




DEBRE BERHAN UNIVERSITY
COLLEGE OF COMPUTING
DEPARTMENT OF INFORMATION SYSTEMS

***TEXT-BASED LANGUAGE IDENTIFICATION FOR TYPOLOGICALLY
RELATED ETHIOPIAN LANGUAGES USING DEEP LEARNING***

BY
MIKRE GETU MIHRETE

DEBRE BERHAN, ETHIOPIA

JUNE 26, 2023



DEBRE BERHAN UNIVERSITY
COLLEGE OF COMPUTING
DEPARTMENT OF INFORMATION SYSTEMS

*TEXT-BASED LANGUAGE IDENTIFICATION FOR TYPOLOGICALLY
RELATED ETHIOPIAN LANGUAGES USING DEEP LEARNING*

BY
MIKRE GETU MIHRETE

A THESIS SUBMITTED TO THE DEPARTMENT OF INFORMATION
SYSTEMS OF DEBRE BERHAN UNIVERSITY IN PARTIAL FULFILMENT
OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN
INFORMATION SYSTEMS

ADVISOR
MICHAEL MELESE (Ph.D.)




DEBRE BERHAN, ETHIOPIA

JUNE 26, 2023

DEBRE BERHAN UNIVERSITY
COLLEGE OF COMPUTING
DEPARTMENT OF INFORMATION SYSTEMS


This is to certify that the thesis prepared by **Mikre Getu Mihrete**, titled: *Text-Based Language Identification for Typologically Related Ethiopian Languages Using Deep Learning* and submitted in partial fulfillment of the requirements for the degree of master of science in information systems complies with the regulations of the university and meets the accepted standards concerning originality and quality.

Name and signature of members of the examining board:

Name	Signature	Date
Advisor: <u>Michael Melese (Ph.D.)</u>	 _____	<u>June-15-2023</u>
Internal Examiner: <u>Kindie Biredagn (Ph.D.)</u>	 _____	<u>July-07-2023</u>
External Examiner: <u>Tibebe Beshah (Ph.D.)</u>	 _____	<u>July-07-2023</u>


DECLARATION

This thesis has not previously been accepted for any degree and is not being concurrently submitted in candidature for any degree in any university. I, the undersigned, declare that the thesis is the result of my own investigation unless otherwise stated. I conducted the study independently with the guidance and support of my research advisor. Other sources are acknowledged with citations with clear references.

Signature: _____ 

Mikre Getu, Debre Berhan University

This thesis has been submitted for examination with my approval as a university advisor.

Advisor's Signature: _____ 

Michael Melese (Ph.D.), Addis Ababa University

ACKNOWLEDGMENT

First and foremost, I would like to thank the almighty God, for letting me through all the difficulties. You are the one who let me finish my degree. I will keep on trusting you for my future.

I would like to express my special appreciation and thanks to my advisor, Dr. Michael Melese for his invaluable guidance and support throughout this research work. His expertise and encouragement have helped me to complete this research and write this thesis. I am also thankful for teaching me the vital lessons that helped me to work this research.

I would like to express my sincere gratitude to Dr. Kindie Biredagn and Dr. Million Meshesha for their encouragement and for their valuable suggestions during the proposal defense that helped me to follow the right direction in this research study. I am also thankful for teaching me the fundamental and advanced lessons that helped me to work this research. Thank you for spending your valuable time teaching me. I am also thankful to Alebachew Chiche (Assistant Professor) and Birhanu Ebisa (Doctoral Candidate) for their valuable suggestions and feedback during the defense of the research proposal.

Words cannot express my gratitude to my parents, my father Kes Getu Mihrete and my mother Mrs. Tesfaye G/Kidan. I believe that I live in their prayers. Your prayers for me were what sustained me this far. Their faith in me has kept my spirits and motivation high throughout this process. I am also thankful to my brothers and sisters for encouraging me to complete this research and for everything in my life.

Finally, I would like to thank my classmates and office mates, especially Ermiyas Abebe, for their moral support and for everything. Thanks should also go to the post graduate digital librarians of the Debre Berhan University for their services. I am grateful to everyone who has supported me throughout this process.

Mikre Getu Mihrete



June 26, 2023

ABSTRACT

Today we live in a world where there are more multilingual than monolingual. There is an ever-increasing amount of information on the world wide web that is written in different languages. Ethiopia is a multilingual country par excellence and multiple languages are used as media of administration, education and mass communications. But, these textual contents may not be expressed in a monolithic format. To use such textual resources for various purposes, language identification (LID) is an important preprocessing task for understanding, organizing and analyzing these contents. LID is the detection of the natural language of an input text. It is also the first necessary step to do any language-dependent natural language processing tasks. Although text-based LID has been extensively studied, there is still no comprehensive understanding of the factors that determine its identification accuracy. Factors such as the size of the text fragment to be identified, the amount and variety of training data available, the classification algorithm and the embedding techniques used. LID in very closely related languages is another unsolved problem. Current LID applications and models are unable to accurately identify the language for given text written in the Ge'ez script due to their similarity. The Ethiopic script is an alpha-syllabary or abugida “አቡጊዳ” writing system used for several languages spoken in Ethiopia and Eritrea.

In this work, we presented a LID model for six typologically and phylogenetically related low-resourced Ethiopian languages that use the Ge'ez script as their writing system; namely Amharic, Awngi, Ge'ez, Guragigna, Tigrigna and Xamtanga. The corpus used was collected automatically from various sources including Ethiopian mass media websites, social media, Bibles and related publications. We used the chars2vec embedding technique as a feature and DNN model for classification. To train and evaluate the proposed LID model, the researchers conducted several experiments with sample texts of different lengths using the best hyperparameter setting. Finally, the proposed LID model correctly identified the languages with an accuracy of more than 99% for texts longer than 50 characters and an accuracy of 77.68% for texts 5 characters long. The developed model also performed well for the out-of-vocabulary texts. In cases where languages are closely related and texts are very short, the identification performance of the proposed model was relatively poor. Therefore, it would be of interest to keep exploring LID models that handle closely related languages with short texts in the future.

Keywords: Amharic, Awngi, Bag-of-Characters, Closely Related Languages, Deep Neural Network, Ethiopic Script, Guragigna, Language Identification, Tigrigna, Xamtanga

TABLE OF CONTENTS

DECLARATION	I
ACKNOWLEDGMENT.....	II
ABSTRACT.....	III
LIST OF FIGURES	VII
LIST OF TABLES	VIII
LIST OF ALGORITHMS.....	IX
LIST OF ACRONYMS AND NOTATIONS.....	X
CHAPTER ONE	1
INTRODUCTION.....	1
1.1. Background.....	1
1.2. Motivation.....	3
1.3. Statement of the Problem.....	3
1.4. Objective of the Study	6
1.4.1. General Objective	6
1.4.2. Specific Objectives	6
1.5. Scope and Limitation of the Study.....	6
1.6. Significance of the Study	7
1.7. Organization of the Thesis	8
CHAPTER TWO	9
LITERATURE REVIEW	9
2.1. Overview.....	9
2.2. Natural Language.....	9
2.3. The Ethiopic Script	10
2.4. Typologically Related Ethiopian Languages	11
2.4.1. Amharic (አማርኛ) Language	11
2.4.2. Awnji (አውኒ) Language	12
2.4.3. Ge'ez (ግዕዝ) Language	13
2.4.4. Guragigna (ጉራጌኛ) Language	14
2.4.5. Tigrigna (ትግርኛ) Language	16
2.4.6. Xamtanga (ኧምባኛ) Language	17
2.5. Natural Language Processing	18
2.5.1. Text Classification	18
2.5.2. Language Identification	19

