SCHOOL OF GRADUATE STUDIES

AMHARIC-ENGLISH CODE-MIXED LANGUAGE IDENTIFICATION ON SOCIAL MEDIA USING MACHINE LEARNING AND DEEP LEARNING APPROACH

MSc. THESIS

YETMWORK TESFAYE ASFAW

JANUARY, 2024

WOLKITE, ETHIOPIA

Wolkite University

School of Graduate Studies

Amharic-English Code-Mixed Language Identification on Social Media Using Machine Learning and Deep Learning Approach

A Thesis Submitted to School of Graduate Studies, in Partial Fulfillment of The Requirements for the Degree of Master of Science in Computer Science and Engineering (Specialization: Computer Science)

Yetmwork Tesfaye Asfaw

Major Advisor: Mesfin Abebe (Ph.D.)

Co-Advisor: Ermias Nigatu (M.Sc.)

January, 2024

Wolkite, Ethiopia

APPROVAL SHEET

School of Graduate Studies

Wolkite University

Amharic-English Code-Mixed Language Identification on Social Media Using Machine Learning and Deep Learning Approach

Submitted by:

Yetmwork Tesfaye		04/01/2024
Name of Student	Signature	Date

Approved by:

1. Dr. MESFIN ABEBE		05/01/2024
Major Advisors Name	Signature	Date
2. Emiros Nigatu		05/01/2024
Co- Advisors Name	Signature	Date
3. _____		05/01/2024
Department Head	Signature	Date
4. _____	_____	_____
Name of Chairman, DGC	Signature	Date
5. Teshome Yihunie Birlie		05/01/2024
Name of Dean, SGS	Signature	Date

WOLKITE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

We hereby certify that we have read and evaluated Thesis titled **"Amharic-English Code-Mixed Language Identification on Social Media Using Machine Learning and Deep Learning Approach"** prepared under our guidance by Yetmwork Tesfaye Asfaw. We recommend that the Thesis shall be submitted as fulfilling the requirements for the award of a MSc. degree in computer science and engineering.

Dr. ADUGNA ABEBE		05/01/2024
Major Advisor	signature	date
Gmicas Nigatu		05/01/2024
Co-Advisor	signature	date

As members of the Board of Examiner of the Master of Science Thesis open defense examination, we have read and evaluated this Thesis prepared by **Yetmwork Tesfaye Asfaw** and examined the candidate. We hereby certify that; the thesis is accepted for fulfilling the requirements for the award of the degree of Master of Science (M.Sc.) in computer science and engineering.

1.	Sintayehu H. (PhD)		04-Jan-2024
	Name of external examiner	signature	date
2.	WORKU ALLEWY		04/01/2024
	Name of internal examiner	signature	date
3.	Abdulmenik Tamer		04/01/24
	Name of chairman	signature	date

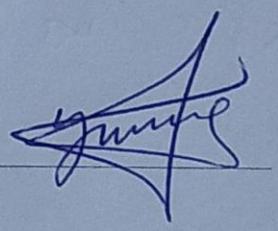
Final approval and acceptance of the Thesis is contingent upon the submission of its final copy to the Council of Postgraduate Program (CPGS) through the candidate's department or school graduate committee (DGC or SGC).

DECLARATION

By my signature below, I declare and affirm that this Thesis is my own work. I have followed all ethical principles of scholarship in the preparation, data collection, data analysis and completion of this thesis. All scholarly matter that is included in the thesis has been given recognition through citation. I affirm that I have cited and referenced all sources used in this document. Every serious effort has been made to Avoid any plagiarism in the preparation of this thesis.

This thesis is submitted in partial fulfillment of the requirement for a degree from the School of Graduate Studies at Wolkite University. The thesis is deposited in the Wolkite University Library and is made available to borrowers under the rules of the library. I solemnly declare that this thesis has been submitted to any other institution anywhere for the award of any academic degree, diploma or certificate.

Brief quotations from this Thesis may be used without special permission provide that accurate and complete acknowledgement of the source is made. Request for permission for extended quotations from, or reproduction of, this thesis in whole or in part may be granted by the Head of the School or Department or the Dean of the School of Graduate Studies when in his or her judgement the proposed use of the material is in the interest of scholarship. In all other instances, however, permission must be obtained from the author of the thesis.

Name: Yetmwork Tesfaye Signature: 

Date: January/4/2023

School/Department: Computer Science and Engineering

ACKNOWLEDGEMENT

First of all, I would like to thank my God who made all things possible. Next, I want to express my sincere thanks to my advisor Dr. Mesfin Abebe for his encouragement and comments from the beginning till the end. Next, I would like to thank my co-advisor, Mr. Ermias Nigatu, for their guidance and support. Next, I would like to thank Mr. Yidnekachew Feleke for helping me to annotate the dataset for my research. Next, I would like to express my gratitude to my family for giving me the chance to become the person I am today. finally, I would like to thank my friends for sharing their knowledge, advice, and encouragement during my research.

LIST OF ABBREVIATIONS

API	Application Programming Interface
BERT	Bidirectional-Encoder Representations from Transformer
Bi-LSTM	Bidirectional Long Short-Term Memory
CM	Code Mixed
CNN	Convolutional Neural Network
CRF	Conditional Random Field
CSV	Comma-Separated Value
ELECTRA	Efficiently Learning an Encoder that Classifies Token Replacements Accurately
GB	Gigabyte
GHz	Gigahertz
LID	Language Identification
LSTM	Long Short-Term Memory
MaxEnt	Maximum Entropy
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
POS	Parts Of Speech
Ref.	Reference
RF	Random Forest

ROBERTA	Robustly Optimized Bidirectional Encoder Representations Transformers Approach
SOTA	State-Of-The-Art
SVM	Support Vector Machine
TF-IDF	Term Frequency-Inverse Document Frequency
WSGI	Web Server Gateway Interface
XGBoost	Extreme Gradient Boosting
XLM	Cross-Lingual Language Model

TABLE OF CONTENTS

APPROVAL SHEET ii

DECLARATION iv

ACKNOWLEDGEMENT v

LIST OF ABBREVIATIONS vi

LIST OF TABLES xv

LIST OF FIGURES xvi

LIST OF EQUATIONS xviii

LIST OF FIGURES IN THE APPENDIX xix

ABSTRACT xx

CHAPTER ONE 1

1. INTRODUCTION 1

1.1. Background of the Study 1

1.2. Motivation of the Study 3

1.3. Statement of the Problem 3

1.4. Objective of the Study 5

1.4.1. General Objective 5

1.4.2. Specific Objectives 5

1.5. Significance of the Study 5

1.6. Scope and Limitation of the Study 6

1.7. Organization of the Study 6

CHAPTER TWO 7

2. LITERATURE REVIEW AND RELATED WORK 7

2.1. Social Media 7

2.2. Natural-Language Processing 8

2.3. Amharic Language 9
2.3.1. Amharic Alphabet 9
2.3.2. Amharic Morphology 12
2.3.3. Amharic Phrase 14
2.3.4. Amharic Sentence Structure 15
2.3.5. Amharic Punctuation Mark 16
2.4. English Language 16
2.4.1. English Alphabet 16
2.4.2. English Morphology 16
2.4.3. English Phrase 18
2.4.4. English Sentence Structure 19
2.4.5. English Article 19
2.4.6. English Punctuation Mark 20
2.5. Code-Mixing 20
2.5.1. Types of Code-Mixing 21
2.6. Language Identification 22
2.7. Machine Learning Approach 23
2.7.1. Naive Bayesian 23
2.7.2. Support Vector Machine 24
2.7.3. Random Forest 24
2.7.4. Decision Tree 24
2.7.5. Logistic Regression 25
2.7.6. Maximum Entropy 25
2.7.7. XGBoost 25
2.7.8. Conditional Random Field 26

2.8. Deep Learning Approach 26
2.8.1. Convolutional Neural Network 27
2.8.2. Multi-Layer Perceptron 27
2.8.3. Long Short-Term Memory Networks 28
2.8.4. Bidirectional Long Short-Term Memory Networks 28
2.9. Related Work 28
CHAPTER THREE 34
3. RESEARCH METHODOLOGY 34
3.1. Overview 34
3.2. Amharic-English Code-Mixed Dataset 34
3.2.1. Sources of Data 35
3.2.2. Dataset Preparation 36
3.2.3. Dataset Annotation 36
3.3. Amharic-English Language Identification Text Pre-Processing Techniques 37
3.3.1. Data Cleaning 37
3.3.2. Word Tokenization 38
3.3.3. Morphology-Based POS Tagging 39
3.4. Amharic-English Language Identification Feature Extraction Techniques 39
3.4.1. Count Vectorizer 39
3.4.2. TF-IDF Vectorization 40
3.4.3. Word to Vector 41
3.4.4. Bi and Tri Gram Vectorization 42
3.5. Selected Models Algorithm 42
3.6. Classification Metrics 45
3.6.1. Confusion Metrics 46

3.6.2. Accuracy 46
3.6.3. Precision 46
3.6.4. Recall 46
3.6.5. F1-Score 47
3.7. Tools 47
3.7.1. Data Preparation Tool 47
3.7.2. Data Processing Tools 47
3.7.3. Package Manager and Environments Tools 48
3.7.4. Modeling and Documentation Tools 48
3.7.5. Hardware Tools 50
3.7.6. Deployment Tools 50
CHAPTER FOUR 51
4. PROPOSED APPROACH 51
4.1. Preprocessing techniques 52
4.2. Proposed Models for Amharic-English Language Identification 53
4.2.1. Feature Extraction and CNN Model 53
4.2.2. Feature Extraction and LSTM Model 54
4.2.3. Feature Extraction and Bidirectional LSTM Model 55
4.2.4. Feature Extraction and Multi-Layer Perceptron Model 56
4.3. Model Evaluation 56
4.4. Model Deployment 57
CHAPTER FIVE 58
5. IMPLEMENTATION OF CODE-MIXED LANGUAGE IDENTIFICATION MODEL 58
5.1. Data loading 58

5.2. Data Pre-Processing 58
5.2.1. Cleaning 58
5.2.2. Tokenization 59
5.2.3. Part-of-Speech (POS) Taggers 59
5.2.4. Split Data 60
5.2.5. Texts_to_sequences 60
5.2.6. Sequence Padding 60
5.3. Feature Extraction 61
5.3.1. Count Vectorizer 61
5.3.2. TF-IDF Vectorizer 61
5.3.3. Word to Vector 62
5.3.4. Bi and Tri Gram 62
5.4. Machine Learning Techniques for Amharic-English Code-Mixed Language Identification 63
5.4.1. Logistic Regression 63
5.4.2. Naïve Bayes Classifier 63
5.4.3. Decision Tree 64
5.4.4. Support Vector Machine 64
5.4.5. Random Forest 65
5.4.6. Maximum Entropy Classifier 65
5.4.7. XGBoost Classifier 65
5.4.8. Conditional Random Field 66
5.5. Build Deep Learning Models for Amharic-English Code-Mixed Language Identification 68
5.5.1. Convolutional Neural Network 68
5.5.2. Multi-Layer Perceptron 68

5.5.3. Long Short-Term Memory 69
5.5.4. Bidirectional Long Short-Term Memory 70
5.6. Models Layer and Parameters 70
5.6.1. Dropout 70
5.6.2. Dense Layer 71
5.6.3. Optimizer 71
5.6.4. Activation Function 71
5.7. Model Deployment 71
CHAPTER SIX 73
6. RESULTS AND DISCUSSIONS 73
6.1. Performance Evaluation of Machine learning and Deep Learning Models 73
6.1.1. Confusion Matrix of Machine Learning Models 73
6.1.2. Classification Report of Machine Learning Models 80
6.1.3. Result of Deep Learning Models 83
6.2. Discussions 95
CHAPTER SEVEN 100
7. CONCLUSION AND FUTURE WORK 100
7.1. Conclusion 100
7.2. Future Work 101
References 102
APPENDICES 110
Appendix A. Confusion Matrix 110
Appendix A.1. Confusion matrix of Naïve Bayes Classifier with count vector, TFIDF, Word to Vector and bi and tri gram. 110
Appendix A.2. Confusion matrix of XGBoost Classifier with count vector, TFIDF, bi and tri gram, and Word to Vector 111

Appendix A.3. Confusion matrix of Maximum Entropy Classifier with count vector, TFIDF, Word to Vector and bi and tri gram. 112

Appendix A.4. Confusion matrix of Conditional Random Field Classifier 113

Appendix B. Classification Report 114

Appendix B.1. Classification Report of Naïve Bayes Classifier with count vector, TFIDF, Word to Vector and bi and tri gram. 114

Appendix B.2. Classification Report of XGBoost Classifier with count vector, TFIDF, Word to Vector and bi and tri gram. 115

Appendix B.3. Classification Report of Maximum Entropy Classifier with count vector, TFIDF, Word to Vector and bi and tri gram. 116

Appendix B.4. Classification Report of Conditional Random Field Classifier 117

LIST OF TABLES

Table 2.1.The Amharic Fidel 11
Table 2.2. List of English Punctuations 20
Table 2.3. Related Work 30
Table 3.1. Data sources 36
Table 3.2. Statistic of Datasets 37
Table 3.3. English Language Morphology based POS Tagging 39
Table 5.1. Models Hyper Parameter 71
Table 6.1. Model Summary with Count Vector 92
Table 6.2. Model Summary with TFIDF Vector 93
Table 6.3. Model Summary with Word to Vector 94
Table 6.4. Model Summary with Bi&Tri Gram 94
Table 6.5. Deep Learning Models Summary 99

LIST OF FIGURES

Figure 2.1. Amharic Levelized Characters 10
Figure 3.1. Research Flow 34
Figure 3.2. dataset preparation 35
Figure 3.3. Text Preprocessing 37
Figure 4.1. Proposed Model System Architecture 51
Figure 4.2. Proposed Convolutional Neural Network Model 53
Figure 4.3. Proposed Long Short Term Memory Model 54
Figure 4.4. Proposed Bidirectional Long Short Term Memory Model 55
Figure 4.5. Proposed Multi-Layer Perceptron Model 56
Figure 4.6. Deployment Architecture of Code-Mixed Language Identification 57
Figure 5.1. Data Loading Sample Code 58
Figure 5.2. Data Cleaning Sample Code 59
Figure 5.3. Tokenization Sample Code 59
Figure 5.4. POS tagging Sample Code 59
Figure 5.5. Split the Dataset Sample Code 60
Figure 5.6. Pad Sequence Sample Code 61
Figure 5.7. Count Vectorizer Sample Code 61
Figure 5.8. TFIDF Vectorizer Sample Code 62
Figure 5.9. Word to Vector Vectorizer Sample Code 62
Figure 5.10. Bi and Tri gram Vectorizer Sample Code 62
Figure 5.11. Word Feature Extraction Sample Code for CRF 67
Figure 5.12. Amharic-English Code-Mixed Language Identification Interface 72
Figure 5.13. Amharic-English Code-Mixed Language Identification Prototype 72
Figure 6.1. Confusion Matrix of SVM 74
Figure 6.2. Confusion Matrix of Decision Tree 76
Figure 6.3. Confusion Matrix of Random Forest 77
Figure 6.4. Confusion Matrix of Logistic Regression 79
Figure 6.5. Classification Report of SVM 80
Figure 6.6. Classification Report of Decision Tree 81

Figure 6.7. Classification Report of Random Forest 82
Figure 6.8. Classification Report of Logistic Regression 83
Figure 6.9. LSTM Model Summary 84
Figure 6.10. LSTM Model Training Vs Validation Accuracy 84
Figure 6.11. Confusion Matrix of LSTM 85
Figure 6.12. Classification Report of LSTM 85
Figure 6.13. CNN Model Summary 86
Figure 6.14. CNN Model Training Vs Validation Accuracy 86
Figure 6.15. Confusion Matrix of CNN 87
Figure 6.16. Classification Report of CNN 87
Figure 6.17. MLP Model Summary 88
Figure 6.18. MLP Model Training Vs Validation Accuracy 88
Figure 6.19. Confusion Matrix of MLP 89
Figure 6.20. Classification Report of MLP 89
Figure 6.21. Bi-LSTM Model Summary 90
Figure 6.22. Bi-LSTM Model Training Vs Validation Accuracy 90
Figure 6.23. Confusion Matrix of Bi-LSTM 91
Figure 6.24. Classification Report of Bi-LSTM 91
Figure 6.25. Models Accuracy Result Comparison Chart 96

LIST OF EQUATIONS

Equation 3.1 TF 41
Equation 3.2 IDF 41
Equation 3.3 TF-IDF 41
Equation 3.4 Accuracy 46
Equation 3.5 Precision 46
Equation 3.6 Recall 46
Equation 3.7 F1-Score 47

LIST OF FIGURES IN THE APPENDIX

1. Confusion matrix of Naïve Bayes Classifier with count vector, TFIDF, Word to Vector and bi and tri gram 110
2. Confusion matrix of XGBoost Classifier with count vector, TFIDF, bi and tri gram, and Word to Vector. 111
3. Confusion matrix of Maximum Entropy Classifier with count vector, TFIDF, Word to Vector and bi and tri gram. 112
4. Confusion matrix of Conditional Random Field Classifier 113
5. Classification Report of Naïve Bayes Classifier with count vector, TFIDF, Word to Vector and bi and tri gram. 114
6. Classification Report of XGBoost Classifier with count vector, TFIDF, Word to Vector and bi and tri gram. 115
7. Classification Report of Maximum Entropy Classifier with count vector, TFIDF, Word to Vector and bi and tri gram. 116
8. Classification Report of Conditional Random Field Classifier 117

ABSTRACT

In recent years, the study of identifying languages in social media text has been a fascinating area of research. Earlier English language was dominantly used in social media communication, but code-mixed text in social media is prevalent in non-English-speaking states. In code-mixed data, a single sentence combines two different languages, making it challenging for people to determine the language used in the text. Therefore, the use of language identification for processing such code-mixed sentences is essential in language processing tasks. In this proposed work we employed different machine learning and deep learning techniques to identify Amharic-English code-mixed text from social media platforms like Facebook, YouTube, TikTok, and Telegram. For the proposed models we prepared 5021 sentences from these social media, and we applied scraping, cleaning, filtration, and then tokenization and POS tagging using NLTK. After tokenization was performed, we labeled each language tag and Amharic POS tag manually, the total number of words was thirty-one thousand three hundred five (31,305) prepared. The process of categorizing Amharic-English code-mixed data into Amharic, English, Named Entity, and Universal was executed. In this study, we employed machine learning and deep learning methods, we trained our models with count vector, TF-IDF vector, bi and tri gram, and word to vector. After model training was finished, we evaluated the effectiveness of each model using accuracy, precision, recall, and F1-Score, and for user-friendly interaction, we saved the model and deployed using the Flask web Framework. The result shows SVM, logistic regression, and naïve bayes achieves an accuracy of 96%, 86%, and 86%, with TF-IDF and count vector respectively, and decision tree, XGBoost, random forest, and MaxEnt performs an accuracy of 94%, 93%, 93% and 87% with word to vector respectively. And LSTM, CNN, Bi-LSTM, MLP gained accuracy of 84%,87%,87%, and 87%. Generally, SVM is best fit for our code-mixed language identification. From deep learning algorithm MLP is best in terms of f1-score of 0.90,0.95,0.64, and 0.55 for Amharic, English, named entity and universal language label.

Keywords: *SVM, Code-Mixed, Language identification, MLP, Decision tree, Machine learning*

CHAPTER ONE

1. INTRODUCTION

1.1. Background of the Study

In recent times, a huge volume of data has been generated on social media during individuals engage in conversations about their interests, hobbies, product reviews, movies, and more. While these platforms primarily used the English language in the past, it's now a prevalent trend to blend various languages together[1]. The usage of multiple languages in everyday conversations led to the generation of Code-Mixed content. The produced code-mixed data can be used to extract essential knowledge like emotion and news. The practice of mixing languages, known as code-mixing, is commonly observed on social media platforms as a way to convey opinions about specific subjects[2], [3].

Automatic handling and comprehension of content from social media platforms is an interesting area in the natural language processing research community[4]. When linguistic elements from one language are present in a text written in another language, it's defined as code-mixing. As speakers blend languages within a conversation, sentence, or even phrase, the need for automatic language identification becomes more significant to enable smoother processing[5].

Amharic belongs to the Afro-Asiatic language family and belongs to the southwest Semitic language group, and after Arabic, it is the second most widely spoken Semitic language on a global scale[6]. On social media, Amharic-English code-mixed comments are widely available due to people speaking both two languages. using these two languages in conversation text causes language identification challenges.

Identifying the language used in Amharic-English code-mixed content on social media is one of the tasks within the field of Natural Language Processing field and predict languages from conversation or sentences before using the information for the purpose we want; it is important to know the language. Word-level code-mixed language identification refers to the task of determining the languages exist in a text that contains a mixture of multiple languages at the

word level. This task is commonly encountered in multilingual societies or online platforms where users mix languages within their text.

Machine learning and deep learning is instrumental in language identification by automating the process of determining the language of a given text[7]. Through the analysis of large datasets, machine learning and deep learning algorithms can learn patterns and features specific to different languages. This enables the development of accurate and efficient language identification models, benefiting tasks such as multilingual document processing and information retrieval. By leveraging supervised and unsupervised learning techniques, machine learning and deep learning facilitates automated language identification, improving accuracy and efficiency in language-related applications. There are many machine learning techniques like Nave Bayes, Random Forest, Logistic Regression, Decision Tree, Support Vector Machines, Conditional Random Field, XGBoost, and MaxEnt and also deep learning techniques such as Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (Bi-LSTM), Convolutional Neural Network (CNN) and Multi-Layer Perceptron (MLP) are used in this research.

Feature extraction is a critical component in language identification, as it allows for the transformation of raw text data into numerical representations that capture the essential characteristics of different languages. Methods such as Word2Vec, TF-IDF, Count Vectorization, and others enable the conversion of textual data into feature vectors, which serve as input for language identification models. By capturing semantic similarities, term frequencies, and word counts, these techniques facilitate the identification of language-specific patterns and linguistic properties. Effective feature extraction enhances the accuracy and efficiency of language identification systems.

In this study, we have built a language identification model for Amharic-English Code-Mixed text by utilizing machine learning and deep learning methods. We have analyzed and identified the language in each word. To our knowledge, this represents our initial achievement in performing language identification for Latin-script Amharic-English code-mixed text on social media.

1.2. Motivation of the Study

In today's interconnected world, people increasingly share diverse ideas and information through social media, playing a crucial role in the advancement and enrichment of our global civilization. When individuals express their thoughts on these platforms, they do so in a multitude of languages. Notably, in Ethiopia, the practice of using Latin letters to convey messages that blend Amharic with English has become prevalent. This linguistic fusion presents a unique challenge for comprehension due to variations in letter usage and informal writing conventions among different users.

One of the primary motivating factors behind this research is the notable gap in the existing literature; there is a lack of comprehensive work that specifically identifies and analyzes Amharic and English code-mixed language. Furthermore, it's worth noting that various countries have developed methods to distinguish their native language from English within code-mixed content, underscoring the practical relevance of this research.

In this study, our objective is to identify and delve into the complexities of code-mixed Amharic-English language, thereby shedding light on its distinct linguistic features and the challenges it poses. By doing so, we contribute to a deeper understanding of the dynamics of multilingual communication, which has become increasingly vital in our interconnected and globalized society.

1.3. Statement of the Problem

Language identification from social media text is essential in natural language processing. Amharic-English code-mixed comments or posts on social media like Facebook, YouTube, and others are a challenging task because people cannot understand the posts or tweets that are written in code-mixed language.

Usually, people write both Amharic and English code-mixed texts on social media. The reason for this type of writing is that people try to convey a message that has an Amharic meaning by using Latin letters because they know both languages[8]. When writing a sentence that combines two languages, it can be challenging for the reader to discern the language in use.

This difficulty arises because different individuals employ distinct Latin letters to convey a message with Amharic significance[9].

The Amharic phrase is transcribed using the Latin script rather than its native Amharic script. This introduces the potential for phonetic variations since the Latin script lacks a one-to-one relationship between pronunciation and script. This, in turn, increases the complexity when processing code-mixed text computationally[10]. In natural language processing, code-mixed language identification has a great challenge. The existing papers like [11],used Morphological dictionary-based models often struggle with out-of-vocabulary words, which are words that do not appear in the underlying dictionary. out-of-vocabulary words can arise due to new or rare vocabulary, proper nouns, acronyms, foreign words, or misspellings. Since these models rely on dictionary lookup, they fail to generate the morphological structure of out-of-vocabulary words. In paper [12], using machine learning and deep learning like LSTM to identify the language but, LSTM capture sequence of language data only one direction. To capture the dependencies in deep and in both forward and backward direction, in our study we implement Bi-LSTM. In paper [13] , they did not use the sequence level labeling SOTA like attention in LID. Paper [14] and [15] are used BERT and their variant and Bi-LSTM respectively, but from 50K code-mixed corpus with six labels all proper nouns are not distinctly tagged as named entities and didn't compare the result with other deep learning algorithms.

As far as we are aware, no prior research has been conducted in the context of identifying languages in code-mixed Amharic-English social media text written in the Latin script. To address the above challenges, machine learning and deep learning-based language identification from social media text was used.

This study will investigate and address the following research queries in order to find a resolution to the previously mentioned issue.

- **RQ1:** Which machine learning techniques best fit for code-mixed Amharic-English language identification?
- **RQ2:** Which feature extraction method is best suited for the selected machine learning algorithms for our dataset?
- **RQ3:** Which deep learning techniques suitable for our dataset?

1.4. Objective of the Study

1.4.1. General Objective

The general objective of this study is to build an Amharic-English code-mixed language identification on social media text using machine learning and deep learning techniques.

1.4.2. Specific Objectives

In pursuit of the overarching goal, this study encompasses a sequence of particular tasks that are carried out throughout its entirety.

- To collect Amharic-English code-mixed dataset
- To preprocess the collected Amharic-English code-mixed dataset
- To build and train machine learning and deep learning models for Amharic-English code-mixed language identification
- To measure the performance of machine learning and deep learning models for our code-mixed Amharic-English language identification

1.5. Significance of the Study

Code-mixed Language identification is a fundamental task in natural language processing, entails the precise categorization of text data into specific languages from a predefined set, encompassing languages like Amharic, English, Named Entity, and Universal. It is a crucial process that underpins various applications in the research and technology domains. Code-mixed Language identification plays a pivotal role in predicting the languages used in written sentences, serving as a preliminary step before utilizing the text data for specific purposes. This step is particularly vital in multilingual and code-mixed contexts, where language boundaries are blurred.

Code-mixed language identification, a subset of language identification, holds significant relevance, especially in technology and healthcare sectors. In situations where expressions in the native language may be limited or entirely absent, code-mixed data poses a challenge. However, language identification of code-mixed data comes to the rescue, enabling the successful execution of numerous natural language processing tasks. These tasks encompass

Sentiment Analysis, Text Classification, Information Retrieval, Spell Checking, Named Entity Recognition, Parts-Of-Speech tagging, Semantic Parsing, and more[16]. The accurate determination of languages in code-mixed data is not only a technological feat but also a bridge that connects diverse languages and cultures, facilitating communication, understanding, and inclusivity.

In addition to its technological implications, proficient code-mixed language identification enhances the quality of healthcare services by ensuring accurate interpretation of medical information for diverse patient populations. It also optimizes search engines, making user queries more efficient and improving content discovery and accessibility, ultimately enhancing user experiences. Furthermore, it aids in the analysis of social and political discourse, offering deeper insights into diverse linguistic dialogues, which can inform decision-making processes.

1.6. Scope and Limitation of the Study

This study is focused on the task of identifying languages from Amharic-English code-mixed social media text using machine learning and deep learning techniques. On the social media conversation multiple languages are code mixed, addressing all code-mixed languages at this level is difficult, because of not being able to speak the languages and not knowing how to use the languages. This study mainly focused on two languages, Amharic and English; in this work, we have done code mixed Amharic-English language identification on the social media contents.

1.7. Organization of the Study

The remainder of the thesis is structured in the following manner. The second chapter presents a literature review and related work, where various concepts and methods associated with code-mixed language identification. Chapter Three presented the proposed methodology of Amharic-English code-mixed language identification, Chapter Four presented the proposed Amharic-English Code-mixed language identification approach, Chapter Five presented the implementation of code-mixed Amharic-English language identification algorithms and techniques, Chapter Six presented the result and discussion. Finally, the last chapter presents a conclusion and future work.

CHAPTER TWO

2. LITERATURE REVIEW AND RELATED WORK

This chapter discusses the concept of code-mixing language identification and theoretical ideas related to Amharic-English code-mixing in a social media text. It includes an introduction to social media, Natural Language Processing, and Amharic Language, under the Amharic Language section: Amharic Part of Speech and Amharic grammar presented. Also, discusses the English language under the English language section: grammar structure and Part of Speech. The other contents are code-mixing concepts, language identification and machine learning and deep learning approach. Finally, related works on code-mixing are discussed in detail.

2.1. Social Media

The phrase social media was initially introduced in 1994 within a Tokyo internet media environment known as Matisse. During the early days of the commercial Internet, the inaugural social media platforms were created and introduced. As time passed, both the quantity of social media platforms and the count of engaged social media users experienced substantial growth, solidifying it as one of the internet's most crucial applications[17].

Social media constitutes a mode of communication, offering users the capability to engage in conversations, exchange data, and generate online content. Various forms of social media exist, including blogs, wikis, social networks, microblogs, platforms for sharing photos, instant messaging, video-sharing sites, Facebook, Twitter, G-mail, YouTube, Telegram, Instagram and much more, the existence of a wide array of both independent and integrated social media services at present presents challenges to the concept of social media[18], [19], [20].

In today's society, social media has become major benefits for every activity. Social media also used for social networking, decision-making and news etc[21]. At an equally rapid pace, businesses have shifted their marketing requirements towards social media platforms. The coexistence of both enterprises and consumers on social media has transformed the way firms engage with their clientele, who now assume an active role in their interactions with the company, as opposed to being limited to a passive role[22]. Customers provide input, raise

inquiries, and anticipate prompt and tailored solutions to their issues. Moreover, customers share text, pictures, and videos. Managers recognize that the shift towards branding through social media necessitates a fundamental transformation in the customer relationship, wherein the customer assumes a role as a partner rather than a mere observer[22], [23].

2.2. Natural-Language Processing

NLP is a field within computer science that deals with the interaction and communication between human language and computers. It falls within the broader domains of Artificial Intelligence and Linguistics. NLP's primary aim is to enable computers to understand and interpret human language, making it more user-friendly for individuals who want to interact with computers in a natural, human-like manner. This field can be divided into two main components: Natural Language Generation and Natural Language Understanding, both of which involve the creation and comprehension of text[24],[25].

Natural Language Understanding pertains to a computer's ability to comprehend and interpret human language. NLU involves comprehending natural language and dissecting it by extracting concepts, emotions, entities, keywords, and more. This technology finds applications in client support systems where it aids in comprehending and addressing concerns raised by clients, whether communicated verbally or in written form[24]. we can be used for many applications such as chatbots, voice assistants and automated translation services[26].

Natural Language Generation (NLG) encompasses the creation of phrases, sentences, and paragraphs that effectively convey meaning and expression based on an internal representation[24]. Natural language generation (NLG) constitutes an element in the broader discipline of NLP. While NLU primarily centers on enhancing a computer's reading understanding abilities, NLG empowers computers to generate written content.

NLG involves the procedure of generating people language text responses based on input data, which can subsequently be transformed into spoken language through text-to-speech services[27]. The term natural language pertains to written or spoken texts that naturally occur and can be examined across multiple levels of linguistic scrutiny, encompassing aspects like morphology, syntax, semantics, discourse, and pragmatic considerations. Additionally, various techniques are available to carry out specific types of linguistic analysis, encompassing

methods that utilize rules, statistical techniques, and machine learning algorithms, and neural networks[28]. The main purpose of NLP is to complete a natural language using every level of linguistic analysis to include a variety of tasks and applications. For instance, language identification, Spelling Correction, grammar checking, part of speech tagger, machine translation, name entity recognition, information retrieval, information extraction, etc.[29].

2.3. Amharic Language

Amharic is grouped under the Semitic branch Afro-Asiatic language. It is the mother tongue language for Amhara in Ethiopia. The language serves as the official working language for Ethiopia and holds the status of an authorized or working language in several federal-level states [22]. Unlike Syrian, Arabic, Hebrew, Amharic is a left-to-right written language. This linguistic has its letters, words, phrases, sentences, grammatical structure, and dialect. Amharic language is a morphologically rich language [13].

