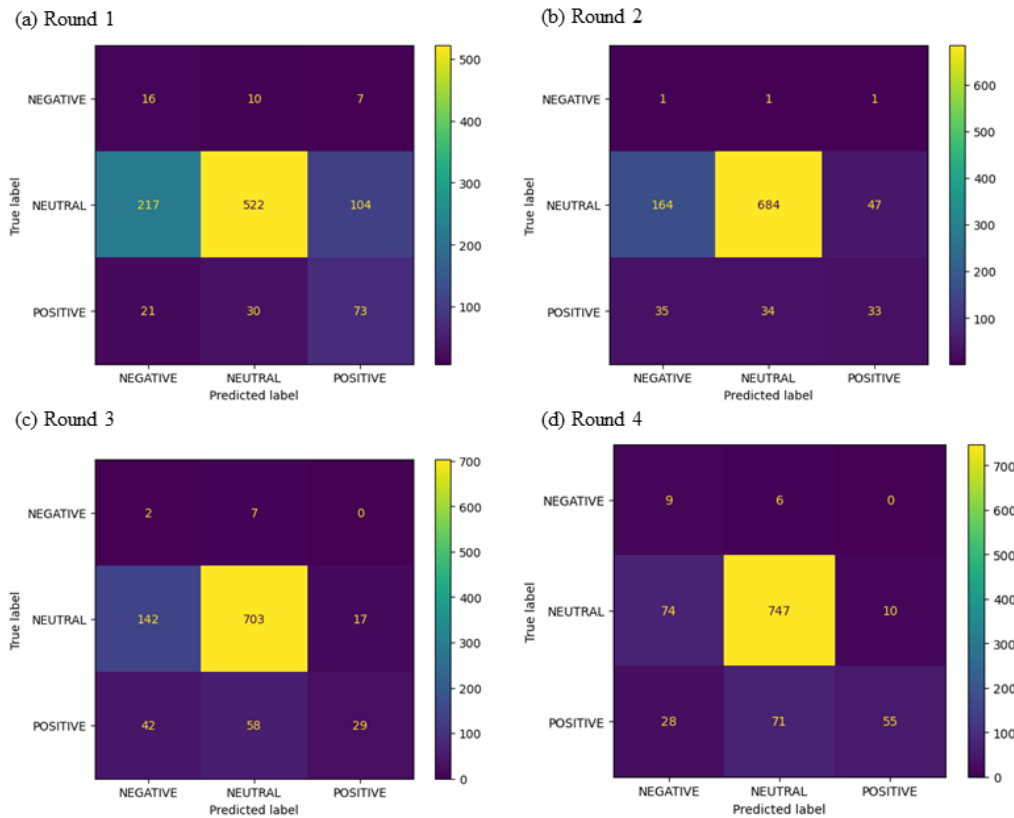


Figure 4.1: Confusion Matrices for All Training Rounds

Figure 4.1 showed all confusion matrices for all training rounds of the bootstrapping technique. Figure 4.1(a) showed the confusion matrix for the first round. For negative class, 16 tweets were correctly classified as negative (true positive). 238 tweets had been incorrectly classified as negative (false positive). However, 17 negative tweets had been

incorrectly classified as neutral and positive tweets (false negative). Then, 729 tweets were correctly classified as non-negative tweets (true negative). For neutral class, 522 tweets were correctly classified as neutral (true positive). 40 tweets were incorrectly classified as neutral (false positive). However, 321 neutral tweets had been incorrectly classified as negative and positive tweets (false negative). Then, 117 tweets were correctly classified as non-neutral tweets (true negative). For positive class, 73 tweets were correctly classified as positive (true positive). 111 tweets were incorrectly classified as positive (false positive). However, 51 positive tweets had been incorrectly classified as negative and neutral tweets (false negative). Then, 765 tweets were correctly classified as non-positive tweets (true negative).

Figure 4.1(b) showed the confusion matrix for the second training round. For negative class, 1 tweet were correctly classified as negative (true positive). 199 tweets had been incorrectly classified as negative (false positive). However, 2 negative tweets had been incorrectly classified as neutral and positive tweets (false negative). Then, 798 tweets were correctly classified as non-negative tweets (true negative). For neutral class, 684 tweets were correctly classified as neutral (true positive). 35 tweets were incorrectly classified as neutral (false positive). However, 211 neutral tweets had been incorrectly classified as negative and positive tweets (false negative). Then, 70 tweets were correctly classified as non-neutral tweets (true negative). For positive class, 33 tweets were correctly classified as positive (true positive). 48 tweets were incorrectly classified as positive (false positive). However, 69 positive tweets had been incorrectly classified as negative and neutral tweets (false negative). Then, 850 tweets were correctly classified as non-positive tweets (true negative).

Figure 4.1(c) showed the confusion matrix for the third training round. For the negative class, 2 tweets were correctly classified as negative (true positive). 184 tweets had been incorrectly classified as negative (false positive). However, 7 negative tweets had been

incorrectly classified as neutral and positive tweets (false negative). Then, 807 tweets were correctly classified as non-negative tweets (true negative). For neutral class, 703 tweets were correctly classified as neutral (true positive). 65 tweets were incorrectly classified as neutral (false positive). However, 159 neutral tweets had been incorrectly classified as negative and positive tweets (false negative). Then, 73 tweets were correctly classified as non-neutral tweets (true negative). For the positive class, 29 tweets were correctly classified as positive (true positive). 17 tweets were incorrectly classified as positive (false positive). However, 100 positive tweets had been incorrectly classified as negative and neutral tweets (false negative). Then, 854 tweets were correctly classified as non-positive tweets (true negative).

Figure 4.1(d) showed the confusion matrix for the fourth training round. For the negative class, 9 tweets were correctly classified as negative (true positive). 102 tweets had been incorrectly classified as negative (false positive). However, 6 negative tweets had been incorrectly classified as neutral and positive tweets (false negative). Then, 883 tweets were correctly classified as non-negative tweets (true negative). For neutral class, 747 tweets were correctly classified as neutral (true positive). 77 tweets were incorrectly classified as neutral (false positive). However, 84 neutral tweets had been incorrectly classified as negative and positive tweets (false negative). Then, 92 tweets were correctly classified as non-neutral tweets (true negative). For the positive class, 55 tweets were correctly classified as positive (true positive). 10 tweets were incorrectly classified as positive (false positive). However, 99 positive tweets had been incorrectly classified as negative and neutral tweets (false negative). Then, 836 tweets were correctly classified as non-positive tweets (true negative).

Table 4.4 showed the statistics for tweets with different sentiment labels in all training rounds. These tweets were the misclassified tweets according to confusion matrices of four training rounds. These tweets had different sentiment labels in the Test Set

(MESocSentiment) and the prediction from the Support Vector Machine in each round. From 389 misclassified tweets in the first round of bootstrapping, the biggest change was 217 neutral tweets had been classified as negative by the model. The smallest difference was 7 negative tweets had been classified as positive tweets. The second round of bootstrapping showed that 282 tweets had been misclassified. The biggest difference in this round was that 164 neutral tweets had been classified as negative tweets. Meanwhile, the smallest difference was only 1 negative tweet had been misclassified as a positive tweet by the model.

Table 4.4 showed that the model had misclassified 266 tweets in the third training round. From the table, no negative tweet had been misclassified as positive. Meanwhile, 142 neutral tweets had been misclassified as negative tweets. From the table, 189 tweets had been misclassified by the model in the fourth training round. Same as in the third training round, no negative tweets had been misclassified as positive. Meanwhile, 74 tweets had been misclassified as negative tweets instead of neutral. The number of neutral tweets classified as negative and positive tweets by the Support Vector Machine model was decreasing with each training round. This situation had happened even though the test set in each round contained different tweets from the MESocSentiment corpus. However, the number of misclassified neutral tweets was still higher than the number of misclassified negative and positive tweets in the first three training rounds. This situation suggested that some neutral tweets in the MESocSentiment corpus might not actually have been neutral.

Table 4.4: Statistics for Tweets with Different Sentiment Labels in All Training Rounds

Training Round	Test Set (MESocSentiment)	Test Set (SVM Model Prediction)	Number of Tweets
1	NEGATIVE	NEUTRAL	10
	NEGATIVE	POSITIVE	7

Table 4.4: continued

	NEUTRAL	NEGATIVE	217
	NEUTRAL	POSITIVE	104
	POSITIVE	NEGATIVE	21
	POSITIVE	NEUTRAL	30
2	NEGATIVE	NEUTRAL	1
	NEGATIVE	POSITIVE	1
	NEUTRAL	NEGATIVE	164
	NEUTRAL	POSITIVE	47
	POSITIVE	NEGATIVE	35
	POSITIVE	NEUTRAL	34
3	NEGATIVE	NEUTRAL	7
	NEGATIVE	POSITIVE	0
	NEUTRAL	NEGATIVE	142
	NEUTRAL	POSITIVE	17
	POSITIVE	NEGATIVE	42
	POSITIVE	NEUTRAL	58
4	NEGATIVE	NEUTRAL	6
	NEGATIVE	POSITIVE	0
	NEUTRAL	NEGATIVE	74
	NEUTRAL	POSITIVE	10
	POSITIVE	NEGATIVE	28
	POSITIVE	NEUTRAL	71

In the post-editing of each round, the ground truth of the test set was obtained, where the tweets in the test set were manually edited. The results of test sets (MESocSentiment), bootstrapping using Support Vector Machine models, and the test sets of post-edited tweets (ground truth) for each round were compared for analysis. Table 4.5 showed the summary of test sets (MESocSentiment), prediction of test sets by Support Vector Machine Models, and post-edited test sets for all bootstrapping rounds in terms of sentiment categories.

In the first training round, the bootstrapped model outputs were closer to the post-edited tweets than Test Set 1 (MESocSentiment), with 215 tweets between the bootstrapped model outputs and the post-edited tweets. However, this was only for neutral and negative sentiment categories. The number of positive tweets in Test Set 1 (MESocSentiment) was closer to the post-edited tweets set compared to the bootstrapped model outputs with a difference of 7 tweets. Instead, the difference between positive tweets in bootstrapped output and post-edited tweets of the first training round was 53.

In the second training round, the bootstrapped model outputs were nearer to the post-edited tweets compared to Test Set 2 (MESocSentiment), with a difference of 46 tweets between the bootstrapped output and the post-edited tweets. The bootstrapped model also gave better outputs label which were nearer to ground truth compared to the test set from the MESocSentiment in the third training round. The difference between ground truth test set and prediction by Support Vector Machine model was 64 tweets.

Same as in the previous three training rounds, the bootstrapped model outputs were nearer to the post-edited tweets than Test Set 4 (MESocSentiment), with 86 tweets between the bootstrapped output and the post-edited tweets. In all rounds, the bootstrapped model generally performed better, providing outputs closer to the ground truth compared to the test set from MESocSentiment. The result showed that the bootstrapping technique was able to

make the manual checking process faster during building the training data for sentiment modelling.

Table 4.5: Test Set Statistics for MESocSentiment, SVM Model Prediction and Post-Edited Tweets for All Training Rounds

Training Round	Sentiment	Test Set (MESocSentiment)	Test Set Prediction by SVM Model	Test Set Post-Edited Tweets	Prediction vs Post-Edited
1	NEUTRAL	843	562	696	134
	NEGATIVE	33	254	173	81
	POSITIVE	124	184	131	53
2	NEUTRAL	895	719	742	23
	NEGATIVE	3	200	182	18
	POSITIVE	102	81	76	5
3	NEUTRAL	862	768	736	32
	NEGATIVE	9	186	191	5
	POSITIVE	129	46	73	27
4	NEUTRAL	831	824	781	43
	NEGATIVE	15	111	130	19
	POSITIVE	154	65	89	24

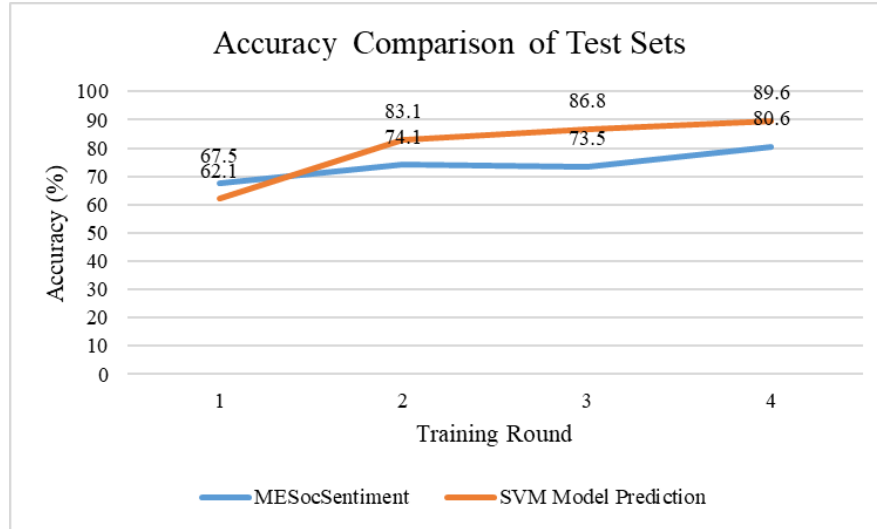


Figure 4.2: Accuracy Comparison for Test Set MESocSentiment and SVM Model Prediction to Ground Truth

Figure 4.2 showed the summary of accuracy comparisons of test sets from MESocSentiment and the prediction of Support Vector Machine models to the ground truth (post-edited tweets) for each bootstrapping training round. The blue line showed the accuracy of the tweets from the MESocSentiment corpus before bootstrapping compared to the ground truth of test sets in each training round. Meanwhile, the orange line showed the accuracy of the tweets after bootstrapping using SVM models compared to the ground truth of test sets in each training round.

From the graph in Figure 4.2, accuracy of prediction result by the bootstrapping Support Vector Machine model to the ground truth of the test set (62.1%) was slightly lower than the test set from MESocSentiment (67.5%), with a difference of 5.4% in the first training round. Nevertheless, the accuracy of both test sets was not good because none of the test sets achieved 70.0% accuracy. The situation changed for test sets in the second training round, where accuracy of the Support Vector Machine model to the ground truth (83.1%) was marginally higher compared to Test Set 2 (MESocSentiment) to ground truth (74.1%). The difference was 9%.

As shown in Figure 4.2, there was a significant difference of 13.1% of between the accuracy of both test sets in the third training round. The accuracy of predicted outputs by the Support Vector Machine model to the ground truth (86.8%) is significantly higher than test set from MESocSentiment to the ground truth (73.5%). In the last round of bootstrapping, the accuracy of both test sets was more than 80.0 % as shown in Figure 4.2. As in the previous two rounds, the accuracy of predicted results from the Support Vector Machine model to the ground truth (89.6%) was slightly higher than the accuracy of Test Set 4 (MESocSentiment) to ground truth (80.6%). The difference was 9.0%.

From Figure 4.2, accuracy of test sets of SVM model prediction significantly increased in the second training round, then steadily increased until the last training round. Meanwhile, the accuracy of test sets of MESocSentiment steadily increased in the second training round before slightly decreased in the third training round. However, the accuracy of the test set for MESocSentiment slightly increased in the fourth training round.

The training rounds of the bootstrapping technique stopped at the fourth round due to the imbalance of data. The limitation was most tweets in the training set for the comparison of models' results were neutral tweets. This training set comprised of post-edited tweets of the fourth training round and the Training Set 4. Yet, the imbalance of data situation was still happening after finishing the fourth training round of bootstrapping.

Figure 4.3 showed the composition of tweets in the training set for cben in terms of sentiments. It could be seen that the training set had a data imbalance where more than half of the dataset were neutral tweets. From 6000 tweets in the training set, 3621 tweets were neutral tweets followed by 1343 negative tweets and 1036 positive tweets.

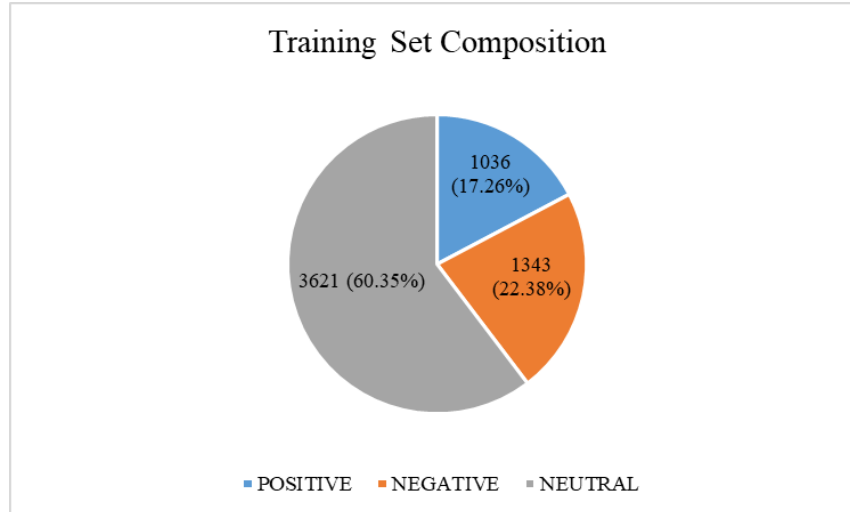


Figure 4.3: Composition of Training Set for Benchmarking Sentiment Models

The imbalance of data in this dataset was remarkable where one sentiment category made up more than half of the training set. Meanwhile, the minority classes only contributed to 39.65% of the training set. The imbalanced dataset may lead machine learning models to perform poorly because the machine-based methods tend to train on the majority class, resulting in skewed performance (Mujahid et al., 2024). Oversampling was used on the training set to rectify this situation of data imbalance in the train set. This technique was chosen to handle the imbalanced training set because it kept all the tweets from the majority and minority sentiment classes, hence there is no loss of original information.

SMOTE was chosen as the oversampling technique in this research because it achieved quite good results in the work by Mujahid et al. (2024). Figure 4.4 shows the composition of the training set after oversampling using SMOTE. The resampled training set contained 10,863 tweets, with each sentiment category having 3,621 tweets. This happened because the majority class of the original training set contained 3621 tweets. Hence, synthetic samples were created and added to the dataset for the negative and positive classes until each sentiment category had 3,621 tweets.

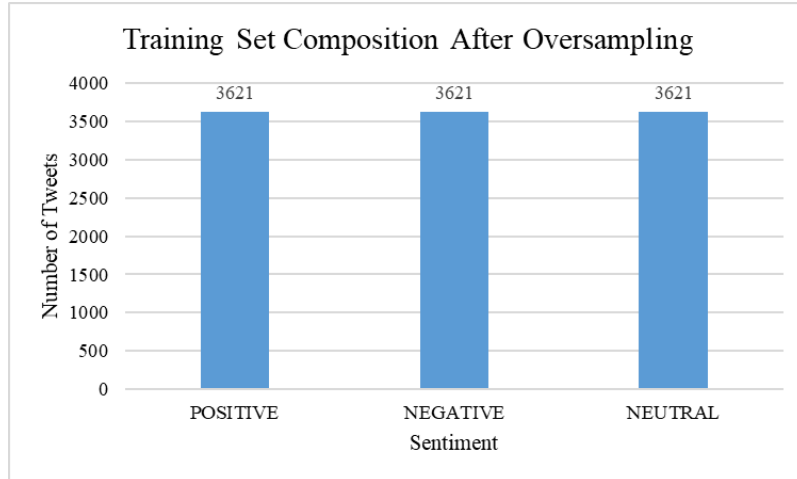


Figure 4.4: Composition of Training Set for Benchmarking Sentiment Models after Oversampling

4.3.2 Results of Benchmarking Sentiment Models

After the bootstrapping technique, a new train set and test set were created for the benchmarking code-switched sentiment models. The training set was the combination of Training Set 4 and the ground truth of Test Set 4, containing 6000 tweets. Due to tweets imbalance, the train set underwent oversampling with the resampled train set contains 10,863 tweets. This resampled train set was used to train on six sentiment models that are based on chosen three machine learning algorithms and three deep learning models. Machine learning algorithms used were Support Vector Machine (SVM), Naïve Bayes, and Random Forest (RF). Deep learning models used were Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and Bidirectional Encoder Representations from Transformers (BERT). These sentiment models were used to predict the sentiment of the ground truth of the test set of 2000 tweets. The prediction results of the test set from all sentiment models were compared for the performance's benchmarking.

Table 4.6 shows the accuracy results for the benchmarking between the SVM, Naïve Bayes, and Random Forest models. From the table, the SVM model achieved the highest accuracy (76.55%) compared to the Random Forest and Naive Bayes models. The Naive

Bayes model achieved the lowest accuracy of 61.4%. The difference between Support Vector Machine and Naive Bayes models was quite significant, with a difference of 15.15% accuracy. However, the accuracy difference between Random Forest and Support Vector Machine models was quite small with 1.40% difference.

Table 4.6: Accuracy of Machine Learning Models

Model	Accuracy (%)
Support Vector Machine	76.55
Random Forest	75.15
Naïve Bayes	61.4

Table 4.7 showed the accuracy of all deep learning models. From the table, the mBERT model achieved the highest accuracy with 69.30% accuracy. The difference was quite huge between mBERT and LSTM models with 33.05% accuracy. Yet, the difference between CNN and LSTM models was slim with a difference of 1.15% accuracy.

Table 4.7: Accuracy of Deep Learning Models

Model	Accuracy (%)
Convolutional Neural Networks (CNN)	35.1
Long Short-Term Memory (LSTM)	36.25
Multilingual Bidirectional Encoder Representations from Transformers (mBERT)	69.30

Figure 4.5 showed the accuracy of all our models compared to the ground truth of the test set. The model that performed best in terms of accuracy was the Support Vector Machine. The model that performed worst in terms of accuracy was the CNN model. From

the figure, deep learning models, which are CNN and LSTM, performed badly, where both models did not even achieve 40.0% accuracy. However, the BERT model achieved nearly 70% accuracy. From 18 BERT models that were fine-tuned using hyperparameter values and settings suggested in the paper by Devlin et al. (2018), four models produced 69.30% accuracy. Meanwhile, Table 4.8 showed the hyperparameter values and settings used in fine-tuning four models.

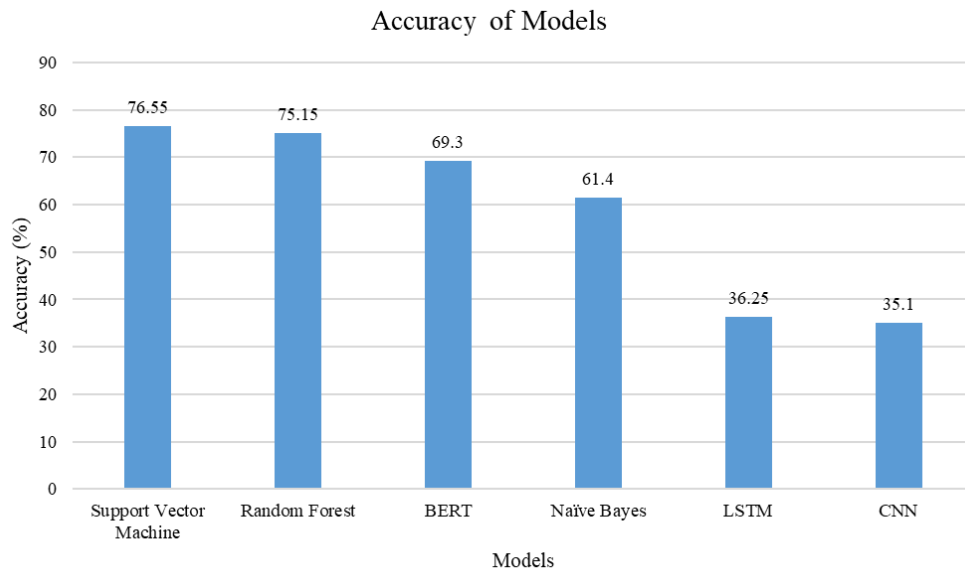


Figure 4.5: Accuracy Results of Code-Switched Sentiment Models

Table 4.8: BERT models with hyperparameter values and settings

Model	Optimizer	Batch Size	Learning Rate	Epoch
A	Adam	16	2e-5	3
B	Adam	16	2e-5	4
C	Adam	32	2e-5	3
D	Adam	32	3e-5	4

4.4 Analysis and Discussion of Results

Results from both framework and strategy were analysed and discussed in this section. Additionally, the MESocSentiment corpus was also analysed and discussed in this section.

4.4.1 Analysis of Findings from MESocSentiment

The definition of the code-switched sentence in this research is that even if there is one foreign word from another language in a sentence, the sentence is code-switched. For example, even if there is only one English word in a sentence and other words are in Malay, it will be considered a Malay-English code-switched sentence.