2.6. Language Identification Approaches	19
2.6.1. Embedding Techniques.....	20
2.6.2. Classification Techniques	25
2.6.3. Evaluation Techniques.....	37
2.7. Related Works.....	40
2.7.1. Summary of Related Works.....	43
2.8. Research Gaps.....	45
CHAPTER THREE	46
METHODOLOGY	46
3.1. Overview.....	46
3.2. The Proposed Research Design	46
3.3. The Proposed System Architecture.....	47
3.3.1. Data Collection	49
3.3.2. Text Preprocessing.....	50
3.3.3. Text Representation	54
3.3.4. Data Scaling	57
3.3.5. Dataset Splitting.....	59
3.3.6. The Proposed Deep Neural Network Model.....	61
3.3.7. Model Training	65
3.3.8. Model Evaluation.....	68
CHAPTER FOUR.....	69
EXPERIMENTAL RESULTS AND DISCUSSION	69
4.1. Overview.....	69
4.2. Experimental Setup.....	69
4.2.1. Hardware and Software Tools	69
4.2.2. Dataset.....	70
4.3. Experiments and Discussion of Results to Select Best Hyperparameter Set.....	70
4.3.1. Experiment One Using AdaGrad Optimizer.....	70
4.3.2. Experiment Two Using Adam Optimizer	72
4.3.3. Experiment Three Using RMSprop Optimizer	74
4.3.4. Summary of the Experiments.....	76
4.4. Model Evaluation Using Different Text Lengths	77
4.4.1. Experiment One Using Text Length of 5 Characters.....	78
4.4.2. Experiment Two Using Text Length of 10 Characters	80
4.4.3. Experiment Three Using Text Length of 50 Characters	82

4.4.4. Experiment Four Using Text Length of 100 Characters.....	84
4.4.5. Summary of the Experiments.....	86
4.5. Prototype and Predictions	87
4.5.1. Language Predictions on Out-of-Vocabulary Texts	91
4.6. Answering Research Questions	93
CHAPTER FIVE	94
CONCLUSION AND RECOMMENDATIONS	94
5.1. Overview.....	94
5.2. Conclusion	94
5.3. Contributions.....	96
5.4. Recommendations.....	96
5.5. Future works	97
REFERENCES.....	98
APPENDIXES	106
Appendix I: Closely Related Ethiopic and South Arabian Abjad Scripts	106
Appendix II: The Current (HaLeHaMe “ሀለሐመ”) Arrangement of the Ge'ez Alphabet	107
Appendix III: The Earlier (ABeGeDe “አበገደ”) Arrangement of the Ge'ez Alphabet.....	108
Appendix IV: The Ge'ez Numbers and Punctuation Marks	108
Appendix V: The Amharic Alpha-syllabic (Fidel)	109
Appendix VI: The Awngi Alpha-syllabic (Fidel).....	110
Appendix VII: The Guragigna Orthography (Fidel).....	111
Appendix VIII: The Tigrinya Alpha-syllabic (Fidel)	114
Appendix IX: The Xamtanga Alpha-syllabic (Fidel)	115
Appendix X: Sample Python Source Code.....	116

LIST OF FIGURES

Figure 1.1 Google language detector for Ethiopic-based texts	4
Figure 2. 1 The structure of CBOW and Skip-gram model [60]	24
Figure 2. 2 Structural similarity of biological and artificial neurons [82]	26
Figure 2. 3 A simple multilayer neural network	28
Figure 2. 4 Deep neural network with multiple number of hidden layers [86].....	29
Figure 2. 5 The architecture of a character-level CNN model [68]	31
Figure 2. 6 A standard RNN with a feedback connection inside the hidden layer	32
Figure 2. 7 Neural network with dropout and without dropout [97].....	37
Figure 3. 1 General architecture of the proposed language identification model	48
Figure 3. 2 Lexical similarity between the Geez-based languages at character and word-level....	54
Figure 3. 3 The reference of all unique alphanumeric and symbols	56
Figure 3. 4 The occurrences of symbols and the total number of input size for a given sample text ...	56
Figure 3. 5 Sample data before data scaling	58
Figure 3. 6 Sample data after applying data scaling technique.....	59
Figure 3. 7 Summary of the proposed deep neural network model architecture	61
Figure 3. 8 The proposed DNN model training process with backpropagation algorithm	66
Figure 4. 1 Learning curves of accuracy and loss over 10 epochs with the AdaGrad optimizer.....	71
Figure 4. 2 Learning curves of accuracy and loss over 15 epochs with the Adam optimizer.....	73
Figure 4. 3 Learning curves of accuracy and loss over 15 epochs with the RMSprop optimizer.....	75
Figure 4. 4 Confusion matrix of the model evaluated on sample text length 5 characters	78
Figure 4. 5 Classification report of the model evaluated with 5 chars of sample text length	79
Figure 4. 6 Confusion matrix of the model evaluated on sample text length 10 characters	81
Figure 4. 7 Classification report of the model evaluated with 10 chars of sample text length	82
Figure 4. 8 Confusion matrix of the model evaluated on sample text length 50 characters	83
Figure 4. 9 Classification report of the model evaluated with 50 chars of sample text length	84
Figure 4. 10 Confusion matrix of the model evaluated on sample text length 100 characters	85
Figure 4. 11 Classification report of the model evaluated with 100 chars of sample text length	86
Figure 4. 12 Accuracy and loss of the proposed model with different character length of texts	87
Figure 4. 13 Prediction result for the given short and long Amharic text.....	88
Figure 4. 14 Prediction result for the given short and long Awnji text.....	88
Figure 4. 15 Prediction result for the given short and long Geez text	89
Figure 4. 16 Prediction result for the given short and long Guragigna text.....	89
Figure 4. 17 Prediction result for the given short and long Tigrigna text.....	90
Figure 4. 18 Prediction result for the given short and long Xamtanga text	90
Figure 4. 19 Prediction results for the given Amharic Out-Of-Vocabulary texts.....	91
Figure 4. 20 Prediction results for the given Guragigna Out-Of-Vocabulary texts	92

LIST OF TABLES

Table 2. 1 Example of text snippets with ISO 639-3 code for six Geez-based languages	18
Table 2. 2 Confusion Matrix.....	37
Table 2. 3 Summary of related works	43
Table 3. 1 The corpus size of each language	51
Table 3. 2 Word and character distributions of each language in the corpus	53
Table 3. 3 The alphanumeric and special characters distribution of the six languages	53
Table 3. 4 Label encoding of each language.....	57
Table 3. 5 One-hot-encoding of each language	58
Table 3. 6 Distribution of sample text dataset	60
Table 3. 7 The distribution of dataset splitting	60
Table 4. 1 Hyperparameters for experiment one	70
Table 4. 2 A summary of experimental results using AdaGrad optimizer	72
Table 4. 3 Hyperparameters for experiment two	72
Table 4. 4 A summary of experimental results using Adam optimizer	74
Table 4. 5 Hyperparameters for experiment three	74
Table 4. 6 A summary of experimental results using RMSprop optimizer	76
Table 4. 7 The chosen hyperparameter settings.....	77
Table 4. 8 Test accuracy and loss results of the proposed model with variety of sample text lengths .	86

LIST OF ALGORITHMS

Algorithm 3. 1: Algorithm for data cleaning	51
Algorithm 3. 2: Algorithm for word tokenization	52
Algorithm 3. 3: Algorithm for character tokenization	52
Algorithm 3. 4: Algorithm for bag-of-characters model	55
Algorithm 3. 5: Algorithm for Adam optimizer	63
Algorithm 3. 6: Algorithm for AdaGrad optimizer	64
Algorithm 3. 7: Algorithm for RMSprop optimizer	65
Algorithm 3. 8: Backpropagation algorithm for training the proposed DNN model.....	67