2.3.1. Amharic Alphabet

The Amharic writing system, known as Fidel, utilizes the Geez script. Fidel is categorized as an abugida, where each character represents a combination of both a vowel and a consonant. However, the fundamental shape of each character is primarily determined by the consonant, with modifications made to accommodate the vowel. Additionally, some consonant phonemes are represented by multiple series of characters. Amharic Fidel comprises 34 fundamental characters, each capable of forming seven distinct vowel-consonant combinations. Furthermore, there are 28 labialized characters, including lwa, ḥwā, mwa, śwā, rwa, swa, shwa, k’wa, bwa, twa, chwa, nwa, nywa, vwa, ḫwa, e, ḵā, zwa, zhwa, dwa, jwa, gwa, t’wa, ch’wa, p’wa, ts’wa, fwa, and pwa[30].

ቈ ḳʷä	ቊ ḳʷi	ቋ ḳʷa	ቌ ḳʷe	ቍ ḳʷə
ኈ ḫʷä	ኊ ḫʷi	ኋ ḫʷa	ኌ ḫʷe	ኍ ḫʷə
ኰ kʷä	ኲ kʷi	ኳ kʷa	ኴ kʷe	ኵ kʷə
ጐ gʷä	ጒ gʷi	ጓ gʷa	ጔ gʷe	ጕ gʷə
ሏ lwa	ሗ hwa	ሟ mwa	ሧ szwa	ሯ rwa
ሷ swa	ሿ shwa	ቇ qwa	ቧ bwa	ቯ vwa
ቷ twa	ቿ cwa	ኇ xwa	ኗ nwa	ኟ ñwa
ኧ 'wa	ኳ kwa	ዟ zwa	ዧ zhwa	ዯ ywa
ዷ dwa	ጷ phwa	ጇ jwa	ጎ gwa	ጧ thwa
ፏ fwa	ጯ chwa	ጿ tswa	ፇ tzwa	ፗ pwa

Figure 2.1. Amharic Levelized Characters

Amharic script is distinct from the Latin alphabet and comprises 34 fundamental characters, each of which plays a crucial role in forming the written language. What sets Fidel apart is its ability to create seven distinct vowel-consonant combinations from each of these characters, resulting in a rich and intricate script that is both beautiful and functional.

The term Fidel translates to letters or characters in Amharic, and it serves as the foundation of written communication in the language. These 34 characters are grouped into basic consonant sounds, with variations for each character based on the addition of vowel sounds. In essence, Fidel combines consonants and vowels into one coherent character, allowing for a seamless representation of spoken Amharic[30].

Table 2.1.The Amharic Fidel

ሀ Hā	ሁ hu	ሂ hī	ሃ ha	ሄ hē	ህ hi	ሆ ho
ለ Le	ሉ lu	ሊ lī	ላ la	ሌ lē	ል li	ሎ lo
ሐ ḥā	ሑ ḥu	ሒ ḥī	ሓ ḥa	ሔ ḥē	ሕ ḥi	ሖ ḥo
መ me	ሙ mu	ሚ mī	ማ ma	ሜ mē	ም mi	ሞ mo
ሠ še	ሡ šu	ሢ šī	ሣ ša	ሤ šē	ሥ ši	ሦ šo
ረ re	ሩ ru	ሪ rī	ራ ra	ሬ rē	ር ri	ሮ ro
ሰ se	ሱ su	ሲ sī	ሳ sa	ሴ sē	ስ si	ሶ so
ሸ she	ሹ shu	ሺ shī	ሻ sha	ሼ shē	ሽ shi	ሾ sho
ቀ k'e	ቁ k'u	ቂ k'ī	ቃ k'a	ቄ k'ē	ቅ k'i	ቆ k'o
በ be	ቡ bu	ቢ bī	ባ ba	ቤ bē	ብ bi	ቦ bo
ተ te	ቱ tu	ቲ tī	ታ ta	ቴ tē	ት ti	ቶ to
ቸ che	ቹ chu	ቺ chī	ቻ cha	ቼ chē	ች chi	ቾ cho
ኀ ḥā	ኁ ḥu	ኂ ḥī	ኃ ḥa	ኄ ḥē	ኅ ḥi	ኆ ḥo
ነ ne	ኑ nu	ኒ nī	ና na	ኔ nē	ን ni	ኖ no
ኘ nye	ኙ nyu	ኚ nyī	ኛ nya	ኜ nyē	ኝ nyi	ኞ nyo
አ ā	ኡ u	ኢ ī	ኣ a	ኤ ē	እ i	ኦ o
ከ ke	ኩ ku	ኪ kī	ካ ka	ኬ kē	ክ ki	ኮ ko
ኸ ẖe	ኹ ẖu	ኺ ẖī	ኻ ẖa	ኼ ẖē	ኽ ẖi	ኾ ẖo
ወ we	ዉ wu	ዊ wī	ዋ wa	ዌ wē	ው wi	ዎ wo
ዐ ʻā	ዑ ʻu	ዒ ʻī	ዓ ʻa	ʻዔ ē	ዕ ʻi	ዖ ʻo
ዘ ze	ዙ zu	ዚ zī	ዛ za	ዜ zē	ዝ zi	ዞ zo
ዠ zhi	ዡ zhu	ዢ zhī	ዣ zha	ዤ zhē	ዥ zhi	ዦ zho
የ ye	ዩ yu	ዪ yī	ያ ya	ዬ yē	ይ yi	ዮ yo
ደ de	ዱ du	ዲ dī	ዳ da	ዴ dē	ድ di	ዶ do
ጀ je	ጁ ju	ጂ jī	ጃ ja	ጄ jē	ጅ ji	ጆ jo
ገ ge	ጉ gu	ጊ gī	ጋ ga	ጌ gē	ግ gi	ጎ go
ጠ t'e	ጡ t'u	ጢ t'ī	ጣ t'a	ጤ t'ē	ጥ t'i	ጦ t'o
ጨ ch'e	ጩ ch'u	ጪ ch'ī	ጫ ch'a	ጬ ch'ē	ጭ ch'i	ጮ ch'o
ጰ p'e	ጱ p'u	ጲ p'ī	ጳ p'a	ጴ p'ē	ጵ p'i	ጶ p'o

ጸ ts’e	ጹ ts’u	ጺ ts’ī	ጻ ts’a	ጼ ts’ē	ጽ ts’i	ጾ ts’o
ፀ ts̲’e	ፁ ts̲’u	ፂ ts̲’ī	ፃ ts̲’a	ፄ ts̲’ē	ፅ ts̲’i	ፆ ts̲’o
ፈ fe	ፉ fu	ፊ fī	ፋ fa	ፌ fē	ፍ fi	ፎ fo
ፐ pe	ፑ pu	ፒ pī	ፓ pa	ፔ pē	ፕ pi	ፖ po
ቨ ve	ቩ vu	ቪ vī	ቫ va	ቬ vē	ቭ vi	ቮ vo

These combinations, along with the other base consonants and their respective vowel forms, allow for the written representation of Amharic words with remarkable precision. The script's versatility enables writers to capture the intricate phonetic nuances of the language.

Amharic Fidel is not only a writing system but also an integral part of Ethiopian culture and identity. It has been used for centuries to document the country's rich history, literature, and traditions. Learning Fidel is an essential skill for anyone wishing to read and write in Amharic, and it continues to play a vital role in preserving the language's heritage.

2.3.2. Amharic Morphology

Amharic morphology deals with structure of Amharic words. Amharic has a complex inflectional morphology, mainly for verbs, paying not even prefixes and suffixes but also alterations of the typical Semitic pattern type and consonant-root. Morphology (šine k’alati consists of the following part of speech (yek’alati kifilochi), stem(iribata), Root(mišireta), affix(k’it’ili) and combining form (t’imera)[31],[32].

Part of speech (yek’alati kifilochi) is a linguistic category of words that describe how to used words in a sentence. Amharic part of speech (ye’āmarinya k’alati kifilochi) is usually classified into 8 lexical categories: noun(simi), pronoun (tiwilat’e simi), adjective(k’its̲’ili), verb(gisi), adverb (tiwisake gisi), preposition(mesitewadidi), conjunction(mesitets̲’amiri) and interjection (k’ali āgano)[33].

- ***noun*:** known as simi, is a linguistic term used to signify the name of an entity, whether it be a location, individual, object, or abstract concept. For instance, names like Abebe, Weniberi, and ānibesa exemplify nouns, which can exist in either singular or plural forms.

- ***pronoun*:** referred to as tewilat'e simi, is a linguistic element employed to substitute a noun or noun phrase within a sentence. Amharic encompasses a set of pronouns, including inē, ānite, ānichi, inanite, inya, isu, iswa, inesu, and more, which fulfill this role effectively.
- ***An adjective*:** denoted as k'its'ili, serves the purpose of providing descriptions for nouns or pronouns. Examples of adjectives in Amharic include words like tinishi, tilik'i, wefirami, k'ech'ini, t'ibebenya, tenikolenya, irezhimi, wet'ati, ārogī, and more, which enhance the details associated with the noun or pronoun they modify.
- ***Verb*:** referred to as gisi, functions as a linguistic element representing an action. In Amharic, a variety of verbs are used, including merot'i, megwazi, manibebi, mebilati, met'et'ati, mech'aweti, mefak'eri, nenyi, nahi, nachihu, newi, neberiku, nebere/chi, neberini, neberiki/shi, and many others, to express different actions and conditions.
- ***An adverb*:** known as tewisake gisi, functions as a word that provides information about the timing, manner, or location of an action. For instance, in Amharic, adverbs like bemech'eresha, betech'emarī, befit'ineti, bezigita, bet'inik'ak'ē, bet'ibik'i, begidileshineti, bek'eni, be'āmeti, bedinigeti, and others play the role of indicating when, how, or where an action occurred.
- ***Preposition*:** referred to as mesitewadidi, is a grammatical element used to express the direction or relationship between things. In Amharic, prepositions such as ye, sile, wisit'i, be... layi, wede wisit'i, be... wisit'i, ke... goni, wede, and gari serve this purpose by defining the spatial or directional aspects of objects or concepts.
- ***Conjunction*:** known as mesitets'amiri, plays the role of linking sentences and coordinating words within the same clause. In Amharic, conjunctions like ina, negeri gini, sile, weyimi, āyidelemi, silezīhi, honomi, mikinyatumi, kehone, inidehone, kalihone, inide, selehone, sale, and others are used for this purpose, facilitating the connection of ideas and elements within sentences.
- ***An interjection*:** referred to as k'ali āgano, serves as a word used to emphasize an idea and is typically inserted at the end of a word or sentence, often accompanied by an exclamation mark (!).
- ***Root*:** termed as mišireta, is a linguistic element that represents the base or citation form for a group of related words. For example, the root word wi-ri-di serves as the

foundation for various related words like wiredi, āwiridi, āweraridi, weredi, wiredi, āsiweridi, teweraredi, āweraridi, meweraredi, and more. These roots provide the essential structure for building related vocabulary.

- ***Stem(*iribata*)*:** is subbing part of word that never changes even when morphologically inflected. For instance, wi-ri-di is the stem word for silewiridi ፣ bewiridi ፣ lewiridi ፣ wiridetu ፣ wiridetachihu ፣ wirideti ፣ wiridetē ፣ wiridetami ፣ kiwirideti ፣ yewirideti ፣ silewiriditu ፣ yalewiridi ፣ inidewiridi ፣ inidewiridetu ፣ inidewiridetwa ፣ wiridetamichi ፣ wiridetami ፣ wiridetachini ፣ wiridetachiwi etc.
- ***Combining Forms (*t'imera*)*:** is also a part of words which are the morphemes that are formed from two bound or free-like roots. For instance, foto girafi ፣ t'eji bēti ፣ 'alemi-āk'efi ፣ k'edo t'igena ፣ wefi-zerashi ፣ ḫige menigišiti ፣ hode sefī etc.
- ***Affixes (*k'it'ili*)*:** are part of words that are bound morphemes either precede(prefix), follow (suffix), or are inserted inside (infix) the root or stem. For instance, in āliseberimi [ālisebe (ri'i) mi] is an affix that precedes and follows the stem seberi.

2.3.3. Amharic Phrase

A phrase is a structural unit in a language, formed by combining one or multiple words. In Amharic, there are five main types of phrases: noun phrases, verb phrases, adjectival phrases, adverbial phrases, and prepositional phrases. Each phrase can be classified as either simple, where it consists of only one word class, or complex, where it incorporates multiple word classes.

- ***Noun Phrase*:** is a grammatical arrangement where the primary element, known as the head (H), is either a noun or a pronoun. Noun phrases can exhibit simplicity or complexity. For instance, yesari bēti. The head of noun(N) is bēti with specifiers(S) yesari. The structure rule Noun phrase is S+N.
- ***Verb Phrase*:** consists of a central verb and additional components such as complements, modifiers, and specifiers. For instance, in the phrase wede sirawi hēde, the term sirawi forms a prepositional phrase (PP) that modifies the verb hēde. The structural formula for a verb phrase can be expressed as =PP+V.

- ***Adjectival Phrase*:** The formation of an adjectival phrase in Amharic follows a structure similar to that of noun phrases and verb phrases. It typically consists of an adjective as the head, along with other elements like complements, modifiers, and specifiers. For example, in the phrase yachi ijigi k'onijo, iswa serves as a specifier, ijigi acts as a modifier that describes the head k'onijo (beautiful). The structural rule governing this phrase is Specifier + Adverb + Adjective.
- ***Prepositional Phrase*:** It is formed by combining a preposition (P) as the primary element and another constituent, which could be a noun, noun phrase, verb, verb phrase, and so on. For instance, inide inisisa beduri. inide(like) and be(an) is a preposition, which combines the noun inisisa(animal) and duri (the forest) This PP is formed by the structural rule: prepositional phrase => PP+ PP.
- ***Adverbial Phrase*:** It is created by using one or multiple adverbs within the language. For instance, kifunya tamalechi. kifunya(severely) (head) is the sole adverb that composes the adverb phrase (AdvP) and adheres to the specified rule: adverbial phrase => Adv.

2.3.4. Amharic Sentence Structure

The Amharic sentence structure follows a Subject + Object + Verb pattern, in contrast to English, which uses a Subject + Verb + Object arrangement. This can be illustrated with the following example. Abebe misa bela.

- ***Simple Sentence*:** In a simple Amharic sentence, you have a noun phrase serving as the subject, and it is succeeded by a verb phrase that forms the predicate. For instance, beru kifiti newi.
- ***Complex Sentence*:** In Amharic, complex sentences are made up of intricate phrases, including noun phrases, verb phrases, or adjective phrases. These combinations can include a complex noun phrase paired with a simple verb phrase, a simple noun phrase combined with a complex verb phrase, or both a complex noun phrase and a complex verb phrase. For instance, girimawi yegezawi komipīwiteri tebelashibeti. girimawi wede yonīverisitī selalefe tedesete.

2.3.5. Amharic Punctuation Mark

Punctuation marks are characters employed to structure and enhance the clarity of written language. It's important to note that English and Amharic employ distinct punctuation marks. The following symbols are used in Amharic language: - ārati net'ibi(።), net'ela serezi(፣), diribi serezi(፤), timihirite silak'i(!), k'ali āgano(!), timihirite t'ik'isi("), t'iyak'ē milikiti(?), k'inifi(()), serezi(-).

2.4. English Language

The English language is a European language under grouped in west Germanic language. It is an international working language. It has 26 alphabet letters, and also it has words, phrases, sentence structures.

2.4.1. English Alphabet

The English alphabet letters consist of 26 letters, these are both upper- and lower-case version. upper-case alphabet consists A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, and Z and lower-case alphabet consists a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, and z.

2.4.2. English Morphology

English morphology is one of the main components of English grammar that explores the structure of English words. Morphology deals with a structured word. English Morphology consists of stem or Root, affix, and combining form.

Part of speech describes the usage of words in a sentence. The English POS is usually classified into 8 lexical categories: noun, adjective, pronoun, verb preposition, adverb, conjunction, and interjection.

- ***Noun*:** is a word that describe people, objectives, places, abstract ideas, etc. for example, Yetmwork, Wolkite, market, birthday, stone, research, builder and father, etc. Nouns use for both singular and plural names.

- ***Pronoun*:** is a term that is used the place noun phrase. For examples, I, We, You, He, They, She, It, that, those, these, this, who, which, what, as, each, all, both, either etc. are English pronouns.
- ***Adjective*:** is a word that defines a noun or pronoun. Such as; Large, small, thin, fat, Cunning wise, Young, long, old, etc.
- ***Verb*:** is a word that represent a state or action of a condition. Walking, Running, eating, drinking, playing, am, are, was, were, etc are English verbs.
- ***Adverb*:** is a word that describes a verb Such as: - fast, carefully, slowly, carefully, carelessly, strictly, daily, suddenly, annually, additionally those are English adverbs.
- ***Preposition*:** is POS that indicates the direction of things. Inside, behind, on, under, about, on above, from …. side, with, at etc. Those are English prepositions.
- ***Conjunction*:** is used to coordinate words in the same clause and connect clauses or sentences such as: - is not, however, so, yet, because, And, but, about/in case of, nor, although, even-though, unless, or, if, whether, if not, since, while etc are English conjunctions.
- ***Interjection*:** is a word used to represent an idea, usually at the end of a word with an exclamation mark(!).
- ***Root*:** is one part of words which is the citation form of a set of words. For instance: break is the root word for the break, breaks, breaking, broke, brooked, broken, etc.
- ***Stem*:** is part of the word that never changes, even when morphologically inflected. For instance: - walk is the stem word for a walk, walks, walking, walked, etc.
- ***Combining Morpheme*:** is a part of words which is formed from two bound or free-like root such as: - photography, Wine-house, sometimes, airline, airport, airtime, International, constitution, grandmother, grandfather afternoon, anybody, background. aircraft, blacklist, blackboard, etc.
- ***Affixes*:** are bounded morphemes that either are inserted inside, precede(prefix), follow (suffix), (infix) the stem, such as: - impassable is an affix that precedes and follows the stem pass.

2.4.3. English Phrase

A phrase is a group of words that contributes to the overall meaning of a sentence. In the English language, there are several types of phrases, including noun phrases, verb phrases, adjective phrases, prepositional phrases, and adverbial phrases.

- ***Noun Phrase*:** it's a linguistic structure in which the primary element (H) is a noun or a pronoun. Noun phrases can fall into either the category of simple or complex. For instance, consider the phrase Grass house. In this case, house serves as the noun (N), and Grass functions as a specifier (S). The syntax of a Noun Phrase is S + N.
- ***Verb Phrase*:** it's a linguistic arrangement that consists of a verb at its core, serving as the head. This verb phrase is often accompanied by additional elements such as modifiers, complements, and specifiers. For instance, in the sentence they went to work, the term to work functions as a prepositional phrase (PP) that provides modification to the verb went. The syntax of a verb phrase follows the pattern: PP + V.
- ***Adjective Phrase*:** which falls within the same category as Noun Phrases and Verb Phrases. An Adjective Phrase is constructed around an adjective as its core, enriched by additional elements like modifiers, specifiers, and complements. For instance, in the phrase that she was very pretty, that she serves as a specifier, and very acts as a modifier, intensifying the meaning of the core adjective pretty. The structural arrangement of this phrase can be expressed as Spec + Adv + Adj.
- ***Prepositional Phrase*:** this type of phrase is formed by joining a preposition (P) as its core component with other elements, including nouns, noun phrases, verbs, and verb phrases. For instance, in the phrase like a human in the home, both like and in function as prepositions, connecting the noun human and the location home. The structural guideline for constructing a prepositional phrase (PP) adheres to the pattern: PP + PP.
- ***Adverbial Phrase*:** which belongs to the realm of linguistic phrases created using one or more adverbs. Consider the sentence, she is severely ill. Here, the adverb severely takes the forefront, forming the primary component of the Adverbial Phrase (AdvP).

2.4.4. English Sentence Structure

The structure of English sentence is subject + verb + object. In English language. Sentences can be classified into four categories: simple, compound, complex, and compound-complex. The distinction between these sentence types is based on the presence of independent and dependent clauses, conjunctions, and subordinating elements.

- ***Simple sentences:*** are composed of an independent clause that functions independently, devoid of both conjunctions and dependent clauses. An example of a simple sentence is I run every weekend.
- ***Compound sentences*:** it is formed from by joined two independent clauses through conjunctions like and, but, or, for, nor, yet and so as an example, Aster is famous, yet she is very humble.
- ***Complex sentences***: are constructed by combining one independent clause with at least one dependent clause. They are linked together using subordinating conjunctions, which serve to establish a relationship between the independent clause and the dependent clause. These subordinating conjunctions can relate to the subject (e.g., who' which), the sequence or time (e.g., since, while), or causal elements (e.g., because, if) within the independent clause. As an example, Although I was healthy, I was still unhappy.
- ***Compound-complex sentences*:** it contains at least one dependent clause and multiple independent clauses. These sentences will contain both subordinators and conjunctions. As an example, Because I paid care, I got an A on the test and I was so happy.

2.4.5. English Article

Articles are words that indicate whether a noun is specific (definite) or nonspecific (indefinite). In the English language, articles offer three distinct semantic options for selecting the appropriate article:

- ***An indefinite article (a and an)*:** is employed prior to a noun when the noun is generic or when its specific identity remains unknown.
- ***The definite article (The)*:** is used when the reader already knows the specific identity of the noun in question.

- ***No article*:** In some instances, a noun is used without any article at all.

2.4.6. English Punctuation Mark

Punctuation marks are symbols employed to structure and enhance the clarity of written language. In English, grammar uses different punctuation marks. the following are some examples of English punctuations[34].

Table 2.2. List of English Punctuations

No	***Name of punctuation***	***Punctuation symbols***
1	Period (Full Stop)	.
2	Comma	,
3	Question Mark	?
4	Exclamation Mark	!
5	Colon	:
6	Semicolon	;
7	Single Quotation	' '
8	Double Quotation	" "
9	Parentheses	( )
10	Brackets	[ ]
11	Braces	{ }
12	Ellipsis	. . .
13	Apostrophe	'
14	Slash	/

2.5. Code-Mixing

Code-Mixing refers to the integration of elements from one language, including phrases, words, and morphemes, into a statement or conversation conducted in another language[35], [36]. CM is usually an intra-sentential phenomenon. Created by multi-lingual users. Code-

Mixing is defined as the inclusion of language elements from one language into words spoken in another language [37]. Code-Mixing occurs when both lexical (vocabulary) and grammatical aspects from two languages are present within the same sentence, conveying various language elements across different linguistic levels and units, from individual words to complete sentences [38]. It is not only used in the usually used spoken systems of multilingual settings; however, we use it in social media sites through answers, posts, and comments, particularly in chat conversations. Most of the one-to-one chats are received in a linguistically diverse than the individual's native language[39].

Code Mixing, also known as a mixed code, occurs when two languages are used to create a third, new linguistic code. This emerging code integrates elements from the two languages into a structurally definable pattern. According to the code-blending theory, when code-switched languages give rise to the appearance of a third code, this new code exhibits distinctive structural characteristics. Code mixing encompasses both code-switching and code-blending. Code-switching involves the integration of words, phrases, and sentences from two different grammatical systems across sentence boundaries within the same speech event [40]. Code-Mixing is the process of incorporating diverse linguistic components, which can include affixes (bound morphemes), words (unbound morphemes), phrases, and clauses. It often occurs during a collaborative interaction, where participants need to reconcile what they hear with their understanding in order to grasp the intended message [41], [42].

Code-switching is the act of blending two separate grammatical systems or subsystems within a single conversation. On the other hand, Code-Mixing entails the inclusion of linguistic components such as phrases, words, and morphemes from one language into a communication conducted in another language [43]. Code-Switching typically occurs between sentences, whereas Code-Mixing takes place within a single sentence. Linguists suggest that there is a consistent pattern in how vocabulary transitions from one language to another in situations of language contact [35].

2.5.1. Types of Code-Mixing

According to[44], [45], [46] There are three distinct categories of code-mixing, which can be summarized as follows:

- ***Insertion Code-Mixing*:** in Insertion code-mixing, elements from one language are inserted into a sentence or phrase in another language.
- ***Alternation code-mixing*:** in Alternation code-mixing, speakers switch between two languages or alternate between them in a conversation.
- ***Congruent Lexicalization***: is a term that describes a scenario in which two languages share grammatical structures that can be lexically filled with elements from one language. It often involves using words or phrases from one language within the context of another. For example, when writers incorporate English words or phrases into their articles that either lack a direct translation into Amharic or are widely understood in English.

2.6. Language Identification

The term, Language Identification is the procedure of recognizing and determining the language being used from written or spoken input texts. Language identification from different language written text needs attention task. There are various approaches there to recognize the language of the same native language speakers[47]. Language identification encompasses a broad spectrum of research within the realm of Natural Language Processing. This domain covers a wide range of activities, including, but not restricted to, tasks such as machine translation, summarization, question answering, and information retrieval. One of the NLP problems is language identification or recognition. At present, the challenge of language detection is addressed by combining the N-gram approach with traditional classification models, such as the naive Bayes and Support Vector Machine (SVM) classifiers, and others. Some Python libraries helps to detect language with good performance. such as langid and FastText.

- ***Word-Level Language Identification:*** This task involves identifying the language of a text segment within mixed-lingual text, and it often goes beyond the capabilities of standard automatic linguistic identification methods.
- ***N-gram Language Profiling and Pruning:*** One of the most well-known language detection systems is TextCat, which relies on character-based n-gram models. This

approach creates language-specific n-gram profiles based on their frequency in the training corpus**.**

- ***Dictionary-Based Detection:*** Another established method for language identification involves the use of dictionaries containing the most frequently used words. For the current task, a dictionary-based language detection approach was integrated. However, challenges arose in preparing the dictionary, especially considering the presence of noisy text in social media.
- ***SVM-based Word-Language Detection:*** Detecting the language at the word level within code-mixed text is essentially a classification problem. Support Vector Machines (SVM) were selected for this experiment. SVM was chosen due to its reputation as one of the best-performing machine learning techniques across various domains and tasks, including language identification.

2.7. Machine Learning Approach

The capacity to learn from previous observations, coupled with the increasing volume, diversity, and speed of data, has paved the way for automation in text processing techniques, particularly in text classification. In some cases, the automation of the classification task involves establishing a set of logical rules using knowledge-engineering methods, often based on expert input[48]. In the realm of machine learning approaches to text classification, there are conventional machine learning algorithms, including logistic regression, support vector machines, decision trees, random forest trees, rule induction, K-nearest neighbor, K-means, and the Hebbian algorithms, which are widely used. Additionally, newer and highly popular approaches, such as deep learning methods, encompass techniques like Convolutional Neural Networks (CNN), Deep Neural Networks (DNN), and Recurrent Neural Networks (RNN), which have become fundamental and extensively adopted methods in text classification.

2.7.1. Naive Bayesian

This is a straightforward classification approach that relies on Bayes' rule, even when dealing with high-dimensional input data. In this method, Bayes' rule is employed to analyze documents and their associated classes. The model assigns class labels to instances of a problem, which are represented as vectors of feature values. The value of one feature is

considered independently from the value of another feature. The technique follows a known prior probability and checks the posterior probability in the naive Bayesian approach, ultimately assigning the document to the class with the highest posterior probability[49].

2.7.2. Support Vector Machine

This method is a remarkable way to classify data with a high number of dimensions, and it operates based on the principle of Structural Risk Minimization (SRM)[50]. It's been described as a discriminative classifier that surpasses most previous classification models in terms of accuracy. SVM (Support Vector Machine) seeks to find the best hyperplane that effectively separates training data points from various classes by maximizing the margin of separation. Additionally, SVM can handle data points with nonlinear decision boundaries by using a technique known as the kernel method, which maps the input data into a higher-dimensional feature space where a linear separation hyperplane can be established.

2.7.3. Random Forest

Random Forest is a versatile and robust ensemble machine learning algorithm, utilized for both classification and regression tasks. It harnesses the potential of numerous decision trees, each constructed using different subsets of the data and random feature subsets[51]. This deliberate randomness among the trees effectively mitigates the risk of overfitting and amplifies the accuracy of predictions. By consolidating the predictions of these diverse trees, Random Forest provides a final prediction that frequently outperforms the predictive capabilities of a single decision tree. This algorithm enjoys wide-ranging applicability in various domains, from spam detection to stock price forecasting, owing to its ability to adeptly manage intricate datasets and consistently yield high-performance results.

2.7.4. Decision Tree

Decision trees are used to create classification models in a tree-like structure. The goal is to build a model that can predict the value of a target variable using input variables. In these tree structures, the terminal points represent class labels, and the branches represent the characteristics related to these class labels. As the process unfolds, a dataset is divided into smaller subsets, and simultaneously, a decision tree is gradually constructed. The final outcome

is a tree comprising decision nodes and leaf nodes[52]. It is capable of handling both numerical and categorical data.

2.7.5. Logistic Regression

Logistic regression is a supervised learning algorithm commonly employed for binary classification tasks [53]. It is usually employed when we need to determine whether the input goes to one class or another class. Logistic regression forecasts the probability that the input can be considered into one primary class. Therefore, logistic regression is used for binary classification rather than predictive model. It allows us to allocate input data to one of the two class classes based on the chance estimate and a distinct threshold.

2.7.6. Maximum Entropy

The Maximum Entropy classifier is a probabilistic classification method that belongs to the class of exponential techniques. Unlike some other methods, it doesn't assume that the features are conditionally independent of each other. The Maximum Entropy classifier operates based on the Principle of Maximum Entropy and selects the model with the highest entropy from all the possible techniques that are consistent with our training data[54]. The Maximum Entropy classifier is a versatile tool used to tackle various classification problems, including tasks like topic classification, language detection, sentiment analysis, and more. It's particularly useful in situations where we have very little prior knowledge about the underlying distributions and making assumptions could be risky. Additionally, the Maximum Entropy classifier is chosen when we cannot assume feature independence.

2.7.7. XGBoost

XGBoost, short for Extreme Gradient Boosting, is a powerful machine learning algorithm designed to enhance your understanding of data and facilitate informed decision-making. It is an implementation of gradient-boosting decision trees and is widely employed by researchers and data scientists globally to optimize their machine learning models. XGBoost is designed for performance on large datasets, speed and easy to use, it doesn’t require optimization of the tuning, which means that it can be used directly after installation without any configuration[55].

2.7.8. Conditional Random Field

A Conditional Random Field (CRF) is a robust machine learning model tailored for structured prediction tasks, particularly those that entail sequential or structured output data. Unlike traditional Random Forests, CRFs are particularly effective in scenarios were output labels display interdependencies. As a result, they are well-suited for tasks such as named entity recognition, part-of-speech tagging, and sequence labeling, especially in the domain of natural language processing [56]. CRFs operate as an ensemble learning technique, harnessing the power of multiple decision trees to collaboratively predict structured outputs. These models are trained with labeled data, allowing them to learn label assignments while considering intricate dependencies between output labels. CRFs have made substantial contributions in domains where structured data and dependencies are pivotal, including linguistics, bioinformatics, and various other fields.

2.8. Deep Learning Approach

Deep learning is a subset of machine learning methods based on representation learning and artificial neural networks. These techniques teach a machine to do what humans do naturally through example. Deep learning has recently become useful for two key reasons: it needs vast volumes of labeled data and it requires significant computational power[57]. The parallel architecture of high-performance GPUs is suitable for deep learning[58]. Neural networks are used in the majority of deep learning techniques. The structure of a neural network is close to that of a human brain, and it is made up of artificial neurons known as nodes. The input layer, hidden layer, and output layer are the three layers that make up these nodes. Deep learning models are trained to utilize large sets of labeled data and neural network architectures that learn features directly from the data, eliminating the requirement for manual feature extraction. Many deep learning algorithms are used for various NLP applications[58].

Neural Network: a neural network classifier is a network of units where the input units represent concepts and the output units speak for the category. Each unit receives a set of inputs. There are two types of class separators, namely single linear perception and multiple perceptions. Single-layer perceptions do not separate classes. Multiple layers are used to find

non-linear classification boundaries. In the neural network approach, Deep Learning, which includes CNN, RNN, and DNN, is commonly used for text classification.

2.8.1. Convolutional Neural Network

A convolutional Neural Network is a type of neural network that uses convolution to perform mathematical processes between its layers[59]. We call the input layer is convolutional layer and the output layer, the layers that make the system. Input layers accomplish a variety of tasks for instance kernel operation, padding, striding, and other functions. This layer is the building block of the convolutional neural network. The main aim of the convolutional neural network is from text extract the features and the output layer displays the final result.