The aim of the language identification stage was to identify languages of words in the pre-processed tweets. This stage utilized two existing libraries, Malaya (fastText model) and Lingua because of Malay and English identification support. In the result by Malaya's fastText model, probability was only assigned to one category. This result made it easier to filter code-switched Malay-English tweets by taking tweets in the 'Manglish' and 'Rojak' categories. Nevertheless, there was a probability of missing some code-switched Malay-English tweets based on the definition of a code-switched sentence in this research.

From Lingua's language identification result, the presence of code-switched tweets, especially Malay-English tweets, was more frequent in the range of 0.5 to 0.8 confidence value. However, there were also code-switched tweets in the higher range with confidence values of 0.8 to 1.0, but they occurred less frequently. In addition, there were also some outliers, which were other languages besides Malay and English, present due to the coding process. The use of these tools for language identification had reduced the need to find fluent

speakers in Malay and English to classify the tweets. Yet, there were still some noises in the dataset after the task because of the capability of both libraries.

In the sentiment identification, a voting approach using three existing sentiment analysers (VADER, TextBlob, and pysentimiento) was used to classify sentiments of code-switched tweets. These sentiment analysers were used to strengthen the reliability of the sentiment identification process. The MESocSentiment corpus was created after removing ambiguous tweets found after the voting approach using existing sentiment analysers. After the voting approach, ambiguous tweets were in the POSITIVE/NEGATIVE/NEUTRAL category. This category existed because each sentiment analyser gave different sentiment labels to a particular tweet. These tweets were removed because the MESocSentiment corpus must only contain tweets with one sentiment label. The sentiment label was the final sentiment from the voting approach.

Currently, the MESocSentiment is unbalanced, with 78.23% of tweets labelled as neutral. This showed that more than $\frac{3}{4}$ of the dataset belongs to the neutral category. A descriptive analysis of the number of words per tweet had been done on the corpus. The analysis showed that the range length of tweets was 43. The lowest number of words per tweet in the corpus was two words, and the highest number of words per tweet was 45. Next, the mean of the corpus length was 8.92 words. It showed that the average length of tweets in the corpus was around eight to nine words per tweet. Then, the standard deviation for this corpus was 5.56 words. Standard deviation was used to measure the variability of sentence length from the mean. From the result, most of the tweets in the corpus were between three to 15 words per tweet. Tweets in this range were 15 812 tweets, which amounted to 80.21% of the corpus.

Tweets were collected from Twitter using the keyword #Malaysia to get tweets about Malaysia. There were various themes found in the tweets such as sports, politics, art, and festivities. Some tweets in sports were about users congratulating athletes on their achievements in tournaments and championships. Then, some political tweets were about the 2022 Malaysian general election which was held on 19 November 2022. There was also the presence of art-related tweets such as single releases and concerts. Some of festivities-related tweets were about Independence Day, Eid, Christmas, Deepavali, and Lunar New Year.

MESocSentiment was also compared with datasets used in Patwa et al. (2020) in terms of tweets. There were two datasets used in the work, one in Spanish-English (Spanglish) and another in Hindi-English (Hinglish). Every token in each tweet was annotated by languages categories for both datasets. Both datasets also kept some tokens and characters that were removed from the MESocSentiment including special characters, mentions, and URL links.

4.4.2 Advantages and Limitations of MESocSentiment Construction Framework

The advantages of this framework could be seen from the use of a semi-automatic approach in language identification and sentiment identification stages. The use of Malaya and Lingua libraries for language identification minimised the use of fluent Malay and English speakers in identifying languages used in tweets. This approach also managed to reduce the time taken to identify the languages of tweets in collected data.

Plus, chosen existing sentiment analysers were leveraged in a voting approach in sentiment identification, thus made the process faster than manual labelling. The procedure also eliminated the need to find speakers who are fluent in both Malay and English to label

the dataset. It also was able to reduce the time needed to label tweets' sentiment compared to manual sentiment labelling.

Nevertheless, there were some limitations found when using the semi-automatic approach in the framework of getting the MESocSentiment corpus. The use of language identification tools expedited the language identification process but there were still some noises in the dataset due to the limitation of the tools. Plus, there were presence of false tweets in the MESocSentiment corpus after sentiment identification stage. This was because of limitations of sentiment analysers used in the voting approach. This shown that there was still a probability of having false tweets in a dataset even though multiple sentiment analysis tools were used.

4.4.3 MESocSentiment Limitations

Limitations in the above framework have contributed to the shortcomings of the MESocSentiment corpus. The corpus is currently unbalanced and skewed towards neutral tweets, as more than half of the corpus consists of neutral tweets, which may cause bias in sentiment models. Additionally, the presence of noise in the corpus may affect the performance of sentiment models.

4.4.4 Analysis of Supervised Machine Learning Approach for Benchmarking Sentiment Models Results

The results of the sentiment modelling strategy were analysed and discussed in this subsection. Some of the hyperparameter values and settings used for CNN and LSTM models were adopted from the work by Thara and Poornachandran (2022). In terms of accuracy, results from the paper were higher than models in this research for both LSTM and CNN models. From the referenced paper, the CNN model with Word2Vec word embedding achieved 74.55% accuracy. Its LSTM model with FastText word embedding

achieved 73.81% accuracy. The differences between CNN and LSTM models in this research were 39.45% and 37.56% for CNN and LSTM models, respectively. This study's method for CNN and LSTM models was understood to be almost identical to the referenced paper.

Although model training efficiency comparison could have provided additional insights, this aspect was not addressed in the current study. The reason was the constraints of the Google Colab environment, which does not retain execution time data after runtime termination. This limitation made it impossible to retrieve or reference past training durations for all models.

The benchmarking results showed that code-switched sentiment models based on traditional machine learning algorithms outperformed sentiment models based on deep learning models. This may be due to the small size of the training dataset used to train deep learning models which is 6000 tweets. This situation has happened in the sentiment analysis work by Das et al. (2023), where SVM models with the Porter stemming algorithm achieved 82.56% and 86.43% accuracy for English and Bangla datasets, respectively. Yet, the BiLSTM models only managed to achieve 78.10% and 83.72% accuracy for English and Bangla datasets, respectively. The size of each dataset is smaller than our training set, at 2,577 reviews only. This shows that not enough training samples affect the capability of deep learning models to learn the patterns in the training set effectively. This also shows that dataset size constraints do affect the performance of sentiment models.

Meanwhile, in the study by Liu (2020), the author utilised 362, 554 Yelp reviews with ratings to compare the performance of several sentiment models in terms of macro F1-score. The results showed that SVM and Logistic Regression models achieved the highest

macro F1-score (0.757). In contrast, the best-performing deep learning model which was BERT only achieved a macro F1-score of 0.723. Although the number of data points used for training and testing is huge, the traditional machine learning models still outperformed deep learning models. The similarity between this work with our study is in the use of hyperparameter tuning and pre-trained embeddings. This study utilised mBERT pre-trained embeddings compared to GloVe and BERT in Liu (2020). It shows that both factors have a significant impact on the performance of deep learning models in terms of learning and understanding data context as well as optimising model performance.

Besides, in the work by Mammadov and Bakal (2024), the authors utilized 820, 049 publicly available tweets to investigate the capability of machine learning and deep learning models for sentiment analysis of a large amount of social media data. The findings showed that the XG-Boost model performed best with an F1-score of 95.81% using a combination of unigrams and bigrams. Meanwhile, the hybrid CNN-BiGRU model only achieved an average F1-score of 95.42%, which was a little bit lower than the XG-Boost model. The results showed that hyperparameter tuning does affect the capability of deep learning models in terms of optimising model performance. By adjusting parameters, the learning efficiency of deep learning models can be improved, hence improving the performance of deep learning models.

The chosen machine learning and deep learning models were also compared in terms of AUC-ROC metrics. AUC-ROC metrics were used to see the capabilities of the chosen models in distinguishing between sentiment categories. Because the research work was considered a multiclass classification case, there were two focused AUC-ROC values for each model which were micro-average and macro-average.

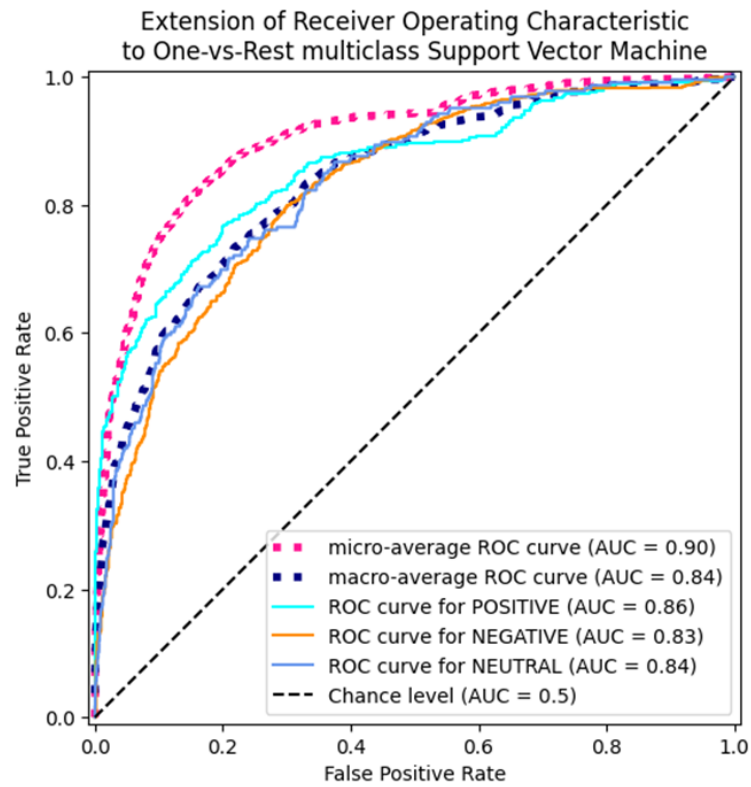


Figure 4.6: ROC graph for SVM Model

Based on Figure 4.6, generally, the SVM model performed well in terms of AUC-ROC metrics. The high micro-average AUC value (0.90) indicated that the model was overall good in distinguishing classes across the dataset. The high macro-average AUC value of 0.84 indicated that the model performed well on average across all classes, regardless of the class sizes. The highest difference between AUC values was between micro-average and negative class with a value of 0.07.

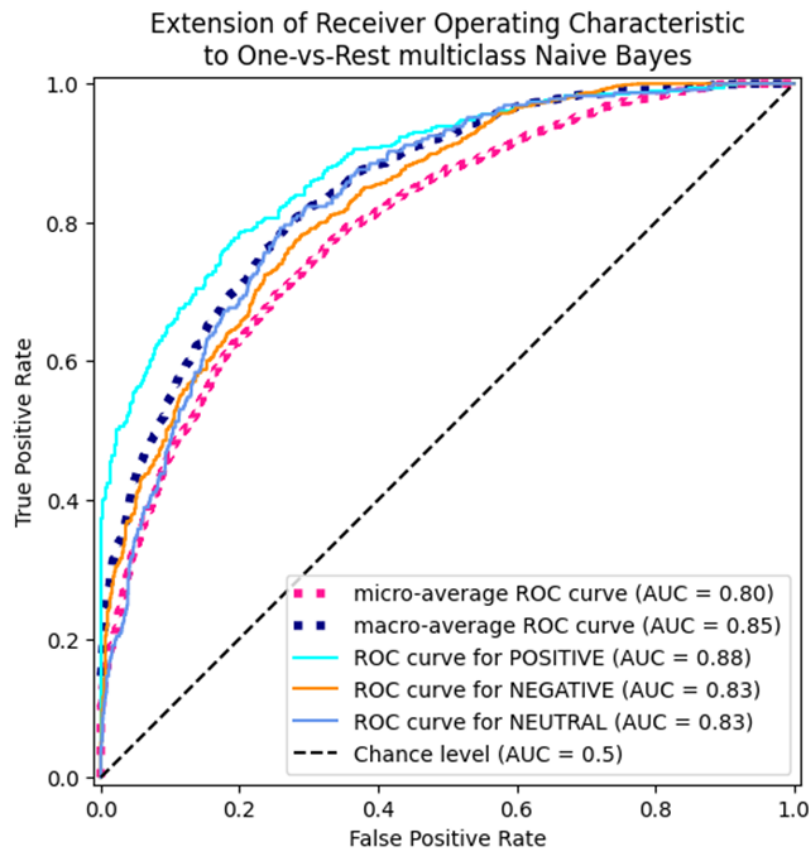


Figure 4.7: ROC graph for Naive Bayes Model

Based on Figure 4.7, the Naïve Bayes model generally performed well in terms of AUC-ROC metrics. The high micro-average AUC value (0.80) indicated that the model was overall good in distinguishing classes across the dataset. The high macro-average AUC value of 0.85 indicated that the model performed well on average across all classes, regardless of the class sizes. The highest difference between AUC values was between micro-average and positive class with a value of 0.08.

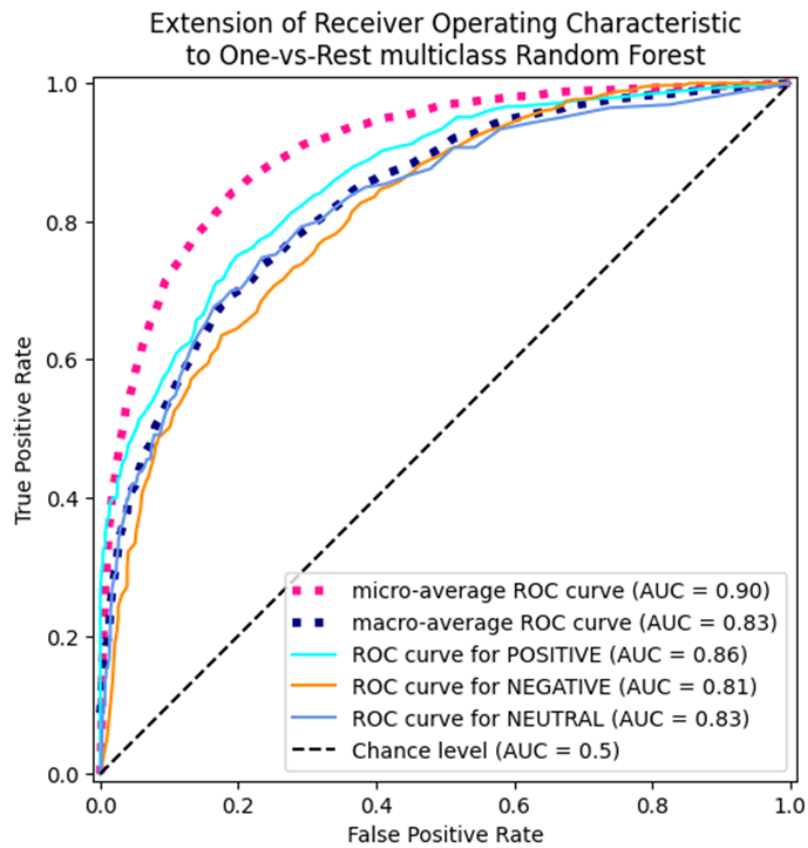


Figure 4.8: ROC graph for Random Forest Model

Based on Figure 4.8, the Random Forest model performed generally well in terms of AUC-ROC metrics. Surprisingly, the model had the same results as the Support Vector Machine model in terms of micro-average, and positive class AUC values. The highest difference between AUC values was between the micro-average and the negative class with a value of 0.09.

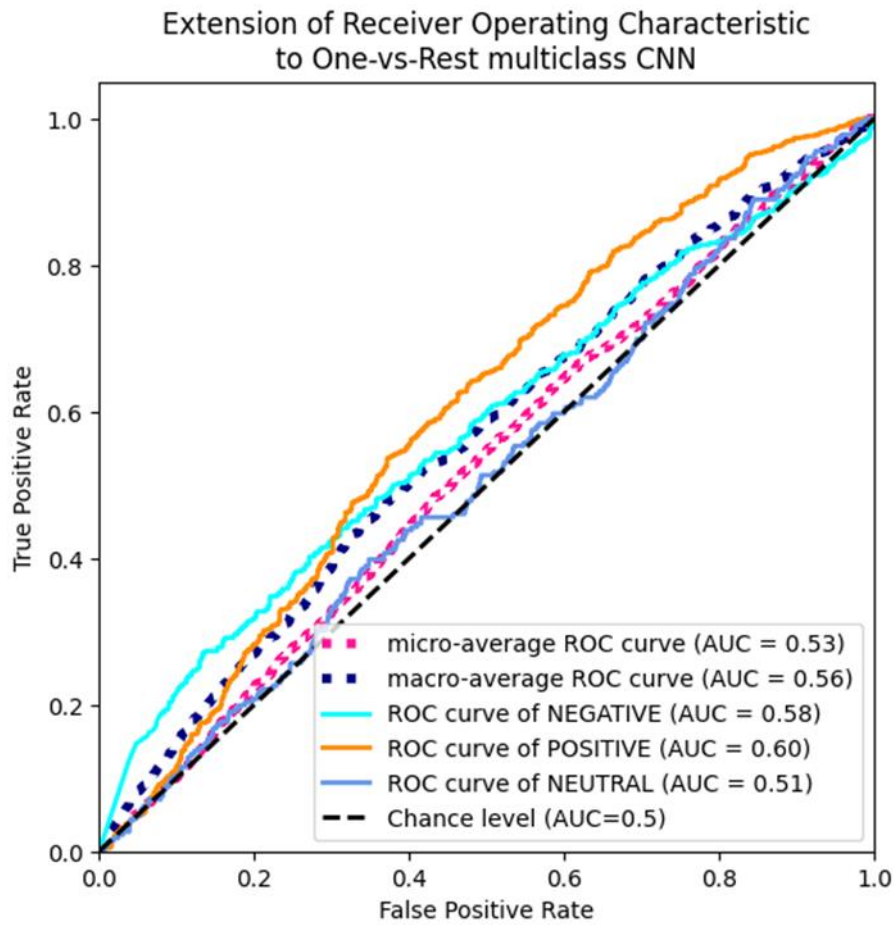


Figure 4.9: ROC graph for CNN Model

Based on Figure 4.9, the model's performance in terms of AUC-ROC metrics was generally not good. The micro-average AUC value of 0.53 shows that the model performed slightly better than random guessing in distinguishing between classes. Plus, the macro-average and all classes' AUC values showed that the model only performed slightly better than random guessing the classes of each tweet.

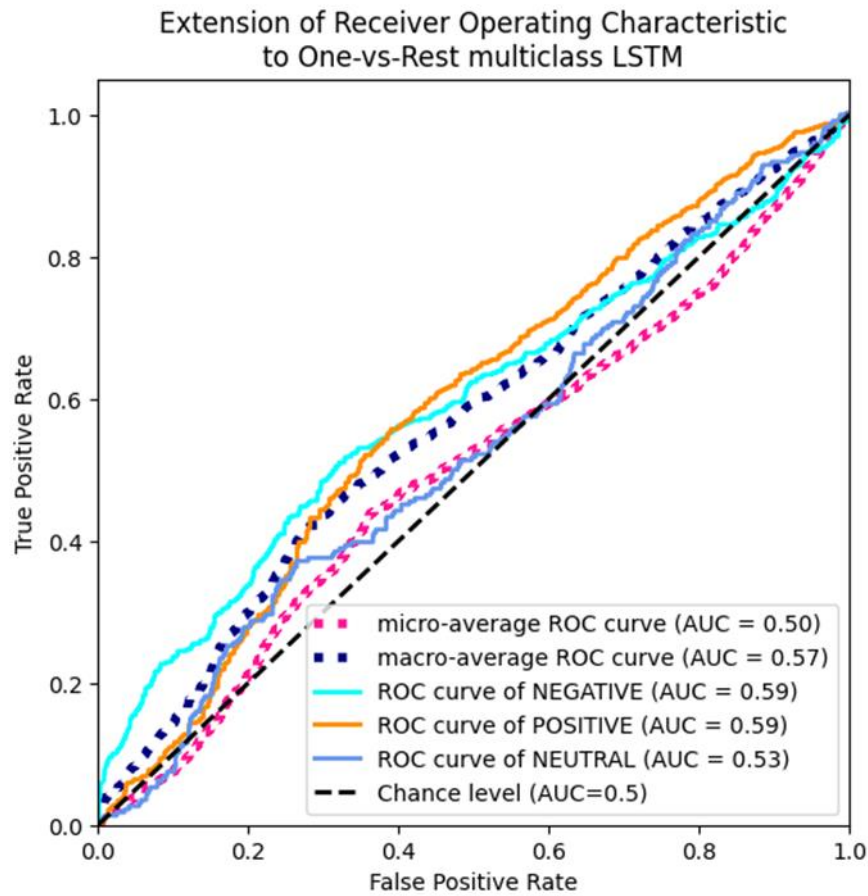


Figure 4.10: ROC graph for LSTM Model

Based on Figure 4.10, the LSTM model did not generally perform well in terms of AUC-ROC metrics. The macro-average AUC value (0.50) showed that the model performed like random guessing in distinguishing between classes. However, the other AUC values showed that the model performed just slightly better than random guessing. This led to the model to perform quite poorly in terms of AUC-ROC metrics.

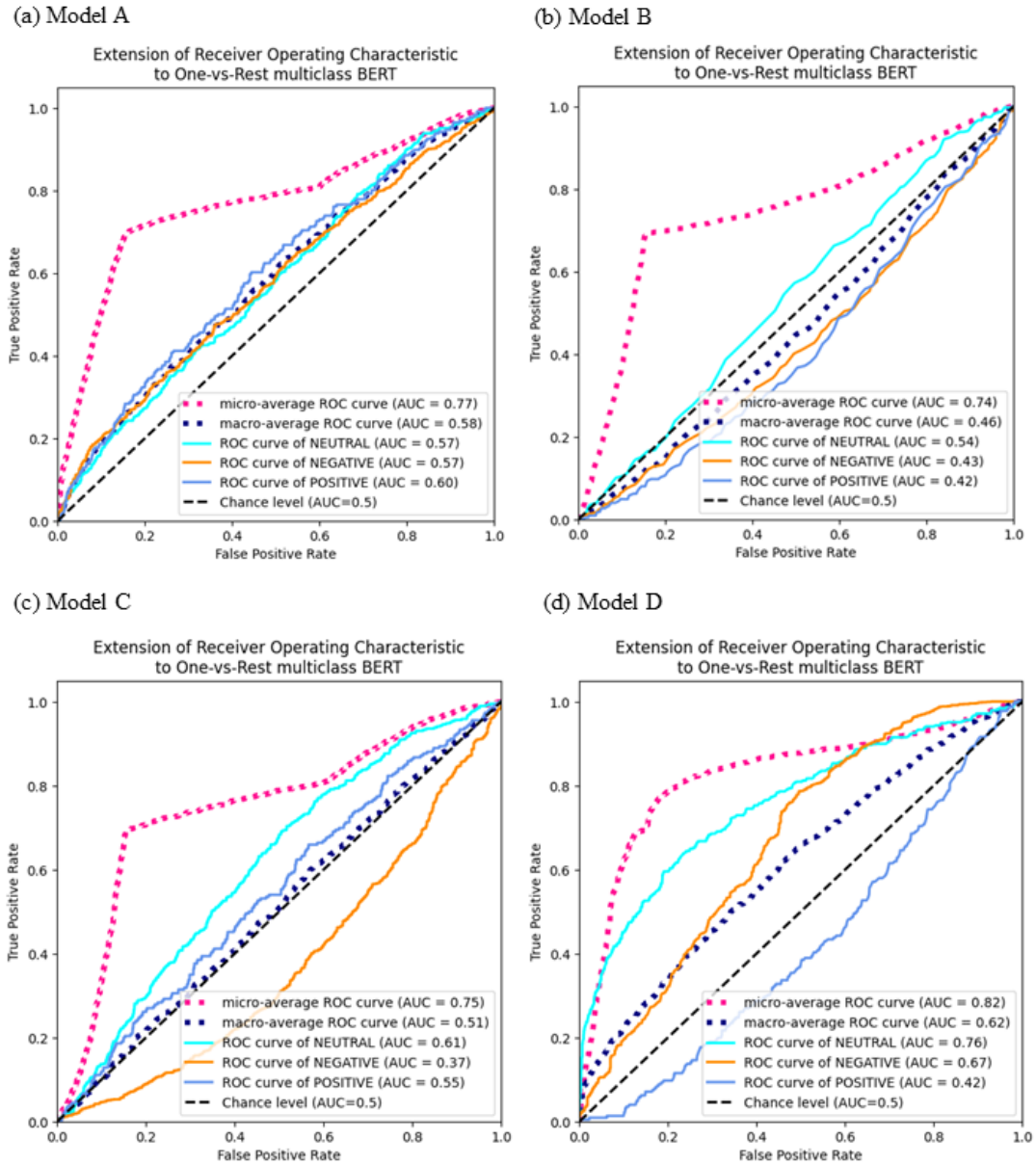


Figure 4.11: ROC graphs for BERT Models

From the experiments, four BERT models produced the same accuracy of 69.30%, as shown in Table 4.7. Figure 4.11 shows ROC graphs for all four BERT models that achieved 69.30% accuracy. Even though all four models achieved the same accuracy, each produced four different ROC graphs with different AUC values. From the figure, Model A had the best ROC curve compared to the other three models. This was because all AUC values in Figure 4.11(a) were more significant than 0.5. The graph indicated that the model

performed well overall in distinguishing between classes, with a relatively high micro-average AUC value. The difference between the lowest and highest AUC values in Figure 4.11(a) was the smallest, at 0.20. However, the other AUC values suggested that the model only had fair performance across specific classes. Although Model C and Model D had higher micro-average AUC values than Model A, both models had AUC values that were less than 0.5. Similarly, Model B had the same micro-average AUC value as Model A but also had several AUC values less than 0.5. Based on the results, the other three models did not perform well in distinguishing across specific classes.