LIST OF ACRONYMS AND NOTATIONS

AdaGrad: Adaptive Gradient	MB: Megabyte
Adam: Adaptive Moment Estimation	MLPs: Multilayer Perceptrons
ANN: Artificial Neural Network	NBC: Naïve Bayes Classifier
AMC: Amhara Media Corporation	NLP: Natural Language Processing
AMH: Amharic	NLTK: Natural Language Toolkit
AWN: Awngi	NN: Neural Network
BOC: Bag-of-Characters	OOV: Out-of-Vocabulary
BOW: Bag-of-Words	POS: Part-of-speech
CBOW: Continuous Bag of Words	ReLU: Rectified Linear Unit
CFA: Cumulative Frequency Addition	RNNs: Recurrent Neural Networks
Chars2vec: Characters to Vector	RMSprop: Root Mean Squared Propagation
CNN: Convolutional Neural Networks	RQ: Research Question
CPU: Central Processing Unit	SGW: Guragigna
Doc.: Document	SOV: Subject-Object-Verb
Exp.: Experiment	SVM: Support Vector Machine
FFNN: Feedforward Neural Networks	TB: Terabyte
GEZ: Geez	TF-IDF: Term Frequency-Inverse Document Frequency
GloVe: Global Vectors	TIR: Tigrigna
HTML: Hypertext Markup language	URLs: Uniform Resource Locators
ISO: International Organization for Standardization	VSO: Verb-Subject-Object
LID: Language Identification	Word2Vec: Word to Vectors
LSTM: Long Short-Term Memory	XAN: Xamtanga
MAX_LEN: Maximum Length of Characters	

CHAPTER ONE

INTRODUCTION

1.1. Background

Ethiopia is the homeland of a remarkable diversity of community, culture, and language [1]. The cultural and linguistic diversity is believed to result from a complex historical background, as well as geographic and social differences. Human language is a communication system that consists of speech, sign language, and written symbols used by people of a particular country or region to speak and write. Ethiopia is a multilingual country where more than 80 languages are spoken by different ethnic groups and up to 200 different dialects are spoken [1] [2]. Ethiopian languages are categorized into four main groups: Semitic, Cushitic, Omotic and Nilotic. Most of these languages belong to the Semitic, Cushitic, and Omotic groups, all part of the Afro-Asiatic language family, while a small number of languages belong to a fourth group Nilotic, which is part of the Nilo-Saharan language family [1] [2].

A language writing system, technically referred to as a script or an orthography, is an organized regular method of information storage and transfer for the communication of messages in a language by visually encoding and decoding with a set of visible symbols, forms or structures called characters [3]. Every language has its own writing system for visual representation of verbal communication, based on script and rules governing its use. Ethiopic script, also known as Ge'ez script, is an alpha-syllabary or abugida “አቡጊዳ” writing system in which each letter represents a consonant-vowel syllable, locally referred to as Fidel “ፊደል”. Ethiopic is the most commonly used script in the writing systems of the majority of languages spoken in Ethiopia and the neighboring country, Eritrea [4].

The Semitic languages use the Ge'ez script for their writing system. Some of the Ethio-Semitic languages spoken in Ethiopia include Adarigna, Amharic, Argoba, Birale, Gafat, Ge'ez, Guragigna, the Chaha group (Chaha, Muher, Ezha, Gumer and Gura), the Inor group (Inor, Enner, Endegegna, Gyeto and Mesemes), Silt'e group (Silt'e, Ulbareg, Enneqor and Walane), Soddo group (Soddo, Gogot and Galila) and Tigrigna [2].

Ethiopians speak a variety of languages from the Cushitic language families like Afaan Oromo, Afar, Awngi, Somali and Xamtanga. Most of the Cushitic languages use the Latin script in their writing system [1]. However, some languages are written in Ethiopic scripts, such as Gedeo, Awngi and Xamtanga. Basketo is one of the Omotic languages which use the Ethiopic script. In general, more than 15 well-known languages¹ use the Geez script as their writing system, such as Amharic, Awngi, Ge'ez, Guragigna, Tigrigna, Xamtanga and others [5].

Human language can be developed naturally or constructed on purpose, but in all cases the defining feature is that it is used to communicate between people through speaking, signing or writing. A subfield within artificial intelligence, computer science and linguistics called natural language processing (NLP) covers the area of how languages can be described, represented, used and constructed in a computational manner. NLP enables computers to process human language and understand meaning and context [6] [7].

Language identification (LID) is an application of NLP that automatically identifies the language in which the contents of the text are written [8] [9]. It is also called language detection or language guessing, it is a well-known research topic in NLP [9]. LID is the task of giving a language label to a text. LID is most commonly modeled as a supervised multi-class single-label, which is generally considered as a special case of text categorization [10] [11].

Language identification can be done by applying two main approaches: statistical approach and non-statistical approach [8] [12] [13]. Non-statistical approaches are basically linguistic approaches that require sufficient knowledge about the rules of the language used, while statistical approaches basically rely on machine learning and deep learning approaches which require less human effort and a large dataset for training and testing the language identification model [13].

Traditionally, the identification of written language was done manually by identifying frequent words and letters known to be characteristic of particular languages [8] [14]. Statistical approaches consist of the training phase and classification phase [13]. Recently, the progress of NLP research on text classification has arrived at the state-of-the-art [15]. It has achieved terrific results, showing deep learning methods as the cutting-edge technology to perform such tasks.

¹ *Languages written with the Ge'ez script:* Aari, Amharic, Argobba, Awngi, Basketo, Blin, Chaha, Dizin, Geddeo, Harari, Harari, Inor, Qimant, Sebat Bet Gurage, Silt'e, Tigre, Tigrinya, Xamtanga and others.
<https://www.ethnologue.com/browse/names>

A set of well-known methods for dealing with language data are using supervised machine learning algorithms, that attempt to infer usage patterns from a set of pre-defined input and output pairs [16]. Deep learning approaches are gaining popularity due to their superior accuracy, ability to represent the data with features extracted on their own, good solution for complex architectures in high-dimensional data, faster and easier interpretation of big data and transform it into meaningful information [15]. Therefore, we implemented this state-of-the-art technology for the proposed research work.

1.2. Motivation

Today we live in the world where there are more bilinguals than monolinguals, however multilingualism does not mean that we have extensive knowledge of multiple languages. There is an ever-increasing amount of information on the world wide web that is written in different languages. In Ethiopia, the number of people using information written in Ge'ez script in administrative, educational, social media, and mass media is growing [3]. Human beings will be interested in retrieving, analyzing, organizing or understanding information written in different languages. However, performing these tasks is challenging unless the users are language experts or one must know the language in which they are written or translate them into someone's mother language. Therefore, it is interesting to develop an automatic language identifier that automatically identifies the language in which the contents of the text are written that considers Ethiopian languages that use Ge'ez script as their writing system.

1.3. Statement of the Problem

The internet is an ever-expanding supply of textual data. A wealth of helpful textual data is available on the web in a multilingual environment like Ethiopia. Social media users, especially in multilingual societies, generate multilingual content in which at least two languages or language varieties are used. Information is usually provided in a context that excludes ambiguity. However, if web users do not recognize the language, it is difficult to understand the information written on the web. In this case, it would be helpful to know the name of the language.

Although textual based language identification problem has been extensively studied, there is still no comprehensive understanding of the factors that determine classification accuracy [4] [13] [15]. Factors such as the size of the text fragment to be identified, the amount and variety of training data available, the classification algorithm, and the embedding techniques used.

Another unsolved issue in language identification is identifying language categories for closely related languages as these languages use the same writing system [4] [15]. Languages written with the Ge'ez script are very closely related, like Amharic, Awngi, Ge'ez, Guragigna, Tigrigna and Xamtanga. Low-resourced languages and closely related languages are generally confused with each other in real-world applications like machine translation, impacting the user experience. Therefore, there is still a considerable room for improvement in terms of such factors.

Research in the area of language identification has grown steadily over the years. However, most researchers have concentrated attention on English and the other European languages for obvious reasons [17]. Natural language models are usually specific to discrete languages. Currently, various language detection applications have been developed for different languages, such as Google Language Detector² and Text Language Detector³, as well as pre-trained multilingual language detection models such as langdetect⁴, lingua-language-detector⁵, langid⁶ [18] and spacy-langdetect⁷. Although some applications can recognize the Amharic and Tigrigna languages, most of these pre-trained models and applications cannot correctly identify the language for the given text written in the Ethiopic script due to their similarity. This suggests that little attention is paid to the Ethiopian languages written in the Ethiopic script. In order to identify the language of the texts written in Ethiopic script, the researchers tested each of the selected languages ten times on Google language detector.