There are two main parts to a CNN architecture

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction.
- The network of feature extraction consists of many pairs of convolutional or pooling layers.
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.
- This CNN model of feature extraction aims to reduce the number of features present in a dataset. It creates new features which summarizes the existing features contained in an original set of features. There are many CNN layers as shown in the CNN architecture diagram.

2.8.2. Multi-Layer Perceptron

A Multi-Layer Perceptron (MLP) is a type of artificial neural network that consists of multiple layers of nodes or neurons. It is a feedforward neural network architecture, meaning that information travels through the network in one direction, from the input layer to the output layer. Each layer, except the input layer, is composed of neurons with an activation function, and connections between neurons have associated weights.

2.8.3. Long Short-Term Memory Networks

Long Short-Term Memory (LSTM) is a recurrent neural network method that can learn and recall long-term dependencies. It is mainly used to solve the vanishing gradient problem in recurrent neural networks. In LSTM, memory cells are used to store information over a long period of time. Since natural language is based on the sequence of words, phonemes, or sentences, RNNs can be used to provide additional attention. Since these networks were created for long-term dependencies, they differ from other types of neural networks in that they can remember information for a long period of time without having taking to relearn it over and over again, making the whole process procedure faster and easier[60].

2.8.4. Bidirectional Long Short-Term Memory Networks

The bidirectional network with long-term memory is the extension of the neural network LSTM. In a long short-term memory network, knowledge flows in only one direction, whereas in a bidirectional long short-term memory network, it flows both backward and forward. A bidirectional long short-term memory (Bi-LSTM), which encodes the context of the input sentence, consists of a forward-directed long short-term memory[61].

2.9. Related Work

In this section, important works of literature related to our title are reviewed to clearly state and further explore the research problem.

Nowadays, Automatic processing and understanding of social media content is an interesting research area in natural language processing. Code mixing is the usage of multiple languages in a single sentence. Recently, different research done to identify languages in code-mixed social media content, but not for Amharic-English Code-mixed text. In 2023 Language Identification of English and Punjabi Code-Mixing and Code-Switching Sentences done by Enjula Uchoi and Mandeep Kaur, they used character N-gram and dictionary-based techniques. Using the character N-gram they achieved 88% accuracy and dictionary-based gained 48%[62]. In 2022 Language Identification at the Word Level in Code-Mixed Texts Using Character Sequence and Word Embedding done by A. Gelbukh et al. They used N-gram, BOW, TFIDF, and word-to-vector feature engineering and RFC, MLP, and LSTM models.

Using RFC with character N-grams, MLP with character N-grams, and LSTM with word2vec performs 74%, 72% and 77% respectively[63].

In 2021 Transformer Based Language Identification for Malayalam-English Code-Mixed Text done by Thara S. et al, used BERT and transformer model and Bidirectional Encoder as well as different BERT variants CamemBERT, DistilBERT, ELECTRA, XLM-ROBERTA. Using the word embedding method they gained BERT 0.9893, ELECTRA 0.9941, CamemBERT 0.9839, XLM-ROBERTA 0.9899, and DistilBERT 0.9880 accuracy[64].In 2021 Corpus Creation and Language Identification in Low-Resource Code-Mixed Telugu-English Text was introduced by Varma K et al, they used a deep learning Model Manually annotated data sets from Twitter and blogs and word level and sentence level word-by-word classification Approach. good thing they did find the best models like BILSTM+LSTM and BILSTM+CRF. Using BLSTM+LSTM and BLSTM+CRF they gained results of 98.53% and 98.35% respectively[65].

In the 2020 Language Identification Framework in Code-mixed social media text based on quantum LSTM the word belongs to which language by Shekhar, et al, they used BILSTM to identify language and special class LSTM network model. Their BILSTM system method performs better for Hindi-English language pairs. Using character level, they gained 93.49% for Hindi, 85.38% for English, and 80.2% for named entity language Also using word level they performed 63.97% for Hindi, 85.71% for English, and 83.9% for named entity language of F measure values[2]. In 2018, Soumil Mandal et al. presented a research paper titled Language Identification in Code-Mixed Data using Multichannel Neural Networks and Context Capture. In their study, they employed multichannel neural networks, which integrated CNN and LSTM for the purpose of identifying languages at the word level. Additionally, they incorporated a BI-LSTM-CRF context capture module into their methodology. This combined approach yielded impressive accuracy rates of 93.28% and 93.32% on their two separate testing sets. This work introduced a novel architecture for language identification, addressing both word-level and contextual aspects[66].

Table 2.3. Related Work

Ref.	***Authors and year***	***Methods and Result Description***	***Research Gaps***
Language Identification of English-Punjabi Code-Mixing and Code-Switching Sentences[62]	Enjula Uchoi and Mandeep kaur (2023)	-They used character N-gram and Dictionary based. -They used 45,628 words -Using character N-gram they achieve 88% accuracy and dictionary based gained 48%.	-They used dictionary-based models, but if the word is out of vocabulary the model not predict the word label.
Language Identification at the Word-Level in Code-Mixed Texts Using Character Sequence and Word Embedding[63]	A. Gelbukh et al. (2022)	- They used N-gram, BOW, TFIDF and word to vector feature engineering and RFC, MLP and LSTM models. - They used 7000 datasets - Using RFC with character N-grams, MLP with character N-grams and LSTM with word2vec performs 74%, 72% and 77% respectively.	-They used different feature extraction technique and machine learning and deep learning like LSTM, but LSTM works for one direction to capture the sequence of words and Language level.

Corpus Creation and Language-Identification in Low-Resource Code-Mixed Telugu-English Text[65]	Varma K et al. (2021)	-They used deep learning Models (including BILSTM+LSTM and BILSTM+CRF), word level and sentence level word-by-word classification Approach and manually annotated data sets from twitter and blogs. - They used 34,061 datasets - Using BILSTM+LSTM and BILSTM+CRF, they gained the result of 98.53% and 98.35% respectively.	-They did not use the sequence level labeling SOTA like attention in LID.
Transformer-Based Language Identification for Malayalam-English Code-Mixed Text[64]	Thara, S et al. (2021)	-They used BERT, transformer, Bidirectional Encoder model and different BERT variants CamemBERT, DistilBERT, ELECTRA, XLM-ROBERTA. - They used 50,000 datasets -Using word embedding method the result shows BERT, ELECTRA, CamemBERT, XLM-ROBERTA and DistilBERT performs 0.9893,0.9941,0.9839,0.9899 and 0.9880 accuracy score respectively.	-From 50K code-mixed corpus with six labels all proper nouns are not distinctly tagged as named entities.

Language Identification Framework in Code-Mixed social media Text Based on Quantum LSTM the word belongs to which language[2]	Shekhar et al. (2020)	-They used BILSTM to identify language and used special class LSTM network model - They used 63,000 datasets -Using character level, they gained 93.49% for Hindi ,85.38% for English and 80.2% for named entity language and also using word level they perform 93.97% for Hindi, 85.71% for English and 83.9% for named entity language F measure values.	- They did not consider Inter-Sentential code-mixing in this study.
Language Identification in Code-Mixed Data using Multichannel Neural Networks and Context Capture[66]	Soumil Mandal et al. (2018)	-They used multichannel neural networks by combining CNN and LSTM for word level language identification. Combining this with a BI-LSTM-CRF context capture module, accuracies of 93.28% and 93.32% is achieved on their two testing sets. - They used 12,000 datasets	-They did not incorporate borrowed tags and collect even more code-mixed data.

In above Table 2.3, we summarized different related works for code-mixed language identification tasks and we found the gaps in all papers. We take techniques that many researchers used and add our language pair techniques to do this study, but in our proposed study, we consider another language pair Amharic-English Code-mixed language identification.

CHAPTER THREE

3. RESEARCH METHODOLOGY

3.1. Overview

This chapter presented the tools and techniques used to carry through our research objectives. The first phase is dedicated to acquiring the data, preparing it, and annotating it to be used in building various machine learning and deep learning models. The second is concerned with the actual procedure used in building and training models, including data preprocessing, feature extraction methods, and model performance evaluation methods as well as model prototyping used in our research.

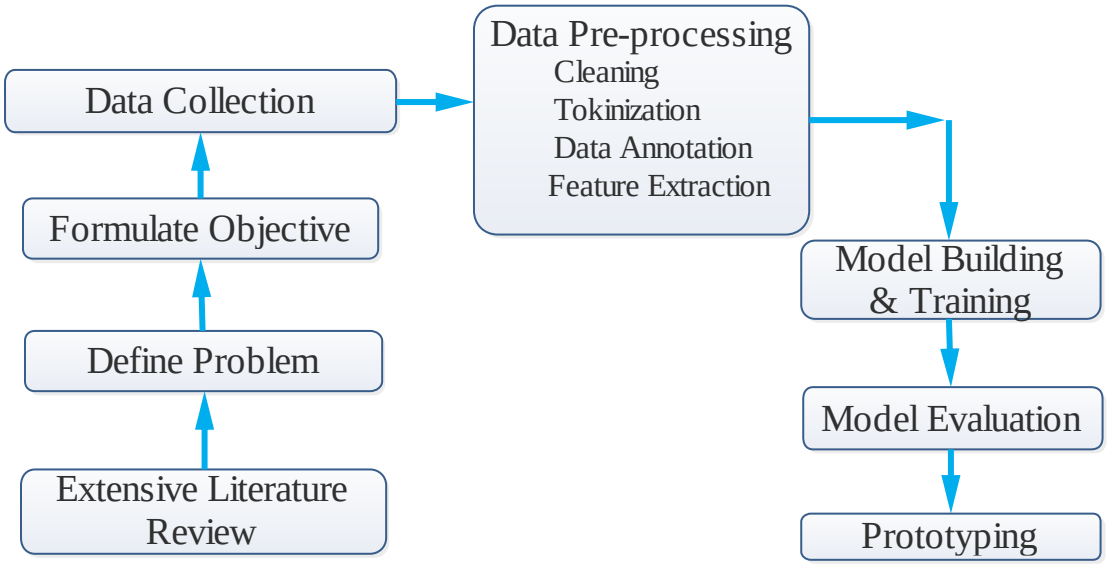


Figure 3.1. Research Flow

3.2. Amharic-English Code-Mixed Dataset

In this section, we discussed the process of gathering data and the sources from which it was obtained. The dataset we collected consists of a blend of two languages: Amharic and English, which is commonly referred to as code-mixing. Amharic serves as the official language of Ethiopia but has relatively limited computational linguistic resources available. Therefore,

people often mix these two languages in their daily activities, especially on social media platforms, where it serves as a mode of communication.

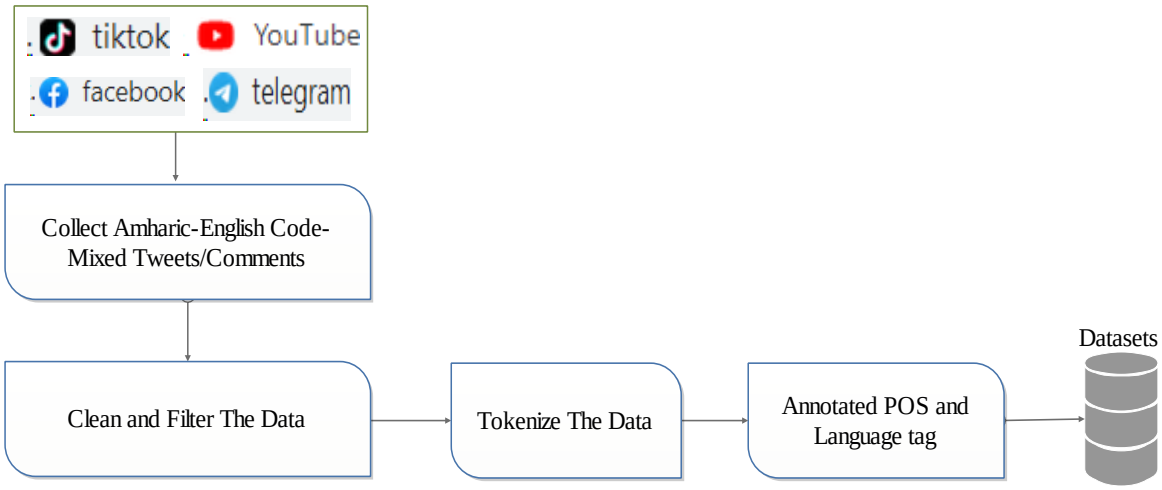


Figure 3.2. dataset preparation

As shown in Figure 3.2, the dataset preparation process contains four steps. Firstly, we collected code-mixed Amharic and English data from social media like Facebook, Telegram, YouTube, and TikTok. By using a face pager, we collected public Facebook data, and Python code for YouTube and TikTok data. we also extracted telegram data. Secondly, we cleaned and filtered the data. Thirdly, we tokenized a sentence to corresponding words and finally manually annotated the code-mixed Amharic and English datasets with their POS and Language tags. Each process is discussed in the following subsection, in detail.

3.2.1. Sources of Data

Nowadays, on social media, people communicating in two mixed languages is very common. for example, on Facebook, TikTok, YouTube comments, and telegram conversations. For this research, we gathered code-mixed data in both Amharic and English from these social media platforms. In general, the data sources are detailed in table 3.1.

Table 3.1. Data sources

No	***Code-Mixed Data Sources***	***Numbers of data collected***
1	Facebook post and Comments	1000
2	TikTok post and Comments	1600
3	YouTube Comments	1850
4	Telegram Conversation	571
Total		**5021**

3.2.2. Dataset Preparation

To collect our dataset, we considered four (4) social media sites such as Facebook, YouTube, Telegram, and TikTok because of these platforms support peoples mix their languages in a conversation also to gain more code-mixed data[67]. These platforms cover general social networking, video sharing, messaging, and short-form video content, offering a comprehensive snapshot of code-mixed language use. The selection aligns with our research goals, considering popularity, user base, and ethical considerations. After scrapping social media text, we filtered code mixed Amharic and English comments and wrote on the computer with Excel and text format.

3.2.3. Dataset Annotation

We prepared five thousand twenty-one (5021) Amharic English code-mixed sentences that bl end, where it is segmented into individual words, and separated by new lines and the dataset c ontains 31,304 tokens. The statistic of dataset as shown in Table 3.2.

Every word in a single sentence is manually annotated, its Language tag and Part of speech tag. Linguistic experts have applied Amharic part-of-speech (POS) tagging, while NLTK has been employed to perform English POS tagging, so, Mr. Yidnekachew Feleke helped me in the annotation of language level and POS tagging, he works in English Language and Literature department, he has more than 13 years' experience in teaching at high school level.

Table 3.2. Statistic of Datasets

No	*Language Label*	*Label Frequency*	*Percentage of Label*
1	Amharic	8158	26.06 %
2	English	19972	63.80 %
3	Universal	2211	7.06 %
4	Named Entity	963	3.07 %

3.3. Amharic-English Language Identification Text Pre-Processing Techniques

Text preprocessing is a vital step in building both machine learning (ML) and deep learning (DL) techniques is determined by the level of data preprocessing in the context of natural language processing (NLP), in the process of model building text preprocessing is the first phase.

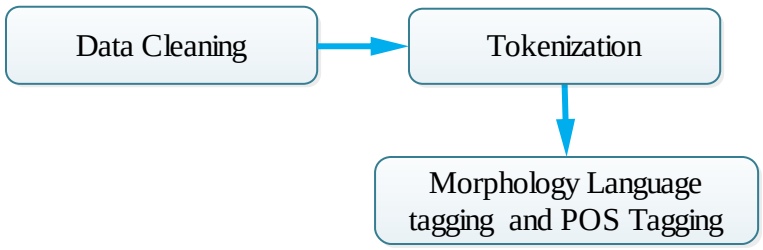


Figure 3.3. Text Preprocessing

3.3.1. Data Cleaning

Data cleaning is one of the text preprocessing steps, it is crucial for any machine learning and deep learning tasks but, more so far natural language processing tasks from this study have applied the following un necessary data removed in the cleaning step. Such emoji and double spaces.

Pseudocode for: Cleaning the dataset

Input: Un_processed_dataset

Output: Clean_dataset

***Step1*:** Read Un_processed_dataset

***Step2*:** While (! End_of_File);

If (Un_processed_dataset contain emojis) then

Remove emoji

If (Un_processed_dataset contain white_space) then

Remove white_space

Return Clean_dataset

End If

***Step 3*:** End

3.3.2. Word Tokenization

Word tokenization is another text preprocessing step that is necessary for splitting a phrase, sentence, or paragraph into words. In this paper, we have used Keras and NLTK API tokenizers to split strings. Keras provides the text sequences to make word sequences. In this process, we split the sentence into words using a space separator. The main job of the tokenizer is cleaning and splitting the input sentence into words.

Pseudocode for: Word Tokenization

Input: Read un_toknize_dataset

Output: Tokenize data

***Step1*:** Read un_toknize_dataset

***Step2*:** While (! End_of_File) do

Word_tokenize ()

Return Tokenize data

3.3.3. Morphology-Based POS Tagging

Morphology-based English language parts of speech tagging are the sub-component of the proposed code-mixed detection text preprocessing. These parts of speech tagging occur after separating a sentence into English tokens. It accepts input tokens and assigns a tag value for each English writing token from the morphologically marked tokens database. To identify if the sentence is Amharic, English, Universal, or Named Entity, first a sentence must be morphologically tagged. The morphology of words may be POS (part of speech), stem, and others. Amharic morphology POS prepared by a language expert. The process of tagging is done by word in, and then every token in the sentence is tagged.

Table 3.3. English Language Morphology based POS Tagging

No	***English POS Tag***	***Description***
1	N	Noun
2	V	Verb
3	ADV	Adverb
4	ADJ	Adjective
5	CON	Conjunction
6	NU	Number
7	Pre	Preposition
8	PRO	Pronoun
9	DET	Determiner
10	PUN	Punctuation

3.4. Amharic-English Language Identification Feature Extraction Techniques

3.4.1. Count Vectorizer

A Count Vectorizer is a technique used in the domain of natural language processing (NLP) and text analysis to convert a collection of text documents into a numerical format that is

suitable for use in machine learning and statistical modeling applications. It serves as a straightforward and widely employed approach for text representation, sometimes denoted as the bag of words models.

Count Vectorization operates as follows: Tokenization: - Initially, each document within the collection undergoes a breakdown into its constituent words or terms. This operation is recognized as tokenization. Frequently, punctuation and common stop words are excluded during this tokenization stage.

Frequency Counting: For every document, the Count Vectorizer proceeds to tally the occurrences of each term (word) present within that document. This results in a numeric depiction of the document, where each term corresponds to a unique dimension, and the value within each dimension signifies how many times that specific term appears in the document.

Vector Representation: After the counting procedure is executed across all documents within the collection, the outcome is a matrix. In this matrix, each row stands for an individual document, while each column represents a distinct term. The values within this matrix indicate the term frequencies in the respective documents.

The resultant count_matrix takes the form of a numerical matrix, where each row corresponds to one of the initial input documents, and each column symbolizes a distinct term found across the entire collection of documents.

3.4.2. TF-IDF Vectorization

TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is a method employed in the realm of natural language processing and information retrieval. Its objective is to convert a collection of text-based documents into numerical vectors, making them suitable for use in machine learning and text analysis tasks.

TF-IDF vectorization operates as follows: Term Frequency (TF): This part of the TF-IDF formula assesses how often each term (word) appears in a document. It assigns greater significance to terms that manifest more frequently within a document. The standard TF formula is expressed as[68]:

$$TF(t,d) = \frac{Number\ of\ occurrences\ of\ term\ t\ in\ document\ d}{Total\ number\ of\ terms\ in\ document\ d} \quad 3.1$$

Inverse Document Frequency (IDF): IDF measures the importance of a term within the entire collection of documents. It bestows a higher weight upon terms that are infrequent across all documents but exist within the document of interest. The typical IDF formula is as follows[8], [68]:

$$IDF(t,D) = \frac{log(Total\ number\ of\ documents\ in\ the\ corpus\ D}{Number\ of\ documents\ containing\ term\ t} \quad 3.2$$

TF-IDF Score: The TF-IDF score for a term in a document is determined by the multiplication of its TF and IDF scores[68].

$$TF - IDF(t,d,D) = TF(t,d)\ *\ IDF(t,D) \quad 3.3$$

Vectorization: Once TF-IDF scores have been computed for all terms across all documents, each document can be depicted as a numerical vector. In this representation, each dimension represents an individual term found in the entire collection of documents, and the value within each dimension is the TF-IDF score associated with that term in the document.

3.4.3. Word to Vector

Word to Vector (Word2Vec) is a widely used word embedding method in the field of natural language processing (NLP) that represents words as compact vectors consisting of real numbers. Unlike traditional methods like Count Vectorization or TF-IDF, Word2Vec captures the semantic meaning of words by learning their representations from large text corpora.

Word2Vec is based on the idea that words that have similar meanings are often found in similar contexts when analyzing a large text dataset. It constructs a dense vector space where words sharing similar meanings are positioned close to each other. There are two primary architectural variations within Word2Vec:

Continuous Bag of Words (CBOW): CBOW endeavors to predict a target word by drawing insights from the surrounding context words. It strives to anticipate a target word based on its contextual companions.

Skip-gram: In contrast, Skip-gram commences with a target word and endeavors to forecast the contextual words that are likely to co-occur with the target word. It aims to forecast the words in the surrounding context based on a provided target word.

3.4.4. Bi and Tri Gram Vectorization

A bigram is a sequence of two consecutive words or characters in a text. In the context of language identification, bigram models consider pairs of adjacent words or characters as the basic unit for analysis. For code-mixed Amharic and English language identification, bigram models would analyze pairs of words or characters to identify patterns and language switches in the text.

Trigram (3-gram): A trigram refers to a series of three consecutive words or characters within a text. Similar to bigrams, trigram models look at sequences of three adjacent words or characters. In the context of language identification, trigram models are more detailed and can capture more complex patterns of language mixing and switching in code-mixed text.

3.5. Selected Models Algorithm

Steps to Train Machine Learning Models

Input: Labeled dataset of code-mixed text

Output: Trained SVM, Decision tree, Random Forest, MaxEnt, XGBoost, Naïve Bayes, Logistic Regression, and CRF Model for language identification

Step 1: Preprocess Features

Step 2: Prepare Training Data

Step 3: Train SVM, Decision tree, Random Forest, MaxEnt, XGBoost, Naïve Bayes, Logistic Regression, and CRF Model

Step 4: Evaluation SVM, Decision tree, Random Forest, MaxEnt, XGBoost, Naïve Bayes, Logistic Regression, and CRF Model

Step 5: Predict Language Labels

Algorithm 1: To Train and Convolutional Neural Network (CNN) Model Amharic-English Code-Mixed Language Identification Task

Start

Input: Amharic-English sequences padded Data

Output: Trained CNN Model

Initialization: epoch, batch size, neuron, dropout rate, verbose

Step1: add embedding Layer with defined vocabulary size and input_length
add convolutional 1D Layer with defined neurons and activation
add GlobalMaxPooling1D Layer
add CNN Layer with defined neurons
add dropout regularization
add dense layer with defined class and SoftMax activation function

Step2: Compile CNN model with sparse_categorical_crossentropy loss function and Adam optimizer

Step3: Fit the X_train_padded and y_train_encoded dataset to CNN Model with initialized hyper parameter
While maximum iteration not reached ***do***
Train CNN Model

Step4: Evaluate CNN model with X_test_padded data

Step5: Predict unseen data by CNN Model

Step6: Measure the CNN Model by accuracy metrics

Stop

Algorithm 2: To Train and Evaluate Multi-Layer Perceptron (MLP) Model Amharic-English Code-Mixed Language Identification Task

Start

Input: Amharic-English sequences padded Data

Output: Trained MLP Model

Initialization: epoch, batch size, neuron, dropout rate, verbose

Step1: Add embedding Layer with defined vocabulary size and input_length

add Flatten Layer
add dense layer with defined neurons and relu activation function
add dropout regularization
add dense layer with defined output class and SoftMax activation function

Step2: Compile MLP model with sparse_categorical_crossentropy loss function and Adam optimizer

Step3: Fit the X_train_padded and y_train_encoded dataset to MLP Model with initialized hyper parameter
While maximum iteration not reached ***do***
Train MLP Model

Step4: Evaluate MLP model with X_test_padded data

Step5: Predict unseen data by MLP Model

Step6: Measure the MLP Model by accuracy metrics

Stop

Algorithm 3: To Train and Evaluate Long Short-Term Memory (LSTM) Model Amharic-English Code-Mixed Language Identification Task

Start

Input: Amharic-English sequences padded Data

Output: Trained LSTM Model

Initialization: epoch, batch size, neuron, dropout rate, verbose

Step1: Add embedding Layer with defined vocabulary size and input_length
add LSTM Layer with defined neurons
add dropout regularization
add dense layer with defined class and SoftMax activation function

Step2: Compile LSTM model with sparse_categorical_crossentropy loss function and Adam optimizer

Step3: Fit the X_train_padded and y_train_encoded dataset to LSTM Model with initialized hyper parameter
While maximum iteration not reached ***do***

Train LSTM Model

Step4: Evaluate LSTM model with X_test_padded data

Step5: Predict unseen data by LSTM Model

Step6: Measure the LSTM Model by accuracy metrics

Stop

Algorithm 4: To Train and Evaluate Bidirectional Long Short-Term Memory (Bi-LSTM) Model Amharic-English Code-Mixed Language Identification Task

Start

Input: Amharic-English sequences padded Data

Output: Trained Bi-LSTM Model

Initialization: epoch, batch size, neuron, dropout rate, verbose

Step1: Add embedding Layer with defined vocabulary size and input_length
add Bi-LSTM Layer with defined neurons
add dropout regularization
add dense layer with defined class and SoftMax activation function

Step2: Compile Bi-LSTM model with sparse_categorical_crossentropy loss function and Adam optimizer

Step3: Fit the X_train_padded and y_train_encoded dataset to Bi-LSTM Model with initialized hyper parameter
While maximum iteration not reached ***do***
Train Bi-LSTM Model

Step4: Evaluate Bi-LSTM model with X_test_padded data

Step5: Predict unseen data by Bi-LSTM Model

Step6: Measure the Bi-LSTM Model by accuracy metrics

Stop

3.6. Classification Metrics

To assess how well our model is performing, various classification metrics are available, including confusion matrices, accuracy, precision, recall, and the F1-score.

3.6.1. Confusion Metrics

The confusion matrix is employed to depict the accuracy of the model. Basic terms in confusion metrics described below.

- ***True positive (TP):*** The actually correct language label classified or predicted as correctly.
- ***True Negative (TN):*** Actually, non-language label predicted as Real non language label.
- ***False positive (FP):*** Incorrect language label classified as positively.
- ***False negative (FN):*** The correct language label predicted as negatively.

3.6.2. Accuracy

Accuracy is determined by adding the number of true positives and true negatives and then dividing this sum by the total of true positives, true negatives, false negatives, and false positives[14].

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \quad 3.4$$

3.6.3. Precision

Precision is computed by dividing the number of true positives by the total of true positives and false positives[14].

$$Precision = \frac{TP}{TP + FP} \quad 3.5$$

3.6.4. Recall

Recall is determined by dividing the number of true positives by the total of true positives and false negatives[14].

$$Recall = \frac{TP}{TP + FN} \quad 3.6$$

3.6.5. F1-Score

The F1-score is computed by taking two times the product of precision and recall and dividing it by the sum of precision and recall[14].

$$\text{F1} - \text{Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad 3.7$$

3.7. Tools

3.7.1. Data Preparation Tool

Data scraping tool is software designed to extract information from websites or web applications.

- ***Face Pager:*** is a data scraping tool used to extract data from Facebook, YouTube, and others. To collect our code-mixed data from public Facebook pages we have used face pager.
- ***Python Tool***: We have collected data from TikTok and YouTube using python code.

3.7.2. Data Processing Tools

Data Processing Tools is a software or utilities used to perform specific tasks, manipulate data, or analyze information in various domains. During preprocessing our Amharic- English code-mixed dataset, we used different preprocessing tools.

- ***Numerical Python (NumPy)*:** is popular for scientific computing tasks. This library provides multi-dimensional metrics and arrays with a collection of mathematical functions. NumPy is an essential tool for numerical tasks in Python.
- ***Pandas***: stands as an open-source Python library that serves as a valuable resource for conducting data analysis and handling data manipulation tasks. Notably, it offers a versatile array of data analysis tools and presents a user-friendly data structure capable of accommodating diverse data types.
- ***scikit-learn***: scikit-learn is a machine learning library that is open-source and built using Python that offers a comprehensive suite of algorithms and utilities catering to diverse machine learning tasks such as classification, clustering, regression, model

selection, and data preprocessing. Notably, this library is underpinned by foundational Python scientific libraries like SciPy, Matplotlib, and NumPy. In essence, scikit-learn not only streamlines the creation of robust and efficient machine learning models but also streamlines the entire machine learning workflow, simplifying the intricate processes involved in model development and evaluation.

3.7.3. Package Manager and Environments Tools

Package Manager is a software for installing, updating, and managing software packages. Environments Tools is utilities for creating and managing isolated software environments. Here we describe the anaconda and Jupyter Notebooks tools:

- ***Anaconda***: is an open-source R and Python programming language for machine learning, data science, and others. It provides multiple pre-installed libraries and tools.
- ***Jupyter Notebooks***: is an interactive web-based environment for sharing code, creating code, and visualizations. It also provides an interactive computational environment and allows users to combine code execution and texts in a single document. Jupyter Notebooks are a combination of cells, which can include either code, markdown text, or raw text.

3.7.4. Modeling and Documentation Tools

Modeling tools is a software or applications used to create, simulate, analyze, or visualize models, which can represent real-world systems, processes, or concepts. In this section we describe the modeling tools that we have used in the experiment.

- ***Python*:** It is a versatile and user-friendly high-level programming language known for its power and readability, often utilized as a valuable tool for machine learning and deep learning. When we compare to other programming languages Python provides a clean syntax, making it easier to write and understand.
- ***TensorFlow*:** It is a machine learning tool that is open-source and deep learning framework, can be utilized for code-mixed language modeling. It offers a range of tools and APIs for building and training neural network models, including language models.

TensorFlow's Keras API provides a high-level interface for developing language models.

- ***Keras*:** Keras is a Python-based open-source deep learning framework that offers a high-level and user-friendly interface for creating and training machine learning and deep learning models. It is designed to be easy to use, modular, and flexible, enabling researchers to swiftly experiment and iterate on their machine learning projects.
- ***PyCRFSuite*:** is widely used in NLP, information extraction, and sequence labeling tasks. It offers efficient and convenient way to incorporate CRF models into Python-based machine learning workflows, and enable researchers to build robust models for sequential data analysis tasks.
- ***Matplotlib*:** is widely used in the systematic, data analysis, and visualization communities due to its functionality, flexibility and integration with different python packages. For creating professional quality visualization and plots matplotlib provides powerful toolset.
- ***Seaborn*:** is popular in statistical analysis communities and data science. It has the ability to produce aesthetically informative visualizations. It is powerful toolkit for meaningful visualizations. Seaborn complements matplotlib by simplifying statistical plots and offers additional styles.
- ***Natural Language Toolkit (NLTK)*:** is A well-liked library commonly used for natural language processing. It serves a valuable resource for variety of text processing, from simple to complex application. It provides set of tools for tokenization, tagging, parsing, stemming and much more.
- ***Microsoft Word:*** is a versatile application widely employed for document creation and writing tasks.
- ***Microsoft Excel:*** is a powerful tool for dataset management and analysis, providing a robust platform for organizing and manipulating data.
- ***EdrawMax:*** is a powerful drawing tool, offering an extensive array of features, templates, and examples to empower users to accomplish diverse tasks with unparalleled versatility and efficiency.

- ***Mendeley Reference Manager:*** is a complimentary web and desktop application, serves to streamline the reference management process, enhancing workflow efficiency.
- ***Snipping Tool:*** is an integrated screenshot utility found in Windows Vista and later iterations, capable of capturing screenshots through diverse methods.

3.7.5. Hardware Tools

For the experiment, we utilized an HP personal computer equipped with an Intel(R) Core (TM) i7-8565U CPU running at 1.80GHz (turbo boosted to 1.99GHz), 8.00 GB of physical memory (with 7.89 GB usable), a 1000 Gigabyte hard disk for storage, and a 64-bit x64-based processor, all running on the Windows 10 Pro operating system.