Among all our six models, the SVM model performed best in terms of AUC-ROC metrics. This was because all its AUC values ranged between 0.90 and 0.83, indicating that the model performed well in distinguishing between classes. Although the Random Forest model had the highest AUC value of 0.90 for the micro-average AUC, its lowest AUC value was 0.81. This resulted in a difference of 0.11 between its highest and lowest AUC values, compared to 0.07 for the Support Vector Machine model. In this research, machine learning models generally outperformed deep learning models, which struggled to effectively learn patterns in the data. This underperformance may be attributed to insufficient training data, particularly for deep learning models.

4.4.5 Strength and Limitations of Bootstrapping Technique in the Supervised Machine Learning Approach for Benchmarking Sentiment Models

The strength of the bootstrapping technique in this research was it enabled the sentiment modelling process to become faster. By incrementally increasing the training data each round, it made the manual checking for ground truth for each set faster because only needed shorter time to focus on a set. Then, the model performances could be monitored more to improve the model's performance.

However, this was also depending on the condition of the dataset. Because the MESocSentiment corpus was created from Malay-English tweets, it could be quite difficult to manually checked the test set of certain rounds. The informal condition of code-switched tweets such as shortform usage along with the presence of noise made it difficult to determine the content and context of a tweet, hence making sentiment labelling for ground truth more challenging. But then, the small size of the test set per round made it easier to focus while checking for the manual ground truth.

The limitation of this technique is it could affect the performance of the last sentiment models. This was quite crucial especially if there were uses of deep learning models during sentiment modelling after doing the bootstrapping technique. Deep learning models in general need to have a huge amount of data for model training to function better. However, if the dataset at the end of the bootstrapping technique was not sufficiently large, the training data used in training sentiment models for benchmarking performance may be inadequate.

In this research, the number of used tweets was 8000 tweets at the end of the comparison of the models' performance stage. Thus, there were around 11, 000 remaining unused tweets from the MESocSentiment at the end of the sentiment modelling strategy. In the future, the bootstrapping technique could be improved by either increasing the number of tweets in each test set or increasing the number of rounds used in the bootstrapping process. This would allow for more data to be utilized and tested as training data for model development. However, this approach may also consume more time during the sentiment modelling stage.

4.5 Chapter Summary

This chapter presented results from the framework of obtaining the MESocSentiment corpus and the sentiment modelling strategy. These results were shown and discussed in this chapter. Section 4.3 presented the results from the framework of obtaining the MESocSentiment corpus. Section 4.2 showcased the results from the sentiment modelling strategy. Then, Section 4.3 analysed and discussed the results from both frameworks and the MESocSentiment corpus.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 Conclusion

This research aimed to address some of the challenges of sentiment analysis for Malay-English code-switched social media data which are the shortage of publicly available Malay-English code-switched sentiment corpus especially from social media data, and a need to find a quicker approach for sentiment modelling strategy for sentiment modelling especially for sentiment models trained on code-switched data. To solve these problems, two research questions have been created to guide this research. The questions were RQ1: how to design a framework to build a Malay-English code-switched corpus using social media data for a sentiment analysis task? and RQ2: how can a supervised machine learning approach be effectively designed to benchmark sentiment analysis models trained on the Malay-English code-switched corpus?

To answer these questions, two research objectives were set: to design a framework for building a Malay-English code-switched corpus from Twitter data using language identifiers and existing sentiment analysers (RO1), and to develop a supervised machine learning approach that includes a bootstrapping technique for benchmarking sentiment models using the constructed Malay-English code-switched corpus (RO2). Hence, all research questions and research objectives were fulfilled. The study provided a synthesis of literature topics related to the research. In the methodology, a framework to construct a Malay-English code-switched sentiment corpus (MESocSentiment) using language identifiers and existing sentiment analysers and a supervised machine learning approach that

incorporated a bootstrapping technique to benchmark sentiment models using data from the MESocSentiment corpus were developed.

The developed MESocSentiment construction framework utilized collected tweets along with existing language identifiers and sentiment analysers to construct the MESocSentiment corpus. The MESocSentiment corpus is comprised of 19,714 tweets. The proposed supervised machine learning approach for benchmarking showed the bootstrapping technique's promising ability to build training data for code-switched sentiment models. Then, the benchmark results found that sentiment models based on traditional machine learning algorithms, particularly the SVM model outperformed deep learning-based sentiment models. The SVM model achieved 76.55% accuracy and also the best-performing model in terms of AUC-ROC metrics. In conclusion, this study contributes to sentiment analysis, especially in terms of the development of methodologies and a data resource for the Malay-English code-switching sentiment analysis research area.

5.2 Research Contribution

Several contributions have been made from this research. This includes:

5.2.1 Framework for Building Malay-English Code-Switched Sentiment Corpus Using Semi-Automatic Approaches

Using semi-automatic approaches, we have successfully designed a framework to build a Malay-English code-switched corpus from social media data for sentiment analysis. The framework contains four stages: data collection, pre-processing, language identification, and sentiment identification.

5.2.2 The Malay-English Code-Switched Sentiment Corpus (MESocSentiment) from Social Media Data

We have successfully built a Malay-English code-switched corpus (MESocSentiment) for sentiment analysis using the previous framework's collected tweets. This corpus contains 19 714 tweets and is skewed towards the neutral sentiment. The corpus is an imbalance with 78.23% neutral, 16.33% positive, and 5.44% negative tweets. The MESocSentiment corpus and training notebooks for each model are available on GitHub via this link (<https://github.com/afifahms/MESocSentiment>).

5.2.3 A Supervised Approach with Bootstrapping Technique Implementation for Benchmarking Sentiment Models

We have successfully designed a sentiment modelling strategy using data from the MESocSentiment corpus. The first stage of this strategy was the bootstrapping technique to build training data for sentiment models. In the final round of bootstrapping, training data was 5000 tweets with 1000 test data. Thus, the training dataset for benchmarking code-switched sentiment models contained 6000 tweets. This was followed by the training of code-switched sentiment models based on the chosen machine learning techniques and deep learning models, which are Naïve Bayes, SVM, Random Forest, LSTM, CNN, and mBERT. However, the train set contained 6000 tweets and had imbalanced composition. After oversampling, the resampled train set contained 10, 863 tweets. The resampled train set was used to train sentiment models from chosen machine learning and deep learning techniques and models. The performance of each code-switched sentiment model was benchmarked for analysis in terms of accuracy and AUC-ROC metrics. Overall findings showed that the SVM model was the best-performing model in terms of accuracy and AUC-ROC metrics.

5.3 Research Limitations

However, the research has some limitations or weaknesses despite successfully proposing the research framework. Firstly, we did not fully utilise the data from the MESocSentiment corpus due to time constraints and the data condition during the bootstrapping technique. This dataset constraint has affected the performance of code-switched sentiment models due to insufficient training data, especially on sentiment models based on deep learning models.

Furthermore, in this research, especially for LSTM and CNN models, the models adopted some hyperparameter settings and values from the paper by Thara and Poornachandran (2022). Thus, some hyperparameter values and settings were still not used and tested in the sentiment modelling strategy for LSTM and CNN models. For this reason, the performance of the code-switched models based on LSTM and CNN was negatively affected and constrained because of the lack of time to explore all hyperparameter values and settings used in Thara and Poornachandran (2022).

Besides, the findings of this study are based on collected Malay-English code-switched tweets, which the data has its own characteristics including short text length and informal sentences and expressions. Therefore, the applicability of the proposed framework may be constrained when applied to other data sources of code-switched texts such as speech transcripts as well as online forums and blogs. Data from these sources may have longer sentences, more formal structures, and different code-switching patterns.

Lastly, a few recent machine learning techniques, especially deep learning models, were not included in the sentiment modelling strategy. This was because the techniques used were chosen from the results of the review paper on trends of sentiment analysis on code-

switched social media data in the literature review. This limitation may be considered for future works of this research.

5.4 Future Works

There are several improvements or future works for this research. The first possible future improvement is using more data from MESocSentiment corpus by either increasing the number of training rounds or the number of tweets used per training round. As also indicated in Section 5.4, the CNN and LSTM models in the sentiment modelling strategy can be trained using different hyperparameter values and settings in the future to see if these models' performance can be improved. This is because the performance of a deep learning model may be affected by its hyperparameter values and settings.

Another potential future work for this research is to explore aspect-based sentiment analysis using the proposed hybrid model in Kabir et al. (2023) with the MESocSentiment corpus. This plan aims to extract aspects inside tweets in the MESocSentiment corpus and their sentiments. The identified aspects can be used to uncover popular topics in the corpus and the sentiments mostly associated with them. Moreover, another potential future work for this research is to expand the data sources used in the framework. This is intended to assess the robustness and generalisability of the framework, hence expanding its applicability beyond social media sites.

Lastly, as mentioned in the research limitations section, more recent machine learning techniques, especially deep learning models, can be used in the supervised approach for benchmarking sentiment models in the future. Hence the data from the corpus constructed in this research can be utilised to train code-switched sentiment models based on some recent deep learning models such as XLNet and XLM-RoBERTa (XLM-R). A

promising direction is the use of transfer learning, such as fine-tuning XLNet and XLM-R on a dataset built from the MESocSentiment corpus for sentiment classification. These models' performance then can be benchmarked to determine their capabilities in handling sentiment analysis for Malay-English code-switched text.

The potential of XLNet in code-switching sentiment analysis is the model utilize permutation language modelling that allows bidirectional context learning (Yang et al., 2019). This architecture makes the model has a little bit similarity with BERT. The model also has outperformed BERT model in various tasks including sentiment analysis in the paper by Yang et al. (2019). The study by Banerjee et al. (2020) found that the XLNet model performed better on the Malayalam-English dataset compared to the Tamil-English (Tanglish) dataset, despite the latter having more samples. This difference in performance was attributed to the higher noise levels in the Tamil-English data, which affected the results negatively.

The potential of using XLM-R in code-switching sentiment analysis is the model is a multilingual masked language model like mBERT (Conneau et al., 2020). Next, the XLM-RoBERTa model was used in the study by Ou and Li (2020) along with the k-folding method on Malayalam-English and Tamil-English datasets. The findings showed that F-score of XLM-RoBERTa was 0.74 for Malayalam-English dataset and F-Score of 0.63 for Tamil-English dataset.

REFERENCES

- Aattouchi, I., Elmendili, S., & Elmendili, F. (2021). Sentiment analysis of health care. In *E3S Web of Conferences* (Vol. 319, p. 01064). EDP Sciences.
- Abdul Rahman, A. N. C., Abdullah, I. H., Zainudin, I. S., Tiun, S., & Jaludin, A. (2021). Domain-specific stop words in Malaysian parliamentary debates 1959-2018. *GEMA Online Journal of Language Studies*, 21(2).
- Agüero-Torales, M. M., Salas, J. I. A., & López-Herrera, A. G. (2021). Deep learning and multilingual sentiment analysis on social media data: An overview. *Applied Soft Computing*, 107, 107373.
- Aguilar, G., Kar, S., & Solorio, T. (2020). LinCE: A centralized benchmark for linguistic code-switching evaluation. *arXiv preprint arXiv:2005.04322*.
- Ahuja, R., Chug, A., Kohli, S., Gupta, S., & Ahuja, P. (2019). The impact of features extraction on the sentiment analysis. *Procedia Computer Science*, 152, 341-348.v
- Alamoodi, A. H., Baker, M. R., Albahri, O. S., Zaidan, B. B., Zaidan, A. A., Wong, W. K., ... & Baqer, M. J. (2022). Public sentiment analysis and topic modeling regarding COVID-19's three waves of total lockdown: A case study on movement control order in Malaysia. *KSII Transactions on Internet and Information Systems (TIIS)*, 16(7), 2169-2190.
- Aliyu, Y., Sarlan, A., Danyaro, K. U., Rahman, A. S. B., & Abdullahi, M. (2024). Sentiment analysis in low-resource settings: a comprehensive review of approaches, languages, and data sources. *IEEE Access*, 12, 66883-66909.
- Altawaier, M. M., & Tiun, S. (2016). Comparison of machine learning approaches on arabic twitter sentiment analysis. *International Journal on Advanced Science, Engineering*

and Information Technology, 6(6), 1067-1073.

- Aryal, S. K., Prioleau, H., & Washington, G. (2022). Sentiment classification of code-switched text using pre-trained multilingual embeddings and segmentation. *arXiv preprint arXiv:2210.16461*.
- Back, B. H., & Ha, I. K. (2019). Comparison of sentiment analysis from large Twitter datasets by Naïve Bayes and natural language processing methods. *Journal of Information and Communication Convergence Engineering*, 17(4), 239-245.
- Bahrin, U. F. M., & Jantan, H. (2024, August). Hybrid PSO-CNN model for cross-domain adaptation sentiment analysis. In *2024 5th International Conference on Big Data Analytics and Practices (IBDAP)* (pp. 156-160). IEEE.
- Banerjee, S., Jayapal, A., & Thavareesan, S. (2020). NUIG-Shubhanker@ Dravidian-CodeMix-FIRE2020: sentiment analysis of code-mixed dravidian text using XLNet. *arXiv preprint arXiv:2010.07773*.
- Behl, S., Rao, A., Aggarwal, S., Chadha, S., & Pannu, H. S. (2021). Twitter for disaster relief through sentiment analysis for COVID-19 and natural hazard crises. *International Journal of Disaster Risk Reduction*, 55, 102101.
- Bhagya, S. S., & Nadera, B. S. (2022, June). Analyzing performance of pre-trained models in detecting sentiments of code mixed Manglish Text Reviews from Social Media. In *2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS)* (pp. 1-6). IEEE.
- Biau, G., & Scornet, E. (2015). A random forest guided tour. *arXiv preprint arXiv:1511.05741*.
- Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media Inc.

- Birjali, M., Kasri, M., & Beni-Hssane, A. (2021). A comprehensive survey on sentiment analysis: Approaches, challenges and trends. *Knowledge-Based Systems*, 226, 107134.
- Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992, July). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory* (pp. 144-152).
- Breiman, L. (2001). Random forests. *Machine learning*, 45, 5-32.
- Chakravarthi, B. R., Jose, N., Suryawanshi, S., Sherly, E., & McCrae, J. P. (2020). A sentiment analysis dataset for code-mixed Malayalam-English. *arXiv preprint arXiv:2006.00210*.
- Chakravarthi, B. R., Muralidaran, V., Priyadharshini, R., & McCrae, J. P. (2020). Corpus creation for sentiment analysis in code-mixed Tamil-English text. *arXiv preprint arXiv:2006.00206*
- Chakravarthi, B. R., Priyadharshini, R., Muralidaran, V., Jose, N., Suryawanshi, S., Sherly, E., & McCrae, J. P. (2022). Dravidiancodemix: Sentiment analysis and offensive language identification dataset for dravidian languages in code-mixed text. *Language Resources and Evaluation*, 56(3), 765-806.
- 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.
- Chekima, K., & Alfred, R. (2016). An automatic construction of malay stop words based on aggregation method. In *Soft Computing in Data Science: Second International Conference, SCDS 2016, Kuala Lumpur, Malaysia, September 21-22, 2016, Proceedings 2* (pp. 180-189). Springer Singapore.

- Christen, P., Hand, D. J., & Kirielle, N. (2023). A review of the F-measure: its history, properties, criticism, and alternatives. *ACM Computing Surveys*, 56(3), 1-24.
- Chundi, R., Hulipalled, V. R., & Simha, J. B. (2020, October). SAEKCS: Sentiment analysis for English–Kannada code switchtext using deep learning techniques. In *2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)* (pp. 327-331). IEEE.
- Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 8440–8451). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2020.acl-main.747>
- Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20, 273-297.
- Das, R. K., Islam, M., Hasan, M. M., Razia, S., Hassan, M., & Khushbu, S. A. (2023). Sentiment analysis in multilingual context: Comparative analysis of machine learning and hybrid deep learning models. *Heliyon*, 9(9), e20281.
- Dang, N. C., Moreno-García, M. N., & De la Prieta, F. (2020). Sentiment analysis based on deep learning: A comparative study. *Electronics*, 9(3), 483.
- Deng, X., Bashlovkina, V., Han, F., Baumgartner, S., & Bendersky, M. (2023, April). Llms to the moon? reddit market sentiment analysis with large language models. *Companion Proceedings of the ACM Web Conference 2023*, 1014-1019. <https://doi.org/10.1145/3543873.3587605>
- 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.

- Douzas, G., Bacao, F., & Last, F. (2018). Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE. *Information Sciences*, 465, 1-20.
- Drus, Z., & Khalid, H. (2019). Sentiment analysis in social media and its application: Systematic literature review. *Procedia Computer Science*, 161, 707-714.
- Fan, W., & Gordon, M. D. (2014). The power of social media analytics. *Communications of the ACM*, 57(6), 74-81.
- FasterCapital. (2024, June 25). *Understanding Sentiment Analysis Metrics*. <https://fastercapital.com/content/Understanding-Sentiment-Analysis-Metrics.html#Types-of-Sentiment-Analysis-Metrics>
- Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, 27(8), 861-874.
- Firdaus, A. A., Yudhana, A., & Riadi, I. (2024). Indonesian presidential election sentiment: Dataset of response public before 2024. *Data in Brief*, 52, 109993.
- Fu, Y., Lin, N., Yang, Z., & Jiang, S. (2021). An open-source dataset and a multi-task model for malay named entity recognition. *arXiv preprint arXiv:2109.01293*.
- Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4), 193-202.
- Gallastegui, L. M. G., Forradellas, R. R., & Alonso, S. L. N. (2024). Applying advanced sentiment analysis for strategic marketing insights: A case study of BBVA using machine learning techniques. *Marketing*, 20(2), 100-115.
- García-Gonzalo, E., Fernández-Muñiz, Z., Garcia Nieto, P. J., Bernardo Sánchez, A., &

- Menéndez Fernández, M. (2016). Hard-rock stability analysis for span design in entry-type excavations with learning classifiers. *Materials*, 9(7), 531.
- Gholamalinezhad, H., & Khosravi, H. (2020). Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*.
- Girija, V. R., Sudha, T., & Cheriyan, R. (2023). Analysis of sentiments in low resource languages: Challenges and solutions. In *2023 IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE)* (pp. 1–6). IEEE. <https://doi.org/10.1109/RASSE60029.2023.10363469>
- Google. (2016, June 24). *Compact Language Detector v3 (CLD3)* [Computer software]. GitHub. <https://github.com/google/cld3>
- Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*.
- Graves, A., & Schmidhuber, J. (2005, July). Framewise phoneme classification with bidirectional LSTM networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005*. (Vol. 4, pp. 2047-2052). IEEE.
- Han, H., Wang, W. Y., & Mao, B. H. (2005, August). Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing* (pp. 878-887). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Hassan, A., Abbasi, A., & Zeng, D. (2013, September). Twitter sentiment analysis: A bootstrap ensemble framework. In *2013 International Conference on Social Computing* (pp. 357-364). IEEE.
- He, H., Bai, Y., Garcia, E. A., & Li, S. (2008, June). 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.
- Hegde, A., Anusha, M. D., Coelho, S., Shashirekha, H. L., & Chakravarthi, B. R. (2022, June). Corpus creation for sentiment analysis in code-mixed Tulu text. In *Proceedings of the 1st annual meeting of the ELRA/ISCA special interest group on under-resourced languages* (pp. 33-40).
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.
- Hutto, C., & Gilbert, E. (2014, May). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 8, No. 1, pp. 216-225).
- IBM. (2024, December 19). *What Is Random Forest?* | IBM. <https://www.ibm.com/cloud/learn/random-forest>
- Invento, A. C. B., Lerias, T. J., & Ceniza, A. M. (2017, December). Sentiment analysis on the impact of K-12 program in the Philippines using Naive Bayes and lexicon approach with code switching. In *Proceedings of the 2017 International Conference on Information Technology* (pp. 103-106).
- Iosifidis, V., & Ntoutsi, E. (2020). Sentiment analysis on big sparse data streams with limited labels. *Knowledge and Information Systems*, 62(4), 1393-1432.s
- Ismail, M. I., & Abdullah, N. E. (2023). An analysis of code-switching in Hitz FM Instagram comments. *e-BANGI Journal*, 20(4).
- Ismail, N. H., Liu, N., Du, M., He, Z., & Hu, X. (2020). A deep learning approach for identifying cancer survivors living with post-traumatic stress disorder on Twitter. *BMC Medical Informatics and Decision Making*, 20, 1-11.

- Jakkula, V. (2006). Tutorial on support vector machine (SVM). *School of EECS, Washington State University*, 37(2.5), 3.
- Jamatia, A., Swamy, S. D., Gambäck, B., Das, A., & Debbarma, S. (2020). Deep learning based sentiment analysis in a code-mixed English-Hindi and English-Bengali social media corpus. *International Journal on Artificial Intelligence Tools*, 29(05), 2050014.
- Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.
- Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., & Mikolov, T. (2016). Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.
- Juan, S. (2015). *Exploiting resources from closely-related languages for automatic speech recognition in low-resource languages from Malaysia* [Doctoral dissertation, Université Grenoble Alpes]. HAL Thèses. <https://theses.hal.science/tel-01314120v1>
- Joblib Developers. (2021). *Joblib: running Python functions as pipeline jobs*. <https://joblib.readthedocs.io/en/stable/>
- Kabir, M. M., Othman, Z. A., Yaakub, M. R., & Tiun, S. (2023). Hybrid syntax dependency with lexicon and logistic regression for aspect-based sentiment analysis. *International Journal of Advanced Computer Science and Applications*, 14(10), 797-804.
- Kasmuri, E., & Basiron, H. (2019). Building a Malay-English code-switching subjectivity corpus for sentiment analysis. *Int. J. Advance Soft Compu. Appl*, 11(1), 112-130.
- Kastrati, Z., Ahmedi, L., Kurti, A., Kadriu, F., Murtezaj, D., & Gashi, F. (2021). A deep learning sentiment analyser for social media comments in low-resource languages.

Electronics, 10(10), 1133.

Keras Team. (n.d.). *Keras documentation: About Keras 3*. <https://keras.io/about/>

Khalil, S. M., & Firda, M. S. Z. M. S. (2018). Inter-sentential and intra-sentential code switching in parliamentary debate. *International Journal of Modern Languages And Applied Linguistics*, 2(4), 29-35.

Khan, M. Y., Qayoom, A., Nizami, M. S., Siddiqui, M. S., Wasi, S., & Raazi, S. M. K. U. R. (2021). Automated prediction of Good Dictionary EXamples (GDEX): a comprehensive experiment with distant supervision, machine learning, and word embedding-based deep learning techniques. *Complexity*, 2021, 1-18.

Khanuja, S., Dandapat, S., Srinivasan, A., Sitaram, S., & Choudhury, M. (2020). GLUECoS: An evaluation benchmark for code-switched NLP. *arXiv preprint arXiv:2004.12376*.

Kit, Y. H., & Mokji, M. M. (2024). Language threshold for multilingual sentiment analysis system. *ELEKTRIKA-Journal of Electrical Engineering*, 23(1), 96-102.

Kong, J. T., Juwono, F. H., Ngu, I. Y., Nugraha, I. G. D., Maraden, Y., & Wong, W. K. (2023). A mixed Malay–English language COVID-19 Twitter dataset: A sentiment analysis. *Big Data and Cognitive Computing*, 7(2), 61.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25.