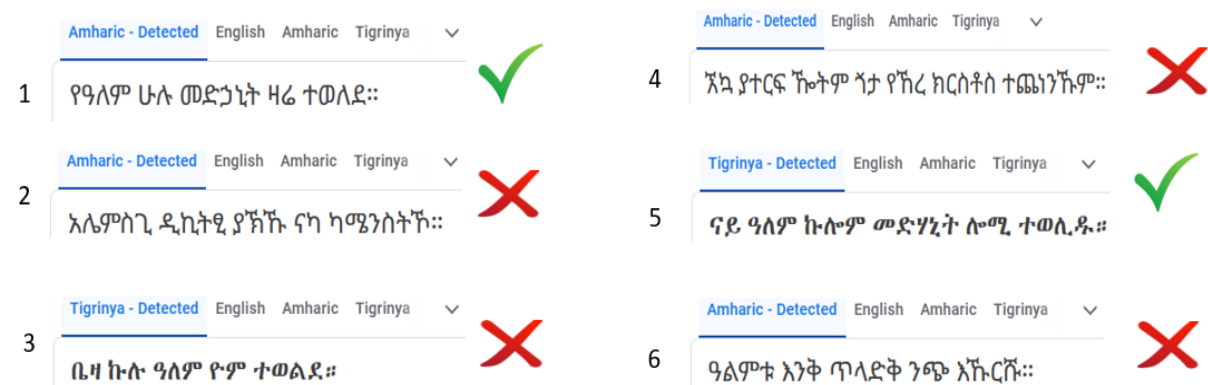


Figure 1.1 Google language detector for Ethiopic-based texts

² Google Language Detector: <https://translate.google.com/?sl=auto&tl=am&op=translate>
³ Text Language Detector: <https://www.dcode.fr/language-recognition>
⁴ langdetect: <https://pypi.org/project/langdetect/>
⁵ lingua-language-detector: <https://pypi.org/project/lingua-language-detector/>
⁶ Langid: <https://github.com/saffsd/langid.py>
⁷ Spacy-langdetect: <https://pypi.org/project/spacy-langdetect/>

For example, Figure 1.1 shows the language identification result of the Google language detector application for the given six Ethiopic-based texts. The Google language detector correctly identifies the given Amharic and Tigrigna texts, while as shown in the second screenshot the actual language of the text was Awngi but the Google language detector classified it as Amharic. The third, fourth, and sixth screenshots also show the incorrect classifications. Thus, it is one of the real problems of language identification in Geez-based languages due to their similarity. Therefore, there is a need to develop a language identification model for Ethiopian languages that use the Ge'ez script as the writing system.

The literature indicates that various approaches have been explored to address the language identification problems. To develop a LID model for different Ethiopian languages, most previous researchers Biruk Tadesse [19], Legesse Wedajo [20], Rediat Bekele [21], Fitsum Gaim [4] and Kidst Ergetie [22] have applied related statistical classification approaches and N-gram embedding techniques. However, the word-level N-gram model has limitations in handling out-of-vocabulary (OOV) words [23]. Intuitively, character N-grams should work better than word N-grams for LID, especially for languages with rich morphology [23]. However, regular morphological features like suffixes and prefixes repeat much more often than entire words, and can be representative of a language. They are more likely to be learned during the training and be indicators when predicting the language for a new unseen text.

To the best of our knowledge, there is no previous research work on LID using a deep learning approach with character-level embedding technique for Geez-based Ethiopian languages. However, currently, the progress of NLP research on LID has arrived state-of-the-art deep learning approach. Classical machine learning often requires researchers to write hand-crafted algorithms to extract and represent high-dimensional features from the raw data. Deep learning models, on the other hand, extract and manipulate these complex features automatically. Deep neural network approaches have been explored to remove the limitations due to the use of handcrafted features, and have achieved unprecedented success by achieving human-level performance when training with large datasets. Finally, the study aims to answer the research questions presented below:

- **RQ1:** To what extent does the proposed language identification model correctly identify Geez-based Ethiopian languages?
- **RQ2:** How robust is the language identifier when tested against a prototype for out-of-vocabulary texts?

1.4. Objective of the Study

In an attempt to develop the proposed language identification model, the general and specific objectives of the study formulated as follows.

1.4.1. General Objective

The general objective of the study is to design and develop an automated text-based language identification model for typologically related Ethiopian languages using a deep neural network.

1.4.2. Specific Objectives

In order to achieve the general objective of the study and to answer the research questions, the following specific objectives are formulated as follows:

- Explore the orthographic and phylogenetic characteristics of all selected languages.
- Collect monolingual corpus for each target language from different sources.
- Perform appropriate text preprocessing tasks.
- Design the general architecture of a deep neural network model for the proposed study.
- Design several hyperparameters and experimental setups.
- Conduct experiments with different hyperparameter settings and select the best set.
- Conduct experiments with sample texts of different character lengths using the selected hyperparameter set.
- Evaluate the performance of the developed LID model for OOV texts using a prototype.
- Forward recommendations for future research direction based on the findings.

1.5. Scope and Limitation of the Study

In a broad sense, language identification refers to all forms of language, including speech, gestures language, and written texts. The main focus of this study is to develop a textual-based language identification model for Geez-based Ethiopian languages using a deep learning approach.

The first step is to identify some reliable content sources for each language. There are more than 15 known languages⁸ that use the Geez script in their writing system [5], however many of them have small corpora and are not in digital content. Of these languages, Amharic (አማርኛ), Awngi (አዊጊ), Ge'ez (ግዕዝ), Guragigna (ጉራጊኛ), Tigrigna (ትግርኛ) and Xamtanga (ክምጣጎ) are available on digitization platforms. Therefore, we chose these six Ethiopic-based languages because of their easy access and sufficient data for training and testing. In other words, the study does not cover the issue of identifying Ethiopian languages that use non-Ethiopic scripts as their writing system.

Language identification aims to detect the language(s) given a text in one or more languages. In this study, we focused on multi-class, single-label classification problems. One of the limitations of this study, which could be addressed in future research, is that the model cannot correctly recognize the texts of code-switching languages. In this study, we trained and evaluated the proposed model on the sample text lengths of five and more characters. Because the corpus distribution analysis showed that the average word length of all languages is about 5 characters.

1.6. Significance of the Study

Language identification is an increasing field of importance in today's world, especially for multilingual eminent people like Ethiopia. Automatic LID becomes an important tool due to the increasing variety of textual data on the web that are written in different languages. Therefore, by using the LID application, users can easily understand these textual contents and finally apply them for the intended purpose. The ability to accurately identify the language in which a document is written is a fundamental technology that increases the accessibility of data and has a wide variety of applications. For example, it has been found that presenting information in a user's native language is a key factor in attracting website visitors.

⁸*Languages written with the Ge'ez script:* Aari, Amharic, Argobba, Awngi, Basketo, Blin, Chaha, Dizin, Geddeo, Harari, Harari, Inor, Qimant, Sebat Bet Gurage, Silt'e, Tigre, Tigrinya, Xamtanga and others.

<https://www.ethnologue.com/browse/names>

Solving automatic LID problem is necessary for many tools that work with multilingual data. Multilingual systems generally work on multiple languages because they use universal features of natural languages. LID is an important pre-processing task to develop multilingual NLP applications such as machine translation, sentiment analysis, spell and grammar checkers, spam filtering, plagiarism detection and information retrieval. For instance, in multilingual machine translation, the language in which the original text is written must be known in order to convert an unknown original language text into the desired target. Once it is detected, then using a machine translation system it can be translated into the required target language. In short, it is used to detect the language of the written text before machine translation.

Due to the diversity of information on the Internet, language detection is an important task for retrieving the right information. LID is a process present in many web services today. When searching the web, many websites have a language detection for the typed text in the search bar and the most relevant search results are then displayed first.

Spam filtering services that support multiple languages must identify the language that emails, online comments, and other input are written in before applying true spam filtering tasks. Without such detection, content originating from specific countries, regions, or areas suspected of generating spam may not be adequately eliminated from online platforms. Finally, this research work could serve as a basis for future research studies and future research in this area could benefit from the collected and prepared corpus.