3.7.6. Deployment Tools

We have deployed our best performing machine learning models with Flask to create a powerful language identification tool for code-mixed content. With this deployment, users can effortlessly input text and receive rapid and accurate predictions regarding the languages present in code-mixed language data. Flask's flexibility in handling web requests and seamlessly integrating with machine learning models has allowed me to transform my models into a practical and accessible solution for code-mixed language identification.

CHAPTER FOUR

4. PROPOSED APPROACH

In this chapter, we presented proposed models or approaches and techniques that are directly applied to our research. Here below seated the proposed model and system architecture diagram.

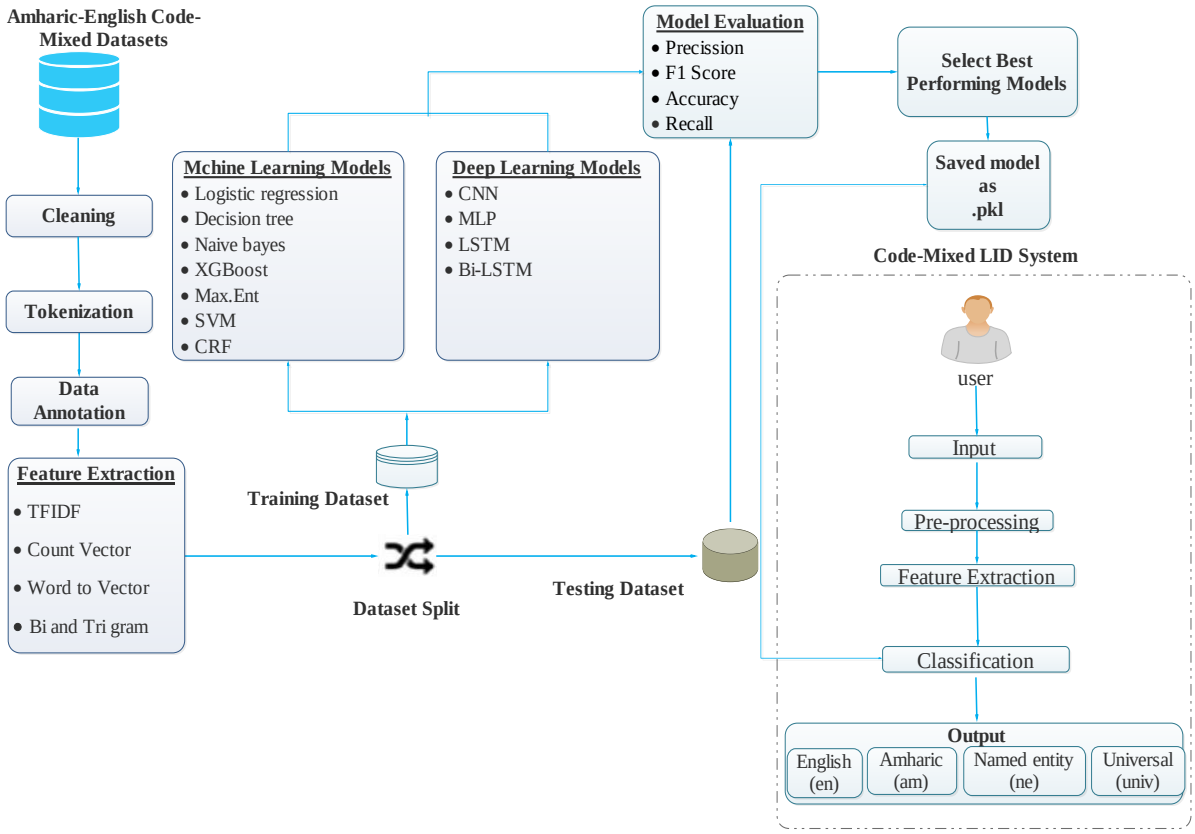


Figure 4.1. Proposed Model System Architecture

As shown in above Figure 4.1, we seated the general proposed model system process. Firstly, we collected Amharic-English code-mixed datasets from social media including (Facebook, YouTube, TikTok, and Telegram). The collected code-mixed data needs preprocessing, we preprocessed the dataset, this step includes (cleaning, filtering, tokenization, and POS tagging) and we divide the data into training and testing sets. Subsequently, we applied feature extraction techniques such as Word2Vec, Count Vectorization, TF-IDF, as well as bigram and trigram vectorization. Following the completion of this phase, we constructed various machine

learning models, including logistic regression, decision trees, naive Bayes, random forests, maximum entropy, support vector machines, conditional random field, and XGBoost and deep learning models like convolutional neural network, multi-layer perceptron, Long Short-Term Memory, and Bidirectional Long Short-Term Memory. These models were trained by training dataset and tested by testing dataset, and their performance was evaluated using metrics like F1-score, recall, precision, and accuracy. Ultimately, we identified the most effective models and deployed them through a Flask web interface for language tag prediction.

4.1. Preprocessing techniques

Tokenization, a key step in natural language processing (NLP), is the process of splitting text into smaller units known as tokens. NLTK (Natural Language Toolkit) is a robust Python library designed for various NLP tasks. One of its notable capabilities includes functions for both word tokenization and part-of-speech (POS) tagging, which are crucial components for comprehending and scrutinizing textual data. In our pursuit of code-mixed language identification, we employ a diverse set of feature extraction techniques to effectively capture the linguistic nuances present in the multilingual text. We begin with Count Vectorization, which transforms our text into a sparse matrix, representing the frequency of each word. Moving forward, we delve into the realm of TF-IDF (Term Frequency-Inverse Document Frequency) at the word level, a method that not only considers the frequency of words but also their importance within the corpus. As code-mixed text often involves the seamless blending of multiple languages, we venture into n-gram analysis, specifically focusing on bi and tri grams. By extracting these sequences of two or three consecutive words, we can capture meaningful phrases and linguistic patterns that may transcend language boundaries. Bi gram (2-gram) is a sequence of two consecutive words or characters in a text. A Tri gram (3-gram) is a sequence of three consecutive words or characters in a text. These features are instrumental in discerning the language dynamics within code-mixed content and contribute significantly to the accuracy of language identification models. Also, we used Word2Vec (word to vector), which is a technique used to change words to vectors, by capturing their semantic similarity, and association with adjacent text.

4.2. Proposed Models for Amharic-English Language Identification

In our comprehensive language identification research, we have harnessed the power of various machine learning and deep learning models, ranging from logistic regression and decision trees to naïve Bayes, maximum entropy, support vector machines, conditional random fields, random forest, and XGBoost and deep learning including CNN, MLP, LSTM, and Bi-LSTM. This diverse ensemble of models has allowed us to explore and leverage different learning strategies and classification approaches, enhancing the depth and breadth of our linguistic analysis. To empower these models, we have incorporated effective methods for feature extraction, which include Count Vectorization and TF-IDF (Term Frequency-Inverse Document Frequency), word to vector and Bi and Tri gram of word techniques. These techniques enable us to represent our text data in a structured and informative manner, capturing the essence of the languages and named entities present within the code-mixed content, thus enriching the foundation of our research. Deep learning techniques depicted in the following sections.

4.2.1. Feature Extraction and CNN Model

In the following figure shows the architecture of Convolutional Neural Network.

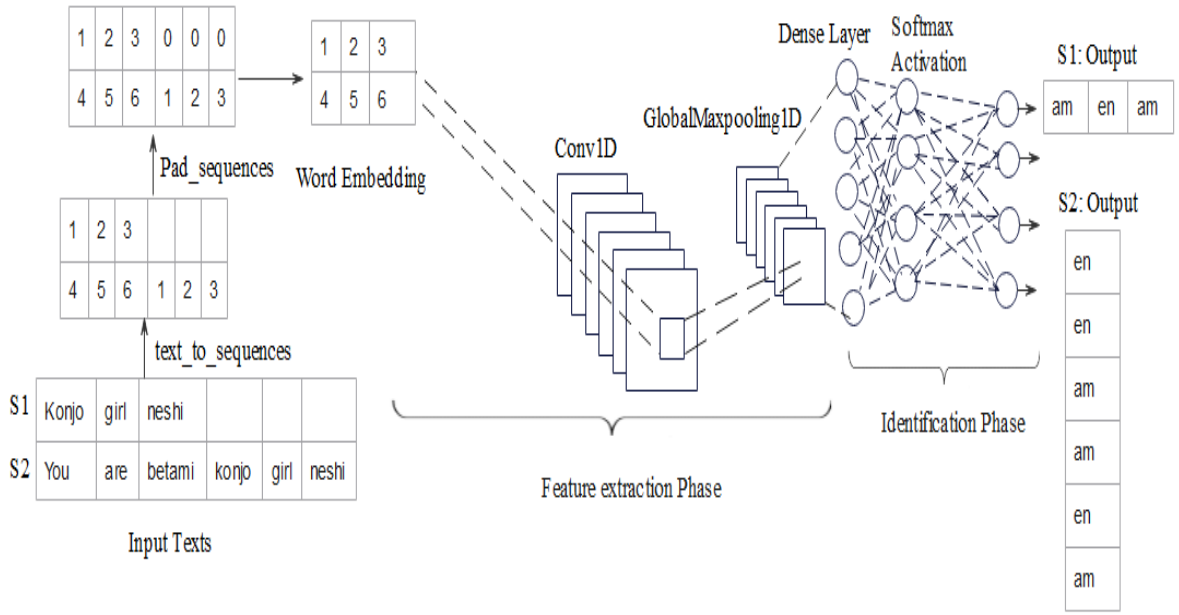


Figure 4.2. Proposed Convolutional Neural Network Model

As shown in Figure 4.2, Proposed CNN model the input text must be sequenced and padded and then passes to the word embedding layer to extract relevant feature from word and the embedded word pass to convolution 1D layer and Global Maximum Pooling layer to extract maximum feature from pattern of convolution word feature, the feature pass to dense layer with activation function to produce output finally produce output value for given input text.

4.2.2. Feature Extraction and LSTM Model

As shown in Figure 4.3, Proposed LSTM model the given input sentence tokenized using NLTK library and the tokenized words given to word embedding to capture the pattern of word feature after that LSTM take the feature of words with multiple neurons, dense layer, dropout layer and activation function finally display the corresponding language label of each word as output.

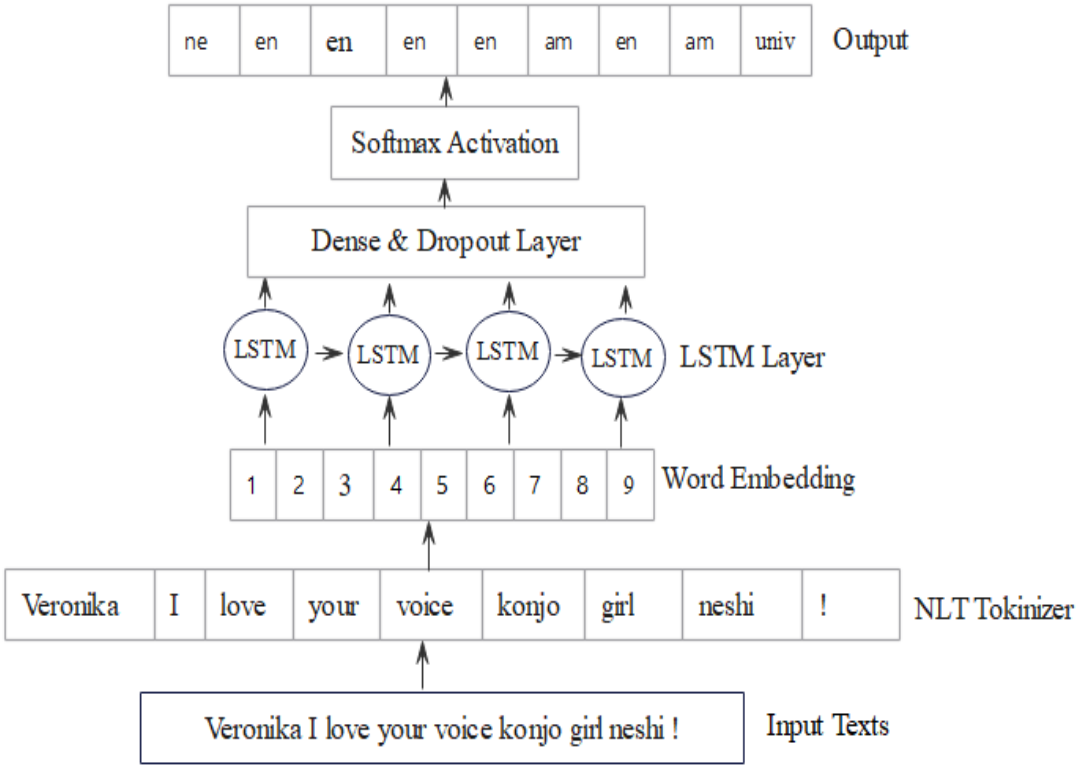


Figure 4.3. Proposed Long Short Term Memory Model

4.2.3. Feature Extraction and Bidirectional LSTM Model

As shown in Figure 4.4, Proposed LSTM model the given input sentence tokenized using NLTK library and the tokenized words given to word embedding to capture the pattern of word feature after that Bi-LSTM take the feature of words in both forward and backward direction with multiple neurons, dense layer, dropout layer and activation function finally display the corresponding language label of each word as output.

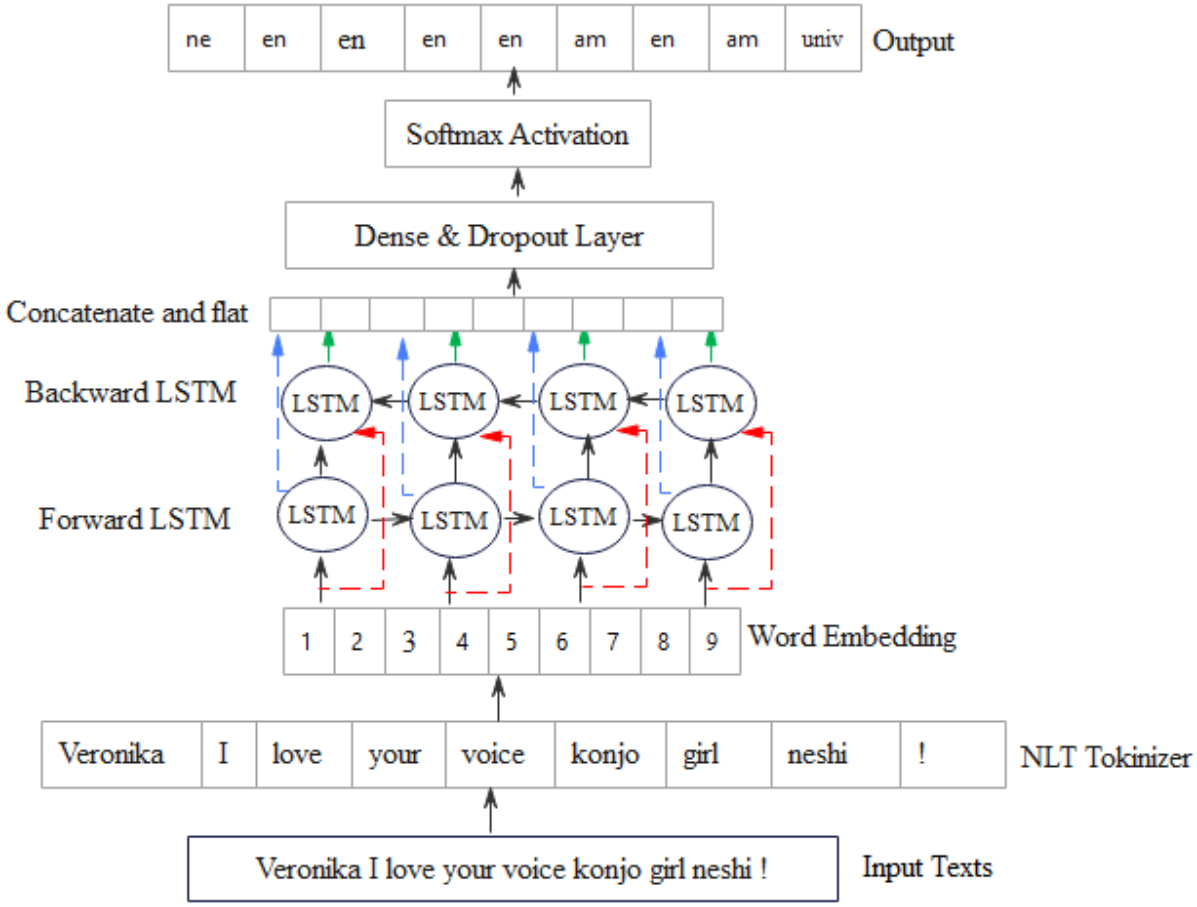


Figure 4.4. Proposed Bidirectional Long Short Term Memory Model

4.2.4. Feature Extraction and Multi-Layer Perceptron Model

As shown in Figure 4.5, Proposed MLP model, using word embedding the input text converted in to vector to extract relevant feature from the word and the embedded word vector pass to flatten layer and the feature pass to multiple dense layers with activation function finally produce output value for given input text.

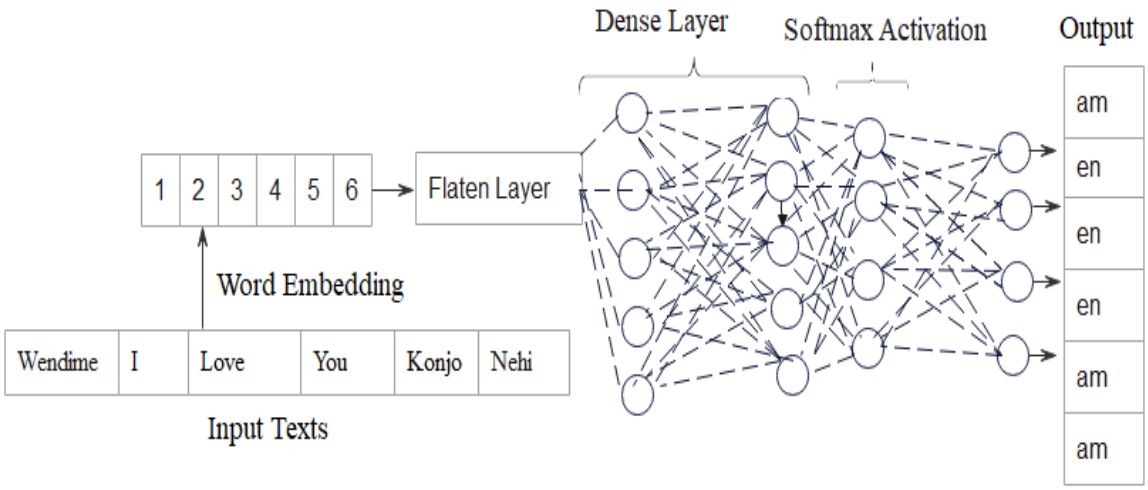


Figure 4.5. Proposed Multi-Layer Perceptron Model

4.3. Model Evaluation

In our rigorous evaluation of code-mixed Amharic-English language identification models, we employ a suite of performance metrics that provide a comprehensive understanding of their effectiveness. The cornerstone of our evaluation is the confusion matrix, a powerful tool that offers understanding the true positives, true negatives, false positives, and false negatives provides us with the means to evaluate the model's accuracy in correctly categorizing languages. Furthermore, we explore precision, recall, accuracy, and the F1-score, which are four essential metrics in the field of classification. Precision evaluates the precision of positive predictions, recall assesses the model's capability to accurately detect all pertinent instances, and the F1-score combines both precision and recall in a balanced manner, providing a comprehensive assessment of the model's performance.

4.4. Model Deployment

After we trained various machine learning and deep learning models, the top-performing models were incorporated into the Flask web framework. Flask is a Python-based, lightweight, and adaptable microweb framework that simplifies the rapid development of web applications, requiring minimal boilerplate code. Flask adheres to the WSGI (Web Server Gateway Interface) standard, facilitating its compatibility with a wide range of web servers.

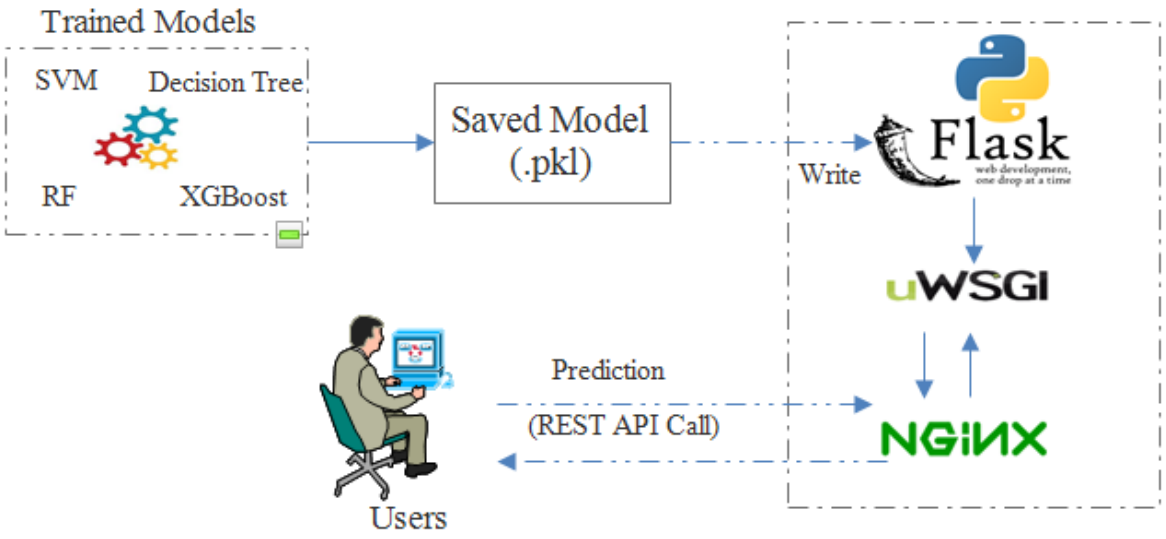


Figure 4.6. Deployment Architecture of Code-Mixed Language Identification

CHAPTER FIVE

5. IMPLEMENTATION OF CODE-MIXED LANGUAGE IDENTIFICATION MODEL

In this chapter, we introduce the machine learning and deep learning methods employed for the task of identifying code-mixed language in Amharic and English.

5.1. Data loading

Data loading refers to the process of reading and importing data from various sources into a Python program or environment for further processing, analysis, or manipulation. Python offers a wealth of libraries and modules designed to streamline the data loading process, facilitating the retrieval of data from a multitude of sources such as various file formats, databases, APIs, or online resources. A prime example of such a resourceful library is Pandas, renowned for its utility in data manipulation and analysis. Within Pandas, there exist dedicated functions expressly designed for reading data stored in the widely used CSV (Comma-Separated Values) format, ensuring seamless data access and integration for users.

```
# Load the dataset
data = pd.read_csv('lid_dataset.csv')
# Split the dataset into features and labels
X = data['Text']
y = data['Language_tag']
```

Figure 5.1. Data Loading Sample Code

5.2. Data Pre-Processing

5.2.1. Cleaning

In our datasets we have an emoji and unnecessary double spaces. So, we cleaned before feeding the data to the models. The following code shows data cleaning that we have applied.

```
def preprocess_text(text):
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'[\U0001F600-\U0001F64F\U0001F300-\U0001F5FF\U0001F680-\U0001F6FF\U0001F700-\U0001F77F\U0001F780-\U0001F7FF\U0001F800-
\U0001F8FF\U0001F900-\U0001F9FF\U0001FA00-\U0001FA6F\U0001FA70-\U0001FAFF\U0001FAB0-\U0001FABF\U0001FAC0-\U0001FAFF\U0001FAD0-
\U0001FADF\U0001FAE0-\U0001FAEF\U0001FAF0-\U0001FAFF\U0001FB00-\U0001FBFF\U0001FC00-\U0001FCFF\U0001F004-\U0001F0CF\U0001F170-
\U0001F19A\U0001F201-\U0001F251\U0001F300-\U0001F5FF\U0001F600-\U0001F64F\U0001F680-\U0001F6FF\U0001F700-\U0001F77F\U0001F780-
\U0001F7FF\U0001F800-\U0001F8FF\U0001F900-\U0001F9FF\U0001FA00-\U0001FA6F\U0001FA70-\U0001FAFF\U0001FAB0-\U0001FABF\U0001FAC0-
\U0001FAFF\U0001FAD0-\U0001FADF\U0001FAE0-\U0001FAEF\U0001FAF0-\U0001FAFF\U0001FB00-\U0001FBFF\U0001FC00-\U0001FCFF\U0001F004-
\U0001F0CF\U0001F170-\U0001F19A\U0001F201-\U0001F251]+', '', text)
    text = text.strip()

    return text

data['text'] = data['text'].apply(preprocess_text)
```

Figure 5.2. Data Cleaning Sample Code

5.2.2. Tokenization

Tokenization of the data that has been cleaned should be tokenized into words to easily assign the language and POS tag to words. To achieve this, we used the NLTK tokenizer.

```
#Using NLTK
import nltk
data_frame['Tweet']
for sent in data_frame['Tweet']:
    tokenize=nltk.word_tokenize(sent)
    for i,tok in enumerate(tokenize):
        print(i,tok)
```

Figure 5.3. Tokenization Sample Code

5.2.3. Part-of-Speech (POS) Taggers

POS tagging tools can assign part-of-speech labels to words in code-mixed text. By tagging words with their respective parts of speech, you can identify the language based on the grammatical patterns observed. Libraries such as NLTK or spaCy provide POS tagging capabilities for several languages.

```
tokenize=nltk.word_tokenize(sent)
   for i,pos in enumerate(tokenize):
          print(i,nltk.pos_tag([pos]))
```

Figure 5.4. POS tagging Sample Code

5.2.4. Split Data

Split the randomized dataset, which includes code-mixed language samples, into two separate groups: one for training and the other for testing. In this partitioning scheme, the training set is allocated the larger portion of the data, serving as the primary data source for model training and learning. In contrast, the testing set is reserved to represent a set of unseen data, strategically designated for the purpose of evaluating and assessing the model's performance, ensuring an objective and rigorous examination of its capabilities.

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 5.5. Split the Dataset Sample Code

- ***X:*** represents the input data, a feature matrix, or a list of input samples
- ***Y:*** represents the corresponding labels or language tags associated with the input data
- ***train_test_split:*** is a sci-kit-learn's model_selection module function that performs the data split
- ***test_size:*** It represents the portion of the data reserved for testing purposes. In this scenario, it's configured at 0.2, indicating that 20% of the data will be designated for testing, and the remaining 80% will be employed for training.
- ***random_state:*** to ensure reproducibility by fixing the random seed. Using the same random state value guarantees that the data split will be consistent across different runs.

5.2.5. Texts_to_sequences

texts_to_sequences () function converts each text in the texts list into a sequence of integers. The integers represent the unique words in the texts, based on the order in which they appeared during the fitting of the tokenizer.

5.2.6. Sequence Padding

Padding the sequences ensures that all input sequences have the same length, which is required when feeding the data into the model. It allows for efficient batch processing and consistent input sizes.

```
max_seq_length = 100  # Maximum sequence length
X_train_padded = pad_sequences(X_train_tokenized, maxlen=max_seq_length, padding='post')
X_test_padded = pad_sequences(X_test_tokenized, maxlen=max_seq_length, padding='post')
```

Figure 5.6. Pad Sequence Sample Code

Figure 5.6 show, pad_sequences () is a function from Keras that pads sequences to a specified length. It takes the X_train_tokenized sequences as input and pads them to have a maximum length of max_seq_length. The padding=post argument specifies that padding should be added at the end of the sequences. X_test_tokenized sequences to have the same length as max_seq_length by adding padding at the end of the sequences.

5.3. Feature Extraction

Feature extraction is the process of extracting relevant feature from the dataset. Vectorization is one of the feature extraction techniques and it is a fundamental stage in natural language processing (NLP) activities. It encompasses the transformation of text-based data into a numerical format that machine learning and deep learning models can understand and work with.

5.3.1. Count Vectorizer

A Count Vectorizer is a technique that is used to convert a collection of text documents into a numerical format that can be used for machine learning and statistical modeling. It is a simple and commonly used method for text vectorization.

```
vectorizer = CountVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

Figure 5.7. Count Vectorizer Sample Code

5.3.2. TF-IDF Vectorizer

TF-IDF, which stands for Term Frequency-Inverse Document Frequency, is a frequently employed method in natural language processing (NLP) for assessing the importance of words in a collection of text. It is used for extracting features from text. Initially, we

transform the input code-mixed words into vectors that represent the TF (Term Frequency) and IDF (Inverse Document Frequency).

```
# Vectorize the text data
tfidf_vec = TfidfVectorizer(analyzer='word', token_pattern=r'\w{1,}', max_features=5000)
X_train_tfidf_vector = tfidf_vec.fit_transform(X_train)
X_test_tfidf_vector = tfidf_vec.transform(X_test)
```

Figure 5.8. TFIDF Vectorizer Sample Code

5.3.3. Word to Vector

Word2Vec, short for word to vector, serves as a technique employed to convert words into vectors. This conversion process helps in capturing the inherent meaning, semantic similarity, and relationships that words share within their surrounding textual context.

```
# Train Word2Vec model on the training data
# Make sure to adjust Word2Vec parameters like window size, vector size, etc.
sentences = [text.split() for text in X_train]
word2vec_model = Word2Vec(sentences, vector_size=100, window=5, min_count=1, sg=0)

# Convert text data to word embeddings using the trained Word2Vec model
def text_to_word_embeddings(text, model):
    words = text.split()
    embeddings = [model.wv[word] for word in words if word in model.wv]
    if embeddings:
        return np.mean(embeddings, axis=0)
    else:
        return np.zeros(model.vector_size)  # Use zero vector for out-of-vocabulary words

X_train_word_embeddings = np.array([text_to_word_embeddings(text, word2vec_model) for text in tqdm(X_train)])
X_test_word_embeddings = np.array([text_to_word_embeddings(text, word2vec_model) for text in tqdm(X_test)])
```

Figure 5.9. Word to Vector Vectorizer Sample Code

5.3.4. Bi and Tri Gram

A bigram is a pair of two consecutive words or characters found within a text. On the other hand, a trigram (also referred to as a 3-gram) is a series of three consecutive words or characters within a text.