Kundu, R. (2023, May 11). *Precision vs. Recall: Differences, Use Cases & Evaluation*. V7. <https://www.v7labs.com/blog/precision-vs-recall-guide>

Kusampudi, S. S. V., Sathineni, P., & Mamidi, R. (2021, September). Sentiment analysis in code-mixed Telugu-English text with unsupervised data normalization.

- In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)* (pp. 753-760).
- Lal, Y. K., Kumar, V., Dhar, M., Shrivastava, M., & Koehn, P. (2019, July). De-mixing sentiment from code-mixed text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop* (pp. 371-377).
- Lawrence, S. J. (2023, April 18). What is LSTM? - Introduction to Long Short-Term Memory - Scaler Topics. *Scaler Topics*. <https://www.scaler.com/topics/deep-learning/lstm/>
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.
- Lemaitre G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17), 1-5.
- Lim, H. T., Huspi, S. H., & Ibrahim, R. (2021, July). A conceptual framework for Malay-English mixed-language question answering system. In *2021 International Congress of Advanced Technology and Engineering (ICOTEN)* (pp. 1-8). IEEE.
- Lin, A., Wu, H., Liang, G., Cardenas-Tristan, A., Wu, X., Zhao, C., & Li, D. (2020). A big data-driven dynamic estimation model of relief supplies demand in urban flood disaster. *International Journal of Disaster Risk Reduction*, 49, 101682.
- Liu, B. (2015). *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press.
- Liu, S. (2020). Sentiment analysis of Yelp reviews: A comparison of techniques and models. *arXiv preprint arXiv:2004.13851*.

- Loria, S. (2024, February 15). *TextBlob: Simplified Text Processing — TextBlob 0.18.0.post0 documentation*. <https://textblob.readthedocs.io/en/dev>
- Lubis, I. S., Surya, S., & Muka, A. U. (2017). The use of code-switching among the late adolescents in social media Facebook. *CaLLs (Journal of Culture, Arts, Literature, and Linguistics)*, 3(2), 83-96.
- Lui, M., & Baldwin, T. (2012, July). langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations* (pp. 25-30).
- Maity, K., Kumar, A., & Saha, S. (2022). A multitask multimodal framework for sentiment and emotion-aided cyberbullying detection. *IEEE Internet Computing*, 26(4), 68-78.
- Mammadov, A., & Bakal, G. (2024, October). From traditional to deep: Evaluating sentiment analysis models on a large-scale tweet dataset. In *2024 9th International Conference on Computer Science and Engineering (UBMK)* (pp. 1-6). IEEE.
- Mandal, S., Mahata, S. K., & Das, D. (2018). Preparing bengali-english code-mixed corpus for sentiment analysis of indian languages. *arXiv preprint arXiv:1803.04000*.
- Mao, Y., Liu, Q., & Zhang, Y. (2024). Sentiment analysis methods, applications, and challenges: A systematic literature review. *Journal of King Saud University-Computer and Information Sciences*, 102048.
- Maskat, R., Azman, N. A., Nulizairos, N. S. S., Zahidin, N. A., Mahadi, A. H., Norshamsul, S. R., ... & Mahdin, H. (2024). A bi-annotated Malay-English code-switching (Manglish) dataset of X posts for biological gender identification and authorship attribution. *Data in Brief*, 52, 110034.
- Matrane, Y., Benabbou, F., & Banou, Z. (2024). WeVoTe: A weighted voting technique for automatic sentiment annotation of Moroccan dialect comments. *IEEE Access*, 12, 16276-16298.

- Mave, D., Maharjan, S., & Solorio, T. (2018, July). Language identification and analysis of code-switched social media text. In *Proceedings of the third workshop on computational approaches to linguistic code-switching* (pp. 51-61).
- Mohammed, A., & Kora, R. (2023). A comprehensive review on ensemble deep learning: Opportunities and challenges. *Journal of King Saud University-Computer and Information Sciences*, 35(2), 757-774.
- Molina, G., AlGhamdi, F., Ghoneim, M., Hawwari, A., Rey-Villamizar, N., Diab, M., & Solorio, T. (2019). Overview for the second shared task on language identification in code-switched data. *arXiv preprint arXiv:1909.13016*.
- Mudinas, A., Zhang, D., & Levene, M. (2018). Bootstrap domain-specific sentiment classifiers from unlabeled corpora. *Transactions of the Association for Computational Linguistics*, 6, 269-285.
- Mujahid, M., Kina, E. R. O. L., Rustam, F., Villar, M. G., Alvarado, E. S., De La Torre Diez, I., & Ashraf, I. (2024). Data oversampling and imbalanced datasets: an investigation of performance for machine learning and feature engineering. *Journal of Big Data*, 11(1), 87.
- Nguyen, H. M., Cooper, E. W., & Kamei, K. (2011). Borderline over-sampling for imbalanced data classification. *International Journal of Knowledge Engineering and Soft Data Paradigms*, 3(1), 4-21.
- NumPy Developers. (2024). *NumPy documentation - Version 2.1 (stable)*. <https://numpy.org/doc/stable/index.html>
- O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.
- Ou, X., & Li, H. (2020, December). YNU@ Dravidian-CodeMix-FIRE2020: XLM-

- RoBERTa for Multi-language Sentiment Analysis. In *FIRE (Working Notes)* (pp. 560-565).
- Pandas. (2024, September 20). pandas documentation — pandas 2.2.1 documentation. <https://pandas.pydata.org/docs/index.html>
- Patwa, P., Aguilar, G., Kar, S., Pandey, S., Pykl, S., Gambäck, B., Chakraborty, T., Solorio, T., & Das, A. (2020). Semeval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. *arXiv preprint arXiv:2008.04277*.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12, 2825-2830.
- Pérez, J. M., Rajngewerc, M., Giudici, J. C., Furman, D. A., Luque, F., Alemany, L. A., & Martínez, M. V. (2021). pysentimiento: a python toolkit for opinion mining and social NLP tasks. *arXiv preprint arXiv:2106.09462*.
- Phung, V. H., & Rhee, E. J. (2018). A deep learning approach for classification of cloud image patches on small datasets. *Journal of Information and Communication Convergence Engineering*, 16(3), 173-178.
- Poplack, S. (1980). Sometimes i'll start a sentence in spanish y termino en espanol: toward a typology of code-switching¹.
- Poplack, S. (2015). Code Switching: Linguistic. In J. D. Wright (Ed.), In *International Encyclopedia of the Social and Behavioral Sciences* (2nd ed, pp. 918-925). Oxford: Elsevier. <https://doi.org/10.1016/B978-0-08-097086-8.53004-9>
- Poria, S., Hazarika, D., Majumder, N., & Mihalcea, R. (2020). Beneath the tip of the iceberg: Current challenges and new directions in sentiment analysis research. *IEEE*

- Transactions on Affective Computing*, 14(1), 108-132.
- Pravalika, A., Oza, V., Meghana, N. P., & Kamath, S. S. (2017, July). Domain-specific sentiment analysis approaches for code-mixed social network data. In *2017, the 8th International Conference on Computing, communication and Networking Technologies (ICCCNT)* (pp. 1-6). IEEE.
- Python Software Foundation. (2024). *csv — CSV File Reading and Writing*. Python Documentation. <https://docs.python.org/3/library/csv.html>
- Ragini, J. R., Anand, P. R., & Bhaskar, V. (2018). Big data analytics for disaster response and recovery through sentiment analysis. *International Journal of Information Management*, 42, 13-24.
- Rahmanti, A. R., Chien, C. H., Nursetyo, A. A., Husnayain, A., Wiratama, B. S., Fuad, A., Yang, H. C., & Li, Y. C. J. (2022). Social media sentiment analysis to monitor the performance of vaccination coverage during the early phase of the national COVID-19 vaccine rollout. *Computer Methods and Programs in Biomedicine*, 221, 106838.
- Ranjan, S., Mekala, D., & Shang, J. (2022, December). Progressive Sentiment Analysis for Code-Switched Text Data. In *Findings of the Association for Computational Linguistics: EMNLP 2022* (pp. 1155-1167).
- Raschka, S. (2014). Naive bayes and text classification i-introduction and theory. *arXiv preprint arXiv:1410.5329*.
- Raveendrarasa, V., & Amalraj, C. R. J. (2020, December). Sentiment analysis of Tamil-English code-switched text on social media using sub-word level LSTM. In *2020 5th International Conference on Information Technology Research (ICITR)* (pp. 1-5). IEEE.
- Ren, Q., Cheng, H., & Han, H. (2017, March). Research on machine learning framework

- based on random forest algorithm. In *AIP Conference Proceedings* (Vol. 1820, No. 1). AIP Publishing.
- Ren, Y., Kaji, N., Yoshinaga, N., & Kitsuregawa, M. (2014). Sentiment classification in under-resourced languages using graph-based semi-supervised learning methods. *IEICE TRANSACTIONS on Information and Systems*, 97(4), 790-797.
- Roesslein, J. (2023, April 24). *Tweepy Documentation — Tweepy 4.14.0 documentation*. Tweepy. <https://docs.tweepy.org/en/stable/>
- Romadhona, N. P., Lu, S. E., Lu, B. H., & Tsai, R. T. H. (2022, October). BRCC and SentiBahasaRojak: The first Bahasa Rojak corpus for pretraining and sentiment analysis dataset. In *Proceedings of the 29th International Conference on Computational Linguistics* (pp. 4418-4428).
- Rosenthal, S., Farra, N., & Nakov, P. (2019). SemEval-2017 task 4: Sentiment analysis in Twitter. *arXiv preprint arXiv:1912.00741*.
- Roslan, A. N. M., Mahmud, M. M., & Ismail, O. (2021). Why code-switch on WhatsApp? A quantitative analysis of types and influences of code-switching. *Asian Social Science*, 17(10), 43-52.
- Routledge, R. (2018, February 7). *Bayes's theorem*. Encyclopedia Britannica. <https://www.britannica.com/topic/Bayess-theorem>
- Sabri, N., Edalat, A., & Bahrak, B. (2021, March). Sentiment analysis of persian-english code-mixed texts. In *2021 26th International Computer Conference, Computer Society of Iran (CSICC)* (pp. 1-4). IEEE.
- Sabty, C., Mesabah, I., Çetinoğlu, Ö., & Abdennadher, S. (2021). Language identification of intra-word code-switching for arabic–english. *Array*, 12, 100104.
- Samah, K. A. F. A., Jailani, N. S., Hamzah, R., Aminuddin, R., Abidin, N. A. Z., & Riza, L.

- S. (2024). Aspect-based classification and visualization of twitter sentiment analysis towards online food delivery services in Malaysia. *Journal of Advanced Research in Applied Sciences and Engineering Technology*, 37(1), 139-150.
- Sarkar, P. (2024, January 19). *Naive Bayes in Machine Learning [Examples, Models, Types]*. KnowledgeHut. <https://www.knowledgehut.com/blog/data-science/naive-bayes-in-machine-learning#what-is-na%C3%AFve-bayes?%C2%A0>
- Serip Mohamad, N. H., & Shafie, H. (2023). ‘Know know married’: Playfulness of Manglish in social media platforms. *Asian Englishes*, 26(1), 140–155. <https://doi.org/10.1080/13488678.2023.2216862>
- Shalini, K., Ravikurnar, A., Reddy, A., & Soman, K. P. (2018, January). Sentiment analysis of Indian languages using convolutional neural networks. In *2018 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1-4). IEEE.
- Shalini, K., Ganesh, H. B., Kumar, M. A., & Soman, K. P. (2018, September). Sentiment analysis for code-mixed Indian social media text with distributed representation. In *2018 International Conference on Advances in Computing, communications and Informatics (ICACCI)* (pp. 1126–1131). IEEE.
- Singh, M., Goyal, V., & Raj, S. (2019, November). Sentiment analysis of English-Punjabi code mixed social media content for agriculture domain. In *2019 4th International Conference on Information Systems and Computer Networks (ISCON)* (pp. 352-357). IEEE.
- Singh, O. M., Timilsina, S., Bal, B. K., & Joshi, A. (2020, December). Aspect based abusive sentiment detection in Nepali social media texts. In *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)* (pp. 301-308). IEEE.

- Shailendra, A., Bengani, C., Kumari, K. S., Senthilraja, P., Prithivi, A., & Ramesh, S. (2023). Suspicious activity detection based on audio detecting methodology using deep learning. *International Conference on Recent Trends in Data Science and Its Applications*, 683–687. <https://doi.org/10.13052/rp-9788770040723.131>
- Shamsuddin, A. M., Juan, S. S., Chua, S., & Bramantoro, A. (2024). Semi-automatic sentiment identification for Malay-English code-switched data. *Journal of Advanced Research Design*, 123(1), 198-212.
- Shanmugavadivel, K., Sampath, S. H., Nandhakumar, P., Mahalingam, P., Subramanian, M., Kumaresan, P. K., & Priyadharshini, R. (2022). An analysis of machine learning models for sentiment analysis of Tamil code-mixed data. *Computer Speech & Language*, 76, 101407.
- Shirazi, G. G., Azmi, R., & Shakibian, H. (2023, May). A semi-automated labeled data generation approach based on deep learning to improve sentiment analysis in the Persian language. In *2023 9th International Conference on Web Research (ICWR)* (pp. 242-246). IEEE.
- Solorio, T., Blair, E., Maharjan, S., Bethard, S., Diab, M., Ghoneim, M., Hawwari, A., AlGhamdi, F., Hirschberg, J., Chang, A., & Fung, P. (2014, October). Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching* (pp. 62-72).
- SR, M. K., Kumar, L., & Malapati, A. (2022, May). Sentiment Analysis on code-switched Dravidian languages with kernel based extreme learning machines. In *Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages* (pp. 184-190).
- Srinivasan, R., & Subalalitha, C. N. (2023). Sentimental analysis from imbalanced code-

- mixed data using machine learning approaches. *Distributed and Parallel Databases*, 41(1), 37-52.
- Srivastava, V., & Singh, M. (2020, December). IIT Gandhinagar at SemEval-2020 Task 9: Code-mixed sentiment classification using candidate sentence generation and selection. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation* (pp. 1259-1264).
- Srivastava, V., & Singh, M. (2021, June). Challenges and limitations with the metrics measuring the complexity of code-mixed text. In *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching* (pp. 6-14).
- Stahl, P. M. (2023, March 2). *Lingua-Py* [Computer software]. GitHub. <https://github.com/pemistahl/lingua-py>
- Stopwords-ISO. (2016, October 8). Stopwords Malay (MS). GitHub. <https://github.com/stopwords-iso/stopwords-ms>
- Suandi, F., Anam, M. K., Firdaus, M. B., Fadli, S., Lathifah, L., Yumami, E., Saleh, A., & Hasibuan, A. Z. (2024, December). Enhancing sentiment analysis performance using SMOTE and majority voting in machine learning algorithms. In *Proceedings of the 7th International Conference on Applied Engineering (ICAE 2024)* (pp. 126-138). Atlantis Press.
- Suhaimin, M. S. M., Hijazi, M. H. A., & Moun, E. G. (2024). Annotated dataset for sentiment analysis and sarcasm detection: Bilingual code-mixed English-Malay social media data in the public security domain. *Data in Brief*, 55, 110663.
- Suresh, S. K., Surana, T., Hao, L. Z., & Chng, E. S. (2025). CS-Sum: A benchmark for code-switching dialogue summarization and the limits of large language models. *arXiv preprint arXiv:2505.13559*.

- Taufek, T. E., Nor, N. F. M., Jaludin, A., Tiun, S., & Choy, L. K. (2021). Public perceptions on climate change: A sentiment analysis approach. *GEMA Online Journal of Language Studies*, 21(4).
- TensorFlow. (2023). *Tutorials*. <https://www.tensorflow.org/tutorials>
- Thara, S., & Poornachandran, P. (2022). Social media text analytics of Malayalam–English code-mixed using deep learning. *Journal of Big Data*, 9(1), 45.
- The Pennsylvania State University. (n.d.). *11.2.1 - Bootstrapping Methods | STAT 500*. PennState: Statistics Online Courses. Retrieved September 1, 2024, from <https://online.stat.psu.edu/stat500/lesson/11/11.2/11.2.1>
- Tho, C., Heryadi, Y., Kartowisastro, I. H., & Budiharto, W. (2021, October). A comparison of lexicon-based and transformer-based sentiment analysis on code-mixed of low-resource languages. In *2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI)* (Vol. 1, pp. 81-85). IEEE.
- Ou, X., & Li, H. (2020, December). YNU@ Dravidian-CodeMix-FIRE2020: XLM-RoBERTa for multi-language sentiment analysis. In *FIRE (Working Notes)* (pp. 560-565).
- Utomo, H. R., & Romadhony, A. (2023, April). Sentiment analysis on Indonesia-English code-mixed data. In *2023 IEEE 8th International Conference for Convergence in Technology (I2CT)* (pp. 1-6). IEEE.
- Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, 53(8), 5929-5955.
- Van Kooten, P. (2022, November 15). *contractions*. GitHub. <https://github.com/kootenpv/contractions>
- Vilares, D., Alonso, M. A., & Gómez-Rodríguez, C. (2016, May). En-es-es: An english-

- spanish code-switching twitter corpus for multilingual sentiment analysis. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)* (pp. 4149-4153).
- Wang, Q., Zhang, W., Cai, H., & Cao, Y. (2020). Understanding the perceptions of Chinese women of the commercially available domestic and imported HPV vaccine: A semantic network analysis. *Vaccine*, 38(52), 8334-8342.
- Wang, L., & Carvalho, L. (2024). Multiclass ROC. *arXiv preprint arXiv:2404.13147*.
- Xie, S., & Wang, T. (2013, December). Dividing for combination: a bootstrapping sentiment classification framework for micro-blogs. In *2013 International Conference on Information Science and Cloud Computing* (pp. 78-84). IEEE.
- Yadav, K., Lamba, A., Gupta, D., Gupta, A., Karmakar, P., & Saini, S. (2020, December). Bi-LSTM and ensemble based bilingual sentiment analysis for a code-mixed Hindi-English social media Text. In *2020 IEEE 17th India Council International Conference (INDICON)* (pp. 1-6). IEEE.
- Yadav, A., & Vishwakarma, D. K. (2020). Sentiment analysis using deep learning architectures: a review. *Artificial Intelligence Review*, 53(6), 4335-4385.
- Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4), 611-629.
- Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237*.
- Yann, L. M., Zahri, N. A. H., Amir, A., Romli, R., Ghazali, N. H., Anwar, S. A., & Hashim, N. M. Z. (2024, February). Sentiment analysis on Malay-English mixed language

- text using artificial neural network. In *AIP Conference Proceedings* (Vol. 2898, No. 1). AIP Publishing.
- Younas, A., Nasim, R., Ali, S., Wang, G., & Qi, F. (2020, December). Sentiment analysis of code-mixed Roman Urdu-English social media text using deep learning approaches. In *2020 IEEE 23rd International Conference on Computational Science and Engineering (CSE)* (pp. 66-71). IEEE.
- Zabha, N. I., Ayop, Z., Anawar, S., Hamid, E., & Abidin, Z. Z. (2019). Developing cross-lingual sentiment analysis of Malay Twitter data using lexicon-based approach. *International Journal of Advanced Computer Science and Applications*, 10(1), 346-351.
- Zhou, Z. H. (2012). *Ensemble methods: Foundations and algorithms* (1st ed.). Chapman and Hall/CRC. <https://doi.org/10.1201/b12207>
- Zirker, Kelly Ann Hill. *Intrasentential vs. intersentential code switching in early and late bilinguals* [Master's thesis, Brigham Young University]. BYU Scholars Archive. <https://scholarsarchive.byu.edu/etd/927>
- Zolkepli, H. (2018). *Malaya: Natural-Language-Toolkit library for Bahasa Malaysia, powered by PyTorch* [Computer software]. GitHub. <https://github.com/mesolitica/malaya>

APPENDICES

Appendix A: Table for chosen articles for the systematic review

Author(s)	Year	Title	Techniques	Proposed approach	Language	Dataset(s)	Data Source	Result(s)
Invento, A. C. B., Lerias, T. J., & Ceniza, A. M.	2017	Sentiment Analysis on the Impact of K-12 Program in the Philippines using Naïve Bayes and Lexicon Approach with Code Switching	Multinomial Naïve-Bayes, Count Vectorizer	Multinomial Naïve-Bayes, Count Vectorizer	English, Cebuano (M)	Own	Facebook, Twitter, News outlets	Accuracy: 0.7226753 Avg precision: 0.73 Avg recall: 0.72 Avg F1-score: 0.71
Pravalika, A., Oza, V., Meghana, N. P., & Kamath, S. S.	2017	Domain-specific sentiment analysis approaches for code-mixed social network	Dictionary, Naïve Bayes, Multilayer Perceptron, Random Tree, Decision Tree	SVM, Naïve Bayes, Multilayer Perceptron, Random	English, Hindi (CS)	Own	Facebook	<u>Lexicon-based approach</u> <u>(All classes)</u> Correct: 176 Incorrect: 14 Total: 200

	data		Tree, Decision Tree						<u>Machine learning classifier (Precision)</u> Naïve Bayes: 0.725 Multilayer Perceptron: 0.695 Random Tree: 0.752 SVM: 0.718 Decision Tree: 0.701
Shalini, K., 2018 Ravikurnar, A., Reddy, A., & Soman, K. P.	Sentiment analysis of Indian languages using convolutional neural networks	Convolutional Neural Networks (CNN)	Use CNN for Bengali-English and Telugu sentiment analysis	English, Bengali, (CS), Telugu	SAIL @ Twitter, Online movie reviews ICON 2017 (Patra et al., 2018), Own				<u>Bengali-English (Overall)</u> Precision: 0.342 Recall: 0.340 F-score: 0.318
Shalini, K., 2018 Ganesh, H.	Sentiment analysis for	Doc2Vec, FastText,	Doing sentiment	English, Bengali,	Own, SAIL @ Twitter ICON 2017		Facebook, Twitter		<u>Accuracy (%)</u> <u>Doc2Vec (best)</u>

B., Kumar, M. A., & Soman, K. P.	code-mixed Indian social media text with distributed representation	BiLSTM, CNN	analysis on Hindi, code-mixed Kannada Indian languages social media corpus	(Patra et al., 2018)	Window size = 3 : EN – HI: 46.19 (D= 300) EN- BE: 41.00 (D= 100) EN – KA: 45.29 (D= 400) Window size = 5 : EN – HI: 46.27 (D= 200) EN- BE: 41.60 (D= 100) EN – KA: 44.86 (D= 400) <u>FastText (best)</u> Window size = 3 : EN – HI: 58.10 (D= 500) EN- BE: 69.80 (D= 100) EN – KA: 54.64
--	---	-------------	---	-------------------------	---

(D= 300)

Window size = 5 :

EN – HI: 58.87

(D= 300)

EN- BE: 70.20 (D= 100)

EN – KA: 54.71

(D= 100)

BiLSTM (best)

EN-HI: 60.22

EN-BE: 72.20

EN-KA: 69.24

CNN (best)

EN-HI: 59.56

EN-BE: 71.19

EN-KA: 71.50

Singh, M., 2019	Sentiment analysis of English-Punjabi code mixed	Support Vector Machine (SVM), Naïve-Bayes (NB)	Use and Bayes sentiment	SVM English, Own Punjabi (CS)	Facebook, Twitter, YouTube	Unigram: SVM: 80% NB: 82%
-----------------	--	--	-------------------------	-------------------------------	----------------------------	---------------------------

social media analysis
content for
agriculture
domain

Bigram:
SVM: 83%
NB: 85%

Trigram:
SVM: 85%
NB: +_ 86%

Lal, Y. K., 2019	De-mixing	CMSA, SVM, Propose	English, Joshi et Facebook	<u>Accuracy (%)</u>
Kumar, V.,	sentiment from	NBSVM, hybrid	Hindi al. (2016)	CMSA: 83.54
Dhar, M.,	code-mixed text	Multinomial Naive approach		SVM: 52.96
Shrivastava,		Bayes (MNB), (CMSA) to		NBSVM: 62.5
M., & Koehn,		Lexicon Lookup, get sentiment		MNB
P.		Char-LSTM, from code-		(Uni+Bigram):
		Subword LSTM, mixed text.		66.36
		FastText, SACMT		MNB (TF-IDF):
				63.53
				Lexicon Lookup:
				51.15
				Char-LSTM: 59.8
				Subword LSTM:

								69.7
								FastText: 46.39
								SACMT: 77.3
Aguilar, G., 2020	LinCE: A	BiLSTM, ELMo,	Centralized	Spanish,	SemEval-	Twitter		<u>Accuracy</u>
Kar, S., &	centralized	ML-BERT	benchmark	English	2020			BiLSTM: 45.39
Solorio, T.	benchmark for		for code-	(CS)	Task 9			ELMo: 49.97
	linguistic code-		switching		(Patwa et			ML-BERT: 60.20
	switching		data for		al., 2020)			
	evaluation.		several NLP					
			tasks					
Chundi, R., 2020	SAEKCS:	Naïve-Bayes,	SAEKCS	English,	Own	YouTube		Accuracy
Hulipalled,	Sentiment	BiLSTM,	hybrid model	Kannada				NB: 39.6%
V. R., &	analysis for	Subword-LSTM	(Sub word	(CS)				BiLSTM: 55.9%
Simha, J. B.	English–		level					Subword-LSTM:
	Kannada code		embedding					64.8%
	switch text using		approach,					SAEKCS: 77.6%
	deep learning		CNN,BiLST					
	techniques		M)					
Jamatia, A., 2020	Deep learning	Naïve Bayes (NB),	A study of	English,	Joshi et	Facebook,		<u>ICON2017 EN-BN</u>

Swamy, S. D., Gambäck, B., Das, A., & Debbarma, S.	based sentiment analysis in a code-mixed English-Hindi and English- Bengali social media corpus.	Sequential Minimal Optimization (SMO), Random Forest (RF), BiLSTM, CNN, Double BiLSTM, Attention-based Bidirectional RNN, GLoVe, BERT	sentiment analysis using deep learning models for code-mixed English- Hindi and English- Bengali datasets	Hindi, Bengali (CS)	al. (2016), Twitter Patra et al. (2018)	(Accuracy) NB: 42.6 SMO: 44.2 RF: 44.2 BiLSTM-CNN: 65.2 Double BiLSTM: 61.5 Attention Model: 62.7 GLoVe BiLSTM- CNN: 59.0 GLoVe Double BiLSTM: 67.1 GLoVe Attention Model: 62.9 BERT _(base, uncased) : 66.2 BERT _(base, domain, uncased) : 63.3
---	--	---	---	---------------------------	---	---

ICON2017 EN-HI

(Accuracy)

NB: 46.8

SMO: 49.1

RF: 48.9

BiLSTM-CNN:

57.6

Double BiLSTM:

57.9

Attention Model:

58.2

GLoVe BiLSTM-

CNN: 57.4

GLoVe Double

BiLSTM: 60.1

GLoVe Attention

Model: 59.8

BERT_(base, uncased):

61.0

BERT_{(base, domain,}

uncased): 59.1

Joshi et al. (2016)

EN-HI (Accuracy)

BiLSTM-CNN:

68.6

Double BiLSTM:

60.6

Attention Model:

71.9

GLoVe BiLSTM-

CNN: 65.2

GLoVe Double

BiLSTM: 69.1

GLoVe Attention

Model: 67.9

BERT_(base, uncased):

67.4

BERT_{(base, domain,}

uncased): 63.3

Khanuja, S., 2020	GLUECoS: An	MUSE, BiSkip,	Evaluation	English,	Vilares et	Twitter	<u>English-Hindi (F-1</u>
Dandapat, S.,	evaluation	BiCVM, GCM,	benchmark	Hindi,	al. (2016),		<u>Score)</u>
Srinivasan,	benchmark for	mBERT, modified	for NLP tasks	Spanish	SAIL @		Baseline: 50.44
A., Sitaram,	code-switched	mBERT	using code-		ICON		Unsupervised
S., &	NLP.		switching		2017		MUSE: 48.37
Choudhury,			data.		(Patra et		Supervised MUSE:
M. (2020).					al., 2018)		51.27
arXiv							BiSkip: 48.84
preprint							BiCVM: 49.56
arXiv:2004.1							GCM: 50.01
2376.							mBERT: 58.24
							Modified mBERT:
							59.35
							<u>English-Spanish</u>
							<u>(F-1 Score)</u>
							Baseline: 50.62
							Unsupervised
							MUSE: 58.73

										Supervised MUSE: 58.44 BiSkip: 60.4 BiCVM: 62.62 GCM: 62.89 mBERT: 66.03 Modified mBERT: 69.31	
Singh, O. M., 2020	Aspect based	BERT, BiLSTM,	Aspect-based	Nepali,	Own	YouTube	BERT	(not concatenated)	Accuracy: 0.789		
Timilsina, S.,	abusive	CNN. SVM	abusive	English							
Bal, B. K., &	sentiment		sentiment	(CS)							
Joshi, A.	detection in		detection								
	Nepali social						BiLSTM	(not concatenated)	Accuracy: 0.805		
	media texts										
							CNN	(not concatenated)	Accuracy: 0.787		

SVM (not
concatenated)
Accuracy: 0.684

BERT
(concatenated)
Accuracy: 0.800

BiLSTM
(concatenated)
Accuracy: 0.815

CNN
(concatenated)
Accuracy: 0.811

SVM
(concatenated)
Accuracy: 0.714

Raveendirara	2020	Sentiment	Subword	LSTM,	Sub-Word	Tamil,	Own	Facebook	Subword-LSTM:
--------------	------	-----------	---------	-------	----------	--------	-----	----------	---------------

sa, V., & Amalraj, C. R. J.	Analysis of Lexicon Lookup, LSTM Tamil-English SVM, Multinomial Code-Switched Naive Bayes, TF- Text on Social IDF, NBSVM Media Using Sub-Word Level LSTM	English (CS)	0.728 Lexicon lookup: 0.252 SVM(Uni+Bigram) : 0.5262 SVM (Unigram): 0.5232 MNB(TF-IDF): 0.4783 MNB(Uni+Bigram) : 0.6046 MNB (Unigram): 0.6143 NBSVM(Uni+Big ram): 0.5375 NBSVM (Unigram): 0.5335
Yadav, K., 2020 Lamba, A.,	Bi-LSTM and Ensembled ensemble based classifier, Bi-	Ensembled Hindi, Own, classifier, Bi- English IITH	Facebook, Average accuracy: social Multinomial Naïve

Gupta, D., Gupta, A., Karmakar, P., & Saini, S.	bilingual sentiment analysis for a code-mixed Hindi-English social media text	LSTM, Multinomial Naïve Bayes, Stochastic Gradient Descent, Xtreme Gradient Boosting (XGB), LSTM	LSTM (CS)	Codemix media ed sites, Sentiment interview Dataset transcripts (Joshi et al., 2016)	Bayes: 0.71 Support Vector Classifier: 0.72 Stochastic Gradient Descent: 0.72 Xtreme Gradient Boosting: 0.59 LSTM: 0.59 Ensembled classifier: 0.74 Bi-LSTM:0.73
Younas, A., 2020 Nasim, R., Ali, S., Wang, G., & Qi, F.	Sentiment Analysis of Code-Mixed Roman Urdu- English Social Media Text using Deep Learning	BERT, XLM-R	mBERT, XLM- RoBERTa	Urdu, English (CS)	Own Twitter mBERT (Without fine-tuning of parameters) Accuracy: 0.65 XLM-R (Without fine-tuning of parameters)

Approaches									Accuracy: 0.49
									mBERT (With fine-tuning of parameters)
									Accuracy: 0.69
									XLM-R (With fine-tuning of parameters)
									Accuracy: 0.71
Sabri, N., 2021	Sentiment	BERT, Bi-LSTM,	Multilingual	Persian,	Own	Twitter	Proposed model:		
Edalat, A., &	Analysis of	Naïve Bayes,	BERT	English			66.17		
Bahrak, B.	Persian-English	Random Forest	embedding,	(CS)			Naïve Bayes:		
	Code-mixed		Bi-LSTM				60.88		
	Texts		ensemble				Random Forest:		
			model				62.12		
Tho, C., 2021	A Comparison	Sentiwordnet,	Sentiwordnet	Indonesia	Own	Twitter	Accuracy:		
Heryadi, Y.,	of Lexicon-	VADER, BERT	, VADER, n,				Sentence-BERT:		

Kartowisastro , I. H., & Budiharto, W.	based and Transformer- based Sentiment Analysis on Code-mixed of Low-Resource Languages	Sentence- BERT, Google Translate	Javanese (CS)	83% Lexicon-VADER: 67% Lexicon- Sentiwordnet: 61%
Bhagya, S. S., 2022 & Nadera, B. S.	Analyzing Performance of Pre-trained Models in Detecting Sentiments of Code Mixed Manglish Text Reviews from Social Media	XLM, XLNet, RoBERTa, DistilBERT, BERT	Use several pre-trained models for m sentiment analysis of Malayalam- English corpus.	English, Malayala rti et al (2020a)
Chakravarthi, B. R., 2022	Dravidiancodem ix: Sentiment	RF, SVM, MNB, KNN, Decision	Developing sentiment	English, Tamil, Own YouTube <u>F-Score (Tamil)</u> RF:0.53

Priyadharshini, R., Muralidaran, V., Jose, N., Suryawanshi, S., Sherly, E., & McCrae, J. P. (2022). Language Resources and Evaluation, 56(3), 765-806.	analysis and offensive language identification dataset for dravidian languages in code-mixed text.	Tree, LR, BERT, CharBERT, DistillBERT, ALBERT, RoBERTa, XLM, XLNet, XLM-R	analysis and offensive language identification code-mixed dataset for Dravidian languages	Kannada, Malayalam	SVM: 0.41 MNB: 0.47 KNN: 0.42 DT: 0.53 LR: 0.56 BERT: 0.60 CharBERT: 0.59 DistillBERT: 0.61 ALBERT: 0.57 RoBERTa: 0.58 XLM: 0.60 XLNet: 0.58 XLM-R: 0.60
					<u>F-Score</u> <u>(Malayalam)</u> RF: 0.56 SVM: 0.21 MNB: 0.46 KNN: 0.38 DT: 0.51

LR: 0.57
BERT: 0.63
CharBERT: 0.62
DistillBERT: 0.58
ALBERT: 0.31
RoBERTa: 0.60
XLM: 0.62
XLNet: 0.63
XLM-R: 0.60

F-Score (Kannada)

RF: 0.55
SVM: 0.30
MNB: 0.48
KNN: 0.43
DT: 0.52
LR: 0.57
BERT: 0.59
CharBERT: 0.59
DistillBERT: 0.56
ALBERT: 0.55

									RoBERTa: 0.34 XLM: 0.59 XLNet: 0.58 XLM-R: 0.41
Maity, K., 2022	A Multi-task BERT, Vec-Map	Multi-task Hindi, Own, Twitter	Multi-modal English (CS) Maity & Saha (2021)	framework for sentiment and emotion-aided cyberbully detection					The result stated is only for the sentiment analysis task on single task section (ST) uni-model data (text) only. ST-BERT (%): 73.51 (+/-1.17) ST-VecMap (%): 75.64 (+/- 1.50) ST-BERT+VecMap (%): 75.53 (+/- 1.76)
Shanmugava 2022	An analysis of RF, LSVC, MNB, Using		Tamil, Chakrava	YouTube					<u>Accuracy</u>

divel, K., Sampath, S. H., Nandhakuma r, P., Mahalingam, P., Subramanian, M., Kumaresan, P. K., & Priyadharshin i, R. (2022)..	machine learning models for sentiment analysis of Tamil code- mixed data	LR, LSTM, BiLSTM, BiGRU, CNN, CNN+LSTM, LSTM+CNN, CNN+BiLSTM, BiLSTM+CNN, IndicBERT	machine learning, deep learning and hybrid models for Tamil code- mixed data sentiment analysis	English rthi et al (2022)	RF: 0.63 LSVC: 0.63 MNB: 0.64 LR: 0.65 LSTM: 0.63 BiLSTM: 0.64 BiGRU: 0.65 CNN: 0.65 CNN+LSTM: 0.65 LSTM+CNN: 0.65 CNN+BiLSTM: 0.66 BiLSTM+CNN: 0.64 IndicBERT: 0.62
SR, M. K., 2022 Kumar, L., & Malapati, A.	Sentiment Analysis on Code-Switched Dravidian Languages with	Extreme Learning Machines (ELM) kernels (RBF kernel, Polynomial kernel,	Sentiment Analysis using Kernel Based Extreme	English, Tamil, Kannada, Malayala m	Chakrava rti et al (2020a), Chakrava ti et al. <

Kernel Based Linear kernel)
Extreme
Learning
Machines.

Learning
Machines.

(2020b),
Hande et
al. (2020)

0.68

PCA+Imbalanced

- RBF: 0.67
- LIN: 0.67
- POLY:
0.67

All+SMOTE

- RBF: 0.56
- LIN: 0.51
- POLY:
0.57

PCA+SMOTE

- RBF: 0.49
- LIN: 0.46
- POLY:
0.49

Malayalam-

English

All+Imbalanced

- **RBF: 0.67**

- **LIN: 0.64**
- **POLY:**
0.67

PCA+Imbalanced

- RBF: 0.61
- LIN: 0.61
- POLY:
0.61

All+SMOTE

- RBF: 0.63
- LIN: 0.54
- POLY:
0.64

PCA+SMOTE

- RBF: 0.48
- LIN: 0.43
- POLY:
0.48

Kannada-English

All+Imbalanced

- RBF 0.71
- **LIN: 0.59**
- **POLY:**
0.70

PCA+Imbalanced

- RBF: 0.60
- LIN:0.55
- POLY:0.59

All+SMOTE

- **RBF: 0.74**
- LIN: 0.53
- POLY:
0.69

PCA+SMOTE

- RBF: 0.57
- LIN: 0.51
- POLY:
0.56

Thara, S., & 2022	Social media	CNN,	LSTM,	Sentiment	English,	FIRE	YouTube	<u>CNN (accuracy)</u>
Poornachandr	text analytics of	GRU,	BiLSTM,	analysis and	Malayala	2020,		Word2Vec: 0.7455

an, P. (2022). <i>Jour nal of big Data</i> , 9(1), 45.	Malayalam– English code- mixed using deep learning	BiGRU, Word2Vec, FastText	offensive m language identification on code- switched data	EACL 2021	FastText: 0.7329 <u>LSTM (accuracy)</u> Word2Vec: 0.6995 FastText: 0.7381 <u>GRU (accuracy)</u> Word2Vec: 0.7603 FastText: 0.7633 <u>BiLSTM (accuracy)</u> Word2Vec: 0.7225 FastText: 0.7284 <u>BiGRU (accuracy)</u> Word2Vec: 0.6965 FastText: 0.7351 <u>BiLSTM+CNN (accuracy)</u>
--	---	---------------------------------	--	--------------	--

Word2Vec: 0.6810

FastText: 0.7396

BiGRU+CNN

(accuracy)

Word2Vec: 0.7106

FastText: 0.7203

LSTM+CNN

(accuracy)

Word2Vec: 0.7151

FastText: 0.7292

GRU+CNN

(accuracy)

Word2Vec: 0.7121

FastText: 0.7255

Transformer based
classification
model (accuracy)

Appendix B: Python code for data collection stage in the framework of getting MESocSentiment corpus.

```
1 import tweepy
2 import csv
3 import datetime
4
5 # Twitter API credentials
6 consumerKey = 'eUiCQzouBv5wv8AltEcZ7imz'
7 consumerSecret = 'JDYw3sqFE7w1z3OzmKXsA8vB1Xosg3x4v5IOBLJHFtXiKJviQx'
8 accessToken = '2171781990-2EaNHhL3QNeU2tGKtisnDaTrWydM0sRzzwrw4'
9 accessTokenSecret = 'LcRdWYd3Qu7UKsC3BMzs2VzsWi9jymzt5NjT1ErqWbjmx'
10
11 my_bearer_token="AAAAAAAAAAAAAAAAAAncQcEAAAAA8wUV%2F4yCXoPhLs%2BHKpF1EHKXN4E%3DEntD7KmAncDYusrTC6LhuSF5UnJdLRFCoNa0tdmHa8mzCR10H"
12
13 csvFile= open('tweepy Malaysia 15 May 2023 ori.csv','a')
14
15 csvWriter = csv.writer(csvFile)
16
17 client = tweepy.Client(bearer_token=my_bearer_token)
18
19 query = '#Malaysia -is:retweet'
20
21 tweets = tweepy.Paginator(client.search_recent_tweets, query=query,
22                             tweet_fields=['context_annotations', 'created_at'], max_results=100).flatten(limit=100000)
23
24 for tweet in tweets:
25     #print(tweet.text)
26     #if len(tweet.context_annotations) > 0:
27         #print(tweet.context_annotations)
28
29     csvWriter.writerow([tweet.created_at, tweet.text.encode('utf-8')])
30 #print ([tweet.created_at, tweet.text])
31
32 csvFile.close()
```

Appendix C: Python code for data pre-processing stage in the framework of getting MESocSentiment corpus

```
10 !pip install contractions
11 !pip install nltk
12
13 # import library
14 import contractions
15 import pandas as pd
16 import numpy as np
17 import re
18 import unicodedata
19 import nltk
20 #import pandas and regex module
21
22 file_name= "12 Aug 2022 15 May 2023 raw.csv"
23 file_name_output= "data clean v5.csv"
24
25 df=pd.read_csv(file_name)
26
27 len(df)
28
29 #print name of columns
30 print(df.columns.tolist())
31
32 #drop all columns except points and blocks
33 df = df[['Tweets']]
34 print(df.columns.tolist())
35
36 df.head(100)
37
38 #remove 1st char in the rows (to remove b flag sign) using regex
39 df['Tweets'] = df['Tweets'].str[1:]
40
41 #remove invalid characters using regex
42 df.replace(to_replace=[r"xcc|xd7|xde|xca|xf3|xc9|xc5|xe1|xc6|xc7|xe6|xc9|x89|xe5|xa7|x90|xe5|xa7|x90|xe5|x92|x8c|xe5|xa6|xb9|xe5|xa6|
    xb9|xed|xbb|xac|x83|xb3|xe7|xbb|x99|x88|x91|xe5|xb0|x84|xe7|x9a|x84|xe5|x98|x9b|xef|xbc|x9f|xb1|x8f|x96|x87|xe8|x81|x8a|xe9|xaa|
    xa9|x9d|xa5|xa5|xbf|xba|xf0|xa4|xe2|xb8|xa1|x97|xb2|xbe|x80|xd8|xd9|x85|xad|xaf|xb7|xa8|xc3|x94|x95|xe3|xab|x82|xa0|xed|xeb|x9e|
    xec|xb4|x9e|xea|xb5|x93|x8d|xe0|xdb|xda|x86|xae|xa3|xa2|x8e|xbd|x8b|xc2|xb6|xd0|xd1|xcf|xce|xc4"], value="", regex=True,
    inplace=True)
43
44 #use to make lowercase to all sentences
45 df['Tweets'] = df['Tweets'].str.lower()
46
47 #remove hashtags (with words too)
48 df['Tweets'] = df['Tweets'].replace(r"#[A-Za-z0-9_]+", "", regex=True)
49
50 #remove mentions
51 df['Tweets'] = df['Tweets'].str.replace(r"\s*@w+", "", regex=True)
52 df['Tweets'] = df['Tweets'].str.replace(r"\s*@B@w+", "", regex=True)
53 df['Tweets'] = df['Tweets'].str.replace(r"\s*@S+", "", regex=True)
54 df['Tweets'] = df['Tweets'].str.replace(r"\s*@S+b", "", regex=True)
55
56 #remove url
57 df['Tweets'] = df['Tweets'].replace(r"http\S+", "", regex=True).replace(r"www\S+", "", regex=True)
58
59 #remove numbers(boleh guna)
60 df['Tweets'] = df['Tweets'].replace(r'[0-9]', "", regex=True)
61
62 #replace return and tab
63 df.replace(to_replace=[r"\t|\r|\n", "\t|\r|\n"], value=["", ""], regex=True, inplace=True)
64
65 #remove underscore(boleh guna)
66 df['Tweets'] = df['Tweets'].replace(r'_', "", regex=True)
67
68 #contractions and expansions
69 expanded_words=[]
70 for sentence in df['Tweets']:
71     # using contractions.fix to expand the shortened words
72     expanded_words.append(contractions.fix(sentence))
73
74 df['Tweets'] = expanded_words
75
76 #remove stopwords using NLTK English stopwords and Stopwords ISO Malay Language stop words list
77
78 nltk.download('stopwords')
79 from nltk.corpus import stopwords
80
81 sw = pd.read_csv('iso_malay.txt', header=None, names=["stop words"])
82
83 stop_words_isomalay = sw['stop words'].tolist()
84
```

```

85 print(stop_words_isomalay)
86 print(len(stop_words_isomalay))
87
88 stop_words = stopwords.words('english')
89
90 print(stop_words)
91 print(len(stop_words))
92
93 final_stop_words = stop_words + stop_words_isomalay
94
95 print(final_stop_words)
96 print(len(final_stop_words))
97

```

```

98 #stop_words.extend
99 df['Tweets'] = df['Tweets'].apply(lambda x: ' '.join([word for word in x.split() if word not in (final_stop_words)]))
100
101 #remove punctuations(boleh guna) ganti dengan space
102 df['Tweets'] = df['Tweets'].replace(r'&#', '&', regex=True)
103 df['Tweets'] = df['Tweets'].replace(r'^\w\s', '', regex=True)
104
105 #remove excess whitespace using split and join
106 df['Tweets'] = (df['Tweets'].str.split()).str.join(' ')
107
108 #remove one word sentence
109 wantedRows = df[df['Tweets'].str.split().str.len()==1].index
110 df['Tweets'] = df.drop(wantedRows, axis = 0)
111
112 #df.drop_duplicates(subset=None, inplace=True)
113 df['Tweets'] = df.drop_duplicates(subset=None, keep='first')
114
115 #drop empty rows
116 emptyRows = df[df['Tweets'].str.split().str.len()==0].index
117 df['Tweets'] = df.drop(emptyRows, axis = 0)
118
119 #df.fillna('DROP', inplace = True)
120 df.dropna(inplace = True)
121
122 df.head(200)
123
124 df.to_csv(file_name_output, index=False)

```

Appendix D: Python code for language identification in the framework of getting MESocSentiment corpus (Malaya)

```
10 !pip install fasttext
11 !pip install malaya
12 !pip install youtokentome
13
14 import malaya
15 import fasttext
16
17 malaya.language_detection.label
18
19 #using fasttext model
20
21 import tensorflow as tf
22 import pandas as pd
23 import numpy as np
24 import malaya
25 import logging
26
27 # To read a CSV file
28 df = pd.read_csv('data clean v5.csv')
29
30 fast_text = malaya.language_detection.fasttext()
31
32 tweet_list = df['Tweets'].tolist()
33
34 fast_text.predict_proba( tweet_list)
35
36 df['probability'] = fast_text.predict_proba( tweet_list)
37
38 print(df.head(5))
39
40 df.to_csv('data clean v5 malaya fast-text.csv', index=False)
```

Appendix E: Python code for language identification in the framework of getting MESocSentiment corpus (Lingua)

```
10 !pip install lingua language-detector
11
12 from lingua import Language, LanguageDetectorBuilder
13 import pandas as pd
14 import numpy as np
15
16 languages = [Language.MALAY, Language.ENGLISH]
17 detector = LanguageDetectorBuilder.from_languages(*languages).build()
18
19 df = pd.read_csv('data_clean_v5.csv')
20
21 df['Tweets'] = df['Tweets'].astype(str)
22
23 #confidence_values = detector.compute_language_confidence_values(sentence)
24
25 df['poly_obj'] = df['Tweets'].apply(lambda x: detector.compute_language_confidence_values(x))
26
27 df.head(5)
28
29 df.to_csv('data_clean_v5_lingua.csv')
```

Appendix F: Python code for combining chosen code-switched tweets from Malaya and Lingua for language identification in the framework of getting MESocSentiment corpus

```
10 import pandas as pd
11
12 df1=pd.read_csv('data clean v5 lingua (0,5-0.7).csv')
13 len(df1)
14
15 df2=pd.read_csv('data clean v5 malaya fast-text manglish rojak.csv')
16 len(df2)
17
18 df = pd.DataFrame()
19
20 lingua = df1['Tweets'].values.tolist()
21 malaya = df2['Tweets'].values.tolist()
22
23 lingua_malaya = lingua + malaya
24
25 lingua_malaya = list(dict.fromkeys(lingua_malaya))
26
27 df['Tweets']=pd.DataFrame(lingua_malaya)
28
29 df.head()
30
31 df.drop_duplicates(subset=None, inplace=True)
32 df['Tweets'] = df.drop_duplicates(subset=None, keep='first')
33
34 len(df)
35
36 df.to_csv("data clean v5 combine v3.csv")
```