1.7. Organization of the Thesis

The remaining section of this research is organized as follows: The second chapter provides a brief description of the Ethiopian languages that use the Ge'ez script as their writing system, language identification approaches, and related works. Chapter three covers the proposed research methodology. In the fourth chapter, the experimental results and discussions are described in detail. Finally, the paper ends with a general conclusion and recommendations.

CHAPTER TWO

LITERATURE REVIEW

2.1. Overview

This section extensively provides a conceptual discussion about natural language in general and specifically Ethiopic script writing system, Ethiopian languages that use Geez script as their writing system, natural language processing, and language identification approaches (classification, word embedding, and evaluation metrics). In addition to this, the conceptual discussion of deep learning besides the related work attempted for language identification for different languages.

2.2. Natural Language

Natural language is a human language that represents the universal means of communication and is constantly changing to meet the needs of users as individuals interacting with a changing world. Language is a set of rules or symbols in which symbols are combined to convey information [24]. Humans express themselves as members of a social group and as participants in a community using a system of conventional spoken, signed or written symbols.

Language Writing Systems: Written languages use visual symbols to represent the sounds of spoken languages. A language writing system, technically referred to as a script or an orthography, is an organized regular method of information storage and transfer for the communication of messages in a language by visually encoding and decoding with a set of visible symbols, forms or structures called characters [25]. Writing system can be divided into six categories namely Alphabetic, Abjads, Abugidas, Syllabic, Logographic (logo-syllabaries) and Featural writing system [26] [27].

Abugida Writing System: The name abugida (አቡጊዳ) is derived from the first four characters of an order of the Ge'ez script specifically, አ(a), ቡ(bu), ጊ(gi) and ዳ(da). An abugida, sometimes known as alpha-syllabary, syllabic alphabets or pseudo-alphabet, is a segmental writing system in which consonant-vowel sequences are written as units, each unit is based on a consonant letter and vowel notation is secondary. In abugidas writing system each character represents a consonant and vowel pairing [3]. For example, the current writing system of the

Ethiopic script used in Ethiopia and the Devanagari script used in South East Asia (modern times to write Hindi) is an alpha-syllabary writing system [24].

2.3. The Ethiopic Script

The script refers to the visual appearance of a writing system. Ethiopic script is known by various names: The Ge'ez (ግዕዝ) script, Abyssinian, Ethiopian or Amharic writing system. Ethiopic script is also called Fidel (ፊደል) or Saba scripts. Fidel is the local name of the writing system widely used in Ethiopia and Eritrea. The Ethiopic script has a long history, in the course of which it was modified in several ways. There are two pieces of historical literature on the origins of Ethiopic script.

The first acceptance, according to various researchers [24] states that the Ethiopic script derives from the South Arabic abjad or consonantal script, but soon morphed into an alpha-syllabary script likely inspired by Indic scripts which also included additional graphemes and numerals because of Greek influence. The original Ge'ez script was an Abjad writing system which means the script represented only consonant ones like, አ, ቢ, ገ, ደ, so forth. The ancient South Arabian script and Ethiopic abjad script are closely related as depicted in Appendix I. There are 24 correspondences of Ge'ez and the ancient South Arabian writing system [24].

According to the beliefs of the Ethiopian and Eritrean Orthodox Tewahido Church, the script was divinely revealed to Enos, grandson of the first man, Adam. The era of Henos witnessed the inception of the alphabet. Henos was a faithful and righteous servant of God. He was rewarded for his honest work through a divine gift of the alphabet, which would serve as his instrument for codifying the law. That is, the heavens opened their gates to him, and the scriptures were revealed to him. From then on he used the alphabet as a medium of literature. It has been in use since the fifth century BC and is currently used as a writing system for several local languages [28] [29].

The earlier Ge'ez script was written from A (አ) to P (ፑ) and the revised script still in Ethiopia starts with H (ዘ) and ends with P (ፑ). The current and former arrangement of the Ge'ez alphabet is described in Appendix II and Appendix II, respectively. The Ge'ez script is the only actively used native African writing system and one of the oldest in the world. The current Ethiopic script is classified as abugida writing system, used for several Afro-Asiatic and Nilo-Saharan languages of Ethiopia and Eritrea in the Horn of Africa [24]. Its writing system is left-to-right

in horizontal lines. There are no uppercase or lowercase letters in the Ethiopic script. Ethiopic is a modification-based script where the modifiers ('vowels') are usually added to the base character to give a derived vocal sound. Sometimes, the modification can also be achieved by slightly deforming the shape of the base character. It is typically done by adding a horizontal line at the top of a similar-sounding consonant. Some letters were modified to create additional consonants for use in languages other than Ge'ez. The pattern is most commonly used to mark a palatalized version of the original consonant. Some of the new symbols represent phonological processes such as palatalization and labialization [24] [28]. The Ge'ez script has been adapted to several modern languages of Ethiopia and Eritrea, frequently requiring additional letters. The script has since been extended for other languages.

2.4. Typologically Related Ethiopian Languages

This research paper discusses the six closely related Ethiopian languages that use the Ethiopic script as their writing system, namely Amharic, Awngi, Geez, Guragigna, Tigrigna and Xamtanga.

2.4.1. Amharic (አማርኛ) Language

Amharic language, also called Amarigna is an Ethio-Semitic language family derived from Ge'ez. Amharic language uses a writing system called Ge'ez script (alpha-syllabic), which is written left-to-right and its basic word order is Subject-Object-Verb (SOV). It is one of the five official languages of Ethiopia and a widely spoken. Amharic is the primary language of the Amhara region and also the most widely spoken language in Addis Ababa, the country's capital city [30] [31]. In most regions, it is the primary second language in the school curriculum. Of all the Ethio-Semitic languages, Amharic has the most speakers.

Amharic is written using a slightly modified form of the Ge'ez script known as Fidel (ፊደል). The Ge'ez similar characters have no difference in the Amharic writing system except for trends taken from Ge'ez. For example, the word Ethiopia is spelled in the same way as ኢትዮጵያ. There are 33 basic characters in the Amharic language, each of which denotes 7 vowels namely, ä (አ), u (አሁ), i (አሁ), a (አ), e (አ), ə/i (አ) and o (አ), making a total of 231 characters [32] [33]. The first order represents the base character and other orders are modifications that represent vocalized sounds of the base character. There are also 4 labialized consonants and 5 vowels with a total of 20 (4x5) characters.

The 26 of the 33 characters derived from the Ge'ez language. Alphabets are closely similar, only differing the addition of letters in Amharic language. For instance, Amharic consists of additional letters that are not found in Ge'ez language, such as ሸ (She), ረ (Che), ቸ (Ce), ጸ (Je), ኸ (Gne), ኼ (zhe) and ኸ (He). This is done by placing a small bar at the top or bottom of 7 characters inherited from Ge'ez namely, ሰ (Se), ጠ (Te), ተ (Te), ደ (De), ነ (Ne), ዘ (Ze) ከ (Ke) and ቤ (Be). Besides, Amharic uses the letter ቬ (V) to represent sounds that it acquired from Cushitic or other languages.

In Amharic languages there is an additional vowel symbol that can be combined with the first order consonants to produce an eighth form for the labializing “wa” or “oa”. Labializing is also known as a gliding vowel a sound formed by the combination of two vowels in a single syllable, in which the sound begins as one vowel and moves towards another. Example, ላ, ግ, ሲ, ሯ, ሳ, ግ, ግ, ሳ, ሳ, ሳ, ሳ, ሳ etc. Therefore, the recently standardized alphabet of the Amharic language has about 283 syllables including 238 core characters and 45 specialized pronunciations as depicted in Appendix V. Amharic has no uppercase or lowercase letters in the writing system.

The Amharic language occasionally uses the Ge'ez numbers and usually uses Arabic numerals. Amharic language shares the Ge'ez punctuation marks and uses symbols taken from European writing systems such as, the question mark (?), a dot or period (.), the exclamation mark (!), mockery mark (¡), ellipsis (. . .), double quotation mark (“”) or (« »), single quotation mark (‘’), forward slash (/), parenthesis () and hyphen (-).