```
Biandtri-gram = TfidfVectorizer(analyzer='word', token_pattern=r'\w{1,}', ngram_range=(2,3), max_features=5000)
X_train_Biandtri-gram=Biandtri-gram.fit_transform(X_train)
X_test_Biandtri-gram=Biandtri-gram.transform(X_test)
```

Figure 5.10. Bi and Tri gram Vectorizer Sample Code

5.4. Machine Learning Techniques for Amharic-English Code-Mixed Language Identification

After we preprocess our dataset, we train various machine learning and deep learning models for the task of Language Identification. This task involves assigning a language identification label to individual words in a sentence, considering their context. We have established machine learning models including Logistic Regression, Decision Tree, Support Vector Machine, Naive Bayes Classifier, Random Forest and Conditional Random Field and deep learning including MLP, CNN, LSTM, and Bi-LSTM. In the following sub section, we have defined each model used in our research.

5.4.1. Logistic Regression

In our Amharic-English code-mixed dataset, we employed logistic regression for language identification tasks by customizing it to address the unique attributes and complexities associated with code-mixed text, we apply LR, for each Amharic and English as well as non-language-tag like Named entity and universal, for this task we used sklearn. linear model libraries. Logistic Regression Classifier trained with different train vectorized dataset. The vectorizer includes count vectorizer, TFIDF, and bi and tri gram and word to vector. Finally, we predict the language tag based on the vectorized testing dataset also to measure the performance of the model we used the Recall, Precision, F1-Score, and Accuracy metrics.

5.4.2. Naïve Bayes Classifier

In our Amharic-English code-mixed dataset, we implemented a baseline Naive Bayes model for language identification. We approach this language identification task as a form of text classification. Each sentence is treated as a document, and each word within it is viewed as a term. We compute the conditional probability for each class label based on these terms. The key assumption of Naive Bayes is that features are independent of each other. To begin, we convert the input code-mixed words into Count Vectors, Term Frequency-Inverse Document Frequency (TF-IDF) representations, Word2Vec embeddings, as well as bigram and trigram vectors. We then train the model using a set of n word vectors along with their corresponding

labels, and subsequently test the model with the testing data. Finally, we assess the Naive Bayes model using metrics such as the accuracy F1 score, recall, and precision.

5.4.3. Decision Tree

A decision tree is a well-known machine learning algorithm in supervised learning that is applied for tasks involving both classification and regression. It is a flowchart-like model that makes predictions based on a series of decisions and conditions applied to the input features, so in our Amharic-English code-mixed dataset, we apply a decision tree classifier to classify language labels for each Amharic and English as well as non-language tag include named entity and universal for this from sklearn. tree libraries we used. Decision Tree Classifier with the count vectorizer, TFIDF and bi and tri gram, and word to vector, both count vectorizer and TFIDF vectorizer vectorize the input training dataset in then transform the vectorized data using the fit-transform method also we consider the bi and tri gram feature of given words. Finally, we train the transformed training vectorized Amharic-English code-mixed dataset for the Decision Tree classifier and predict the language tag based on the testing dataset also to measure the performance of the model we used the Confusion matrix, Recall, Precision, F1-Score, and Accuracy metrics.

5.4.4. Support Vector Machine

The task of recognizing the language at the word level within code-mixed text can be conceptualized as a classification problem. In our experimentation, we have incorporated the Support Vector Machine (SVM) as an essential component. The basis behind including SVM in our study is its remarkable proficiency as a supervised machine learning algorithm, skilled at addressing both classification and regression tasks. SVM, well-known for its ability in binary classification, extends its capabilities to tackle multi-class classification, a prerequisite for language identification tasks. To accomplish this, we used various vectorization techniques, including count vectorization, TFIDF, bi and tri grams, and word-to-vector vectorization, carefully applied to the Amharic-English code-mixed dataset, and then train an SVM classifier to classify language labels for each Amharic and English as well as non-language tag like named entity and universal. The SVM classifier effectively predicts the language tag based on the testing dataset. For the comprehensive evaluation of our model's performance, we

employed an array of metrics, including the Recall, Precision, F1-Score, and Accuracy, providing a multifaceted assessment of its efficacy.

5.4.5. Random Forest

We leveraged the power of a Random Forest model to tackle the intricacies of our code-mixed Amharic-English dataset. To enhance the model's performance, we applied a variety of pre-processing techniques, including Count Vectorization, TF-IDF (Term Frequency-Inverse Document Frequency), and Word2Vec (Word to Vector) approaches. These techniques transformed our raw textual data into numerical features that the model could understand. Our training data was meticulously prepared using these pre-processing methods, creating training vectors that served as the foundation for our Random Forest model. Subsequently, we employed a separate testing vector to evaluate the model's performance rigorously. We assessed the model's effectiveness using key metrics such as F1 score, precision, accuracy, recall, and visualized the results with a confusion matrix. This comprehensive approach allowed us to gain valuable insights into the model's ability to understand and classify code-mixed content, providing a solid foundation for our research.

5.4.6. Maximum Entropy Classifier

We include also a maximum entropy classifier in our experiment, we used MaxEnt to train a classifier that learns the patterns and count vector, word to vector, bi and tri gram, and TFIDF vector features associated with the code-mixed text. preprocess our data to extract relevant features to train the classifier based on the training dataset and validate the classifier with the testing dataset. We also evaluate the maximum entropy classifier with Recall, Precision, F1-Score, and Accuracy metrics.

5.4.7. XGBoost Classifier

XGBoost, short for Extreme Gradient Boosting, is a powerful machine learning library that employs scalable and distributed gradient-boosted decision trees. It excels in parallel tree boosting and stands as a premier machine learning library, particularly suited for tackling regression, classification, and ranking challenges, we used XGBoost to train a classifier that learns the patterns and count vector, word to vector, bi and tri gram and TFIDF vector features

associated with the code-mixed text. preprocess our data to extract relevant features, train the classifier based on the training dataset and validate the classifier with the testing dataset. We also evaluate the XGBoost classifier with Recall, Precision, F1-Score, and Accuracy metrics.

5.4.8. Conditional Random Field

We have implemented a word-level language identification model using Conditional Random Field (CRF) in conjunction with crfpysuite, an open-source and customizable tool. A significant aspect of this approach is featuring selection. We devised various feature templates based on different combinations of available words, their context, and potential tags. For this model, the following feature sets were employed to train the CRF model:

- ***Current word and its POS tag:*** We take into account the current testing words and their respective part-of-speech (POS) tags to predict the language label.
- ***Next word and its POS tag:*** We consider the next word and its POS tag to capture the relationship between the current word and the subsequent word.
- ***Previous word and its POS tag:*** To extract context related to the current word, we consider the preceding word and its POS tag based on the first-order Markov assumption.
- ***Prefix and suffix of the focal word:*** We extract the prefix and suffix of the current word. If the word lacks either a prefix or a suffix, we include a feature marked as NULL.
- ***Length of a word:*** The length of the word is utilized as a feature.
- ***Starts with a numeric digit*****:** We determine whether the word commences with a numeric digit.
- ***Contains numeric digits:*** We use regular expressions to detect if the words contain any numerical digits.
- ***Starts with a special symbol:*** We check if the words start with special characters such as !, @, $, /, etc.
- ***Starts with a capital letter:*** This feature is set to True if the word begins with a capital letter, otherwise, it is marked as False.

- ***Contains any capital letters:*** We assess whether the word contains any capital letters, like in the case of oFteN (often).
- ***Previous language tag:*** To predict the language label of the current word, we consider the language label assigned to the preceding word.

```
def word2features(sent, i):
word = sent[i][0]
postag = sent[i][1]

features = {
    'bias': 1.0,
    'word.lower()': word.lower(),
    'word[-3:]': word[-3:],
    'word.isupper()': word.isupper(),
    'word.istitle()': word.istitle(),
    'word.isdigit()': word.isdigit(),
    'postag': postag,
    'postag[:2]': postag[:2],
}
if i > 0:
    word1 = sent[i-1][0]
    postag1 = sent[i-1][1]
    features.update({
        '-1:word.lower()': word1.lower(),
        '-1:word.istitle()': word1.istitle(),
        '-1:word.isupper()': word1.isupper(),
        '-1:postag': postag1,
        '-1:postag[:2]': postag1[:2],
    })
else:
    features['BOS'] = True

if i < len(sent)-1:
    word1 = sent[i+1][0]
    postag1 = sent[i+1][1]
    features.update({
        '+1:word.lower()': word1.lower(),
        '+1:word.istitle()': word1.istitle(),
        '+1:word.isupper()': word1.isupper(),
        '+1:postag': postag1,
        '+1:postag[:2]': postag1[:2],
    })
else:
    features['EOS'] = True

return features
```

Figure 5.11. Word Feature Extraction Sample Code for CRF

5.5. Build Deep Learning Models for Amharic-English Code-Mixed Language Identification

5.5.1. Convolutional Neural Network

Convolutional Neural Networks are mostly used for images, but we have seen that CNNs perform on text as well in few applications. In our experiment we include CNN, so here we used the Tokenizer class from TensorFlow. keras. preprocessing. Text is used to tokenize the input text. The fit_on_texts method is used to fit the tokenizer on the training data, and texts_to_sequences are used to convert the text to sequences of tokens. The pad_sequences function from TensorFlow. keras. preprocessing. Sequence is used to pad the tokenized sequences to have the same length. The maxlen parameter is set to max_seq_length to define the maximum sequence length, and padding is set to post to pad the sequences at the end. The LabelEncoder class from sklearn. Preprocessing is used to encode the target labels. The fit transform method is used to fit the label encoder on the training labels and encode them, and transform is used to encode the test labels. The vocabulary size is determined as the length of the tokenizer's word index plus 1. The CNN model is built using the Sequential model from tensor flow. keras. Models. It consists of an embedding layer, a 1D convolutional layer, a global max pooling layer, and two dense layers. The model is compiled with the adam optimizer and the sparse_categorical_crossentropy loss function. The accuracy metric is also specified. The model is trained using the fit method, with the padded training sequences and encoded training labels as inputs. The number of epochs, batch size, and verbosity level are specified.

5.5.2. Multi-Layer Perceptron

A Multi-Layer Perceptron (MLP) is a supervised feed forward neural networks, which consists of an input, an output layer and single hidden layers each layer consists of a set of 100 neurons that receives input from the previous layer and sends output to neurons in the next layer based on activation function. We building a Multilayer Perceptron (MLP) classifier using scikit-learn's MLPClassifier class. Also, we need to import the MLPClassifier class from scikit-learn's neural_network module. The MLPClassifier is an implementation of a feedforward neural network with a multilayer perceptron architecture. To create an instance of the

MLPClassifier class. The hidden_layer_sizes parameter specifies the number of hidden layers and the number of neurons in each hidden layer. In our case, (100,) indicates a single hidden layer with 100 neurons. The activation parameter specifies the activation function to be used in the hidden layers. The relu activation function stands for Rectified Linear Unit, which is commonly used in neural networks. The solver parameter specifies the optimization algorithm to use. adam refers to the Adam optimizer, which is an efficient optimization algorithm commonly used for training neural networks. random_state parameter is helps to assign the random number generator. Setting it to a specific value, such as 42, ensures reproducibility of the results if you run the code multiple times. Once the MLP classifier is built, we can proceed to train and evaluate it using our dataset.

5.5.3. Long Short-Term Memory

The dataset is split into training and testing sets, where the input text is assigned to the variable X and the language labels are assigned to y. Next, the input text is tokenized using the Tokenizer class from a library. The tokenizer is fitted on the training set to build the vocabulary, and the training and testing text sequences are converted to sequences of integers. To ensure equal input lengths, the sequences are padded with zeros using the pad_sequences () function. The maximum sequence length is set to 100. The target labels are encoded using a LabelEncoder, where the training and testing labels are transformed into numeric form. The vocabulary size is determined by adding 1 to the length of the tokenizer's word index. An LSTM model is then constructed using the Sequential model from a library. The model consists of an Embedding layer, which maps the input sequence to dense word vectors. This is followed by an LSTM layer with 128 units and a ReLU activation function. A Dropout layer with a rate of 0.2 is added to mitigate overfitting. Finally, a Dense layer with the number of output classes and a ReLU activation function is included. The model is compiled with the Adam optimizer, using the sparse categorical cross-entropy loss function and accuracy as the metric. Training is performed using the fit () function, providing the padded training sequences and encoded labels. The model is trained for 15 epochs, with a batch size of 32.

5.5.4. Bidirectional Long Short-Term Memory

The Long Short Term Memory model is constructed using the sequential model, starting with an embedding layer. This layer maps the input sequence of words to dense word vectors of size 100. The following layer is a bidirectional LSTM layer with 128 units. This bidirectional layer processes the input sequence in both forward and backward directions, allowing the model to capture contextual information from both directions. This bidirectional aspect can be beneficial for tasks that require a deeper understanding of the input sequence, such as language identification. To mitigate overfitting, a dropout layer with a rate of 0.2 is added. This layer randomly drops out 20% of the connections between LSTM units during training, preventing the model from relying too heavily on specific units and improving its generalization ability. The final layer of the model is a dense layer with the number of output classes and a SoftMax activation function. The SoftMax activation ensures that the model's output is a probability distribution over the different language classes, allowing it to make predictions by selecting the class with the highest probability. The model is compiled with the Adam optimizer, a popular optimization algorithm. The loss function is set to sparse_categorical_crossentropy, which is appropriate for multi-class classification tasks. Additionally, the accuracy metric is specified for evaluation during training. The training process utilizes the fit () function to train the model. The padded training sequences (X_train_padded) and their corresponding encoded labels (y_train_encoded) are passed as inputs. The model is trained for 15 epochs, meaning it iterates over the entire training dataset 15 times. A batch size of 32 is chosen, indicating that the model updates its weights after processing 32 samples at a time. The verbose parameter is set to 1, which displays the training progress for each epoch.

5.6. Models Layer and Parameters

5.6.1. Dropout

Dropout regularization is a technique used to prevent overfitting in neural networks. In this technique, a certain percentage of units or neurons in the network are randomly dropped or ignored during training. In your case, the dropout rate is set to 20%.

5.6.2. Dense Layer

A Dense layer in a neural network is like a connecting bridge that takes input data and transforms it into output data. It is called dense because every neuron in this layer is connected to every neuron in the previous layer. This means that information flows freely and densely between the layers.

5.6.3. Optimizer

In our implementation, we utilized the Adam optimizer, which combines the strengths of two optimization techniques: the Adaptive Gradient Algorithm and Root Mean Square Propagation (RMSprop).

5.6.4. Activation Function

For our models we used two activation functions such as rectified linear unit (relu) and softmax, these activation functions preferable for multi class classification, the model predicted class is four language labels these are Amharic, English, Named Entity and Universal, we considered as multiple class, the reason why use relu and softmax activations.

Table 5.1. Models Hyper Parameter

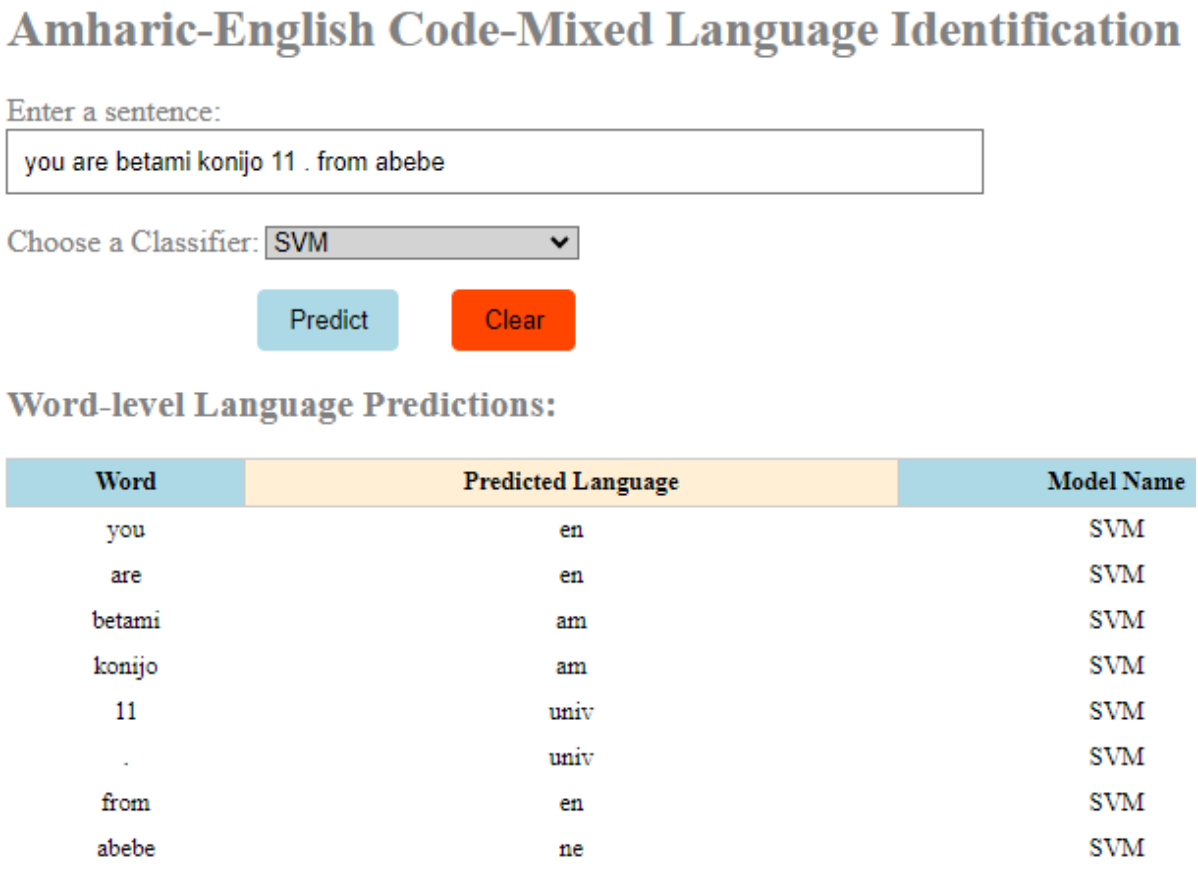
Hyper Parameter	***Value***
Number of units per Layer	128
Epochs	15
Batch Size	32
Dropout	0.2
Loss	sparse_categorical_crossentropy
Activation	relu and softmax
Optimizer	Adam

5.7. Model Deployment

In model deployment, we have deployed high-performing models using the Flask web framework.

Figure 5.12. Amharic-English Code-Mixed Language Identification Interface

In Figure 5.12 represented Amharic-English Code-Mixed Language Identification Interface, we implemented using HTML and CSS.



Word	Predicted Language	Model Name
you	en	SVM
are	en	SVM
betami	am	SVM
konijo	am	SVM
11	univ	SVM
.	univ	SVM
from	en	SVM
abebe	ne	SVM

Figure 5.13. Amharic-English Code-Mixed Language Identification Prototype

In Figure 5.13 depicted Amharic-English Code-Mixed Language Identification deployment, we implemented using HTML and CSS and server side using FLASK web framework.

CHAPTER SIX

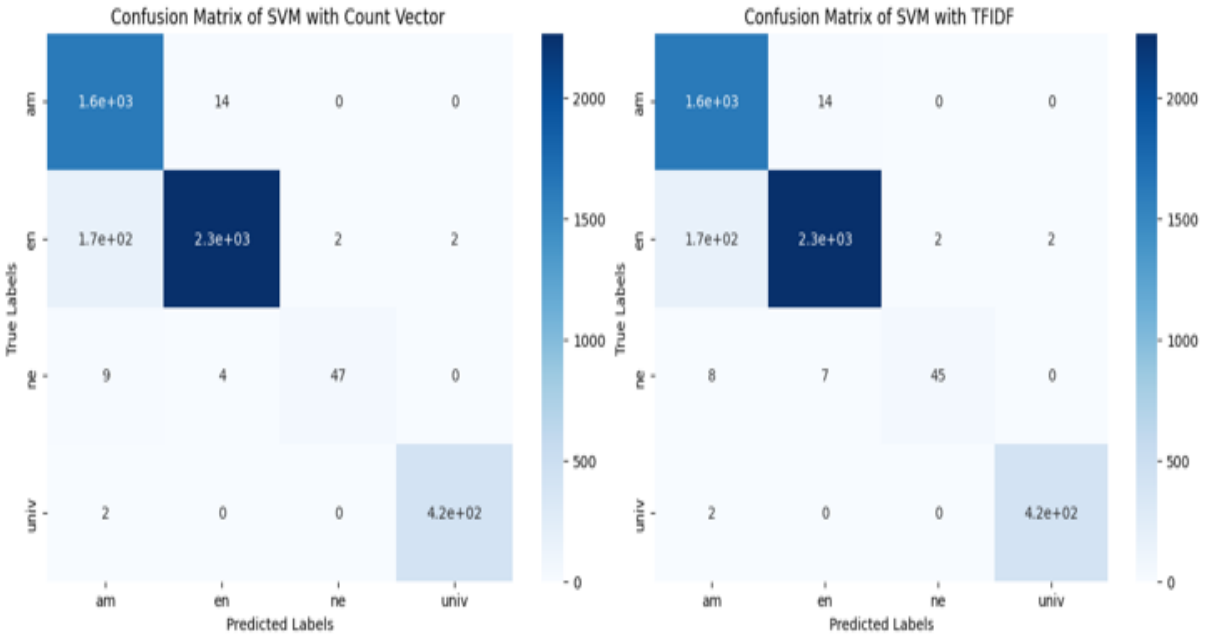
6. RESULTS AND DISCUSSIONS

In this chapter, we showcased the outcomes of our research experiments. Based on the testing data confusion matrix and classification report is generated. We used a sklearn library for the model evaluation reports.

6.1. Performance Evaluation of Machine learning and Deep Learning Models

6.1.1. Confusion Matrix of Machine Learning Models

Here, many machine learning models tested for Amharic-English code-mixed language identification problem.



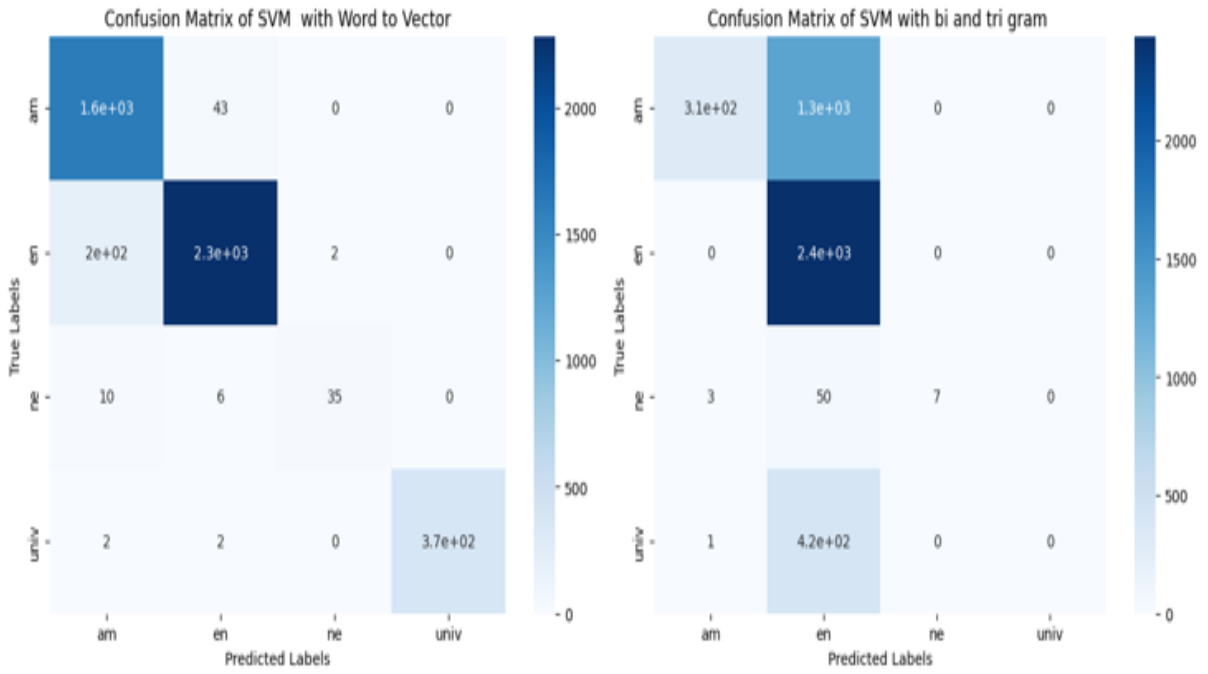


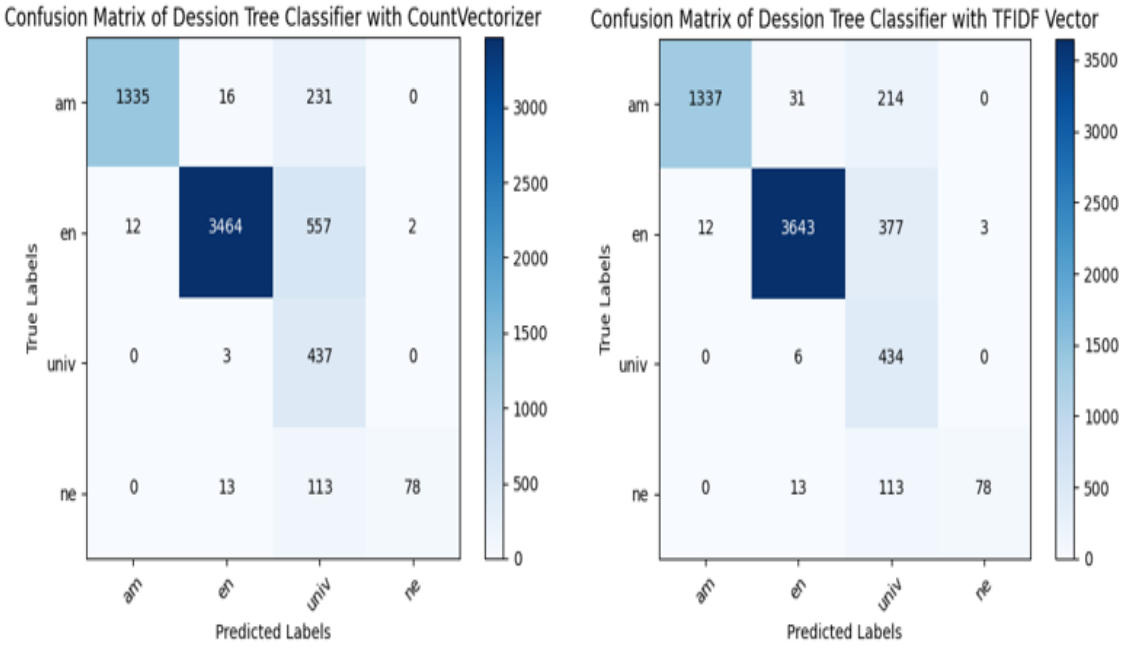
Figure 6.1. Confusion Matrix of SVM

As shown in Figure 6.1, we have a confusion matrix of SVM with the following labels: Amharic, English, Named entity, and Universal for SVM with count vector, TFIDF, word to vector, and bi and tri gram. Within a confusion matrix, actual classes are depicted in rows, while predicted classes are depicted in columns. The entries in the matrix denote the quantities of true positives, false positives, false negatives, and true negatives for each class.

- ***SVM with Count Vector*:** for am, there were 1624 true positives (predicted as am) and 14 false positives (predicted as en). For en, there were 2267 true positives (predicted as en), 170 false negatives (predicted as am), 2 false positives (predicted as ne), and 2 false positives (predicted as univ). For ne, there were 47 true positives (predicted as ne), 4 false negatives (predicted as en), 9 false negatives (predicted as am). For univ, there were 420 true positives (predicted as univ) and 2 false negatives (predicted as am).
- ***SVM with TFIDF*:** for am, there were 1624 true positives (predicted as am) and 14 false positives (predicted as en). For en, there were 2267 true positives (predicted as en), 170 false negatives (predicted as am), 2 false positives (predicted as ne), and 2 false positives (predicted as univ). For ne, there were 45 true positives (predicted as

ne), 7 false negatives (predicted as en), and 8 false negatives (predicted as am). For univ, there were 420 true positives (predicted as univ) and 2 false negatives (predicted as am).

- ***SVM with Word to Vector*:** for am, there were 1614 true positives (predicted as am) and 43 false positives (predicted as en). For en, there were 2282 true positives (predicted as en), 198 false negatives (predicted as am), and 2 false positives (predicted as ne). For ne, there were 35 true positives (predicted as ne), 6 false negatives (predicted as en), 10 false negatives (predicted as am). For univ, there were 367 true positives (predicted as univ), 2 false negatives (predicted as en), and 2 false negatives (predicted as am).
- ***SVM with bi and tri gram*:** for am, there were 310 true positives (predicted as am) and 1328 false positives (predicted as en). For en, there were 2441 true positives (predicted as en). For ne, there were 7 true positives (predicted as ne), 50 false negatives (predicted as en), and 3 false negatives (predicted as am). For univ, there were 421 false negatives (predicted as en) and 1 false negative (predicted as am).



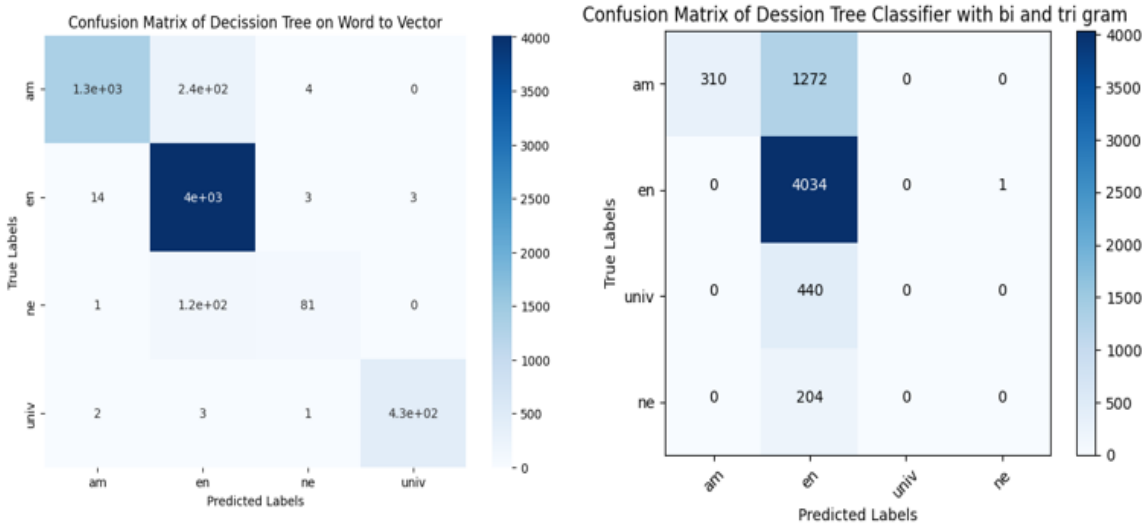


Figure 6.2. Confusion Matrix of Decision Tree

As shown in Figure 6.2, indicate the confusion matrix of the decision tree with count vector, TFIDF, word to vector, and bi and tri gram.

- ***With Count Vector*:** for am, there were 1335 true positives (predicted as am), 16 false positives (predicted as en), and 231 false positives (predicted as univ). For en, there were 3464 true positives (predicted as en), 12 false negatives (predicted as am), 557 false positives (predicted as univ), and 2 false positives (predicted as ne). For univ, there were 437 true positives (predicted as univ) and 3 false negatives (predicted as en). For ne, there were 78 true positives (predicted as ne), 113 false negatives (predicted as univ), and 13 false negatives (predicted as en).
- ***With TFIDF*:** for am, there were 1337 true positives (predicted as am), 31 false positives (predicted as en), and 214 false positives (predicted as univ). For en, there were 3643 true positives (predicted as en), 12 false negatives (predicted as am), 377 false positives (predicted as univ), and 3 false positives (predicted as ne). For univ, there were 434 true positives (predicted as univ) and 6 false negatives (predicted as en). For ne, there were 78 true positives (predicted as ne), 113 false negatives (predicted as univ), and 13 false negatives (predicted as en).
- ***With Word to Vector*:** for am, there were 1336 true positives (predicted as am), 242 false positives (predicted as en), and 4 false positives (predicted as ne). For en, there were 4015 true positives (predicted as en), 14 false negatives (predicted as am), 3 false

positives (predicted as ne), and 3 false positives (predicted as univ). For ne, there were 81 true positives (predicted as ne), 122 false negatives (predicted as en), and 1 false negative (predicted as am). For univ, there were 434 true positives (predicted as univ), 2 false negatives (predicted as am), 3 false negatives (predicted as en), and 1 false negative (predicted as ne).

- ***With Bi and Tri Gram*:** for am, there were 310 true positives (predicted as am) and 1272 false positives (predicted as en). For en, there were 4034 true positives (predicted as en) and 1 false positive (predicted as ne). For univ, there were 440 false negatives (predicted as en). For ne, there were 204 false negatives (predicted as en).

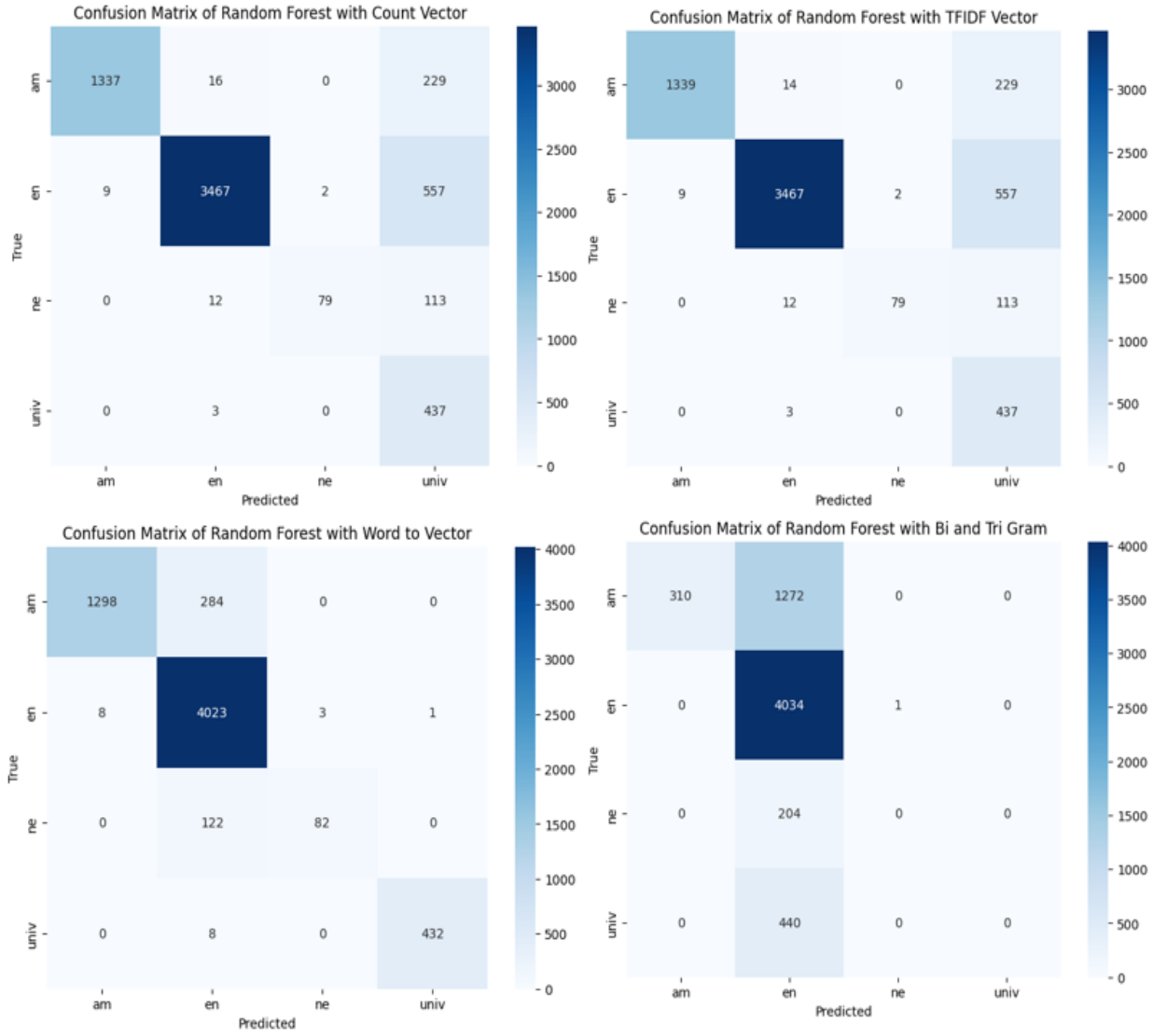


Figure 6.3. Confusion Matrix of Random Forest

As shown in Figure 6.3, indicate the confusion matrix of Random Forest with count vector, TFIDF, word to vector, and bi and tri gram.

- ***With Count Vector*:** for am, there were 1337 true positives (predicted as am), 16 false positives (predicted as en), and 229 false positives (predicted as univ). For en, there were 3467 true positives (predicted as en), 9 false negatives (predicted as am), 2 false positives (predicted as ne), and 557 false positives (predicted as univ). For ne, there were 79 true positives (predicted as ne), 12 false negatives (predicted as en), and 113 false positives (predicted as univ). For univ, there were 437 true positives (predicted as univ) and 3 false negatives (predicted as en).
- ***With TFIDF:*** for am, there were 1339 true positives (predicted as am), 14 false positives (predicted as en), and 229 false positives (predicted as univ). For en, there were 3467 true positives (predicted as en), 9 false negatives (predicted as am), 2 false positives (predicted as ne), and 557 false positives (predicted as univ). For ne, there were 79 true positives (predicted as ne), 12 false negatives (predicted as en), and 113 false positives (predicted as univ). For univ, there were 437 true positives (predicted as univ) and 3 false negatives (predicted as en).
- ***With Word to Vector:*** for am, there were 1298 true positives (predicted as am) and 284 false positives (predicted as en). For en, there were 4023 true positives (predicted as en), 8 false negatives (predicted as am), 3 false positives (predicted as ne), and 1 false positive (predicted as univ). For ne, there were 82 true positives (predicted as ne) and 122 false negatives (predicted as en). For univ, there were 432 true positives (predicted as univ) and 8 false negatives (predicted as en).
- ***With Bi and Tri Gram:*** for am, there were 310 true positives (predicted as am) and 1272 false positives (predicted as en). For en, there were 4034 true positives (predicted as en) and 1 false positive (predicted as ne). For ne, there were 204 false negatives (predicted as en). For univ, there were 440 false negatives (predicted as en).

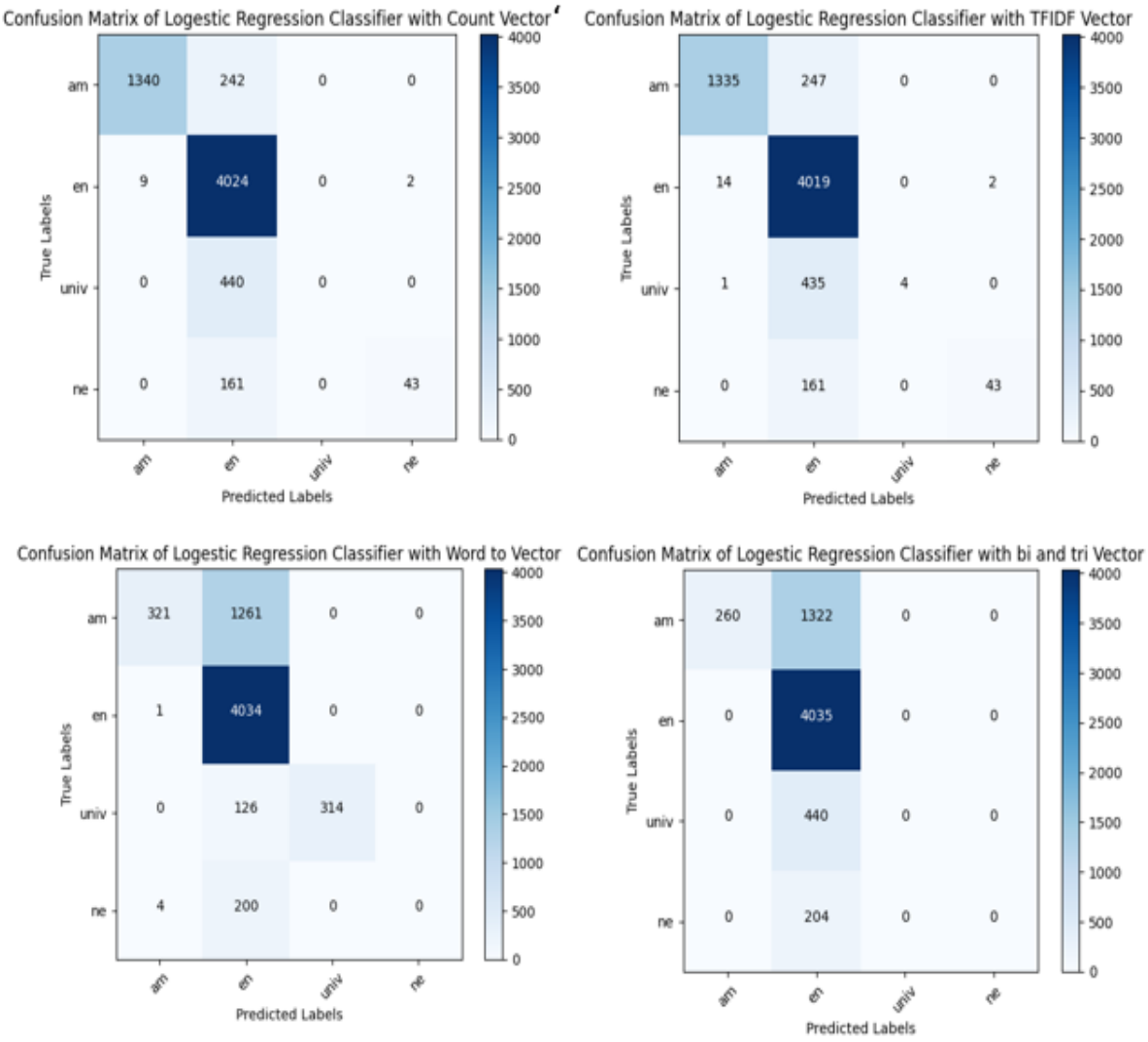


Figure 6.4. Confusion Matrix of Logistic Regression

As shown in Figure 6.4, indicate the confusion matrix of Logistic Regression with count vector, TFIDF, word to vector, and bi and tri gram.

- ***With Count Vector:*** for am, there were 1340 true positives (predicted as am) and 242 false positives (predicted as en). For en, there were 4024 true positives (predicted as en), 9 false negatives (predicted as am), and 2 false positives (predicted as ne). For univ, there were 440 false negatives (predicted as en). For ne, there were 43 true positives (predicted as ne) and 161 false negatives (predicted as en),
- ***With TFIDF:*** for am, there were 1335 true positives (predicted as am) and 247 false positives (predicted as en). For en, there were 4019 true positives (predicted as en), 14

false negatives (predicted as am), and 2 false positives (predicted as ne). For univ, there were 4 true positives (predicted as univ), 435 false negatives (predicted as en), and 1 false negative (predicted as am). For ne, there were 43 true positives (predicted as ne) and 161 false negatives (predicted as en).

- ***With Word to Vector:*** for am, there were 321 true positives (predicted as am) and 1261 false positives (predicted as en). For en, there were 4034 true positives (predicted as en) and 1 false negative (predicted as am). For univ, there were 314 true positives (predicted as univ) and 126 false negatives (predicted as en). For ne, there were 200 false negatives (predicted as en) and 4 false negatives (predicted as am).
- ***With Bi and Tri Gram:*** For am, there were 260 true positives (predicted as am) and 1322 false positives (predicted as en). For en, there were 4035 true positives (predicted as en). For univ, there were 440 false negatives (predicted as en). For ne, there were 204 false negatives (predicted as en).

6.1.2. Classification Report of Machine Learning Models

In classification report we depicted the result of high performing models below.

SVM classification with Count Vector

	precision	recall	f1-score	support
0	0.99	0.91	0.95	1798
1	0.94	0.99	0.96	2342
2	0.61	1.00	0.76	31
3	0.98	0.99	0.99	390
accuracy			0.96	4561
macro avg	0.88	0.97	0.91	4561
weighted avg	0.96	0.96	0.96	4561

SVM classification with TFIDF

	precision	recall	f1-score	support
0	0.99	0.91	0.95	1801
1	0.94	0.99	0.96	2339
2	0.61	1.00	0.76	31
3	0.98	0.99	0.99	390
accuracy			0.96	4561
macro avg	0.88	0.97	0.91	4561
weighted avg	0.96	0.96	0.96	4561

SVM classification with Word to Vector

	precision	recall	f1-score	support
0	0.98	0.90	0.94	1743
1	0.93	0.98	0.95	2394
2	0.57	0.93	0.71	29
3	0.99	1.00	0.99	395
accuracy			0.95	4561
macro avg	0.87	0.95	0.90	4561
weighted avg	0.95	0.95	0.95	4561

SVM classification with bi and tri Gram

	precision	recall	f1-score	support
0	0.22	0.99	0.36	367
1	1.00	0.59	0.74	4189
2	0.10	1.00	0.18	5
3	0.00	0.00	0.00	0
accuracy			0.62	4561
macro avg	0.33	0.64	0.32	4561
weighted avg	0.94	0.62	0.71	4561

Figure 6.5. Classification Report of SVM

As shown in Figure 6.5, classification report of SVM, we have four classes Amharic, English, named entity, and universal, the report indicates every class precision, recall, and f1-score, and in the code-mixed Amharic-English dataset SVM archives the highest F1 score for most language labels in count vector, TFIDF and word to vector. but in bigram and trigram, the F1 score value drops. This indicates that bigram and trigram are not suitable for SVM for our dataset. SVM achieves an accuracy of 96%,96%,95%, and 62% with count vector, TFIDF, word to vector, and bi and tri gram, respectively.

Decission Tree classification with Count Vector

	precision	recall	f1-score	support
am	0.99	0.84	0.91	1582
en	0.99	0.86	0.92	4035
ne	0.97	0.38	0.55	204
univ	0.33	0.99	0.49	440
accuracy			0.85	6261
macro avg	0.82	0.77	0.72	6261
weighted avg	0.94	0.85	0.88	6261

Decission Tree classification with TFIDF Vector

	precision	recall	f1-score	support
am	0.99	0.84	0.91	1582
en	0.99	0.90	0.94	4035
ne	0.96	0.38	0.55	204
univ	0.38	0.99	0.55	440
accuracy			0.88	6261
macro avg	0.83	0.78	0.74	6261
weighted avg	0.94	0.88	0.89	6261

Decission Tree classification with word to Vector

	precision	recall	f1-score	support
am	0.99	0.85	0.91	1582
en	0.92	0.99	0.95	4035
ne	0.95	0.40	0.56	204
univ	0.99	0.99	0.99	440
accuracy			0.94	6261
macro avg	0.96	0.81	0.85	6261
weighted avg	0.94	0.94	0.93	6261

Decission Tree classification with bi and tri gram

	precision	recall	f1-score	support
am	1.00	0.20	0.33	1582
en	0.68	1.00	0.81	4035
ne	0.00	0.00	0.00	204
univ	0.00	0.00	0.00	440
accuracy			0.69	6261
macro avg	0.42	0.30	0.28	6261
weighted avg	0.69	0.69	0.60	6261

Figure 6.6. Classification Report of Decision Tree

As shown in Figure 6.6, classification report of Decision Tree, we have four classes Amharic, English, named entity, and universal, the report indicates every class precision, recall, and f1-score and also in the code-mixed Amharic and English dataset Decision Tree archives the highest F1 score for most language labels in count vector, TFIDF and word to vector. but in bigram and trigram, the F1 score value drops. This indicates that bigram and trigram are not suitable for Decision Tree for our dataset. Decision Tree achieves an accuracy of

85%,88%,94%, and 69% with count vector, TFIDF, word to vector, and bi and tri gram, respectively.

Random Forest classification with Count Vector

	precision	recall	f1-score	support
am	0.99	0.85	0.91	1582
en	0.99	0.86	0.92	4035
ne	0.98	0.39	0.55	204
univ	0.33	0.99	0.49	440
accuracy			0.85	6261
macro avg	0.82	0.77	0.72	6261
weighted avg	0.94	0.85	0.88	6261

Random Forest classification with TFIDF

	precision	recall	f1-score	support
am	0.99	0.85	0.91	1582
en	0.99	0.86	0.92	4035
ne	0.98	0.39	0.55	204
univ	0.33	0.99	0.49	440
accuracy			0.85	6261
macro avg	0.82	0.77	0.72	6261
weighted avg	0.94	0.85	0.88	6261

Random Forest classification with Word to Vector

	precision	recall	f1-score	support
am	0.99	0.82	0.90	1582
en	0.91	1.00	0.95	4035
ne	0.96	0.40	0.57	204
univ	1.00	0.98	0.99	440
accuracy			0.93	6261
macro avg	0.97	0.80	0.85	6261
weighted avg	0.94	0.93	0.93	6261

Random Forest classification with bi and tri gram

	precision	recall	f1-score	support
am	1.00	0.20	0.33	1582
en	0.68	1.00	0.81	4035
ne	0.00	0.00	0.00	204
univ	0.00	0.00	0.00	440
accuracy			0.69	6261
macro avg	0.42	0.30	0.28	6261
weighted avg	0.69	0.69	0.60	6261

Figure 6.7. Classification Report of Random Forest

As shown in Figure 6.7, classification report of Random Forest, we have four classes Amharic, English, named entity, and universal, the report indicates every class precision, recall, and f1-score and also in the code-mixed Amharic and English dataset Random Forest archives better F1 score for most language labels in count vector, TFIDF and word to vector. but in bigram and trigram decrease the F1 score. This indicates that bigram and trigram are not suitable for Random Forest for our dataset. Random Forest achieves an accuracy of 85%,85%,93%, and 69% with count vector, TFIDF, word to vector and bi and tri gram, respectively.

Logistic Regression classification with Count Vector

	precision	recall	f1-score	support
am	0.99	0.85	0.91	1582
en	0.83	1.00	0.90	4035
ne	0.96	0.21	0.35	204
univ	0.00	0.00	0.00	440
accuracy			0.86	6261
macro avg	0.69	0.51	0.54	6261
weighted avg	0.81	0.86	0.82	6261

Logistic Regression classification with TFIDF Vector

	precision	recall	f1-score	support
am	0.99	0.84	0.91	1582
en	0.83	1.00	0.90	4035
ne	0.96	0.21	0.35	204
univ	1.00	0.01	0.02	440
accuracy			0.86	6261
macro avg	0.94	0.51	0.54	6261
weighted avg	0.88	0.86	0.82	6261

Logistic Regression classification with Word to Vector

	precision	recall	f1-score	support
am	0.98	0.20	0.34	1582
en	0.68	1.00	0.81	4035
ne	0.00	0.00	0.00	204
univ	0.00	0.00	0.00	440
accuracy			0.70	6261
macro avg	0.42	0.30	0.29	6261
weighted avg	0.69	0.70	0.61	6261

Logistic Regression classification with bi and tri gram

	precision	recall	f1-score	support
am	1.00	0.16	0.28	1582
en	0.67	1.00	0.80	4035
ne	0.00	0.00	0.00	204
univ	0.00	0.00	0.00	440
accuracy			0.69	6261
macro avg	0.42	0.29	0.27	6261
weighted avg	0.69	0.69	0.59	6261

Figure 6.8. Classification Report of Logistic Regression

As shown in Figure 6.8, classification report of Logistic Regression, we have four classes Amharic, English, named entity, and universal, the report indicates every class precision, recall, and f1-score and also in the code-mixed Amharic and English dataset logistic regression archives has better F1 score for most language labels in count vector and TFIDF but when using word to vector and bigram and trigram decrease the F1 score for Amharic and English labels on the other hand logistic regression to perform named entity and universal labels in this to feature extraction techniques and also logistic regression achieves 86%,86%,70%, and 69% with count vector, TFIDF, word to vector, and bi and tri gram respectively.

6.1.3. Result of Deep Learning Models

LSTM Model Summary Result shown below.

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (None, 3, 300)            1638600

 lstm_3 (LSTM)               (None, 128)               219648

 dropout_5 (Dropout)         (None, 128)               0

 dense_5 (Dense)             (None, 4)                 516

=================================================================
Total params: 1,858,764
Trainable params: 220,164
Non-trainable params: 1,638,600
_________________________________________________________________
```