Appendix G: Python code for sentiment identification in the framework of getting MESocSentiment corpus (VADER)

```
10 !pip install vaderSentiment
11
12 import pandas as pd
13 import numpy as np
14
15 from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
16
17 sid = SentimentIntensityAnalyzer()
18
19 # To read a CSV file
20 df = pd.read_csv('data clean v5 combine v3.csv')
21 print(df.head(5))
22
23 df['scores'] = df['Tweets'].apply(Lambda Tweets: sid.polarity_scores(Tweets))
24
25 df['compound'] = df['scores'].apply(Lambda score_dict: score_dict['compound'])
26
27 print(df.head(5))
28
29 df.loc[df['compound']>0, 'comp_score']='positive'
30 df.loc[df['compound']<0, 'comp_score']='negative'
31 df.loc[df['compound']==0, 'comp_score']='neutral'
32
33 print(df.head(5))
34
35 df.to_csv('data clean v5 combine v3 VADER.csv', index=False)
```

Appendix H: Python code for sentiment identification in the framework of getting MESocSentiment corpus (TextBlob)

```
10 pip install textblob
11 pip install pandas
12
13 import pandas as pd
14 import numpy as np
15
16 # To read a CSV file
17 df = pd.read_csv('data clean v5 combine v3.csv')
18
19 from textblob import TextBlob
20
21 # The x in the lambda function is a row (because I set axis=1)
22 # Apply iterates the function across the dataframe's rows
23 df['polarity'] = df.apply(lambda x: TextBlob(x['Tweets']).sentiment.polarity, axis=1)
24 df['subjectivity'] = df.apply(lambda x: TextBlob(x['Tweets']).sentiment.subjectivity, axis=1)
25
26 df.loc[df['polarity']>0, 'sentiment']='positive'
27 df.loc[df['polarity']<0, 'sentiment']='negative'
28 df.loc[df['polarity']==0, 'sentiment']='neutral'
29 print(df.head(5))
30
31 df.to_csv('data clean v5 combine v3 TextBlob.csv', index=False)
```

Appendix I: Python code for sentiment identification in the framework of getting MESocSentiment corpus (pysentimiento)

```
10 !pip install pysentimiento
11 !pip install transformers accelerate
12
13 import pysentimiento
14 import pandas as pd
15
16 from pysentimiento import create_analyzer
17
18 df = pd.read_csv('data clean v5 combine v3.csv')
19
20 analyzer = create_analyzer(task="sentiment", Lang="en")
21
22 tweet_list = df['Tweets'].tolist()
23
24 df['sentiment'] = analyzer.predict(tweet_list)
25
26 print(df.head(5))
27
28 df.to_csv('data clean v5 combine v3 Pysentimiento.csv', index=False)
```

Appendix J: Python code for compiling all datasets from all sentiment analysers for voting approach in sentiment identification stage in the framework of getting MESocSentiment corpus

```
12 import pandas as pd
13
14 f2 = pd.read_csv("data clean v5 combine v3 TextBlob.csv")
15 f3 = pd.read_csv("data clean v5 combine v3 VADER.csv")
16 f5 = pd.read_csv("data clean v5 combine v3 Pysentimiento.csv")
17
18 print(f2.head(5))
19 print(f3.head(5))
20 print(f5.head(5))
21
22 f6= pd.DataFrame()
23
24 f6['Tweets'] = f1['Tweets']
25 f6['TextBlob'] = f2['sentiment']
26 f6['VADER'] = f3['comp_score']
27 f6['Pysentimiento'] = f5['finalSentiment']
28
29 f6['VADER'] = f6['VADER'].replace(['positive', 'negative', 'neutral'], ['POSITIVE', 'NEGATIVE', 'NEUTRAL'])
30 f6['TextBlob'] = f6['TextBlob'].replace(['positive', 'negative', 'neutral'], ['POSITIVE', 'NEGATIVE', 'NEUTRAL'])
31 f6['Pysentimiento'] = f6['Pysentimiento'].replace(['POS', 'NEG', 'NEU'], ['POSITIVE', 'NEGATIVE', 'NEUTRAL'])
32
33 print(f6.head(5))
34
35 f6.to_csv("data clean v5 combine v3 (VADER, TextBlob, Pysentimiento).csv", index=False)
36
```

Appendix K: Python code for the bootstrapping technique for sentiment modelling strategy

```
10 pip install -U scikit-learn
11 pip install seaborn
12
13 import numpy as np
14 import pandas as pd
15 import re
16 import nltk
17 import matplotlib.pyplot as plt
18 from sklearn.model_selection import train_test_split
19
20 df = pd.read_csv('Train 4 MESocSentiment.csv')
21 df.head()
22
23 df2 = pd.read_csv('Test 4 MESocSentiment.csv')
24 df2.head()
25
26 df2.drop(['Y/N', 'comparison'], axis = 1, inplace = True)
27 df2.head()
28
29 df.head()
30 df2.head()
31
32 X_train = df['Tweets']
33 y_train = df['Sentiment (All)']
34
35 X_test = df2['Tweets']
36 y_test = df2['Sentiment (All)']
37
38 from sklearn.feature_extraction.text import TfidfVectorizer
39
40 tfidf_vect = TfidfVectorizer(max_features = None)
41 tfidf_vect.fit(df['Tweets'])
42 X_train_tfidf_vect = tfidf_vect.transform(df['Tweets'])
43 X_test_tfidf_vect = tfidf_vect.transform(df2['Tweets'])
44
45 tfidf_vect
```

```

47 print(X_train_tfidf_vect)
48 print(X_test_tfidf_vect)
49
50 print(X_train_tfidf_vect.shape)
51 print(X_test_tfidf_vect.shape)
52
53 from sklearn.svm import SVC
54
55 model = SVC(kernel='linear', random_state=0, decision_function_shape='ovr' )
56 model.fit(X_train_tfidf_vect, y_train)
57
58 from sklearn.metrics import accuracy_score
59
60 predictions_SVM = model.predict(X_test_tfidf_vect)
61 test_prediction_SVM = pd.DataFrame()
62 test_prediction_SVM['Tweets'] = X_test
63 #test_prediction_SVM['All'] = predictions_SVM
64 test_prediction_SVM['Sentiment (All)'] = predictions_SVM
65
66 SVM_accuracy = accuracy_score(predictions_SVM, y_test)
67
68 test_prediction_SVM
69 test_prediction_SVM.to_csv(r"Test4_test_prediction_SVM_TF-IDF_linear_SVM3.csv")
70
71 SVM_accuracy
72
73 from sklearn.metrics import classification_report
74
75 print("\nHere is the classification report for Support Vector Machine:")
76 print(classification_report(y_test, predictions_SVM))
77

```

```

78 from sklearn.metrics import confusion_matrix
79
80 print("\nHere is the confusion matrix for Support Vector Machine:")
81 print(confusion_matrix(y_test, predictions_SVM))
82
83 from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
84
85 cm = confusion_matrix(y_test, predictions_SVM)
86 disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model.classes_)
87
88 disp.plot()
89 plt.show()
90
91 #create bar graph
92 import matplotlib.pyplot as plt
93 import seaborn as sns
94
95 # create a barplot
96 sns.countplot(x=test_prediction_SVM['Sentiment (All)'], data=test_prediction_SVM['Sentiment (All)'])
97 # set the title
98 plt.title('Sentiment Analysis')
99 # set the x and y label
100 plt.xlabel('Sentiment')
101 plt.ylabel('Number of Tweets')
102 # show the plot
103 plt.show()
104
105 test_prediction_SVM['Sentiment (All)'].value_counts()
106 df2['Sentiment (All)'].value_counts()

```

Appendix L: Python code for oversampling using SMOTE

```
10 pip install pandas scikit-learn imbalanced-learn
11
12 #latest code (basically number 7)
13 import pandas as pd
14 from sklearn.feature_extraction.text import TfidfVectorizer
15 from sklearn.preprocessing import LabelEncoder
16 from imblearn.over_sampling import SMOTE
17
18 # Load your DataFrame
19 df = pd.read_csv('Train Part 2 MESocSentiment.csv') # Replace with your actual file name
20
21 # Assuming df is your original DataFrame
22 X_text = df['Tweets'] # Original text data
23 y = df['Sentiment (All)'] # Target variable
24
25 # Convert text data to numerical features using TF-IDF
26 vectorizer = TfidfVectorizer()
27 X_text_vectorized = vectorizer.fit_transform(X_text)
28
29 # Encode the target variable into numerical labels
30 label_encoder = LabelEncoder()
31 y_encoded = label_encoder.fit_transform(y)
32
33 # Count the number of tweets in each category in the original data
34 original_counts = pd.Series(y).value_counts()
35 print("Original Data Counts:")
36 print(original_counts)
37
38 # Initialize SMOTE with a random state for reproducibility
39 smote = SMOTE(random_state=42)
40
41 # Apply SMOTE to the feature matrix and target vector
42 X_resampled, y_resampled = smote.fit_resample(X_text_vectorized, y_encoded)
43
44 # Separate the synthetic samples from the original ones
45 original_sample_count = len(X_text)
46 X_resampled_text = vectorizer.inverse_transform(X_resampled)
47 y_resampled_labels = label_encoder.inverse_transform(y_resampled)
48
49 # Count the number of tweets in each category after resampling
50 resampled_counts = pd.Series(y_resampled_labels).value_counts()
51 print("\nResampled Data Counts:")
52 print(resampled_counts)
53
54 # Create a DataFrame for the original data
55 df_original = pd.DataFrame({
56     'Tweets': X_text,
57     'Sentiment (All)': y
58 })
59
60 # Create a DataFrame for the synthetic data
61 df_synthetic = pd.DataFrame({
62     'Tweets': [' '.join(text) for text in X_resampled_text[original_sample_count:]],
63     'Sentiment (All)': y_resampled_labels[original_sample_count:]
64 })
65
66 # Combine the original data with the synthetic data
67 df_resampled = pd.concat([df_original, df_synthetic], ignore_index=True)
68
69 # Save the combined DataFrame to a CSV file
70 df_resampled.to_csv('SMOTE Train Part 2 MESocSentiment.csv', index=False)
```

Appendix M: Python code for Support Vector Machine (SVM) model for sentiment modelling strategy

```
10 pip install -U scikit-learn
11 pip install joblib
12 pip install seaborn
13
14 import numpy as np
15 import pandas as pd
16 import re
17 import nltk
18 import matplotlib.pyplot as plt
19 from sklearn.model_selection import train_test_split
20
21 df = pd.read_csv('SMOTE Train Part 2 MESocSentiment.csv')
22 df.head()
23
24 df2 = pd.read_csv('Test Part 2 Ground Truth.csv')
25 df2.head()
26
27 df2.drop(['Sentiment (MESocSentiment)'], axis = 1, inplace = True)
28 df2.head()
29
30 df.head()
31 df2.head()
32
33 X_train = df['Tweets']
34 y_train = df['Sentiment (All)']
35
36 X_test = df2['Tweets']
37 y_test = df2['Sentiment (Manual)']
38
39
40
41 from sklearn.feature_extraction.text import TfidfVectorizer
42
43 tfidf_vect = TfidfVectorizer(max_features = None)
44 tfidf_vect.fit(df['Tweets'])
45 X_train_tfidf_vect = tfidf_vect.transform(df['Tweets'])
46 X_test_tfidf_vect = tfidf_vect.transform(df2['Tweets'])
47
48 tfidf_vect
49
50 print(X_train_tfidf_vect)
51 print(X_test_tfidf_vect)
52
53 print(X_train_tfidf_vect.shape)
54 print(X_test_tfidf_vect.shape)
55
56 from sklearn.svm import SVC
57
58 model = SVC(kernel='linear', random_state=0, decision_function_shape='ovr' )
59 model.fit(X_train_tfidf_vect, y_train)
60
61 #To save the SVM model
62 import joblib
63
64 # Using joblib
65 joblib.dump(model, 'support_vector_machine_model_joblib.pkl')
```

```

65 from sklearn.metrics import accuracy_score
66
67 predictions_SVM = model.predict(X_test_tfidf_vect)
68 test_prediction_SVM = pd.DataFrame()
69 test_prediction_SVM['Tweets'] = X_test
70 test_prediction_SVM['Sentiment'] = predictions_SVM
71
72 SVM_accuracy = accuracy_score(predictions_SVM, y_test)
73
74 test_prediction_SVM
75
76 test_prediction_SVM.to_csv(r"15Aug2024TestPart2_test_prediction_SVM_TF-IDF_linear_SVM3.csv")
77
78 SVM_accuracy
79
80 # Classification Report
81
82 from sklearn.metrics import classification_report
83
84 print("\nHere is the classification report for Support Vector Machine:")
85 print(classification_report(y_test, predictions_SVM))
86
87 from sklearn.metrics import confusion_matrix
88
89 print("\nHere is the confusion matrix for Support Vector Machine:")
90 print(confusion_matrix(y_test, predictions_SVM))
91
92 from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
93
94 cm = confusion_matrix(y_test, predictions_SVM)
95 disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model.classes_)
96
97 disp.plot()
98
99 plt.show()
100

```

```

101 #create bar graph
102 import matplotlib.pyplot as plt
103 import seaborn as sns
104
105 # create a barplot
106 sns.countplot(x='Sentiment', data= test_prediction_SVM)
107
108 # set the title
109 plt.title('Sentiment Analysis')
110
111 # set the x and y label
112 plt.xlabel('Sentiment')
113 plt.ylabel('Number of Tweets')
114
115 # show the plot
116 plt.show()
117
118 test_prediction_SVM['Sentiment'].value_counts()
119
120 df2['Sentiment (Manual)'].value_counts()
121

```

```

122 """Codes below is to calculate roc auc score and display graphs."""
123
124 n_samples, n_features = X_test_tfidf_vect.shape
125 n_classes = len(np.unique(y_test))
126 target_names = ['POSITIVE', 'NEGATIVE', 'NEUTRAL']
127
128 y_score = model.fit(X_train_tfidf_vect, y_train).decision_function(X_test_tfidf_vect)
129
130 from sklearn.preprocessing import LabelBinarizer
131
132 label_binarizer = LabelBinarizer().fit(y_train)
133 y_onehot_test = label_binarizer.transform(y_test)
134 y_onehot_test.shape # (n_samples, n_classes)
135
136 label_binarizer.transform(["POSITIVE"])
137
138 class_of_interest = "POSITIVE"
139 class_id = np.flatnonzero(label_binarizer.classes_ == class_of_interest)[0]
140 class_id
141
142 from sklearn.metrics import auc, roc_curve
143
144 # store the fpr, tpr, and roc_auc for all averaging strategies
145 fpr, tpr, roc_auc = dict(), dict(), dict()
146 # Compute micro-average ROC curve and ROC area
147 fpr["micro"], tpr["micro"], _ = roc_curve(y_onehot_test.ravel(), y_score.ravel())
148 roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
149
150 print(f"Micro-averaged One-vs-Rest ROC AUC score:\n{roc_auc['micro']:.2f}")
151
152 # store the fpr, tpr, and roc_auc for all averaging strategies
153 #fpr, tpr, roc_auc = dict(), dict(), dict()
154
155 for i in range(n_classes):
156     fpr[i], tpr[i], _ = roc_curve(y_onehot_test[:, i], y_score[:, i])
157     roc_auc[i] = auc(fpr[i], tpr[i])
158
159 fpr_grid = np.linspace(0.0, 1.0, 1000)
160

```

```

161 # Interpolate all ROC curves at these points
162 mean_tpr = np.zeros_like(fpr_grid)
163
164 for i in range(n_classes):
165     mean_tpr += np.interp(fpr_grid, fpr[i], tpr[i]) # linear interpolation
166
167 # Average it and compute AUC
168 mean_tpr /= n_classes
169
170 fpr["macro"] = fpr_grid
171 tpr["macro"] = mean_tpr
172 roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])
173
174 print(f"Macro-averaged One-vs-Rest ROC AUC score:\n{roc_auc['macro']:.2f}")
175
176 from sklearn.metrics import auc, roc_curve
177 import numpy as np
178 import matplotlib.pyplot as plt
179 from itertools import cycle
180 from sklearn.metrics import RocCurveDisplay
181
182 # Store the fpr, tpr, and roc_auc for all averaging strategies
183 fpr, tpr, roc_auc = dict(), dict(), dict()
184
185 # Compute micro-average ROC curve and ROC area
186 fpr["micro"], tpr["micro"], _ = roc_curve(y_onehot_test.ravel(), y_score.ravel())
187 roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
188
189 # Compute ROC curve and ROC area for each class
190 for i in range(n_classes):
191     fpr[i], tpr[i], _ = roc_curve(y_onehot_test[:, i], y_score[:, i])
192     roc_auc[i] = auc(fpr[i], tpr[i])
193
194 fpr_grid = np.linspace(0.0, 1.0, 1000)
195
196 # Interpolate all ROC curves at these points
197 mean_tpr = np.zeros_like(fpr_grid)
198

```

```

199 for i in range(n_classes):
200     mean_tpr += np.interp(fpr_grid, fpr[i], tpr[i]) # linear interpolation
201
202 # Average it and compute AUC
203 mean_tpr /= n_classes
204
205 fpr["macro"] = fpr_grid
206 tpr["macro"] = mean_tpr
207 roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])
208
209 print(f"Micro-averaged One-vs-Rest ROC AUC score:\n{roc_auc['micro']:.2f}")
210 print(f"Macro-averaged One-vs-Rest ROC AUC score:\n{roc_auc['macro']:.2f}")
211

```

```

212 # Plotting
213 fig, ax = plt.subplots(figsize=(6, 6))
214
215 plt.plot(
216     fpr["micro"],
217     tpr["micro"],
218     label=f"micro-average ROC curve (AUC = {roc_auc['micro']:.2f})",
219     color="deeppink",
220     linestyle=":",
221     linewidth=4,
222 )
223
224 plt.plot(
225     fpr["macro"],
226     tpr["macro"],
227     label=f"macro-average ROC curve (AUC = {roc_auc['macro']:.2f})",
228     color="navy",
229     linestyle=":",
230     linewidth=4,
231 )
232

```

```

233 colors = cycle(["aqua", "darkorange", "cornflowerblue"])
234 for i, color in zip(range(n_classes), colors):
235     RocCurveDisplay.from_predictions(
236         y_onehot_test[:, i],
237         y_score[:, i],
238         name=f"ROC curve for {target_names[i]}",
239         color=color,
240         ax=ax,
241         plot_chance_level=(i == 2),
242     )
243
244 ax.set(
245     xlabel="False Positive Rate",
246     ylabel="True Positive Rate",
247     title="Extension of Receiver Operating Characteristic\nto One-vs-Rest multiclass Support Vector Machine",
248 )
249
250 plt.legend(loc="lower right")
251 plt.show()

```

Appendix N: Python code for Naïve Bayes model for sentiment modelling strategy

```
10 !pip install -U scikit-learn
11 !pip install joblib
12 pip install seaborn
13
14 import numpy as np
15 import pandas as pd
16 import re
17 import nltk
18 import matplotlib.pyplot as plt
19 from sklearn.model_selection import train_test_split
20
21 df = pd.read_csv('SMOTE Train Part 2 MESocSentiment.csv')
22 df.head()
23
24 df2 = pd.read_csv('Test Part 2 Ground Truth.csv')
25 df2.head()
26
27 df2.drop(['Sentiment (MESocSentiment)'], axis = 1, inplace = True)
28 df2.head()
29
30 df.head()
31 df2.head()
32
33 X_train = df['Tweets']
34 y_train = df['Sentiment (All)']
35
36 X_test = df2['Tweets']
37 y_test = df2['Sentiment (Manual)']
38
39
40 from sklearn.feature_extraction.text import TfidfVectorizer
41 tfidf_vect = TfidfVectorizer(max_features = None)
42 tfidf_vect.fit(df['Tweets'])
43 X_train_tfidf_vect = tfidf_vect.transform(df['Tweets'])
44 X_test_tfidf_vect = tfidf_vect.transform(df2['Tweets'])
45
46 tfidf_vect
47
48 print(X_train_tfidf_vect)
49 print(X_test_tfidf_vect)
50
51 print(X_train_tfidf_vect.shape)
52 print(X_test_tfidf_vect.shape)
53
54 from sklearn.naive_bayes import MultinomialNB
55
56 MNB = MultinomialNB()
57 MNB.fit(X_train_tfidf_vect, y_train)
58
59 #To save the Multinomial Naive Bayes model
60 import joblib
61
62 # Using joblib
63 joblib.dump(MNB, 'naive_bayes_model_joblib.pkl')
64
65 from sklearn.metrics import accuracy_score
66
67 predictions_MNB = MNB.predict(X_test_tfidf_vect)
68 test_prediction_MNB = pd.DataFrame()
69 test_prediction_MNB['Tweets'] = X_test
70 test_prediction_MNB['Sentiment'] = predictions_MNB
71 MNB_accuracy = accuracy_score(predictions_MNB, y_test)
72
73 test_prediction_MNB
74 MNB_accuracy
75
76 test_prediction_MNB.to_csv(r"15AugTestPart2_test_prediction_MNB_TF-IDF_linear_MNB3.csv")
77
78 # Classification Report
79
80 from sklearn.metrics import classification_report
81
82 print("\nHere is the classification report for Multinomial Naive Bayes:")
83 print(classification_report(y_test, predictions_MNB))
84
85 from sklearn.metrics import confusion_matrix
86
87 print("\nHere is the confusion matrix for Multinomial Naive Bayes:")
88 print(confusion_matrix(y_test, predictions_MNB))
89
```

```

90 from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
91
92 cm = confusion_matrix(y_test, predictions_MNB)
93 disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=MNB.classes_)
94
95 disp.plot()
96
97 plt.show()
98
99 #create bar graph
100 import matplotlib.pyplot as plt
101 import seaborn as sns
102
103 # create a barplot
104 sns.countplot(x='Sentiment', data= test_prediction_MNB)
105
106 # set the title
107 plt.title('Sentiment Analysis')
108
109 # set the x and y label
110 plt.xlabel('Sentiment')
111 plt.ylabel('Number of Tweets')
112
113 # show the plot
114 plt.show()
115
116 test_prediction_MNB['Sentiment'].value_counts()
117
118 df2['Sentiment (Manual)'].value_counts()
119

```

```

120 """Codes below is to calculate roc auc score and display graphs."""
121
122 n_samples, n_features = X_test_tfidf_vect.shape
123 n_classes = len(np.unique(y_test))
124 target_names = ['POSITIVE', 'NEGATIVE', 'NEUTRAL']
125
126 y_score = MNB.fit(X_train_tfidf_vect, y_train).predict_proba(X_test_tfidf_vect)
127
128 from sklearn.preprocessing import LabelBinarizer
129
130 label_binarizer = LabelBinarizer().fit(y_train)
131 y_onehot_test = label_binarizer.transform(y_test)
132 y_onehot_test.shape # (n_samples, n_classes)
133
134 label_binarizer.transform(["POSITIVE"])
135
136 class_of_interest = "POSITIVE"
137 class_id = np.flatnonzero(label_binarizer.classes_ == class_of_interest)[0]
138 class_id
139
140 #First code example for micro-average ROC-AUC
141 from sklearn.metrics import auc, roc_curve
142
143 # store the fpr, tpr, and roc_auc for all averaging strategies
144 fpr, tpr, roc_auc = dict(), dict(), dict()
145 # Compute micro-average ROC curve and ROC area
146 fpr["micro"], tpr["micro"], _ = roc_curve(y_onehot_test.ravel(), y_score.ravel())
147 roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
148
149 print(f"Micro-averaged One-vs-Rest ROC AUC score:\n{roc_auc['micro']:.2f}")
150

```

```

151 #Second code example for micro-average ROC-AUC
152 from sklearn.metrics import roc_auc_score
153
154 micro_roc_auc_ovr = roc_auc_score(
155     y_test,
156     y_score,
157     multi_class="ovr",
158     average="micro",
159 )
160
161 print(f"Micro-averaged One-vs-Rest ROC AUC score:\n{micro_roc_auc_ovr:.2f}")
162
163 #First code example for macro-average ROC-AUC
164 for i in range(n_classes):
165     fpr[i], tpr[i], _ = roc_curve(y_onehot_test[:, i], y_score[:, i])
166     roc_auc[i] = auc(fpr[i], tpr[i])
167
168 fpr_grid = np.linspace(0.0, 1.0, 1000)
169
170 # Interpolate all ROC curves at these points
171 mean_tpr = np.zeros_like(fpr_grid)
172
173 for i in range(n_classes):
174     mean_tpr += np.interp(fpr_grid, fpr[i], tpr[i]) # linear interpolation
175
176 # Average it and compute AUC
177 mean_tpr /= n_classes
178
179 fpr["macro"] = fpr_grid
180 tpr["macro"] = mean_tpr
181 roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])
182
183 print(f"Macro-averaged One-vs-Rest ROC AUC score:\n{roc_auc['macro']:.2f}")
184