2.4.2. Awnji (አውጃ) Language

The Awnji also called Awiya or Awigna is a central Cushitic language that is a branch of the Afro-Asiatic language group. Awnji is one of the four (Xamtanga, Bilen, Kemant, and Awnji) central Cushitic Agew language group [34]. The Agaw people are one of the Cushitic races and the oldest ethnic groups in Ethiopia and Eritrea [35]. It is spoken in the provinces of Agew-Midr by the Awi people, living in Central Gojjam in North Western Ethiopia [36]. Most speakers of the Awnji language live in the Agew Awi Zone Amhara Region, but there are also communities speaking the language in various areas of Metekel Zone of the Benishangul-Gumuz region. Awnji is not the official working language in Awi Administrative Zone of Amhara Region [35]. Injibara is the organizational center of the Agew Awi zone.

The Awngi have been endorsed their Nationality Zone as one in the Amhara National Regional State of Ethiopia and have granted to establish Awngi as the medium of instruction for primary education [37]. Awngi is written with a version of the Ethiopic script writing system. Awngi alphabet which share many alphabetical characters with the Ge'ez script. Awngi and Xamtanga are considered as sister languages [38].

Awngi does not use all of the Geez alphabets, but has special characters that distinguish it from the Ge'ez base alphabets. These distinct characters are ሸ, ቐ, ሸ, ቐ, ጸ, ኸ, ኸ, ቐ, ኸ and ኸ. The Awngi characters ቐ, ኸ and ኸ with their variants are frequently typed characters in the Awngi writing system. According to [39], stated that Awngi contain seven vowel phonemes and 29 base consonant phonemes, of which 5 are labialized. However, the researchers collected 30 base consonants including letter ሸ 'V' as depicted in Appendix VI. Awngi language follows a subject-object-verb (SOV) word order [36]. Awngi usually uses Amharic punctuation marks and European number systems and punctuation marks.

2.4.3. Ge'ez (ግዕዝ) Language

Ge'ez is an ancient Ethiopian language and the predecessor of the Semitic languages with their own script. All the ancient books and works of literature in Ethiopia have been preserved in the Ge'ez language. It is the main language used in the worship services of the Ethiopian and Eritrean Orthodox Churches, though Amharic may be used for sermons. Ge'ez language has 26 consonantal letters and 7 vowels in total 182 (26x7) characters. There are also 4 labialized variant consonants (labiovelars) and 5 vowels with a total of 20 (4x5) characters [28] [40]. Accordingly, the total number of Ge'ez syllables is 202, which is ((26x7) + (4x5)).

Geez alphabet is conveniently written in a tabular format of seven columns (orders) where the first column represents the base character (consonant letter) and other columns represent derived vocal sound (vowel letters) of the base character as depicted in Appendix II. The unmarked set is known as the first order also called the first form Geez (ግዕዝ). Each of the first order consonants can be combined with one of six vowels, to produce a syllograph. The resulting sets of syllograph are known as the second (ካዕብ), third (ሣልስ), fourth (ራብዕ), fifth (ኃምስ), sixth (ሳድስ) and seventh (ሳብዕ) orders [41].

The writing system of the Ge'ez language is case-sensitive for some letters such as (**ሀ/ሐ/ኀ**), (**አ/ዐ**), (**ሰ/ሠ**), and (**ጸ/ፀ**). These are referred to as similar letters, which are phonetically the same letter, but orthographically different. Also, these characters have different significance in the Ge'ez script i.e. words give different meanings. For example, with the letter 'ሠ' we can create two different words. The words **ሠረቀ** (šerek'e) and **ሰረቀ** (serek'e) meaning **ወጣ** (weta) and **ሰረቀ** (sereke) in Amharic and also in English means (He went out) and (He stole) respectively. The basic word order in Ge'ez language is Verb-Subject-Object (VSO).

Ge'ez language has its own numeral system such as, **፩** (1), **፪** (2), **፫** (3) so forth. All Ge'ez numbers have two lines on top and bottom. There is no equivalent character for the number zero in the Ge'ez number system. Punctuation marks are also crucial to clarifying meaning in written languages. Geez punctuation varies greatly from English punctuation marks. For example, the end of sentence mark '፥' (full stop) is used to show when an idea is finished and the punctuation mark '፣' is used as a word separator, however is often replaced by a blank space in current literature. Appendix IV shows more of the Ge'ez numbers and punctuation marks. In addition, the Ge'ez language has 10 tonal mark or signs of St. Yared zema [28] [32].

2.4.4. Guragigna (**ጉራጌኛ**) Language

The Guragigna (**ጉራጌኛ**) language, also called Gurage (**ጉራጌ**), is a dialect variety languages belonging to the Ethio-Semitic branch of the Afro-asiatic language family. The term Guragigna often refers to a language, while the term Gurage refer to an ethnic group. The Gurage people are an ethnic group in Ethiopia. The majority of the Gurage population lives in the Gurage Zone within the larger multi-ethnic SNNPR in central Ethiopia and is also found in various regions and cities of Ethiopia [30] [42]. Gurage is a multilingual area. The Gurage speak a variety of languages belonging to the Ethio-Semitic languages, which also includes Amharic.

There are three dialectically varied Gurage subgroups: Northern, Eastern and Western. Gurage Zone language groups emerge geographically with Soddo (**ሶዶ**) also called Kistane (**ክስታኔ**), Dobbi (**ዶቢ**) also called Gogot (**ጎጎት**), Mesqan (**መስቃን**) spoken in the Northern group, Silt'e (**ሰልጤ**), Zay (**ዛይ**), and Wolane (**ወለኔ**) in the Eastern group and Sebat Bet (**ሰባት ቤት ጉራጌ**) in the Western group [43].

2.4.5. Tigrigna (ትግርኛ) Language

Tigrigna, also spelled Tigrinya, is an Afro-Asiatic language belonging to the Semitic branch. It is mainly spoken in northern Ethiopia Tigray region and Eritrea in the Horn of Africa. Tigrigna is descended from an ancient Semitic language called Ge'ez [49]. Tigrigna is written in the Ge'ez script, originally developed for Ge'ez, also called Ethiopic. Tigrinya is the primary language for over 95% of the population in Tigray [30]. There are also communities of Tigrigna speakers in Sudan, Israel, Saudi Arabia, USA, Germany, Italy, UK, Canada and other countries [37]. Tigrigna is considered the second most widely spoken Semitic language in Ethiopia, after Amharic.

Similar to Amharic and Ge'ez, the writing system of Tigrigna is known as Fidel [49]. Each alphabet represents a consonant-vowel sequence. In the Tigrigna alphabet, letters are organized in a grid system where consonants appear vertically and their vowel-added variants, horizontally. As depicted in Appendix VIII, the way in which consonant characters are combined with vowel signs often follows a general pattern, particularly in the second to sixth orders.

Tigrigna has 37 base consonants with 7 vowels, which change the basic phoneme of each consonant into seven different character orders [50] [51]. The basic first order consonants 32 and the 5 labialized velars, totally there are 37 consonants. However, the researchers collected 39 base consonants including **ጸ**, **ፀ** and **ሠ**. A labialized velar or labiovelar is a velar consonant that is labialized, with a 'W'. Phoneme V (**፳**) is included in the Tigrigna consonant chart which is used for foreign language words. There are about 11 Tigrigna consonants not included in the Ge'ez alpha-syllabary, such as **ሸ**, **ቐ**, **፳**, **ቐ**, **ኘ**, **ኸ**, **ዠ**, **ጸ**, **፩**, **ቐ** and **ኸ** [19].

The Tigrinya language uses all base consonants and labialized velars of Geez, except for letter **ጐ** and **ጐ**. The sound 'Tse' is normally written as **ጸ** or **ፀ** in modern Ethiopian Tigrinya [52]. In this study we used both letters. The basic word order of Tigrigna language is SOV (Subject-Object-Verb). Most of the Tigrigna language punctuation marks is similar to Amharic. The traditional set of numerals used in Tigrinya texts is similar to Ge'ez number system. These numerals have been replaced by the Arabic numerals, that are, the same ones used in English. But we may get them in different writings of Tigrinya [50].