Figure 6.9. LSTM Model Summary

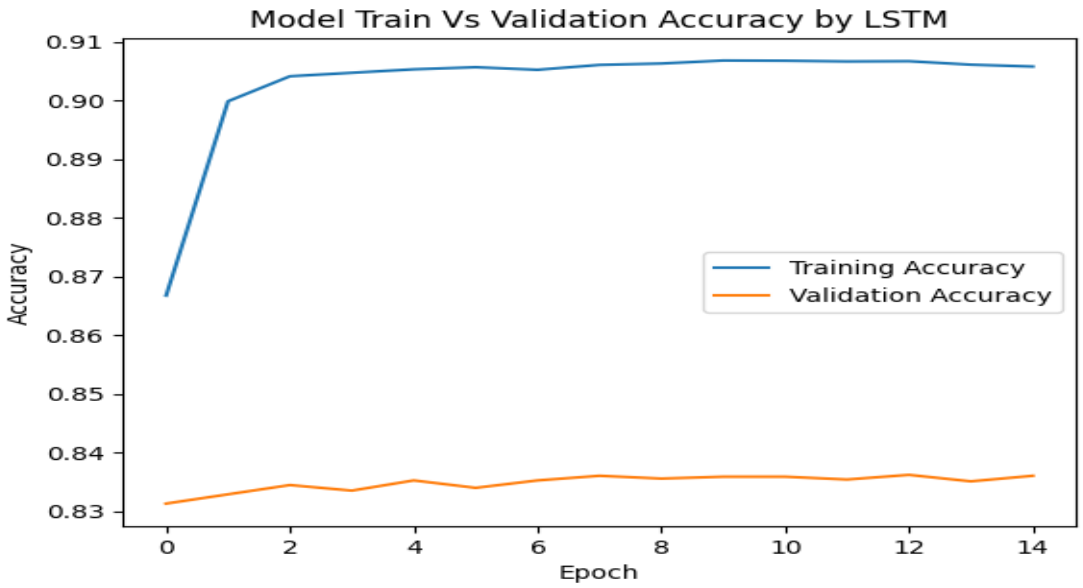


Figure 6.10. LSTM Model Training Vs Validation Accuracy

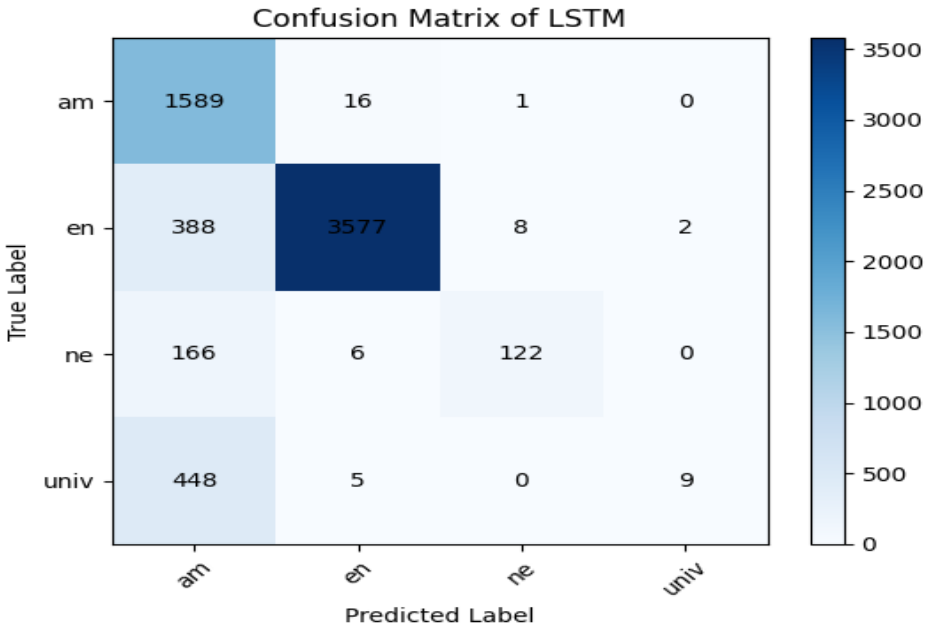


Figure 6.11. Confusion Matrix of LSTM

As shown in Figure 6.11, indicate the confusion matrix of Long Short-Term Memory. for am, there were 1589 true positives (predicted as am), 16 false positives (predicted as en), and 1 false positive (predicted as ne). For en, there were 3577 true positives (predicted as en), 388 false negatives (predicted as am), 8 false positives (predicted as ne), and 2 false positives (predicted as univ). For ne, there were 122 true positives (predicted as ne), 6 false negatives (predicted as en), and 166 false negatives (predicted as am). For univ, there were 9 true positives (predicted as univ), 5 false negatives (predicted as en), and 448 false negatives (predicted as am).

Classification Report of Long Short Term Memory

	precision	recall	f1-score	support
am	0.61	0.99	0.76	1606
en	0.99	0.90	0.94	3975
ne	0.94	0.40	0.56	294
univ	0.85	0.02	0.05	462
accuracy			0.84	6337
macro avg	0.85	0.58	0.58	6337
weighted avg	0.88	0.84	0.81	6337

Figure 6.12. Classification Report of LSTM

As shown in Figure 6.12, classification report of LSTM, we have four classes Amharic, English, named entity, and universal, the report indicates every class precision, recall, and f1-score, and LSTM achieves an accuracy of 84%.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 100)          546200

 conv1d (Conv1D)             (None, 96, 164)           82164

 global_max_pooling1d (Globa  (None, 164)              0
 lMaxPooling1D)

 dense (Dense)               (None, 100)               16500

 dense_1 (Dense)             (None, 4)                 404

=================================================================
Total params: 645,268
Trainable params: 645,268
Non-trainable params: 0
```

Figure 6.13. CNN Model Summary

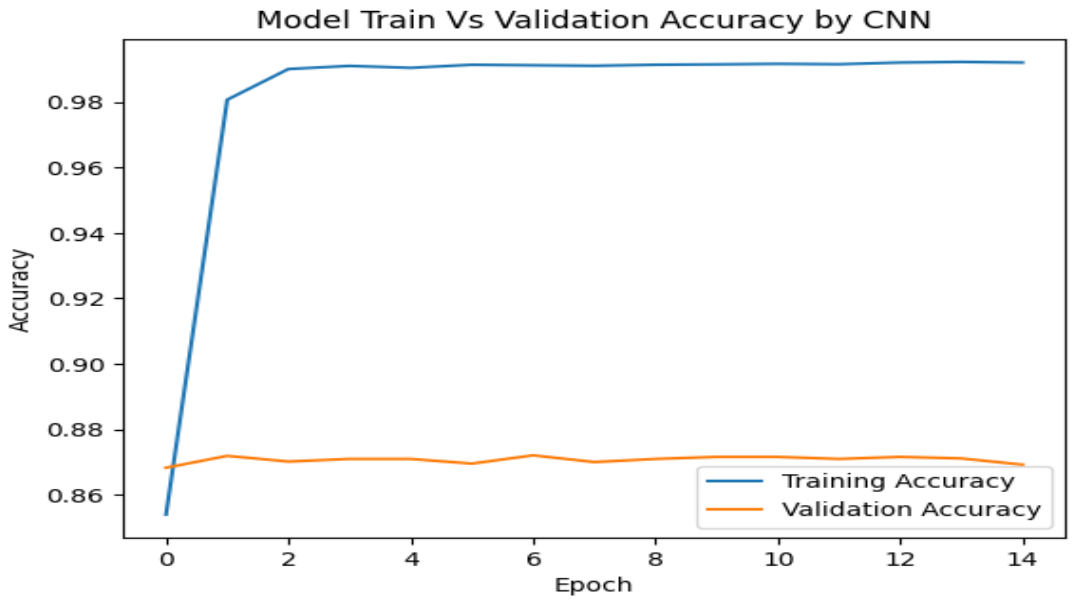


Figure 6.14. CNN Model Training Vs Validation Accuracy

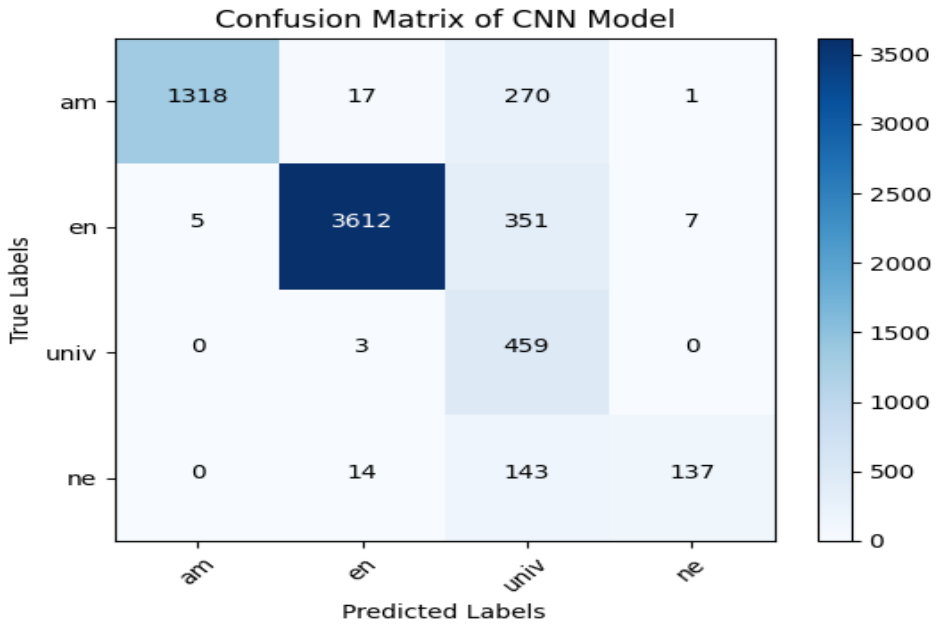


Figure 6.15. Confusion Matrix of CNN

As shown in Figure 6.15, indicate the confusion matrix of CNN. for am, there were 1318 true positives (predicted as am), 17 false positives (predicted as en), 270 false positives (predicted as univ) ¸ and 1 false positive (predicted as ne). For en, there were 3612 true positives (predicted as en), 5 false negatives (predicted as am), 7 false positives (predicted as ne), and 351 false positives (predicted as univ). For ne, there were 137 true positives (predicted as ne), 14 false negatives (predicted as en), and 143 false negatives (predicted as univ). For univ, there were 459 true positives (predicted as univ) and 3 false negatives (predicted as en)..

Classification Report of Convolutional Neural Network

	precision	recall	f1-score	support
am	0.99	0.82	0.90	1606
en	0.99	0.91	0.95	3975
ne	0.95	0.46	0.62	294
univ	0.38	1.00	0.55	462
accuracy			0.87	6337
macro avg	0.83	0.80	0.75	6337
weighted avg	0.94	0.87	0.89	6337

Figure 6.16. Classification Report of CNN

As shown in Figure 6.16, classification report of CNN, for each classes Amharic, English, named entity, and universal, represent precision, recall, and f1-score, and CNN gained an accuracy of 87%.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 100)          546200

 flatten (Flatten)           (None, 10000)             0

 dense (Dense)               (None, 128)               1280128

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 4)                 516

=================================================================
Total params: 1,826,844
Trainable params: 1,826,844
Non-trainable params: 0
_________________________________________________________________
```

Figure 6.17. MLP Model Summary

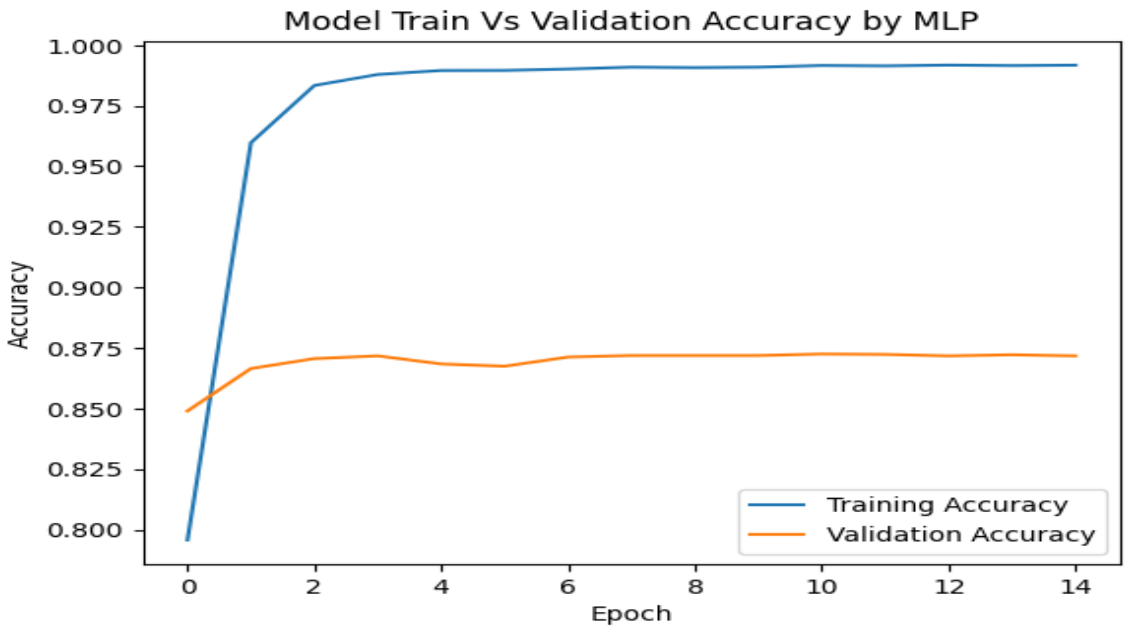


Figure 6.18. MLP Model Training Vs Validation Accuracy

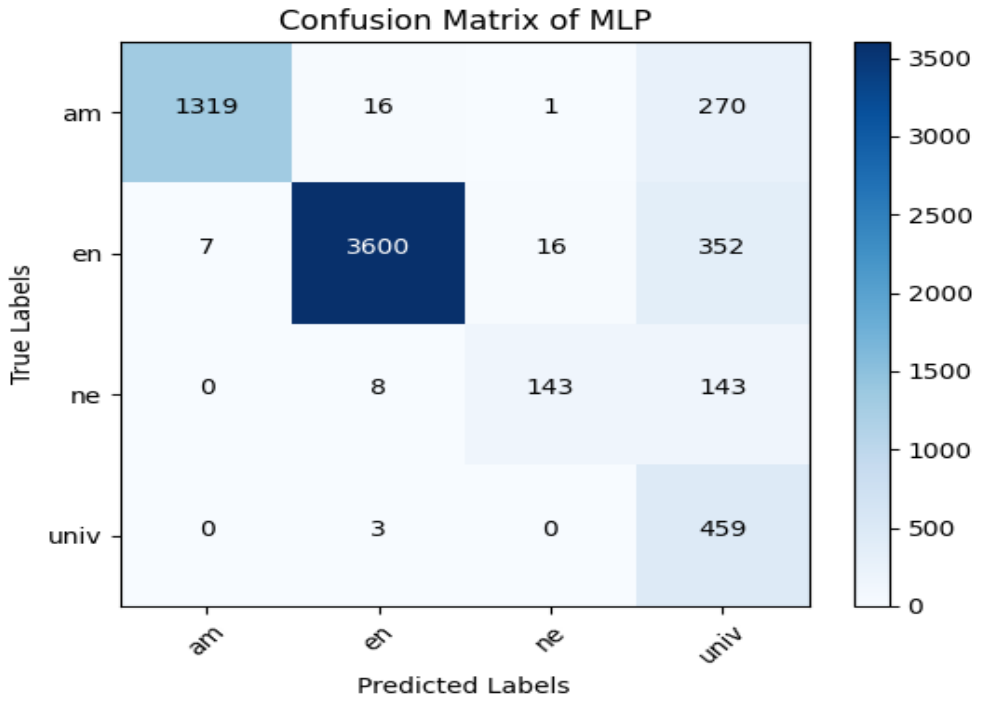


Figure 6.19. Confusion Matrix of MLP

As shown in Figure 6.19, indicate the confusion matrix of MLP. for am, there were 1319 true positives (predicted as am), 16 false positives (predicted as en), and 270 false positives (predicted as univ), and 1 false positive (predicted as ne). For en, there were 3600 true positives (predicted as en), 7 false negatives (predicted as am), 16 false positives (predicted as ne), and 352 false positives (predicted as univ). For ne, there were 143 true positives (predicted as ne), 8 false negatives (predicted as en), and 143 false positives (predicted as univ). For univ, there were 459 true positives (predicted as univ), and 3 false negatives (predicted as en).