```

```

185 #Second code example for macro-average ROC-AUC
186 macro_roc_auc_ovr = roc_auc_score(
187     y_test,
188     y_score,
189     multi_class="ovr",
190     average="macro",
191 )
192
193 print(f"Macro-averaged One-vs-Rest ROC AUC score:\n{macro_roc_auc_ovr:.2f}")
194

```

```

195 from itertools import cycle
196
197 fig, ax = plt.subplots(figsize=(6, 6))
198
199 plt.plot(
200     fpr["micro"],
201     tpr["micro"],
202     label=f"micro-average ROC curve (AUC = {roc_auc['micro']:.2f})",
203     color="deeppink",
204     linestyle=":",
205     linewidth=4,
206 )
207
208 plt.plot(
209     fpr["macro"],
210     tpr["macro"],
211     label=f"macro-average ROC curve (AUC = {roc_auc['macro']:.2f})",
212     color="navy",
213     linestyle=":",
214     linewidth=4,
215 )
216
217 colors = cycle(["aqua", "darkorange", "cornflowerblue"])
218 for class_id, color in zip(range(n_classes), colors):
219     RocCurveDisplay.from_predictions(
220         y_onehot_test[:, class_id],
221         y_score[:, class_id],
222         name=f"ROC curve for {target_names[class_id]}",
223         color=color,
224         ax=ax,
225         plot_chance_level=(class_id == 2),
226     )
227
228 ax.set(
229     xlabel="False Positive Rate",
230     ylabel="True Positive Rate",
231     title="Extension of Receiver Operating Characteristic\nto One-vs-Rest multiclass Naive Bayes",
232 )
233

```

Appendix O: Python code for Random Forest model for sentiment modelling strategy

```
10 pip install -U scikit-learn
11 pip install joblib
12 pip install seaborn
13
14 import numpy as np
15 import pandas as pd
16 import re
17 import nltk
18 import matplotlib.pyplot as plt
19 from sklearn.model_selection import train_test_split
20
21 df = pd.read_csv('SMOTE Train Part 2 MESocSentiment.csv')
22 df.head()
23
24 df2 = pd.read_csv('Test Part 2 Ground Truth.csv')
25 df2.head()
26
27 df2.drop(['Sentiment (MESocSentiment)'], axis = 1, inplace = True)
28 df2.head()
29
30 df.head()
31 df2.head()
32
33 X_train = df['Tweets']
34 y_train = df['Sentiment (All)']
35
36 X_test = df2['Tweets']
37 y_test = df2['Sentiment (Manual)']
38
```

```
39 from sklearn.feature_extraction.text import TfidfVectorizer
40
41 tfidf_vect = TfidfVectorizer(max_features = None)
42 tfidf_vect.fit(df['Tweets'])
43 X_train_tfidf_vect = tfidf_vect.transform(df['Tweets'])
44 X_test_tfidf_vect = tfidf_vect.transform(df2['Tweets'])
45
46 tfidf_vect
47
48 print(X_train_tfidf_vect)
49 print(X_test_tfidf_vect)
50
51 print(X_train_tfidf_vect.shape)
52 print(X_test_tfidf_vect.shape)
53
54 from sklearn.ensemble import RandomForestClassifier
55
56 forest = RandomForestClassifier(n_estimators=100, random_state=1)
57 forest.fit(X_train_tfidf_vect, y_train)
58
59 #To save the Random Forest model
60 import joblib
61
62 # Using joblib
63 joblib.dump(forest, 'random_forest_model_joblib.pkl')
64
65 from sklearn.metrics import accuracy_score
66
67 predictions_RF = forest.predict(X_test_tfidf_vect)
68 test_prediction_RF = pd.DataFrame()
69 test_prediction_RF['Tweets'] = X_test
70 test_prediction_RF['Sentiment'] = predictions_RF
71 RF_accuracy = accuracy_score(predictions_RF, y_test)
72
73 test_prediction_RF
74
75 test_prediction_RF.to_csv(r"15Aug2024TestPart2_test_prediction_RF_TF-IDF_linear_RF3.csv")
76
```

```

77 # Classification Report
78
79 from sklearn.metrics import classification_report
80
81 print("\nHere is the classification report for Random Forest:")
82 print(classification_report(y_test, predictions_RF))
83
84 RF_accuracy
85
86 from sklearn.metrics import confusion_matrix
87
88 print("\nHere is the confusion matrix for Random Forest:")
89 print(confusion_matrix(y_test, predictions_RF))
90
91 from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
92
93 cm = confusion_matrix(y_test, predictions_RF)
94 disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=forest.classes_)
95
96 disp.plot()
97
98 plt.show()
99

```

```

100 #create bar graph
101 import matplotlib.pyplot as plt
102 import seaborn as sns
103
104 # create a barplot
105 sns.countplot(x='Sentiment', data= test_prediction_RF)
106
107 # set the title
108 plt.title('Sentiment Analysis')
109
110 # set the x and y label
111 plt.xlabel('Sentiment')
112 plt.ylabel('Number of Tweets')
113
114 # show the plot
115 plt.show()
116
117 test_prediction_RF['Sentiment'].value_counts()
118
119 df2['Sentiment (Manual)'].value_counts()
120
121 #Preparing for ROC-AUC metrics
122 n_samples, n_features = X_test_tfidf_vect.shape
123 n_classes = len(np.unique(y_test))
124 target_names = ['POSITIVE', 'NEGATIVE', 'NEUTRAL']
125
126 y_score = forest.fit(X_train_tfidf_vect, y_train).predict_proba(X_test_tfidf_vect)
127
128 from sklearn.preprocessing import LabelBinarizer
129
130 label_binarizer = LabelBinarizer().fit(y_train)
131 y_onehot_test = label_binarizer.transform(y_test)
132 y_onehot_test.shape # (n_samples, n_classes)
133
134 #Check encoding of a class
135 label_binarizer.transform(["POSITIVE"])
136
137 class_of_interest = "POSITIVE"
138 class_id = np.flatnonzero(label_binarizer.classes_ == class_of_interest)[0]
139 class_id

```

```

140
141 #First code example for micro-average ROC-AUC
142 from sklearn.metrics import auc, roc_curve
143
144 # store the fpr, tpr, and roc_auc for all averaging strategies
145 fpr, tpr, roc_auc = dict(), dict(), dict()
146 # Compute micro-average ROC curve and ROC area
147 fpr["micro"], tpr["micro"], _ = roc_curve(y_onehot_test.ravel(), y_score.ravel())
148 roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
149
150 print(f"Micro-averaged One-vs-Rest ROC AUC score:\n{roc_auc['micro']:.2f}")
151
152 #Second code example for micro-average ROC-AUC
153 from sklearn.metrics import roc_auc_score
154
155 micro_roc_auc_ovr = roc_auc_score(
156     y_test,
157     y_score,
158     multi_class="ovr",
159     average="micro",
160 )
161
162 print(f"Micro-averaged One-vs-Rest ROC AUC score:\n{micro_roc_auc_ovr:.2f}")
163

```

```

164 #First code example for macro-average ROC-AUC
165 for i in range(n_classes):
166     fpr[i], tpr[i], _ = roc_curve(y_onehot_test[:, i], y_score[:, i])
167     roc_auc[i] = auc(fpr[i], tpr[i])
168
169 fpr_grid = np.linspace(0.0, 1.0, 1000)
170
171 # Interpolate all ROC curves at these points
172 mean_tpr = np.zeros_like(fpr_grid)
173
174 for i in range(n_classes):
175     mean_tpr += np.interp(fpr_grid, fpr[i], tpr[i]) # linear interpolation
176
177 # Average it and compute AUC
178 mean_tpr /= n_classes
179
180 fpr["macro"] = fpr_grid
181 tpr["macro"] = mean_tpr
182 roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])
183
184 print(f"Macro-averaged One-vs-Rest ROC AUC score:\n{roc_auc['macro']:.2f}")
185
186 #Second code example for macro-average ROC-AUC
187 macro_roc_auc_ovr = roc_auc_score(
188     y_test,
189     y_score,
190     multi_class="ovr",
191     average="macro",
192 )
193
194 print(f"Macro-averaged One-vs-Rest ROC AUC score:\n{macro_roc_auc_ovr:.2f}")
195

```

```

196 from itertools import cycle
197
198 fig, ax = plt.subplots(figsize=(6, 6))
199
200 plt.plot(
201     fpr["micro"],
202     tpr["micro"],
203     label=f"micro-average ROC curve (AUC = {roc_auc['micro']:.2f})",
204     color="deeppink",
205     linestyle=":",
206     linewidth=4,
207 )
208
209 plt.plot(
210     fpr["macro"],
211     tpr["macro"],
212     label=f"macro-average ROC curve (AUC = {roc_auc['macro']:.2f})",
213     color="navy",
214     linestyle=":",
215     linewidth=4,
216 )
217
218 colors = cycle(["aqua", "darkorange", "cornflowerblue"])
219 for class_id, color in zip(range(n_classes), colors):
220     RocCurveDisplay.from_predictions(
221         y_onehot_test[:, class_id],
222         y_score[:, class_id],
223         name=f"ROC curve for {target_names[class_id]}",
224         color=color,
225         ax=ax,
226         plot_chance_level=(class_id == 2),
227     )
228
229 - ax.set(
230     xlabel="False Positive Rate",
231     ylabel="True Positive Rate",
232     title="Extension of Receiver Operating Characteristic\nto One-vs-Rest multiclass Random Forest",
233 )

```

Appendix P: Python code for CNN model for sentiment modelling strategy

```
9 The parameter for CNN in this file is based on Ihara, S., & Poornachandran, P. (2022). Social media text analytics of
  Malayalam-English code-mixed using deep learning. Journal of big Data, 9(1), 45.
10
11
12 pip install tensorflow numpy pandas scikit-learn seaborn
13
14 import numpy as np
15 import pandas as pd
16 import matplotlib.pyplot as plt
17 import seaborn as sns
18 import tensorflow
19 from tensorflow.keras.preprocessing.text import Tokenizer
20 from tensorflow.keras.preprocessing.sequence import pad_sequences
21 from tensorflow.keras.models import Sequential
22 from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense, Dropout
23 from tensorflow.keras.callbacks import EarlyStopping
24
25 # Change the name of the train file and test file
26 df = pd.read_csv('SMOTE Train Part 2 MESocSentiment.csv')
27 df2 = pd.read_csv('Test Part 2 Ground Truth.csv')
28
29 df.head()
30
31 df.info()
32
33 df.shape
34
35 # The maximum number of words to be used. (most frequent)
36 MAX_NB_WORDS = 50000
37 # Max number of words in the tweet (using the highest value for maximum sequence length in the paper)
38 MAX_SEQUENCE_LENGTH = 16
39 # using the lowest word embedding dimension from the paper
40 EMBEDDING_DIM = 100
41
```

```
42 tokenizer = Tokenizer(num_words=MAX_NB_WORDS, Lower=True)
43 tokenizer.fit_on_texts(df['Tweets'].values)
44 word_index = tokenizer.word_index
45 print('Found %s unique tokens.' % len(word_index))
46
47 X_train = tokenizer.texts_to_sequences(df['Tweets'].values)
48 X_train = pad_sequences(X_train, maxlen=MAX_SEQUENCE_LENGTH)
49 print('Shape of data tensor:', X_train.shape)
50
51 Y_train = pd.get_dummies(df['Sentiment (All)']).values
52 print('Shape of label tensor:', Y_train.shape)
53
54 df2.head()
55
56 df2.info()
57
58 #df2.drop(['Y/N', 'comparison'], axis = 1, inplace = True)
59 df2.drop(['Sentiment (MESocSentiment)'], axis = 1, inplace = True)
60 df2.head(100)
61
62 df2.shape
63
64 df2.value_counts("Sentiment (Manual)")
65
66 tokenizer.fit_on_texts(df2['Tweets'].values)
67 word_index2 = tokenizer.word_index
68 print('Found %s unique tokens.' % len(word_index2))
69
70 X_test = tokenizer.texts_to_sequences(df2['Tweets'].values)
71 X_test = pad_sequences(X_test, maxlen=MAX_SEQUENCE_LENGTH)
72 print('Shape of data tensor:', X_test.shape)
73
74 Y_test = pd.get_dummies(df2['Sentiment (Manual)']).values
75 print('Shape of label tensor:', Y_test.shape)
76
77 Y_test
78
79 print(X_train.shape, Y_train.shape)
80 print(X_test.shape, Y_test.shape)
81
```

```

82 filters = 256 # Number of filters/kernels in the Conv1D layer (from the paper)
83 kernel_size = 1 # Size of the convolutional kernel (from the paper)
84 strides = 1 # from the paper
85
86 #Define the model
87 model = Sequential()
88
89 #This is for embedding layer
90 model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length=X_train.shape[1]))
91
92 #1-D Convolutional Layer
93 model.add(Conv1D(filters, kernel_size, strides, activation='selu'))#activation function used is based on the paper
94
95 # 1-D Global Max Pooling Layer (from the paper)
96 model.add(GlobalMaxPooling1D())
97
98 #First feedforward neural network
99 model.add(Dense(256, activation='relu'))# dimensionality of the output space (units) and activation are from the paper
100
101 #Dropout layer between two feedforward neural network to prevent overfitting
102 model.add(Dropout(0.5))# using 0.5 value because the paper stated the used parameters is 0 to 1
103
104 #Second feedforward neural network layer
105 #The final feedforward layer is also acts as the classification layer
106 model.add(Dense(3, activation='softmax')) #units are from number of classes and activation is based on paper
107
108 #Compiling the model
109 #optimizer are from the paper
110 #The paper only state that using cross-entropy, but the code using the specific categorical_crossentropy instead
111 #Using accuracy for metrics
112 #learning rate is using default learning rate
113 model.compile(optimizer=tensorflow.keras.optimizers.Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics
114               =['accuracy'])
115 print(model.summary())
116
117 #using early stopping to stop training where there is no improvement in loss in 3 consecutive epochs
118 callback = tensorflow.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
119

```

```

120 # Train the model
121 #batch size and epoch are from the smallest values used in the paper
122 history = model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=10, batch_size=32,callbacks=[callback])
123
124 history.history
125
126 #Save the CNN model
127 from tensorflow.keras.models import load_model, save_model
128
129 # Assuming 'model' is your TensorFlow model
130 model.save('cnn 16 August 2024 model.h5')
131
132 from google.colab import files
133 files.download('cnn 16 August 2024 model.h5')
134
135 # Evaluate the model
136 loss, accuracy = model.evaluate(X_test, Y_test)
137 print(f'Test accuracy: {accuracy * 100:.2f}%')
138 print(f'Test accuracy: {accuracy}%')
139 print(f'Test accuracy: {accuracy:.3f}')
140 print(f'Test accuracy: {accuracy:.4f}')
141
142 # Plot training and validation accuracy
143 plt.plot(history.history['accuracy'], label='Training Accuracy')
144 plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
145 plt.title('Training and Validation Accuracy')
146 plt.xlabel('Epoch')
147 plt.ylabel('Accuracy')
148 plt.legend()
149 plt.show()
150
151 # Plot training and validation loss
152 plt.plot(history.history['loss'], label='Training Loss')
153 plt.plot(history.history['val_loss'], label='Validation Loss')
154 plt.title('Training and Validation Loss')
155 plt.xlabel('Epoch')
156 plt.ylabel('Loss')
157 plt.legend()
158 plt.show()

```

```

159
160 #Graph for loss
161 plt.title('Loss')
162 plt.plot(history.history['loss'], Label='train')
163 plt.plot(history.history['val_loss'], Label='test')
164 plt.legend()
165 plt.show();
166
167 #Graph for accuracy
168 plt.title('Accuracy')
169 plt.plot(history.history['accuracy'], Label='train')
170 plt.plot(history.history['val_accuracy'], Label='test')
171 plt.legend()
172 plt.show();
173
174 #Prediction
175 predictions_CNN = model.predict(X_test)
176 test_prediction_CNN = pd.DataFrame()
177 test_prediction_CNN['Tweets'] = df2['Tweets']
178
179 classes = np.argmax(predictions_CNN, axis=1)
180 print(classes)
181
182 test_prediction_CNN['Classes'] = classes
183
184 #import numpy as np
185
186 labels = ['NEGATIVE', 'POSITIVE', 'NEUTRAL']
187 sentiment = [labels[idx] for idx in np.argmax(predictions_CNN, axis=1)]
188 print(sentiment)
189
190 test_prediction_CNN['Sentiment'] = sentiment
191
192 test_prediction_CNN
193
194 #Change the name of the output file
195 test_prediction_CNN.to_csv(r"16Aug2024_CNN_TestPart2_test_prediction.csv")
196

```

```

197 import pandas as pd
198
199 value_counts = test_prediction_CNN['Sentiment'].value_counts()
200 print(value_counts)
201
202 #create bar graph
203 import matplotlib.pyplot as plt
204 import seaborn as sns
205
206 # create a barplot
207 sns.countplot(x='Sentiment', data= test_prediction_CNN)
208
209 # set the title
210 plt.title('Sentiment Analysis')
211
212 # set the x and y label
213 plt.xlabel('Sentiment')
214 plt.ylabel('Number of Tweets')
215
216 # show the plot
217 plt.show()
218

```

```

219 #ROC-AUC
220 from sklearn.metrics import roc_curve, roc_auc_score
221 import matplotlib.pyplot as plt
222 from itertools import cycle
223 import numpy as np
224
225 # Assuming predictions and true labels are available
226 # predictions_CNN is your predicted probabilities or scores
227 # Y_test is your true labels
228
229 # Calculate ROC AUC for each class
230 n_classes = predictions_CNN.shape[1]
231 fpr = dict()
232 tpr = dict()
233 roc_auc = dict()
234
235 for i in range(n_classes):
236     fpr[i], tpr[i], _ = roc_curve(Y_test[:, i], predictions_CNN[:, i])
237     roc_auc[i] = roc_auc_score(Y_test[:, i], predictions_CNN[:, i])
238
239 # Compute microaverage ROC AUC
240 fpr["micro"], tpr["micro"], _ = roc_curve(Y_test.ravel(), predictions_CNN.ravel())
241 roc_auc["micro"] = roc_auc_score(Y_test, predictions_CNN, average="micro")
242
243 # Compute macroaverage ROC AUC
244 all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))
245 mean_tpr = np.zeros_like(all_fpr)
246 for i in range(n_classes):
247     mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])
248 mean_tpr /= n_classes
249 fpr["macro"] = all_fpr
250 tpr["macro"] = mean_tpr
251 roc_auc["macro"] = roc_auc_score(Y_test, predictions_CNN, average="macro")
252
253 # Class labels
254 labels = ['NEGATIVE', 'POSITIVE', 'NEUTRAL']
255
256 # Plot ROC curves
257 fig, ax = plt.subplots(figsize=(6, 6))
258
259 # Plot microaverage ROC curve
260 plt.plot(fpr["micro"], tpr["micro"],
261         label=f'micro-average ROC curve (AUC = {roc_auc["micro"]:.2f})',
262         color='deeppink', linestyle=':', linewidth=4)
263
264 # Plot macroaverage ROC curve
265 plt.plot(fpr["macro"], tpr["macro"],
266         label=f'macro-average ROC curve (AUC = {roc_auc["macro"]:.2f})',
267         color='navy', linestyle=':', linewidth=4)
268
269 # Plot ROC curves for each class
270 colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
271 for i, color in zip(range(n_classes), colors):
272     plt.plot(fpr[i], tpr[i], color=color, lw=2,
273             label=f'ROC curve of {labels[i]} (AUC = {roc_auc[i]:.2f})')
274
275 # Plot random classifier (chance level)
276 plt.plot([0, 1], [0, 1], color='black', linestyle='--', lw=2, label='Chance level (AUC=0.5)')
277
278 # Plot chance value annotation
279 plt.plot([0, 1], [0.5, 0.5], color='red', linestyle='-.', lw=2)
280
281 plt.xlim([0.0, 1.0])
282 plt.ylim([0.0, 1.05])
283 plt.xlabel('False Positive Rate')
284 plt.ylabel('True Positive Rate')
285 plt.title('Extension of Receiver Operating Characteristic\nto One-vs-Rest multiclass CNN')
286 plt.legend(loc="lower right")
287 plt.show()
288

```

```

289 #Confusion Matrix
290 import numpy as np
291 from sklearn.metrics import confusion_matrix
292 import matplotlib.pyplot as plt
293 import seaborn as sns
294
295 # Assuming predictions_CNN is your CNN model's predictions and Y_test is your true labels
296 predictions = np.argmax(predictions_CNN, axis=1)
297
298 # Define your class labels
299 labels = ['NEGATIVE', 'POSITIVE', 'NEUTRAL']
300
301 # Convert predicted and true labels to sentiment labels
302 predicted_sentiment = [labels[idx] for idx in predictions]
303 true_sentiment = [labels[idx] for idx in np.argmax(Y_test, axis=1)]
304
305 # Compute confusion matrix
306 conf_mat = confusion_matrix(true_sentiment, predicted_sentiment, labels=labels)
307
308 # Plot confusion matrix
309 plt.figure(figsize=(8, 6))
310 sns.heatmap(conf_mat, annot=True, fmt='d', cmap='viridis', xticklabels=labels, yticklabels=labels)
311 plt.xlabel('Predicted label')
312 plt.ylabel('True label')
313 plt.title('Confusion Matrix')
314 plt.show()
315

```

```

316 #Classification Report
317
318 import numpy as np
319 import pandas as pd
320 from sklearn.metrics import classification_report
321
322 # Assuming df2 has your tweets and Y_test has your actual labels
323 # Assuming Y_test is one-hot encoded
324 y_pred = np.argmax(predictions_CNN, axis=1)
325 y_true = np.argmax(Y_test, axis=1)
326
327 # Generate classification report
328 class_names = ['NEGATIVE', 'POSITIVE', 'NEUTRAL'] # Replace with your actual class names
329 report = classification_report(y_true, y_pred, target_names=class_names)
330
331 print(report)
332

```

Appendix Q: Python code for LSTM model for sentiment modelling strategy

```
9 The parameter for LSTM in this file is based on Thara, S., & Poornachandran, P. (2022). Social media text analytics of
  Malayalam-English code-mixed using deep learning. Journal of big Data, 9(1), 45.
10
11
12 pip install tensorflow numpy pandas scikit-learn seaborn
13
14 import numpy as np
15 import pandas as pd
16 import matplotlib.pyplot as plt
17 import seaborn as sns
18 import tensorflow
19 from tensorflow.keras.preprocessing.text import Tokenizer
20 from tensorflow.keras.preprocessing.sequence import pad_sequences
21 from tensorflow.keras.models import Sequential
22 from tensorflow.keras.layers import Embedding, Conv1D, MaxPooling1D, Flatten, Dense, Dropout, LSTM
23 from tensorflow.keras.callbacks import EarlyStopping
24
25 # Change the name of the train file and test file
26 df = pd.read_csv('SMOTE Train Part 2 MESocSentiment.csv')
27 df2 = pd.read_csv('Test Part 2 Ground Truth.csv')
28
29 df.head()
30
31 df.info()
32
33 df.shape
34
35 # The maximum number of words to be used. (most frequent)
36 MAX_NB_WORDS = 50000
37 # Max number of words in the tweet (using the highest value for maximum sequence length in the paper)
38 MAX_SEQUENCE_LENGTH = 16
39 # using the lowest word embedding dimension from the paper
40 EMBEDDING_DIM = 100
41
42 tokenizer = Tokenizer(num_words=MAX_NB_WORDS, Lower=True)
43 tokenizer.fit_on_texts(df['Tweets'].values)
44 word_index = tokenizer.word_index
45 print('Found %s unique tokens.' % len(word_index))
46
```

```
47 X_train = tokenizer.texts_to_sequences(df['Tweets'].values)
48 X_train = pad_sequences(X_train, maxlen=MAX_SEQUENCE_LENGTH)
49 print('Shape of data tensor:', X_train.shape)
50
51 Y_train = pd.get_dummies(df['Sentiment (All)']).values
52 print('Shape of label tensor:', Y_train.shape)
53
54 df2.head()
55
56 df2.info()
57
58 df2.drop(['Sentiment (MESocSentiment)'], axis = 1, inplace = True)
59 df2.head(100)
60
61 df2.shape
62
63 df2.value_counts("Sentiment (Manual)")
64
65 tokenizer.fit_on_texts(df2['Tweets'].values)
66 word_index2 = tokenizer.word_index
67 print('Found %s unique tokens.' % len(word_index2))
68
69 X_test = tokenizer.texts_to_sequences(df2['Tweets'].values)
70 X_test = pad_sequences(X_test, maxlen=MAX_SEQUENCE_LENGTH)
71 print('Shape of data tensor:', X_test.shape)
72
73 Y_test = pd.get_dummies(df2['Sentiment (Manual)']).values
74 print('Shape of label tensor:', Y_test.shape)
75
76 Y_test
77
78 print(X_train.shape, Y_train.shape)
79 print(X_test.shape, Y_test.shape)
80
```

```

81 #Define the model
82 model = Sequential()
83
84 #Define embedding layer
85 model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length=X_train.shape[1]))
86
87 # Add a 1D Spatial Dropout layer
88 model.add(tensorflow.keras.layers.SpatialDropout1D(rate=0.5))# using 0.5 value because the paper stated the used parameters is 0 to 1
89
90 # Add a LSTM layer with tanh activation
91 model.add(LSTM(units=130, activation='tanh'))#hidden units (80-150) and activation used are from the paper
92
93 # Add a Dropout layer
94 model.add(Dropout(rate=0.5))# using 0.5 value because the paper stated the used parameters is 0 to 1
95
96 #Add feedforward neural network layer for classification
97 model.add(Dense(3, activation='softmax')) #units are from number of classes and activation is based on paper
98
99 #Compiling the model
100 #optimizer are from the paper
101 #The paper only state that using cross-entropy, but the code using the specific categorical_crossentropy instead
102 #Using accuracy for metrics
103 #learning rate is using default learning rate
104 model.compile(optimizer=tensorflow.keras.optimizers.Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
105 print(model.summary())
106
107 #using early stopping to stop training where there is no improvement in loss in 3 consecutive epochs
108
109 callback = tensorflow.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
110
111 # Train the model
112 #batch size and epoch are from the smallest values used in the paper
113 history = model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=10, batch_size=32,callbacks=[callback])
114
115 history.history
116
117 #saving the model in HD5 format
118 from tensorflow.keras.models import load_model, save_model