2.4.6. Xamtanga (ጽጌግግ) Language

Xamtanga is one of the four Agew languages (Xamtanga, Bilen, Kemant, and Awngi), classified as Eastern Agew peoples. The Eastern Agew people known as Xamta (or Hamir, Xamir) is part of the Central Cushitic branch of the Afro-Asiatic family [34] [53]. The language the Xamir people speak is called Xamtanga although their language is also known as Agawinya, Khamtanga, Xamta, Chamta, Khamir, Hamt'agna, Himtagne.

Xamtanga is a Central Cushitic language mainly spoken in the North Amhara Region, Sekota, Zikwala and Averegele district, Lasta and Wag Hemra Zones [37]. Languages in the surrounding area are Amharic, Afar and Tigrigna. Xamtanga is used in schools and is known by most of the people, although some also speak Amharic. The Eastern Agaw speakers are bilingual speaking both Xamtanga and Semitic languages (Amharic and Tigrinya). The official language of the Wag Hemra Zone is Amharic, with native language Xamtanga. Xamtanga heritable cultural legacies have mainly existed in the memories of tradition bearers. Wag Hemra is a Zone in the Amhara Region of Ethiopia. The Xamtanga is one of the least researched languages found in Ethiopia [35]. Research [54] approved that Xamtanga is a little documented Central Cushitic language.

Similar to the mentioned languages above, Xamtanga is written with a version of the Ethiopic script and the basic word order is SOV. Xamtanga shares many alphabetical characters with the Ge'ez, Amharic, Tigrigna and Awngi languages. The Awngi and Xamtanga languages are strongly intertwined as they both descend from the same ancestor and both share many characters in their alphabet, differing only by the addition or omission of a few letters. For example, Xamtanga includes the phonemes Te (ጠ) and Che (ጨ) but not in Awngi alphabet. According to [55], Xamtanga has a total of 39 base consonant characters, of which 7 are labialized and each character having seven different forms, usually referred to as orders, depending on the vowel with which the basic symbol is combined [55][56]. Xamtanga⁹ alpha-syllabary writing system took all the symbols and consists some new ones representing sounds not found in Ge'ez. These palatalized letters are ሸ, ጨ, ቐ, ሸ, ቐ, ጀ, ኸ, ኸ, ቐ, ቐ and ኸ. Currently, the orthographic representation of the language is organized as Appendix IX.

⁹ <http://keyboards.ethiopic.org/specification/>, <https://omniglot.com/writing/xamtanga.htm>

In addition to the published articles, the researcher reviews the documents written on the social media page of the Amhara mass media corporation Xamtanga and Awngi site to understand and find out the number of their base consonant letters. The Xamtanga writing system uses both Ethiopic and European number systems and punctuation marks. Table 2.1 summarizes the six Ge'ez-based Ethiopian languages with their ISO 639-3 code and example texts.

Table 2. 1 Example of text snippets with ISO 639-3 code for six Geez-based languages

Language	639-3 Code	Branch	Sample Script	Writing System
Amharic	amh	Semitic	የዓለም ሁሉ መድኃኒት ዛሬ ተወለደ።	Alpha-syllabic
Awngi	awn	Cushitic	አሌምሰጊ ዲኪትጊ ያኸኸኑ ናካ ካሜንስትኸ።	Alpha-syllabic
Geez	gez	Semitic	ቤዛ ኩሉ ዓለም የም ተወልደ።	Alpha-syllabic
Guragigna	sgw	Semitic	ኸኳ ያተርፍ ኸትም ሃታ የኸረ ክርስቶስ ተጨነንኸም።	Alpha-syllabic
Tigrigna	tir	Semitic	ናይ ዓለም ኩሉም መድሃኒት ሎሚ ተወለዱ።	Alpha-syllabic
Xamtanga	xan	Cushitic	ዓልምቱ እንቅ ጥላድቅ ንጭ እኹርኹ።	Alpha-syllabic

2.5. Natural Language Processing

Natural language processing (NLP) is a subfield of artificial intelligence that combines computational linguistics, statistics, machine learning and deep learning models to enable computers to process human language and understand its context, intent and emotion [6] [7]. NLP arose to make things easier for users and to satisfy the desire to communicate with the computer in natural language. With the emergence of several social media platforms and availability of a large amount of text data in them, NLP plays a great role in understanding and generating data today. Applications of NLP lies under several fields like text classification, machine translation, email spam detection, information extraction, text summarization, text extraction, machine translation, speech recognition, text classification, question answering and more [6] [10] [11].

2.5.1. Text Classification

In machine learning, classification is the problem of categorizing a data instance into one or more known classes. Text classification is an integral part of NLP. The data can be originally of different formats, such as text, speech, image, or numeric. Text classification is a special instance of the classification problem, where the input data is text and the goal is to categorize the piece of text into one or more class from a set of pre-defined classes [10] [57]. The text can be of arbitrary length: a character, a word, a sentence, a paragraph, or a full document.

In general, the text classification system contains four different levels of scope that can be applied [8]. In the document level, the algorithm obtains the relevant categories of a full document. In the paragraph level, the algorithm obtains the relevant categories of a single paragraph (a portion of a document). In the sentence level, obtains the relevant categories of a single sentence (a portion of a paragraph). In the sub-sentence level, the algorithm obtains the relevant categories of sub-expressions within a sentence (a portion of a sentence) [8].

Supervised classification approach, including text classification, can be further distinguished into three types based on the number of categories involved: binary, multiclass, and multi-label classification. If the number of classes is two, it's called binary classification. If the number of classes is more than two, it's referred to as multiclass classification. In both binary and multiclass settings, each document belongs to exactly one class from C , where C is the set of all possible classes. In multi-label classification, a document can have one or more labels attached to it.

Text classification is performed on multilingual or monolingual text documents. LID on monolingual texts is called a multiclass classification problem, in which a text is assigned to one of the C classes. Whereas LID for multilingual texts is viewed as a multi-label classification task, where a text can be mapped to a subset of labels from a larger closed label set [57]. The next sub-section discusses a detailed overview of the LID approaches.

2.5.2. Language Identification

Language identification is a special case of text classification. LID on written texts, also known as language detection and sometimes as language guessing, which has been constantly studied over decades [8]. It is a process that attempts to classify text in a language into a pre-defined set of known languages.

2.6. Language Identification Approaches

The most important step of the language identification pipeline is selecting the best embedding techniques, classification techniques and evaluation methods. LID can be done by applying the two main approaches, which are statistical and non-statistical approaches [58] [59]. In addition, it is also possible to apply hybrid approaches, that is combining rule-based and statistical approaches.

Statistical Approaches: It is known as computational approaches, are basically relying on machine learning and deep learning approaches which require less human effort and a large training dataset for each feature to be identified [60]. It's a learning-based solutions using dataset. Statistical approaches follow two successive phases: one is the training phase and the other is classification phase. In the training phase, feature vectorization (word embedding's techniques) is performed from the given training dataset known as the training corpus. There are several statistical classification techniques for LID, such as Naïve Bayes Classifier, SVM and Artificial Neural Networks [60].

Non-statistical Approaches: It is also called non-computational or rule-based approaches are basically linguistic approaches, the researcher must have extensive knowledge of the rules of the language used. Some of the most prominent non-statistical methods are using diacritics and special symbols, most frequent words used, grammatical-based, stopwords-based and lexicon-based approach [61] [62]. In the rule-based approach, texts are separated into an organized group using a set of handicraft linguistic rules to identify the language. In a non-statistical approach, one way to group text is to create a list of words that relate to a specific column, and then evaluate the text based on occurrences of those words [63]. These approaches require a lot of domain knowledge to be comprehensive, take a long time to compile, and are difficult to scale.

Hybrid Approach: Hybrid approach is the process of solving the language identification problem by combining two approaches, rule-based and statistical approaches [64]. First, we discussed embedding techniques as follows:

2.6.1. Embedding Techniques

Machine learning and deep learning algorithms are not capable of processing strings or plain text in their raw form. To convert these text data into numerical data, we need some smart ways which are known as vectorization, or in the NLP world, it is known as embedding techniques [65]. Word embedding is a language modelling technique to represent the words, texts or phrases as vectors of real numbers. Later those vectors are used to build various machine learning and deep learning models. Using word embedding approach, words and documents are represented in the form of numeric vectors allowing similar words to have similar vector representations.