Classification Report of Multi Layer Perceptron

	precision	recall	f1-score	support
am	1.00	0.82	0.90	1606
en	0.99	0.91	0.95	3975
ne	0.93	0.48	0.64	294
univ	0.38	1.00	0.55	462
accuracy			0.87	6337
macro avg	0.82	0.80	0.76	6337
weighted avg	0.95	0.87	0.89	6337

Figure 6.20. Classification Report of MLP

As shown in Figure 6.20, classification report of MLP, for Amharic, English, named entity, and universal class represent the precision, recall, and f1-score values, and MLP gained an accuracy of 87%.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 100)          546200

 bidirectional (Bidirectiona  (None, 256)              234496
 l)

 dropout (Dropout)           (None, 256)               0

 dense (Dense)               (None, 4)                 1028

=================================================================
Total params: 781,724
Trainable params: 781,724
Non-trainable params: 0
```

Figure 6.21. Bi-LSTM Model Summary

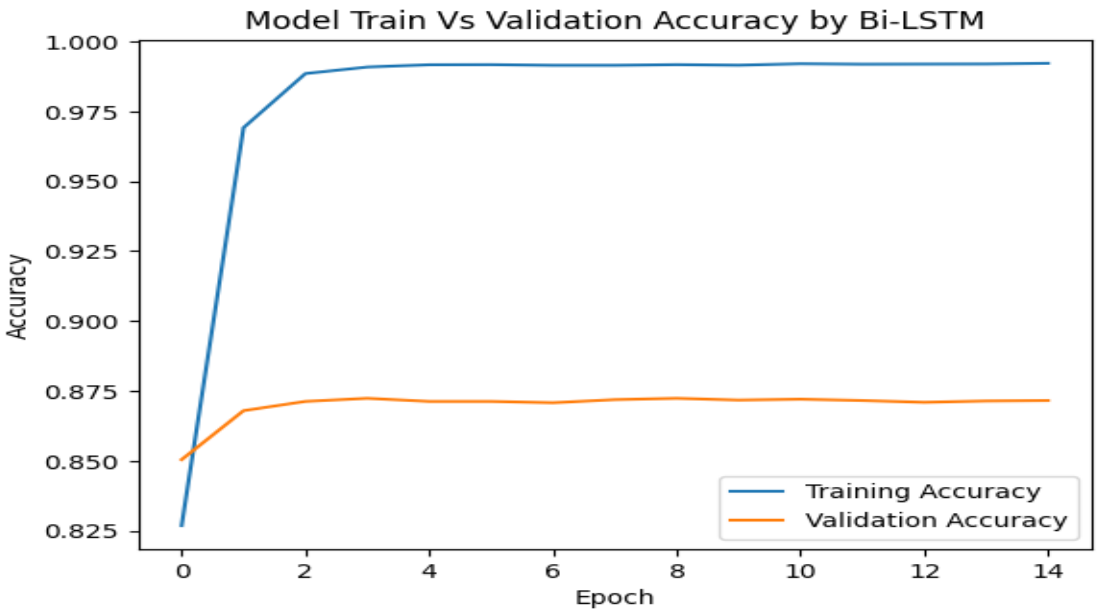


Figure 6.22. Bi-LSTM Model Training Vs Validation Accuracy

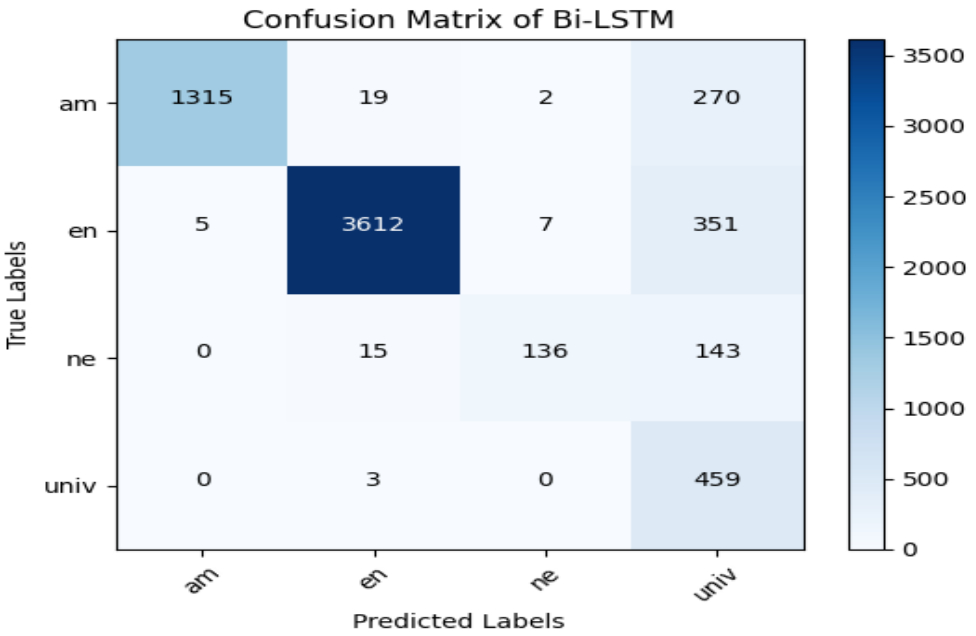


Figure 6.23. Confusion Matrix of Bi-LSTM

As shown in Figure 6.23, indicate the confusion matrix of Bidirectional Long Short-Term Memory. for am, there were 1315 true positives (predicted as am), 19 false positives (predicted as en), 2 false positives (predicted as ne) and 270 false positives (predicted as univ). For en, there were 3612 true positives (predicted as en), 5 false negatives (predicted as am), 7 false positives (predicted as ne), and 351 false positives (predicted as univ). For ne, there were 136 true positives (predicted as ne), 15 false negatives (predicted as en), and 143 false positives (predicted as univ). For univ, there were 459 true positives (predicted as univ) and 3 false negatives (predicted as en).

Classification Report of Bidirectional Long Short Term Memory

	precision	recall	f1-score	support
am	1.00	0.82	0.90	1606
en	0.99	0.91	0.95	3975
ne	0.92	0.48	0.63	294
univ	0.38	0.99	0.54	462
accuracy			0.87	6337
macro avg	0.82	0.80	0.76	6337
weighted avg	0.94	0.87	0.89	6337

Figure 6.24. Classification Report of Bi-LSTM

As shown in Figure 6.24, classification report of Bi-LSTM, for Amharic, English, named entity, and universal class depicted the precision, recall, and f1-score values, and Bi-LSTM gained an accuracy of 87%.

Table 6.1. Model Summary with Count Vector

Language Name	***Label Tag***	***Logistic Regression***	***Naïve Bayes***	***Decision Tree***	***SVM***	***RF***	***XGBoost***	***MaxEnt***
		F1- Score						
Amharic	Am	0.91	0.92	0.91	0.94	0.91	0.58	0.00
English	En	0.90	0.90	0.92	0.96	0.92	0.84	0.78
Named Entity	Ne	0.35	0.13	0.55	0.83	0.55	0.19	0.00
Universal	univ	0.00	0.00	0.49	0.99	0.49	0.00	0.49
Weighted Average		0.82	0.82	0.88	0.96	0.88	0.69	0.53
Accuracy		86%	86%	85%	96%	85%	75%	63%

As shown in Table 6.1, Model Summary with Count Vector represented the F1-Score and Accuracy for each language label using different machine learning algorithms with the Count Vector feature extraction method. The Weighted Average row provides an overall weighted average of the F1-Scores across all language labels, and the Accuracy row shows the accuracy percentage for each algorithm. From this table, we analyze how each algorithm performs on different language labels and the overall performance of each algorithm using Count Vector features. SVM, Decision Tree, and Random Forest have high F1-Scores and Accuracy, while MaxEnt has lower F1-Scores and Accuracy.

As shown in Table 6.2, Model Summary with TFIDF Vector represented the F1-Score and accuracy for each language label using different machine learning algorithms with the TFIDF Vector feature extraction method. The Weighted Average row provides an overall weighted average of the F1-Scores across all language labels, and the Accuracy row shows the accuracy percentage for each algorithm. From this table, we analyze how each algorithm performs on different language labels and the overall performance of each algorithm using TFIDF Vector

features. SVM and Decision Tree seem to perform well across most language labels, while MaxEnt has lower F1-Scores and accuracy.

Table 6.2. Model Summary with TFIDF Vector

Language Name	***Label Tag***	***Logistic Regression***	***Naïve Bayes***	***Decision Tree***	***SVM***	***RF***	***XGBoost***	***MaxEnt***
		F1- Score						
Amharic	Am	0.91	0.91	0.91	0.94	0.91	0.61	0.00
English	En	0.90	0.90	0.94	0.96	0.92	0.84	0.82
Named Entity	Ne	0.35	0.15	0.55	0.84	0.55	0.19	0.00
Universal	univ	0.02	0.01	0.55	0.99	0.49	0.00	0.45
Weighted Average		0.82	0.82	0.89	0.96	0.88	0.70	0.56
Accuracy		86%	86%	88%	96%	85%	76%	65%

As shown in Table 6.3 below, Model Summary with Word2Vector represented the F1-Score and accuracy for each language label using different machine learning algorithms with the Word to Vector feature extraction method. The Weighted Average row provides an overall weighted average of the F1-Scores across all language labels, and the Accuracy row shows the accuracy percentage for each algorithm. From this table, we analyze how each algorithm performs on different language labels and the overall performance of each algorithm using Word to Vector features. Decision Tree, SVM, Random Forest, and XGBoost. Maximum Entropy seems to perform well across most language labels, while naïve bayes has lower F1-Scores and accuracy.

As shown in Table 6.4, Model Summary with Bi and Tri Vector represented the F1-Score and accuracy for each language label using different machine learning algorithms with the Bi&Tri Gram feature extraction method. The Weighted Average row provides an overall weighted average of the F1-Scores across all language labels, and the Accuracy row shows the accuracy percentage for each algorithm. From this table, we analyze how each algorithm performs on different language labels and the overall performance of each algorithm using Bi&Tri Gram features. Decision Tree, Random Forest, and Logistic Regression seem to perform well across

Amharic and English language labels, while MaxEnt has lower F1-Scores and accuracy for most language labels.

Table 6.3. Model Summary with Word to Vector

Language Name	***Label Tag***	***Logistic Regression***	***Naïve Bayes***	***Decision Tree***	***SVM***	***RF***	***XGBoost***	***MaxEnt***
		F1- Score						
Amharic	Am	0.34	0.00	0.91	0.93	0.90	0.89	0.90
English	En	0.81	0.81	0.95	0.95	0.95	0.95	0.95
Named Entity	Ne	0.00	0.00	0.56	0.71	0.57	0.57	0.56
Universal	univ	0.00	0.83	0.99	0.99	0.99	0.99	0.54
Weighted Average		0.61	0.58	0.93	0.94	0.93	0.92	0.89
Accuracy		70%	69%	94%	94%	93%	93%	87%

The above table summaries different machine learning f1-score values and accuracy scores.

Table 6.4. Model Summary with Bi&Tri Gram

Language Name	***Label Tag***	***Logistic Regression***	***Naïve Bayes***	***Decision Tree***	***SVM***	***RF***	***XGBoost***	***MaxEnt***
		F1- Score						
Amharic	Am	0.28	0.20	0.33	0.36	0.33	0.24	0.00
English	En	0.80	0.80	0.81	0.74	0.81	0.80	0.01
Named Entity	Ne	0.00	0.00	0.00	0.19	0.00	0.03	0.00
Universal	univ	0.00	0.00	0.00	0.00	0.00	0.00	0.14
Weighted Average		0.59	0.56	0.60	0.71	0.60	0.58	0.02
Accuracy		69%	67%	69%	62%	69%	68%	7%

6.2. Discussions

The objective of this study is Amharic-English Code-mixed language identification using machine learning and deep learning approach. As far as we are aware, this represents the initial effort or the first attempt Amharic-English Code-mixed language identification from Latin Amharic Scripts, for our code-mixed Amharic-English dataset. This study result signifies for machine learning and deep learning application in Linguistics, to facilitate further natural language processing tasks, support for multilingual social media platforms, improved communication analysis, and enhanced social media user experience. The existing work not compare different feature extraction techniques, this helps to know which feature extraction technique is suitable for code-mixed text and some papers use ether of machine learning and deep learning models. But in our study, we used both machine learning and deep learning models and gained better accuracy. This study also contributes to introduce Amharic-English Code-Mixed corpus We trained different machine, apply different feature extraction techniques to know the best for machine learning and deep learning models, and build different machine learning models including (Naïve Bayes, Decision Tree, Logistic Regression, Conditional Random Field, Support Vector Machine, Random Forest, Maximum Entropy and XGBoost) and deep learning models including LSTM, CNN, MLP and Bi-LSTM. The overall machine algorithm accuracy comparison is shown in Figure 6.25 below.

As shown in the Figure 6.9. Logistic Regression and Naïve Bayes perform similarly across all feature extraction methods. They achieve relatively high accuracy scores with both Count Vector and TF-IDF Vector, slightly outperforming the other methods. However, they have lower accuracy with Word2Vec, and Bi&Tri-Gram. Decision Tree performs well with Word2Vec, Count Vector and TF-IDF Vector, achieving accuracy scores of 94%, 85%, and 88%, respectively. However, its accuracy drops to 69% with Bi&Tri-Gram. SVM achieves the highest accuracy scores of 94%, 96%, and 96% with Word2vec, Count Vector, and TF-IDF Vector, respectively. However, its accuracy significantly drops to 62% with Bi&Tri-Gram, indicating that this feature extraction method may not be suitable for SVM. Random Forest shows consistent performance with Count Vector and TF-IDF Vector, achieving accuracy score of 85%. It performs well with Word2Vec 93% and Bi&Tri-Gram 69% as well. XGBoost exhibits lower accuracy with Count Vector 75% but improves with TF-IDF Vector 76%. It

performs well with Word2Vec 93% and maintains reasonable accuracy with Bi&Tri-Gram 68%. Maximum Entropy has relatively low accuracy across all feature extraction methods, with the lowest score of 7% when using Bi&Tri-Gram. This algorithm might not be suitable for our dataset. In general, the selection of the feature extraction method greatly influences the performance of these algorithms. Count Vector and TF-IDF Vector work well with several algorithms, including Logistic Regression, Naïve Bayes, Decision Tree, and Random Forest. SVM achieves the highest accuracy with these methods but struggles with Bi&Tri-Gram. XGBoost also performs reasonably well across these feature extraction methods. Maximum Entropy consistently underperforms compared to the other algorithms. In addition to Random Forest, we implement a conditional random field, this model uses word to feature extraction techniques so gained an accuracy of 93% for our code-mixed dataset.

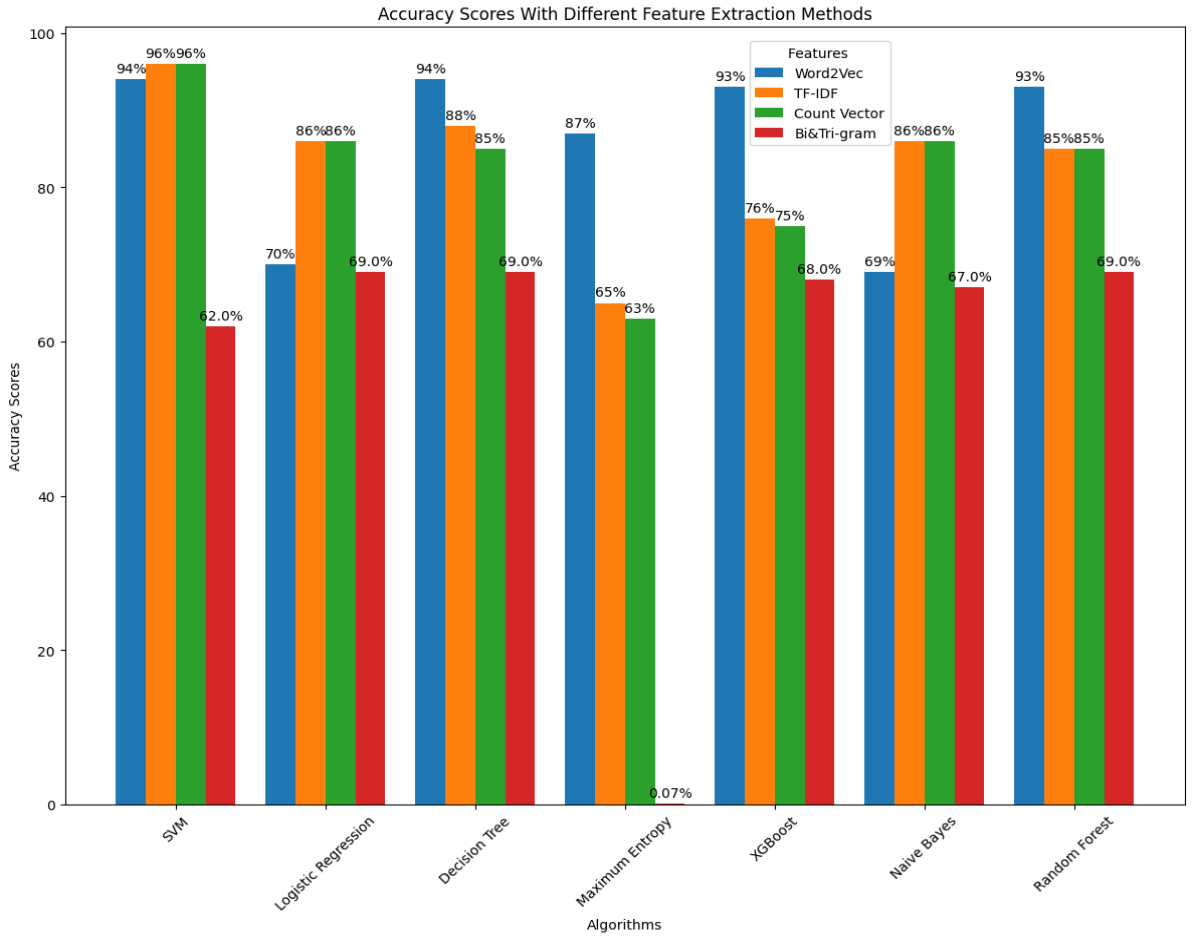


Figure 6.25. Models Accuracy Result Comparison Chart

As the experimental results are shown in Table 6.1, We compared each algorithm used in the study using weighted average values with the count vector feature extraction technique. From this comparison, we can see that the SVM algorithm achieved the highest weighted average accuracy of 0.96, making it the best-performing algorithm on the Count Vector feature extraction method. Decision Tree and Random Forest also performed well with weighted average accuracies of 0.88. In contrast, Maximum Entropy (MaxEnt) had the lowest weighted average accuracy of 0.53 among the algorithms considered.

As the experimental results are shown in Table 6.2, We compared each algorithm used in the study using weighted average values with the TFIDF vector feature extraction technique. From this comparison, we can see that the SVM algorithm achieved the highest weighted average accuracy of 0.96, making it the best-performing algorithm on the TF-IDF Vector feature extraction method. The decision Tree also performed well with a weighted average accuracy of 0.89. In contrast, both XGBoost and Maximum Entropy (MaxEnt) had the lowest weighted average accuracy of 0.70 and 0.56, respectively among the algorithms considered.

As the experimental results are shown in Table 6.3, We compared each algorithm used in the study using weighted average values with word to vector feature extraction technique. From this comparison, we can see that the SVM algorithm achieved the highest weighted average accuracy of 0.94, making it the best-performing algorithm on the Word2Vec feature extraction method. Decision Tree, Random Forest, XGBoost, and Maximum Entropy (MaxEnt) also performed well with high weighted average accuracies of 0.93, 0.93, 0.92, and 0.89 respectively. Logistic Regression and Naïve Bayes had a low weighted average accuracy of 0.61 and 0.58, respectively.

As the experimental results are shown in Table 6.4, We compared each algorithm used in the study using weighted average values with both bigram and trigram word feature extraction techniques. From this comparison, we can see that the SVM algorithm achieved the highest weighted average accuracy of 0.71, making it the best-performing algorithm on the Bi&Tri Gram feature extraction method. Naïve Bayes, XGBoost, Logistic Regression, Decision Tree, and Random Forest had moderate weighted average accuracies around 0.56 to 0.60. Maximum Entropy (MaxEnt) had a very low weighted average accuracy of 0.02.

In another way, we can see the F1_Score of each language label with different feature extraction techniques. From Table 6.1 result scores, these scores represent the F1-Score for each language label using different machine learning algorithms with the Count Vector feature extraction method. F1-Score is a measure of a model's accuracy, with higher values indicating better performance. From this table, you can see how each algorithm performs on different language labels. For example, SVM performs very well on all language labels, while MaxEnt has a low F1-Score for most labels. From Table 6.2 result scores, these scores represent the F1-Score for each language label using different machine learning algorithms with the TF-IDF Vector feature extraction method. From these results, we understand how each algorithm performs on different language labels and their overall performance using TF-IDF Vector features. For example, SVM and Decision Tree achieve high F1-Scores on most language labels, while MaxEnt and XGBoost have lower scores for some labels. From Table 6.3 result scores, these scores represent the F1-Score for each language label using different machine learning algorithms with the Word to Vector feature extraction method. From these results, we understand how each algorithm performs on different language labels and their overall performance using Word to Vector features. For example, the SVM, Random Forest, and Decision Tree algorithms achieve high F1-Scores across most language labels, while other algorithms like MaxEnt and Naïve Bayes have varying performance. From Table 6.4 result scores, these scores represent the F1-Score for each language label using different machine learning algorithms with the Bi&Tri Gram feature extraction method. From these results, we understand how each algorithm performs on different language labels and their overall performance using the Bi&Tri Gram features. For example, the Decision Tree and Random Forest algorithms perform well in English but less effectively in Amharic, while other algorithms have varying performance across different language labels.

Table 6.5. Deep Learning Models Summary

Language Name	***Label Tag***	***CNN***	***MLP***	***LSTM***	***Bi-LSTM***
		F1-Score Values			
Amharic	Am	0.90	0.90	0.76	0.90
English	En	0.95	0.95	0.94	0.95
Named Entity	Ne	0.62	0.64	0.56	0.63
Universal	univ	0.55	0.55	0.05	0.54
Weighted Average		0.89	0.89	0.81	0.89
Accuracy		87%	87%	84%	87%

As shown in Table 6.5, Model Summary of deep learning Models, when we compare four deep learning models in terms of F1-score for each language labels multi-layer perceptron models suitable for this task it has better f1-score for each language label than others model. Next to MLP Bi-LSTM and CNN has better F1-score and LSTM is Lower f1-score specially for universal language label. In terms of Accuracy CNN, MLP and Bi-LSTM has the same accuracy for our dataset and LSTM is lower accuracy than others.

In general, to answer the research question one Support Vector Machine is Best fit for our Amharic-English Code-mixed language identification with higher accuracy of 96% and higher f1-score value for every language label. For research question two TFID feature extraction technique is suitable for different machine learning models, by this feature extraction technique most machine learning models achieve higher accuracy. For research question three Multi-Layer Perceptron is best for each language label in terms of f1-score. Next to MLP Bi-LSTM is achieved good f1-score.

CHAPTER SEVEN

7. CONCLUSION AND FUTURE WORK

7.1. Conclusion

The goal of this study is identification of code-mixed Amharic-English language by using machine learning and deep learning techniques. As far as we are aware, no prior research has been conducted in the context of identifying languages in code-mixed Amharic-English social media text written in the Latin script. Code-mixing, in this context, refers to the practice of intermingling one language with another, on the social media people mix Amharic and English languages with the same script or Latin in a conversation. In this study we used Facebook, Tik Tok, Telegram and YouTube comments in the data collection phase. And includes data preprocessing, data annotation, feature extraction, model building, training, testing and deployment phases.

In preprocessing phase, we filtered, cleaned the data and tokenize a sentence in to words using NLTK library. NLTK also helps to tag English POS. The next phase is data annotation, we included four language tag such as Amharic, English, Universal and Named Entity, and annotated Amharic as am, English as en, universal as univ and Named Entity as ne. The next phase is featuring extraction for this phase we used count vector, TF-IDF vector, word to vector, bi and tri gram techniques. After feature extraction phase the next phase is model building and training, in this phase we build and trained different machine learning and deep learning models including (support vector machine, naïve bayes, conditional random field, decision tree, logistic regression, random forest, maximum entropy and XGBoost) and deep learning Long Short-Term Memory, Bidirectional Long Short-Term Memory, Convolutional Neural Network, and Multi-Layer Perceptron. The other phase is testing or evaluation phase, we evaluate our model's using accuracy, precession, recall, and F1-score using the testing data. For code mixed Amharic-English language identification word to vector, TF-IDF vector, count vector feature extraction techniques well fit for the most applied models. SVM model performs highest Accuracy of 96%,96% and 94% with TF-IDF vector, count vector and word to vector

respectively, Decision tree also achieves accuracy of 94%,88% and 85% with word to vector, TF-IDF vector and count vector respectively, Random Forest perform well 93%,85% and 85% with word to vector, TF-IDF vector and count vector respectively and Logistic Regression achieves accuracy of 70%,86% and 86% with word to vector, TF-IDF vector and count vector respectively. And deep learning models LSTM gained 84% accuracy and Bi-LSTM, CNN and MLP achieves the same accuracy of 87% for our dataset. Finally, we deployed the best performing models with Flask web interface to make it user-friendly. Generally, we conclude from machine learning technique SVM is best for our code-mixed Amharic English language identification and from deep learning almost all models achieve comparable result but in terms of f1-score muli-layer perceptron has better values for all language label.