```

```

119
120 # Assuming 'model' is your TensorFlow model
121 model.save('lstm 1 july 2024 model.h5')
122
123 from google.colab import files
124 files.download('lstm 1 july 2024 model.h5')
125
126 # Evaluate the model
127 loss, accuracy = model.evaluate(X_test, Y_test)
128 print(f'Test accuracy: {accuracy * 100:.2f}%')
129 print(f'Test accuracy: {accuracy}%')
130 print(f'Test accuracy: {accuracy:.3f}')
131 print(f'Test accuracy: {accuracy:.4f}')
132
133 # Plot training and validation accuracy
134 plt.plot(history.history['accuracy'], Label='Training Accuracy')
135 plt.plot(history.history['val_accuracy'], Label='Validation Accuracy')
136 plt.title('Training and Validation Accuracy')
137 plt.xlabel('Epoch')
138 plt.ylabel('Accuracy')
139 plt.legend()
140 plt.show()
141
142 # Plot training and validation loss
143 plt.plot(history.history['loss'], Label='Training Loss')
144 plt.plot(history.history['val_loss'], Label='Validation Loss')
145 plt.title('Training and Validation Loss')
146 plt.xlabel('Epoch')
147 plt.ylabel('Loss')
148 plt.legend()
149 plt.show()
150
151 #Graph for loss
152 plt.title('Loss')
153 plt.plot(history.history['loss'], Label='train')
154 plt.plot(history.history['val_loss'], Label='test')
155 plt.legend()
156 plt.show()
157

```

```

158 #Graph for accuracy
159 plt.title('Accuracy')
160 plt.plot(history.history['accuracy'], Label='train')
161 plt.plot(history.history['val_accuracy'], Label='test')
162 plt.legend()
163 plt.show();
164
165 #Prediction
166 predictions_LSTM = model.predict(X_test)
167 test_prediction_LSTM = pd.DataFrame()
168 test_prediction_LSTM['Tweets'] = df2['Tweets']
169
170 classes = np.argmax(predictions_LSTM, axis=1)
171 print(classes)
172
173 test_prediction_LSTM['Classes'] = classes
174
175 #import numpy as np
176
177 labels = ['NEGATIVE', 'POSITIVE', 'NEUTRAL']
178 sentiment = [labels[idx] for idx in np.argmax(predictions_LSTM, axis=1)]
179 print(sentiment)
180
181 test_prediction_LSTM['Sentiment'] = sentiment
182
183 test_prediction_LSTM
184
185 #Change the name of the output file
186 test_prediction_LSTM.to_csv(r"16August2024_LSTM_TestPart2_test_prediction.csv")
187
188 import pandas as pd
189
190 value_counts = test_prediction_LSTM['Sentiment'].value_counts()
191 print(value_counts)
192
193 #create bar graph
194 import matplotlib.pyplot as plt
195 import seaborn as sns
196

```

```

197 # create a barplot
198 sns.countplot(x='Sentiment', data= test_prediction_LSTM)
199
200 # set the title
201 plt.title('Sentiment Analysis')
202
203 # set the x and y label
204 plt.xlabel('Sentiment')
205 plt.ylabel('Number of Tweets')
206
207 # show the plot
208 plt.show()
209
210 #AUC-ROC
211 from sklearn.metrics import roc_curve, roc_auc_score
212 import matplotlib.pyplot as plt
213 from itertools import cycle
214 import numpy as np
215
216 # Calculate ROC AUC for each class
217 n_classes = predictions_LSTM.shape[1]
218 fpr = dict()
219 tpr = dict()
220 roc_auc = dict()
221
222 for i in range(n_classes):
223     fpr[i], tpr[i], _ = roc_curve(Y_test[:, i], predictions_LSTM[:, i])
224     roc_auc[i] = roc_auc_score(Y_test[:, i], predictions_LSTM[:, i])
225
226 # Compute microaverage ROC AUC
227 fpr["micro"], tpr["micro"], _ = roc_curve(Y_test.ravel(), predictions_LSTM.ravel())
228 roc_auc["micro"] = roc_auc_score(Y_test, predictions_LSTM, average="micro")
229

```

```

230 # Compute macroaverage ROC AUC
231 all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))
232 mean_tpr = np.zeros_like(all_fpr)
233 for i in range(n_classes):
234     mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])
235 mean_tpr /= n_classes
236 fpr["macro"] = all_fpr
237 tpr["macro"] = mean_tpr
238 roc_auc["macro"] = roc_auc_score(Y_test, predictions_LSTM, average="macro")
239
240 # Class labels
241 labels = ['NEGATIVE', 'POSITIVE', 'NEUTRAL']
242
243 # Plot ROC curves
244 fig, ax = plt.subplots(figsize=(6, 6))
245
246 # Plot microaverage ROC curve
247 plt.plot(fpr["micro"], tpr["micro"],
248         label=f'micro-average ROC curve (AUC = {roc_auc["micro"]:.2f})',
249         color='deeppink', linestyle=':', linewidth=4)
250
251 # Plot macroaverage ROC curve
252 plt.plot(fpr["macro"], tpr["macro"],
253         label=f'macro-average ROC curve (AUC = {roc_auc["macro"]:.2f})',
254         color='navy', linestyle=':', linewidth=4)
255
256 # Plot ROC curves for each class
257 colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
258 for i, color in zip(range(n_classes), colors):
259     plt.plot(fpr[i], tpr[i], color=color, lw=2,
260             label=f'ROC curve of {labels[i]} (AUC = {roc_auc[i]:.2f})')
261

```

```

262 # Plot random classifier (chance level)
263 plt.plot([0, 1], [0, 1], color='black', linestyle='--', lw=2, label='Chance level (AUC=0.5)')
264
265 plt.xlim([0.0, 1.0])
266 plt.ylim([0.0, 1.05])
267 plt.xlabel('False Positive Rate')
268 plt.ylabel('True Positive Rate')
269 plt.title('Extension of Receiver Operating Characteristic\nto One-vs-Rest multiclass LSTM')
270 plt.legend(loc="lower right")
271 plt.show()
272
273 #Confusion Matrix
274 import numpy as np
275 from sklearn.metrics import confusion_matrix
276 import matplotlib.pyplot as plt
277 import seaborn as sns
278
279 # Assuming predictions_LSTM is your LSTM model's predictions and Y_test is your true labels
280 predictions = np.argmax(predictions_LSTM, axis=1)
281
282 # Define your class labels
283 labels = ['NEGATIVE', 'POSITIVE', 'NEUTRAL']
284
285 # Convert predicted and true labels to sentiment labels
286 predicted_sentiment = [labels[idx] for idx in predictions]
287 true_sentiment = [labels[idx] for idx in np.argmax(Y_test, axis=1)]
288
289 # Compute confusion matrix
290 conf_mat = confusion_matrix(true_sentiment, predicted_sentiment, labels=labels)
291
292 # Plot confusion matrix
293 plt.figure(figsize=(8, 6))
294 sns.heatmap(conf_mat, annot=True, fmt='d', cmap='viridis', xticklabels=labels, yticklabels=labels)
295 plt.xlabel('Predicted label')
296 plt.ylabel('True label')
297 plt.title('Confusion Matrix')
298 plt.show()
299

```

```

300 #Classification Report
301
302 import numpy as np
303 import pandas as pd
304 from sklearn.metrics import classification_report
305
306 # Assuming df2 has your tweets and Y_test has your actual labels
307 # Assuming Y_test is one-hot encoded
308 y_pred = np.argmax(predictions_LSTM, axis=1)
309 y_true = np.argmax(Y_test, axis=1)
310
311 # Generate classification report
312 class_names = ['NEGATIVE', 'POSITIVE', 'NEUTRAL'] # Replace with your actual class names
313 report = classification_report(y_true, y_pred, target_names=class_names)
314
315 print(report)
316

```

Appendix Q: Python code for BERT model for sentiment modelling strategy

```
10 pip install tensorflow transformers pandas numpy scikit-learn keras
11
12 import os
13
14 # Install tf-keras
15 !pip install tf-keras
16
17 # Set TF_USE_LEGACY_KERAS environment variable
18 os.environ['TF_USE_LEGACY_KERAS'] = '1'
19
20 import tensorflow as tf
21 import pandas as pd
22 from transformers import BertTokenizer, TFBertForSequenceClassification
23 from sklearn.model_selection import train_test_split
24 from sklearn.metrics import accuracy_score
25 import numpy as np
26
27 # Change the name of the train file and test file
28 df = pd.read_csv('SMOTE Train Part 2 MESocSentiment.csv')
29 df2 = pd.read_csv('Test Part 2 Ground Truth.csv')
30
31 df.info()
32 df.head()
33
34 df2.info()
35 df2.head()
36
37 df2.drop(['Sentiment (MESocSentiment)'], axis = 1, inplace = True)
38 df2.head(100)
39
40 df2.info()
41
42 #Text length distribution train set
43 df['sent_len'] = df['Tweets'].apply(Lambda x: len(x.split(" ")))
44 max_seq_len_train = np.round(df['sent_len'].mean() + 2 * df['sent_len'].std()).astype(int)
45 max_seq_len_train
46
47 #Text length distribution test set
48 df2['sent_len'] = df2['Tweets'].apply(Lambda x: len(x.split(" ")))
49 max_seq_len_test = np.round(df2['sent_len'].mean() + 2 * df2['sent_len'].std()).astype(int)
50 max_seq_len_test
51
52 # Create a new variable to hold the highest max_seq_len value
53 highest_max_seq_len = max(max_seq_len_train, max_seq_len_test)
54
55 print("The highest value of max_seq_len is:", highest_max_seq_len)
56
57 # Load the pre-trained multilingual BERT tokenizer
58 tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased')
59
60 #attention mask untuk train set
61 input_sequences_train = []
62 # The attention mask is an optional argument used when batching sequences together.
63 # The attention mask is a binary tensor indicating the position of the padded indices so that the model does not attend to them.
64 attention_masks_train = []
65
66 for text in df['Tweets']:
67     sequence_dict = tokenizer.encode_plus(text, max_length=highest_max_seq_len, pad_to_max_length=True)
68     input_ids = sequence_dict['input_ids']
69     att_mask = sequence_dict['attention_mask']
70
71     input_sequences_train.append(input_ids)
72     attention_masks_train.append(att_mask)
73
74 df['input_sequences_train'] = input_sequences_train
75 df['attention_masks_train'] = attention_masks_train
76 df.head()
77
78 #attention mask untuk test set
79 input_sequences_test = []
80 # The attention mask is an optional argument used when batching sequences together.
81 # The attention mask is a binary tensor indicating the position of the padded indices so that the model does not attend to them.
82 attention_masks_test = []
83
```

```

84 for text in df2['Tweets']:
85     sequence_dict = tokenizer.encode_plus(text, max_length=highest_max_seq_len, pad_to_max_length=True)
86     input_ids = sequence_dict['input_ids']
87     att_mask = sequence_dict['attention_mask']
88
89     input_sequences_test.append(input_ids)
90     attention_masks_test.append(att_mask)
91
92 df2['input_sequences_test'] = input_sequences_test
93 df2['attention_masks_test'] = attention_masks_test
94 df2.head()
95
96 Y_train = pd.get_dummies(df2['Sentiment (All)']).values
97 print('Shape of label tensor:', Y_train.shape)
98
99 Y_test = pd.get_dummies(df2['Sentiment (Manual)']).values
100 print('Shape of label tensor:', Y_test.shape)
101
102 df.info()
103
104 df2.info()
105
106 #input sequences, attention mask, and sentiment (All)--sentiment labels
107 X_train = df['input_sequences_train'].tolist()
108 X_train = tf.constant(X_train)
109 Y_train = tf.constant(Y_train)
110 att_masks_train = df['attention_masks_train'].tolist()
111 att_masks_train = tf.constant(att_masks_train)
112
113
114 X_test = df2['input_sequences_test'].tolist()
115 X_test = tf.constant(X_test)
116 Y_test = tf.constant(Y_test)
117 att_masks_test = df2['attention_masks_test'].tolist()
118 att_masks_test = tf.constant(att_masks_test)
119
120 print(f'Train | X shape: {X_train.shape}, att_mask shape: {att_masks_train.shape}, y shape: {Y_train.shape}')
121 print(f'Test | X shape: {X_test.shape}, att_mask shape: {att_masks_test.shape}, y shape: {Y_test.shape}')
122
123
124 # Build the model
125 model = TFBertForSequenceClassification.from_pretrained('bert-base-multilingual-cased', num_labels=3)
126
127 optimizer = tf.keras.optimizers.Adam(learning_rate=2e-5)
128
129 model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
130
131 model.summary()
132
133 #Model fit
134 history = model.fit([X_train, att_masks_train], Y_train, batch_size=16, epochs=3, validation_data=([X_test, att_masks_test], Y_test))
135
136 history.history
137
138 import matplotlib.pyplot as plt
139
140 # Plot training and validation accuracy
141 plt.plot(history.history['accuracy'], label='Training Accuracy')
142 plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
143 plt.title('Training and Validation Accuracy')
144 plt.xlabel('Epoch')
145 plt.ylabel('Accuracy')
146 plt.legend()
147 plt.show()
148
149 # Plot training and validation loss
150 plt.plot(history.history['loss'], label='Training Loss')
151 plt.plot(history.history['val_loss'], label='Validation Loss')
152 plt.title('Training and Validation Loss')
153 plt.xlabel('Epoch')
154 plt.ylabel('Loss')
155 plt.legend()
156 plt.show()
157
158 #Model evaluation after training
159 loss, metric = model.evaluate([X_test, att_masks_test], Y_test, batch_size=16, verbose=0)
160 print(f"Loss after training: {loss:.4f}, Accuracy after training: {metric:.2%}")
161 print(f"Loss after training: {loss:.4f}, Accuracy after training: {metric:.4f}")

```

```

162
163 #Test Accuracy
164 loss, accuracy = model.evaluate([X_test, att_masks_test], Y_test)
165 print(f'Test accuracy: {accuracy * 100:.2f}%')
166 print(f'Test accuracy: {accuracy:.4f}%')
167 print(f'Test accuracy: {accuracy:%}')
168
169 #Prediction
170 predictions_BERT = model.predict([X_test, att_masks_test])
171 test_prediction_BERT = pd.DataFrame()
172 test_prediction_BERT['Tweets'] = df2['Tweets']
173
174 print(predictions_BERT)
175
176 import numpy as np
177
178 # Assuming predictions_BERT is the TFSequenceClassifierOutput object
179 logits = predictions_BERT.logits
180
181 # Determine the predicted class for each sample
182 predicted_classes = np.argmax(logits, axis=1)
183
184 # Print the predicted classes
185 print(predicted_classes)
186
187 test_prediction_BERT['Classes'] = predicted_classes
188
189 labels = ['NEGATIVE', 'POSITIVE', 'NEUTRAL']
190 sentiment = [labels[idx] for idx in np.argmax(logits, axis=1)]
191 print(sentiment)
192
193 test_prediction_BERT['Sentiment'] = sentiment
194
195 test_prediction_BERT
196
197 #Change the name of the output file
198 test_prediction_BERT.to_csv(r"69.30%16August2024_BERT_TestPart2_test_prediction(mBERT adam 16 2e-5 epoch 3).csv")
199

```

```

200 import pandas as pd
201
202 value_counts = test_prediction_BERT['Sentiment'].value_counts()
203
204 print(value_counts)
205
206 #create bar graph
207 import matplotlib.pyplot as plt
208 import seaborn as sns
209
210 # create a barplot
211 sns.countplot(x='Sentiment', data= test_prediction_BERT)
212
213 # set the title
214 plt.title('Sentiment Analysis')
215
216 # set the x and y label
217 plt.xlabel('Sentiment')
218 plt.ylabel('Number of Tweets')
219
220 # show the plot
221 plt.show()
222
223 #Confusion Matrix
224 import numpy as np
225 import matplotlib.pyplot as plt
226 import seaborn as sns
227 from sklearn.metrics import confusion_matrix
228
229 # Assuming predictions_BERT is the TFSequenceClassifierOutput object
230 logits = predictions_BERT.logits
231
232 # Define your class labels
233 labels = ['NEGATIVE', 'POSITIVE', 'NEUTRAL']
234
235 # Determine the predicted class for each sample and map to class names
236 predicted_classes = np.argmax(logits, axis=1)
237 predicted_sentiment = [labels[idx] for idx in predicted_classes]
238

```

```

239 # Assuming Y_test contains your actual labels
240 # If Y_test is in one-hot encoded format, convert it to class indices
241 Y_test_single = np.argmax(Y_test, axis=1) if len(Y_test.shape) > 1 else Y_test
242 true_sentiment = [labels[idx] for idx in Y_test_single]
243
244 # Calculate confusion matrix
245 conf_matrix = confusion_matrix(true_sentiment, predicted_sentiment, labels=labels)
246
247 # Display confusion matrix
248 plt.figure(figsize=(6, 6))
249 sns.heatmap(conf_matrix, annot=True, cmap='viridis', fmt='d', cbar=False,
250             xtickLabels=labels, ytickLabels=labels)
251 plt.xlabel('Predicted labels')
252 plt.ylabel('True labels')
253 plt.title('Confusion Matrix')
254 plt.show()
255
256 #Classification Report
257 import numpy as np
258 from sklearn.metrics import classification_report
259
260 # Assuming predictions_BERT is the TFSequenceClassifierOutput object
261 logits = predictions_BERT.logits
262
263 # Define the class labels
264 labels = ['NEGATIVE', 'POSITIVE', 'NEUTRAL']
265
266 # Determine the predicted class for each sample
267 predicted_classes = np.argmax(logits, axis=1)
268
269 # Map predicted class indices to class names
270 sentiment = [labels[idx] for idx in predicted_classes]
271
272 # Convert one-hot encoded true labels back to categorical labels
273 true_labels = np.argmax(Y_test, axis=1)
274
275 # Generate the classification report with class names
276 report = classification_report(true_labels, predicted_classes, target_names=labels)
277 print(report)

```

```

278
279 # Print the predicted sentiments
280 print(sentiment)
281
282 #AUC-ROC
283 import numpy as np
284 import tensorflow as tf
285 from sklearn.metrics import roc_curve, auc
286 import matplotlib.pyplot as plt
287 from itertools import cycle
288 import pandas as pd
289
290 # Assuming predictions_BERT is the TFSequenceClassifierOutput object
291 logits = predictions_BERT.logits
292
293 # Determine the predicted class for each sample
294 predicted_classes = np.argmax(logits, axis=1)
295
296 # Print the predicted classes
297 print(predicted_classes)
298
299 # Convert logits to TensorFlow tensor
300 logits_tensor = tf.constant(logits, dtype=tf.float32)
301
302 # Apply softmax to get predicted probabilities
303 predicted_probabilities = tf.nn.softmax(logits_tensor, axis=-1).numpy()
304
305 # Example Y_test using one-hot encoded labels
306 Y_test = pd.get_dummies(df2['Sentiment (Manual)']).values
307
308 # Check if the shapes match
309 print(f"Shape of logits: {logits.shape}")
310 print(f"Shape of predicted_probabilities: {predicted_probabilities.shape}")
311 print(f"Shape of Y_test: {Y_test.shape}")
312

```

```

313 # Calculate ROC AUC for each class
314 fpr = dict()
315 tpr = dict()
316 roc_auc = dict()
317 n_classes = logits.shape[1]
318
319 for i in range(n_classes):
320     fpr[i], tpr[i], _ = roc_curve(Y_test[:, i], predicted_probabilities[:, i])
321     roc_auc[i] = auc(fpr[i], tpr[i])
322
323 # Calculate microaverage ROC curve and AUC
324 fpr["micro"], tpr["micro"], _ = roc_curve(Y_test.ravel(), predicted_probabilities.ravel())
325 roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
326
327 # Calculate macroaverage ROC curve and AUC
328 all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))
329 mean_tpr = np.zeros_like(all_fpr)
330 for i in range(n_classes):
331     mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])
332 mean_tpr /= n_classes
333
334 fpr["macro"] = all_fpr
335 tpr["macro"] = mean_tpr
336 roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])
337
338 # Class labels
339 labels = df2['Sentiment (Manual)'].unique() # Replace with your actual class labels
340
341 # Plot ROC curves
342 fig, ax = plt.subplots(figsize=(6, 6))
343
344 # Plot microaverage ROC curve
345 plt.plot(fpr["micro"], tpr["micro"],
346         label=f'micro-average ROC curve (AUC = {roc_auc["micro"]:.2f})',
347         color='deeppink', linestyle=':', linewidth=4)
348
349 # Plot macroaverage ROC curve
350 plt.plot(fpr["macro"], tpr["macro"],
351         label=f'macro-average ROC curve (AUC = {roc_auc["macro"]:.2f})',
352         color='navy', linestyle=':', linewidth=4)
353
354 # Plot ROC curves for each class
355 colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
356 for i, color in zip(range(n_classes), colors):
357     plt.plot(fpr[i], tpr[i], color=color, lw=2,
358             label=f'ROC curve of {labels[i]} (AUC = {roc_auc[i]:.2f})')
359
360 # Plot random classifier (chance level)
361 plt.plot([0, 1], [0, 1], color='black', linestyle='--', lw=2, label='Chance level (AUC=0.5)')
362
363 plt.xlim([0.0, 1.0])
364 plt.ylim([0.0, 1.05])
365 plt.xlabel('False Positive Rate')
366 plt.ylabel('True Positive Rate')
367 plt.title('Extension of Receiver Operating Characteristic\n\tOne-vs-Rest multiclass BERT')
368 plt.legend(loc="lower right")
369 plt.show()
370
371 # Save the model in the TensorFlow SavedModel format
372 model.save('bert_model_16aug2024_adam_16_2e_5_epoch_3_mBERT', save_format='tf')
373
374 # Zip the saved model directory
375 !zip -r bert_model_16aug2024_adam_16_2e_5_epoch_3_mBERT.zip bert_model_16aug2024_adam_16_2e_5_epoch_3_mBERT
376
377 # Save the model in the H5 format
378 #Change the name of the model
379 model.save_weights('bert_model_16aug2024_adam_16_2e_5_epoch_3_mBERT', save_format='h5')
380
381 # Zip the saved model directory
382 !zip -r bert_model_16aug2024_adam_16_2e_5_epoch_3_mBERT.zip bert_model_16aug2024_adam_16_2e_5_epoch_3_mBERT
383
384 # Download the zipped model
385 from google.colab import files
386 files.download('bert_model_16aug2024_adam_16_2e_5_epoch_3_mBERT.zip')

```