7.2. Future Work

For future work, the following tasks should be considered:

- In the realm of Amharic-English code-mixed language identification, fine-tuning BERT offers better results. We have not implemented this model due to computational resource and time. This model can grasp word meanings within context. There were word ambiguities in code-mixed dataset context capture helps resolve such issues. So, we recommended to implement BERT model for such problem.
- This study paper considers the identification of code-mixing language, we recommend working the language correction.
- This study paper was considering text code-mixing language identification, we recommended speech code mixed detection or video.
- This study paper was used only Amharic-English code-mixed language identification, we recommended extending for others language pair.

References

[1] P. v. Veena, M. A. Kumar, and K. P. Soman, “Character embedding for language identification in Hindi-English code-mixed social media text,” Computacion y Sistemas, vol. 22, no. 1, pp. 65–74, 2018, doi: 10.13053/CyS-22-1-2775.

[2] S. Shekhar, D. K. Sharma, and M. M. S. Beg, “Language identification framework in code-mixed social media text based on quantum LSTM - the word belongs to which language?” Modern Physics Letters B, vol. 34, no. 6, Feb. 2020, doi: 10.1142/S0217984920500864.

[3] E. E.H. and A. M. Mekonnen, “Patterns of Code-switching in the Amharic Media,” Macrolinguistics, vol. 10, no. 17, pp. 125–150, Dec. 2022, doi: 10.26478/ja2022.10.17.6.

[4] J. Wagner, J. Foster, U. Barman, and A. Das, “Code Mixing: A Challenge for Language Identification in the Language of Social Media”, doi: 10.13140/2.1.3385.6967.

[5] M. Zeeshan Ansari, M. Jazib Khan, M. M. Sufyan Beg, G. Wasim, and T. Ahmad, “Language Identification of Hindi-English tweets using code-mixed BERT.”

[6] E. Tekle and A. Mulu, “Acceptance Sentiment Analysis on Amharic Language-Based COVID-19 Discourse from Facebook social media comments,” 2022.

[7] A. Gasparetto, M. Marcuzzo, A. Zangari, and A. Albarelli, “Survey on Text Classification Algorithms: From Text to Predictions,” Information (Switzerland), vol. 13, no. 2, Feb. 2022, doi: 10.3390/info13020083.

[8] C. Sabty, I. Mesabah, Ö. Çetinoğlu, and S. Abdennadher, “Language Identification of Intra-Word Code-Switching for Arabic–English,” Array, vol. 12, p. 100104, Dec. 2021, doi: 10.1016/j.array.2021.100104.

[9] H. L. Shashirekha, F. Balouchzahi, M. D. Anusha, and G. Sidorov, “CoLI-Machine Learning Approaches for Code-mixed Language Identification at the Word Level in Kannada-English Texts,” Nov. 2022, [Online]. Available: http://arxiv.org/abs/2211.09847

[10] D. Patel and R. Parikh, “Language Identification and Translation of English and Gujarati code-mixed data,” in International Conference on Emerging Trends in Information Technology and Engineering, ic-ETITE 2020, Institute of Electrical and Electronics Engineers Inc., Feb. 2020. doi: 10.1109/ic-ETITE47903.2020.410.

[11] E. Uchoi and M. Kaur, “Language Identification of English and Punjabi Code-Mixing and Code-Switching Sentences,” Eur. Chem. Bull, vol. 2023, no. si6, pp. 4119–4123, doi: 10.48047/ecb/2023.12.si6.367.

[12] O. E. Ojo, A. Gelbukh, H. Calvo, A. Feldman, O. O. Adebanji, and J. Armenta-Segura, “Language Identification at the Word Level in Code-Mixed Texts Using Character Sequence and Word Embedding.”

[13] K. S. S. Varma, A. Chaluvadi, and R. Mamidi, “Corpus Creation and Language Identification in Low-Resource Code-Mixed Telugu-English Text,” in International Conference Recent Advances in Natural Language Processing, RANLP, Incoma Ltd, 2021, pp. 744–752. doi: 10.26615/978-954-452-072-4_085.

[14] A. L. Tonja, M. G. Yigezu, O. Kolesnikova, M. S. Tash, G. Sidorov, and A. Gelbuk, “Transformer-based Model for Word Level Language Identification in Code-mixed Kannada-English Texts,” Nov. 2022, [Online]. Available: http://arxiv.org/abs/2211.14459

[15] S. Shekhar, D. K. Sharma, and M. M. S. Beg, “Language identification framework in code-mixed social media text based on quantum LSTM - the word belongs to which language?,” Modern Physics Letters B, vol. 34, no. 6, Feb. 2020, doi: 10.1142/S0217984920500864.

[16] H. L. Shashirekha, F. Balouchzahi, M. D. Anusha, and G. Sidorov, “CoLI-Machine Learning Approaches for Code-mixed Language Identification at the Word Level in Kannada-English Texts,” Nov. 2022, [Online]. Available: http://arxiv.org/abs/2211.09847

[17] T. Aichner, M. Grünfelder, O. Maurer, and D. Jegeni, “Twenty-Five Years of Social Media: A Review of Social Media Applications and Definitions from 1994 to 2019,” Cyberpsychology, Behavior, and Social Networking, vol. 24, no. 4. Mary Ann Liebert Inc., pp. 215–222, Apr. 01, 2021. doi: 10.1089/cyber.2020.0134.

[18] “Introduction to Social Media | University Communications and Marketing.” Accessed:Feb.16,2023.[Online].Available:https://www.usf.edu/ucm/marketing/intro-social-media.aspx

[19] “The rise of social media - Our World in Data.” Accessed: Mar. 08, 2023. [Online]. Available: https://ourworldindata.org/rise-of-social-media

[20] “What Is Social Networking?” Accessed: Mar. 08, 2023. [Online]. Available: https://www.investopedia.com/terms/s/social-networking.asp

[21] “Deep Learning-Based Automatic Amharic-English Code-Switching Detection AndCorrectGirmaw,Anduale,”2022.[Online].Available:http://dspace.orghttp://ir.bdu.edu.et/handle/123456789/14391

[22] “How to Build Your Social Media Marketing Strategy | Sprout Social.” Accessed: Mar. 09, 2023. [Online]. Available: https://sproutsocial.com/insights/social-media-marketing-strategy/

[23] Y. K. Dwivedi et al., “Setting the future of digital and social media marketing research: Perspectives and research propositions,” Int J Inf Manage, vol. 59, Aug. 2021, doi: 10.1016/j.ijinfomgt.2020.102168.

[24] D. Khurana, A. Koli, K. Khatter, and S. Singh, “Natural language processing: state of the art, current trends and challenges,” Multimed Tools Appl, vol. 82, no. 3, pp. 3713–3744, Jan. 2023, doi: 10.1007/s11042-022-13428-4.

[25] D. Khurana, A. Koli, K. Khatter, and S. Singh, “Natural language processing: state of the art, current trends and challenges,” Multimed Tools Appl, vol. 82, no. 3, pp. 3713–3744, Jan. 2023, doi: 10.1007/s11042-022-13428-4.

[26] “What is Natural Language Understanding & How Does it Work? | Simplilearn.” Accessed:Mar.09,2023.[Online].Available:https://www.simplilearn.com/natural-language-understanding-article

[27] “NLP vs. NLU vs. NLG: the differences between three natural language processing concepts - Watson Blog.” Accessed: Mar. 09, 2023. [Online]. Available: https://www.ibm.com/blogs/watson/2020/11/nlp-vs-nlu-vs-nlg-the-differences-between-three-natural-language-processing-concepts/

[28] “10 NLP Techniques Every Data Scientist Should Know.” Accessed: Mar. 09, 2023. [Online]. Available: https://www.projectpro.io/article/10-nlp-techniques-every-data-scientist-should-know/415

[29] “10 Natural Language Processing Applications in 2023 - InData Labs.” Accessed:Mar.09,2023.[Online].Available:https://indatalabs.com/blog/applications-of-natural-language-processing-in-business

[30] “Amharic.” Accessed: Feb. 16, 2023. [Online]. Available: http://www.languagesgulper.com/eng/Amharic.html

[31] M. Abate and Y. Assabie, “LNAI 8686 - Development of Amharic Morphological Analyzer Using Memory-Based Learning.”

[32] “Deep Learning-Based Automatic Amharic-English Code-Switching Detection AndCorrectGirmaw,Andualem,”2022.[Online].Available:http://dspace.orghttp://ir.bdu.edu.et/handle/123456789/14391

[33] “Punctuation Marks in English – Rules and Examples.” Accessed: Feb. 20, 2023. [Online]. Available: https://www.really-learn-english.com/punctuation-marks.html

[34] K. Bali, J. Sharma, M. Choudhury, and Y. Vyas, “‘I am borrowing ya mixing ?’ An Analysis of English-Hindi Code Mixing in Facebook,” 2014.

[35] A. Dutta, “Word-level Language Identification Using Subword Embeddings for Code-mixed Bangla-English Social Media Data.” [Online]. Available: https://github.com/

[36] F. Fatkhu Rohmah, D. Ario Fajar, and I. Ayu Panuntun, “Javanese Code Mixing In Teaching English At Smk Muhammadiyah Doro In Academic Year 2020/2021,” 2023.

[37] K. Eunhee “Reasons and Motivations for Code-Mixing and Code-Switching.” Year 2006 Vol.4 No.1

[38] S. Gundapu and R. Mamidi, “Word Level Language Identification in English Telugu Code Mixed Data,” Oct. 2020, [Online]. Available: http://arxiv.org/abs/2010.04482

[39] M. Sari, A. Arifin, and R. Harida, “Code-Switching And Code-Mixing Used By Guest Star In Hotman Paris Show,” Journal of English Language Learning (JELL), vol. 5, no. 2, pp. 105–112.

[40] A. I. Wibowo, I. Yuniasih, and F. Nelfianti, “Analysis Of Types Code Switching And Code Mixing By The Sixth President Of 5(38%/,&&,1’21(6,$ ¶6 Speech At The National Of Independence Day,” 2017.

[41] B. Setiawan, “Code-Mixing vs Code-Switching: a Study of Grammatical Perspective Through Code-Switching Varieties,” KnE Social Sciences, Apr. 2023, doi: 10.18502/kss.v8i7.13235.

[42] A. Darwis and Mp. STAI DDI Pinrang, “Code-Switching and Code-Mixing In Elt,” vol. 5, no. 1, pp. 1–14, 2023.

[43] J. A. Sagala, A. Sihombing, and G. Rudianto, “The Sociolinguistic Analysis Of Code-Mixing Types In Livy Renata And Nirina Zubir’s Utterances In ‘Ts Media’ Podcast,” 2022.

[44] Z. A. Aziz, D. Achmad, and M. Fadlun, “What Types Of Codes Are Mixed In Indonesia?: An Investigation Of Code Mixing In A Magazine.”

[45] M. Sari, A. Arifin, and R. Harida, “Code-Switching And Code-Mixing Used By Guest Star In Hotman Paris Show,” Journal of English Language Learning (JELL), vol. 5, no. 2, pp. 105–112.

[46] S. Malmasi and M. Dras, “Chinese Native Language Identification,” Association for Computational Linguistics. [Online]. Available: http://comp.mq.edu.au/

[47] G.M.Misganew,“Semantic-Aware Amharic Text Classification Using Deep Learning Approach,” 2020.

[48] P. Cunningham and S. J. Delany, “k-Nearest Neighbour Classifiers: 2nd Edition (with Python examples),” Apr. 2020, doi: 10.1145/3459665.

[49] F. Peng and D. Schuurmans, “Combining Naive Bayes and n-Gram Language Models for Text Classification.”

[50] C. Kruengkrai, P. Srichaivattana, V. Sornlertlamvanich, and H. Isahara, “Language Identification Based on String Kernels.” [Online]. Available: http://cs.haifa.ac.iltshlomo/

[51] B. R. Chakravarthi et al., “DravidianCodeMix: sentiment analysis and offensive language identification dataset for Dravidian languages in code-mixed text,” Lang Resour Eval, vol. 56, no. 3, pp. 765–806, Sep. 2022, doi: 10.1007/s10579-022-09583-7.

[52] C. Kingsford and S. L. Salzberg, “What are decision trees?,” 2008. [Online]. Available: http://www.nature.com/naturebiotechnology

[53] “Logistic regression for binary classification with Core APIs | TensorFlow Core.”Accessed:Aug.28,2023.[Online].Available:https://www.tensorflow.org/guide/core/logistic_regression_core#logistic_regression

[54] “Machine Learning Tutorial: The Max Entropy Text Classifier.” Accessed: Aug. 28, 2023. [Online]. Available: https://blog.datumbox.com/machine-learning-tutorial-the-max-entropy-text-classifier/

[55] “What is XGBoost? An Introduction to XGBoost Algorithm in Machine Learning |Simplilearn.”Accessed:Aug.28,2023.[Online].Available:https://www.simplilearn.com/what-is-xgboost-algorithm-in-machine-learning-article#what_is_xgboost_algorithm

[56] R. van der Goot and Ö. Çetinoğlu, “Lexical Normalization for Code-switched Data and its Effect on POS-tagging,” Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.01175

[57] P. Niraula, J. Mateu, and S. Chaudhuri, “A Bayesian machine learning approach for spatio-temporal prediction of COVID-19 cases,” Stochastic Environmental Research and Risk Assessment, vol. 36, no. 8, pp. 2265–2283, Aug. 2022, doi: 10.1007/s00477-021-02168-w.

[58] “Promise of Deep Learning for Natural Language Processing - MachineLearningMastery.com.” Accessed: Feb. 25, 2023. [Online]. Available: https://machinelearningmastery.com/promise-deep-learning-natural-language-processing/

[59] S. Saumya, A. Kumar, and J. P. Singh, “Proceedings of the First Workshop on Speech and Language Technologies for Dravidian Languages, pages 36-45 Offensive language identification in Dravidian code mixed social media text,” 2021. [Online]. Available: https://fasttext.cc/

[60] I. Sutskever, J. Martens, and G. Hinton, “Generating Text with Recurrent Neural Networks,” 2011.

[61] Y. Ning, S. He, Z. Wu, C. Xing, and L. J. Zhang, “Review of deep learning based speech synthesis,” Applied Sciences (Switzerland), vol. 9, no. 19. MDPI AG, Oct. 01, 2019. doi: 10.3390/app9194050.

[62] “An Overview on Long Short Term Memory (LSTM) - Analytics Vidhya.” Accessed: Feb. 25, 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2022/03/an-overview-on-long-short-term-memory-lstm/

[63] S. Mandal and A. K. Singh, “Language Identification in Code-Mixed Data using Multichannel Neural Networks and Context Capture,” Aug. 2018, [Online]. Available: http://arxiv.org/abs/1808.07118

[64] D. A. Moreno and M. Jabreel, “Deep Learning Models for Paraphrases Identification.”

[65] E. Uchoi and M. Kaur, “Language Identification of English and Punjabi Code-Mixing and Code-Switching Sentences,” Eur. Chem. Bull, vol. 2023, no. si6, pp. 4119–4123, doi: 10.48047/ecb/2023.12.si6.367.

[66] O. E. Ojo, A. Gelbukh, H. Calvo, A. Feldman, O. O. Adebanji, and J. Armenta-Segura, “Language Identification at the Word Level in Code-Mixed Texts Using Character Sequence and Word Embedding.”

[67] S. Thara and P. Poornachandran, “Transformer Based Language Identification for Malayalam-English Code-Mixed Text,” IEEE Access, vol. 9, pp. 118837–118850, 2021, doi: 10.1109/ACCESS.2021.3104106.

[68] K. S. S. Varma, A. Chaluvadi, and R. Mamidi, “Corpus Creation and Language Identification in Low-Resource Code-Mixed Telugu-English Text,” in International Conference Recent Advances in Natural Language Processing, RANLP, Incoma Ltd, 2021, pp. 744–752. doi: 10.26615/978-954-452-072-4_085.

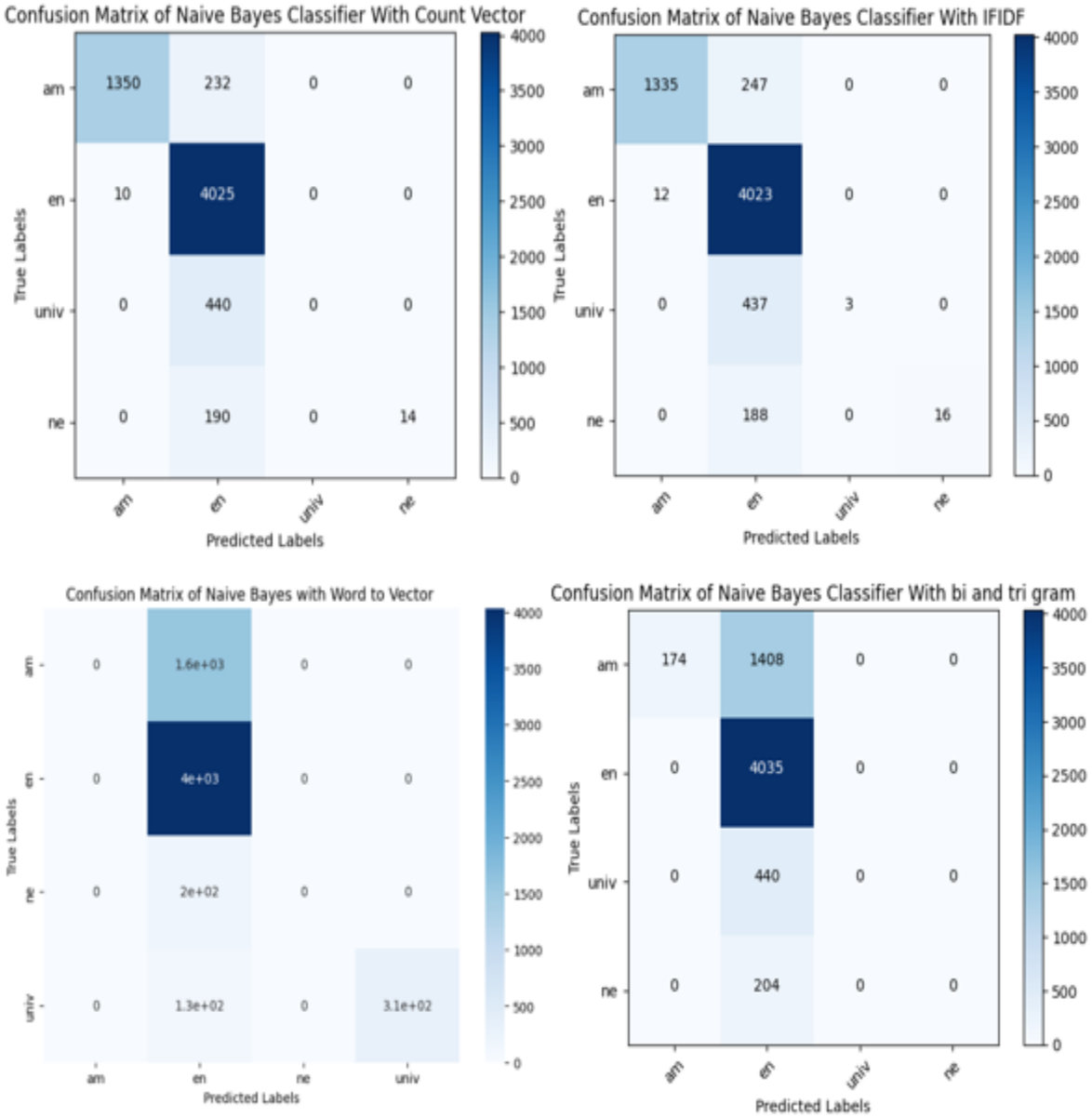
[69] S. Mandal and A. K. Singh, “Language Identification in Code-Mixed Data using Multichannel Neural Networks and Context Capture,” Aug. 2018, [Online]. Available: http://arxiv.org/abs/1808.07118

[70] N. Tarihoran, E. Fachriyah, Tressyalina, and I. R. Sumirat, “The Impact of Social Media on the Use of Code Mixing by Generation Z,” International Journal of Interactive Mobile Technologies, vol. 16, no. 7, pp. 54–69, 2022, doi: 10.3991/ijim.v16i07.27659.

APPENDICES

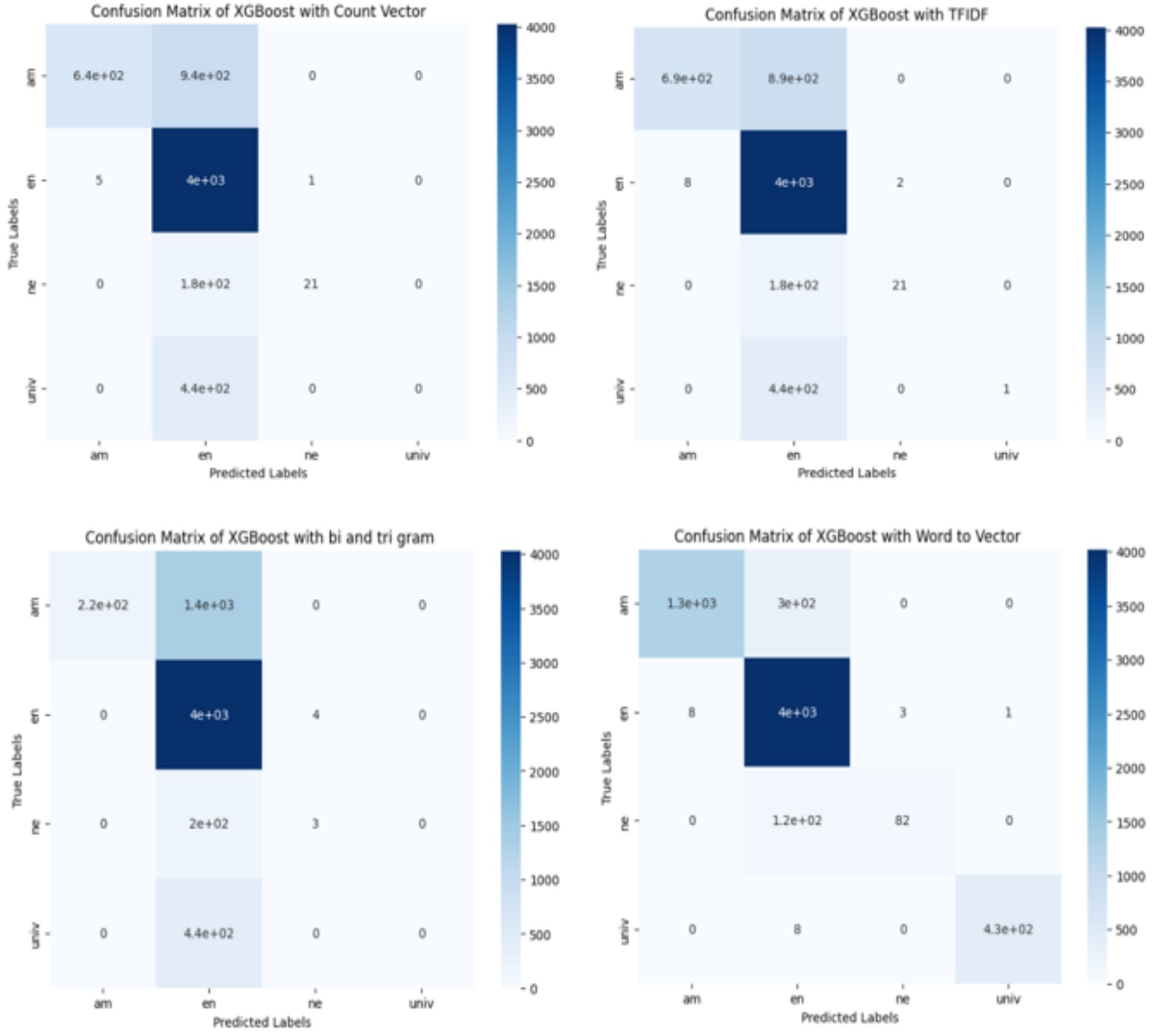
Appendix A. Confusion Matrix

Appendix A.1. Confusion matrix of Naïve Bayes Classifier with count vector, TFIDF, Word to Vector and bi and tri gram.



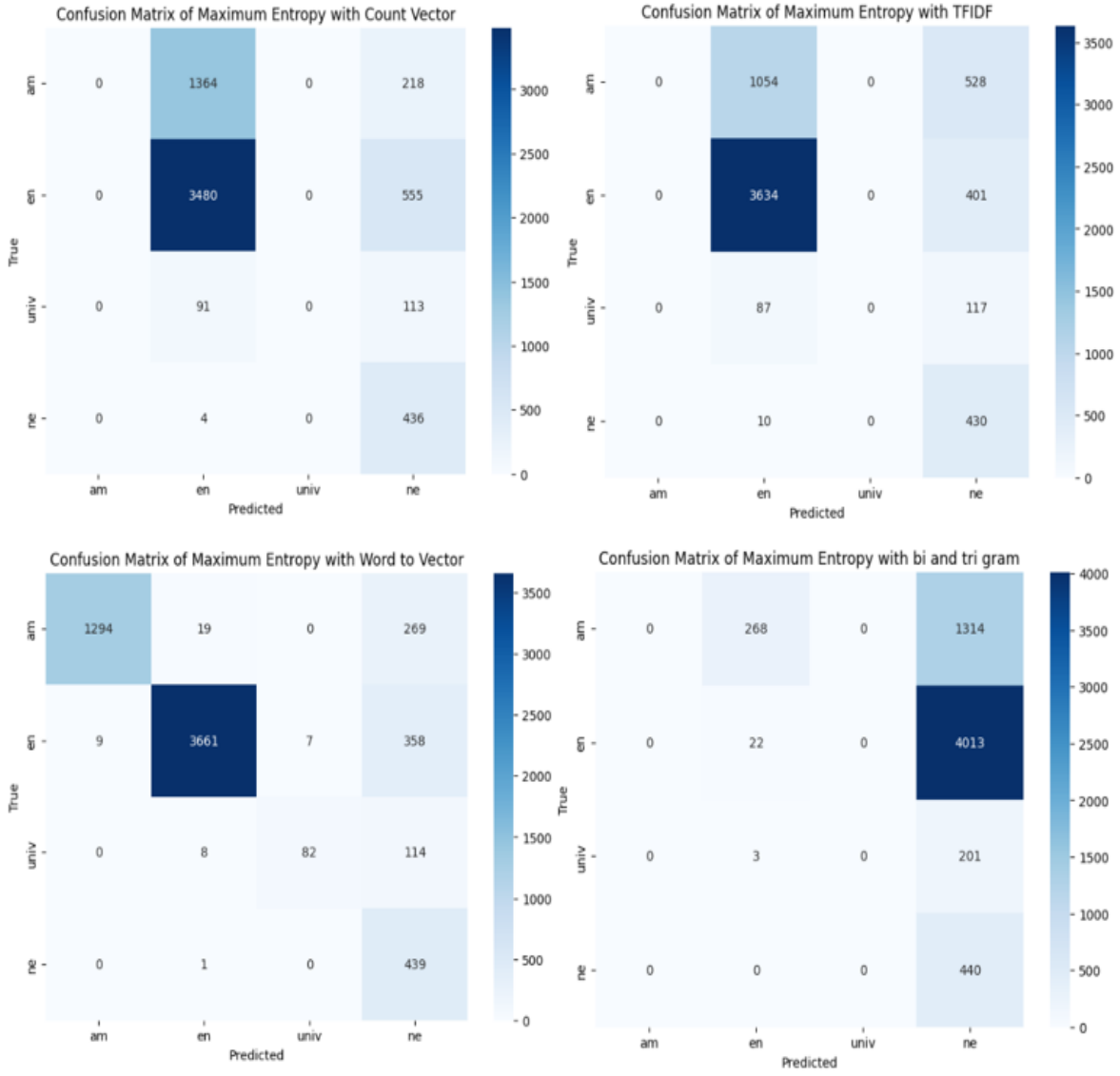
Confusion matrix of Naïve Bayes Classifier with count vector, TFIDF, Word to Vector and bi and tri gram

Appendix A.2. Confusion matrix of XGBoost Classifier with count vector, TFIDF, bi and tri gram, and Word to Vector.



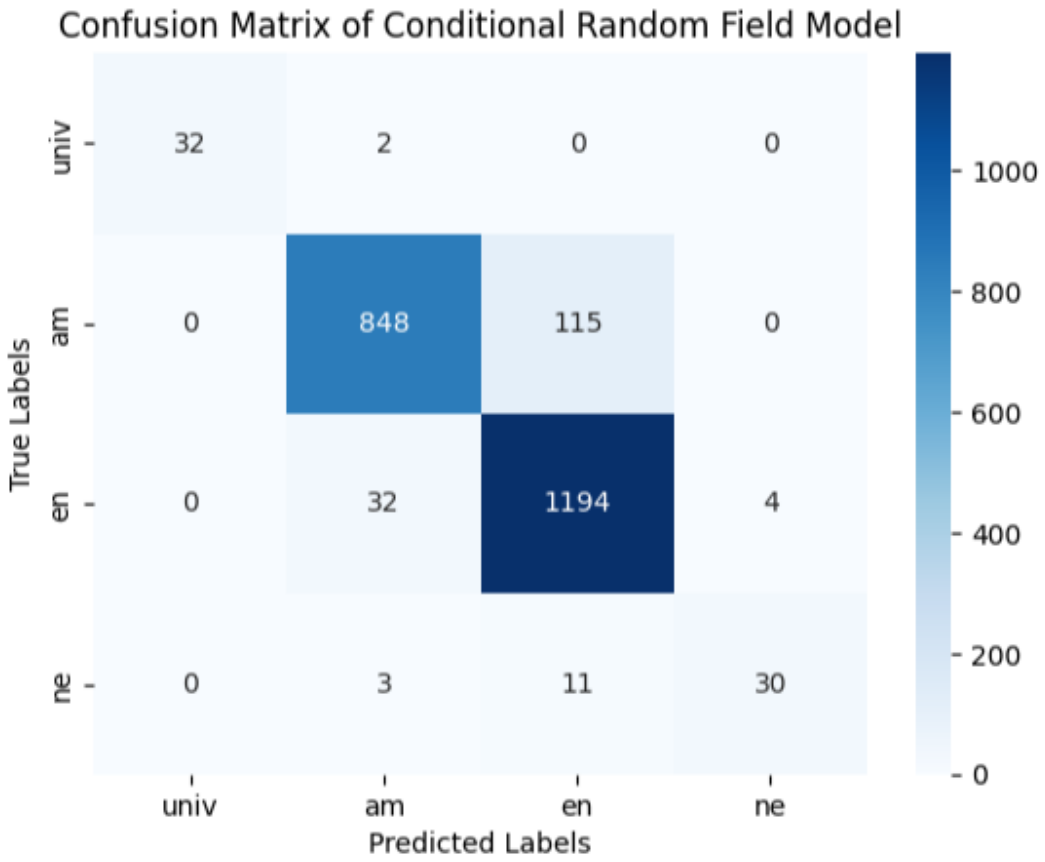
Confusion matrix of XGBoost Classifier with count vector, TFIDF, bi and tri gram, and Word to Vector.

Appendix A.3. Confusion matrix of Maximum Entropy Classifier with count vector, TFIDF, Word to Vector and bi and tri gram.



Confusion matrix of Maximum Entropy Classifier with count vector, TFIDF, Word to Vector and bi and tri gram.

Appendix A.4. Confusion matrix of Conditional Random Field Classifier



Confusion matrix of Conditional Random Field Classifier

Appendix B. Classification Report

Appendix B.1. Classification Report of Naïve Bayes Classifier with count vector, TFIDF, Word to Vector and bi and tri gram.

Naive Bays Classifier with Count Vector

	precision	recall	f1-score	support
am	0.99	0.85	0.92	1582
en	0.82	1.00	0.90	4035
ne	1.00	0.07	0.13	204
univ	0.00	0.00	0.00	440
accuracy			0.86	6261
macro avg	0.70	0.48	0.49	6261
weighted avg	0.81	0.86	0.82	6261

Naive Bays Classifier with TFIDF Vector

	precision	recall	f1-score	support
am	0.99	0.84	0.91	1582
en	0.82	1.00	0.90	4035
ne	1.00	0.08	0.15	204
univ	1.00	0.01	0.01	440
accuracy			0.86	6261
macro avg	0.95	0.48	0.49	6261
weighted avg	0.88	0.86	0.82	6261

Naive Bays Classifier with Word to Vector

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1582
1	0.68	1.00	0.81	4035
2	0.00	0.00	0.00	204
3	1.00	0.71	0.83	440
accuracy			0.69	6261
macro avg	0.42	0.43	0.41	6261
weighted avg	0.51	0.69	0.58	6261

Naive Bays Classifier with bi and tri gram

	precision	recall	f1-score	support
am	1.00	0.11	0.20	1582
en	0.66	1.00	0.80	4035
ne	0.00	0.00	0.00	204
univ	0.00	0.00	0.00	440
accuracy			0.67	6261
macro avg	0.42	0.28	0.25	6261
weighted avg	0.68	0.67	0.56	6261

Classification Report of Naïve Bayes Classifier with count vector, TFIDF, Word to Vector and bi and tri gram.

Appendix B.2. Classification Report of XGBoost Classifier with count vector, TFIDF, Word to Vector and bi and tri gram.

XGBoost Classifier with Count Vector

	precision	recall	f1-score	support
0	0.99	0.41	0.58	1582
1	0.72	1.00	0.84	4035
2	0.95	0.10	0.19	204
3	0.00	0.00	0.00	440
accuracy			0.75	6261
macro avg	0.67	0.38	0.40	6261
weighted avg	0.75	0.75	0.69	6261

XGBoost Classifier with TFIDF Vector

	precision	recall	f1-score	support
0	0.99	0.44	0.61	1582
1	0.73	1.00	0.84	4035
2	0.91	0.10	0.19	204
3	1.00	0.00	0.00	440
accuracy			0.76	6261
macro avg	0.91	0.39	0.41	6261
weighted avg	0.82	0.76	0.70	6261

XGBoost Classifier with Word to Vector

	precision	recall	f1-score	support
0	0.99	0.81	0.89	1582
1	0.90	1.00	0.95	4035
2	0.96	0.40	0.57	204
3	1.00	0.98	0.99	440
accuracy			0.93	6261
macro avg	0.96	0.80	0.85	6261
weighted avg	0.93	0.93	0.92	6261

XGBoost Classifier with bi and tri gram

	precision	recall	f1-score	support
0	1.00	0.14	0.24	1582
1	0.67	1.00	0.80	4035
2	0.43	0.01	0.03	204
3	0.00	0.00	0.00	440
accuracy			0.68	6261
macro avg	0.52	0.29	0.27	6261
weighted avg	0.70	0.68	0.58	6261

Classification Report of XGBoost Classifier with count vector, TFIDF, Word to Vector and bi and tri gram.

Appendix B.3. Classification Report of Maximum Entropy Classifier with count vector, TFIDF, Word to Vector and bi and tri gram.

Maximum Entropy Classifier with Count Vector

	precision	recall	f1-score	support
am	0.00	0.00	0.00	1582
en	0.70	0.86	0.78	4035
ne	0.00	0.00	0.00	204
univ	0.33	0.99	0.49	440
accuracy			0.63	6261
macro avg	0.26	0.46	0.32	6261
weighted avg	0.48	0.63	0.53	6261

Maximum Entropy Classifier with TFIDF

	precision	recall	f1-score	support
am	0.00	0.00	0.00	1582
en	0.76	0.90	0.82	4035
ne	0.00	0.00	0.00	204
univ	0.29	0.98	0.45	440
accuracy			0.65	6261
macro avg	0.26	0.47	0.32	6261
weighted avg	0.51	0.65	0.56	6261

Maximum Entropy Classifier with Word to Vector

	precision	recall	f1-score	support
am	0.99	0.82	0.90	1582
en	0.99	0.91	0.95	4035
ne	0.92	0.40	0.56	204
univ	0.37	1.00	0.54	440
accuracy			0.87	6261
macro avg	0.82	0.78	0.74	6261
weighted avg	0.95	0.87	0.89	6261

Maximum Entropy Classifier with bi and tr gram

	precision	recall	f1-score	support
am	0.00	0.00	0.00	1582
en	0.08	0.01	0.01	4035
ne	0.00	0.00	0.00	204
univ	0.07	1.00	0.14	440
accuracy			0.07	6261
macro avg	0.04	0.25	0.04	6261
weighted avg	0.05	0.07	0.02	6261

Classification Report of Maximum Entropy Classifier with count vector, TFIDF, Word to Vector and bi and tri gram.

Appendix B.4. Classification Report of Conditional Random Field Classifier

Classification Report for CRF

	precision	recall	f1-score	support
univ	1.00	0.94	0.97	34
am	0.96	0.88	0.92	963
en	0.90	0.97	0.94	1230
ne	0.88	0.68	0.77	44
accuracy			0.93	2271
macro avg	0.94	0.87	0.90	2271
weighted avg	0.93	0.93	0.93	2271

Classification Report of Conditional Random Field Classifier