In a broad sense, they require numerical numbers as inputs to perform any sort of task, such as classification, regression, clustering, etc. Language identification is done based on document characteristics, typically at the character or word level. Commonly used text representation techniques used for language modelling fall into two basic categories: frequency-based embedding and prediction-based embedding [65].

Frequency-based or statistical-based word embedding approaches utilize to vectorize the text depending on the frequency of occurrence of the words or characters in the text or document. For example, one-hot encoding, Bag-of-Characters (BoC), Bag-Of-Words (BOW), N-gram and Term Frequency-Inverse Document Frequency (TF-IDF) are frequency-based word embedding approaches. Prediction-based word embedding approaches are pre-trained model that capture the semantic and syntactic meaning of a word as they are trained on large datasets. Word to Vectors (Word2Vec), Global Vectors (GloVe), fastText are an example of pre-trained word embedding models [60].

2.6.1.1. Bag-of-Words Model

Bag-of-words (BoW) model is one of the popular word embedding techniques of text where each value in the vector would represent the count of words in a document or sentence [66]. This method is mostly used in language modeling and text classification tasks. All the words in the corpus are formed into a mapping array. According to the mapping array, a sentence can be represented as a vector. The i -th element in the vector represents the frequency of the i -th word in the mapping array of the sentence. The individual value of the vector denotes the word frequency corresponding to its inherent position in the text.

The BoW model has two main operations. The first operation is tokenization; It's the process of dividing each sentence into words. Before tokenization, all sentences are converted to lowercase. After dividing the sentences into words and creating a list of all unique words, the next operation is to create a vector for each sentence with the frequencies of the words. Let's look at the following sample texts:

Sentence 1: Dara likes to go to cinema. Going to cinema is one of the hobbies of Azad.

Sentence 2: Dara also wants to go to play football and to go to swim.

Based on the above mentioned text sentences, a group of words is produced for the sentences by tokenizing the sentences to produce a dictionary of the words, as follows:

Sentence 1: “Dara”, “likes”, “to”, “go”, “to”, “cinema”, “Going”, “to”, “cinema”, “is”, “one”, “of”, “the”, “hobbies”, “of”, “Azad”

Sentence 2: “Dara”, “also”, “wants”, “to”, “go”, “to”, “play”, “football”, “and”, “to”, “go”, “to” “swim”. Therefore, each bag of the words is represented as following:

$BoW1 = \{“Dara”:1, “likes”:1, “to”:3, “go”:1, “cinema”:2, “Going”:1, “is”:1, “one”:1, “of”:2, “the”:1, “hobbies”:1, “Azad”:1\}$;

$BoW2 = \{“Dara”:1, “also”:1, “wants”:1, “to”:4, “go”:2, “play”:1, “football”:1, “and”:1, “swim”:1\}$;

The representation of the sequences or vectors does not take into account the order of the words in the documents, only the counts of words matter, which is one of the major properties of the BoW model. The union of two text documents or sentences is the combination of the words of the both text documents as shown in equation 2.1.

$$BoW3 = BoW1 \cup BoW2 \tag{2.1}$$

The result of the union of the two sentences will be: Dara likes to go cinema going is one of the hobbies Azad also wants play football and swim. The unique words are represented in the format of BoW as the following:

$BoW3 = BoW3 = \{“Dara”:2, “likes”:1, “to”:7, “go”:3, “cinema”:2, “going”:1, “is”:1, “one”:1, “of”:2, “the”:1, “hobbies”:1, “Azad”:1, “also”:1, “wants”:1, “play”:1, “football”:1, “and”:1, “swim”:1\}$. Vectors for each sentence with the frequency of words is called a sparse matrix.

Below is the sparse matrix of example sentences.

Sentence: 1 [1, 1, 3, 1, 2, 1, 1, 1, 2, 1, 1, 1, 0, 0, 0, 0, 0, 0]

Sentence: 2 [1, 0, 4, 2, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1]

2.6.1.2. Bag-of-Characters Model

A bag-of-characters (BoC) vectorization technique, also called characters-to-vector (chars2vec model), converts the raw input text into numerical format using its letter frequency [67]. It is character embedding techniques. Count the occurrence frequency of all the possible letters in a sample text string. List of such letters is just sum of alphabets of a sample text string, without letters repetitions. This technique generates a vector for each character with the fixed vector size. This vectorization technique converts the text content to numerical feature vectors.

Char2vec takes a document from a corpus and converts it into a numeric vector by mapping each document character to a feature vector for the machine learning or deep learning model. Bag-of-character text representation technique is similar to the BOW model, with the exception that it is using characters and not words for defining the vocabulary [68]. The main advantage of character-level representation is that it naturally deals with out-of-vocabulary. Therefore, the character-based model can provide representations for unseen words and share information on morpheme-level regularities. In addition one-hot-encoding is used to represent the categorical variables as binary vectors [69].

2.6.1.3. N-gram Model

N-gram language model finds the probability distribution over a sequence of words that shows the tendency of frequently following words [70] [71]. It shows how many words or characters are considered for predicting the next word or character in a sentence, N may be 1 words for unigrams, 2 words for 2-gram (bigrams), 3 words for 3-gram (trigrams) model and so on. Language model can be constructed over sequence of characters or words [72] [73]. For example, the word “**አማርኛ**” can be modeled using character level N-grams surrounded by underscores, as follows:

Unigrams: , **አ**, **ማ**, **ር**, **ኛ**,
 Bigrams: **አ**, **አማ**, **ማር**, **ርኛ**, **አ**
 Trigrams: **አማ**, **አማር**, **ማርኛ**, **ርኛ** , **አ**
 Quad-grams: **አማር**, **አማርኛ**, **ማርኛ** , **ርኛ** , **አ**

In word based N-gram language modelling, probability of next word w_n is calculated previous n-1 words $w_1w_2 \dots \dots w_{n-1}$ in the sentence as:

$$p(w_n|h) = p(w_n|w_1w_2w_3 \dots \dots w_{n-2}w_{n-1}) \tag{2.2}$$

where, h represents contextual history of words in the sentence. The probability distribution of N-grams is obtained by using maximum likelihood estimation as:

$$p(w_n|w_1, \dots \dots w_{n-2}w_{n-1}) = \frac{c(w_1, \dots \dots w_{n-2}, w_{n-1}w_n)}{c(w_1, \dots \dots, w_{n-2}, w_{n-1})} \tag{2.3}$$

Thus, N-gram modeling finds the probability of a subset of n characters or words sequence in a long sentence in the text. Here, n is the number of characters or words used and it may vary from 1 to any number of words.

For language identification purposes, sequences of characters or words in the sentence of all languages are modeled separately using N-grams [71] [72]. The N-gram technique captures the profile of a natural language during the training. Thus, profiles of all the languages present in the training dataset are stored in trained N-gram models separately. The profile of the language in the test document is then obtained using the N-gram technique. The distance between the profile of the test document language and profiles of languages in training documents is measured using the similarity metric and finally, the language in the training dataset whose profile is having minimum distance is selected as the language of the test document.

2.6.1.4. Word2Vec

Word2vec (word to vector), as the name suggests, is a tool that converts texts into vector form. The method involves iteration over a corpus of text to learn the association between the words and the vector creation process is performed by determining which words have higher occurrences of the target word [60]. It relies on a hypothesis that the neighboring words in a text have semantic similarities with each other. Actually Word2Vec is pre-trained prediction-based embedding model i.e. it covers algorithms and training on its own data.

Word2vec is a combination of two techniques, namely continuous bag of words (CBOW) and Skip-gram model. These are basically shallow neural networks that have an input layer, a projection layer and an output layer as shown in Figure 2.1. In CBOW, the neural network model takes various words as input and predicts the target word that is closely related to the context of the input words. On the other hand, the Skip-gram architecture takes one word as input and predicts its closely related context words.

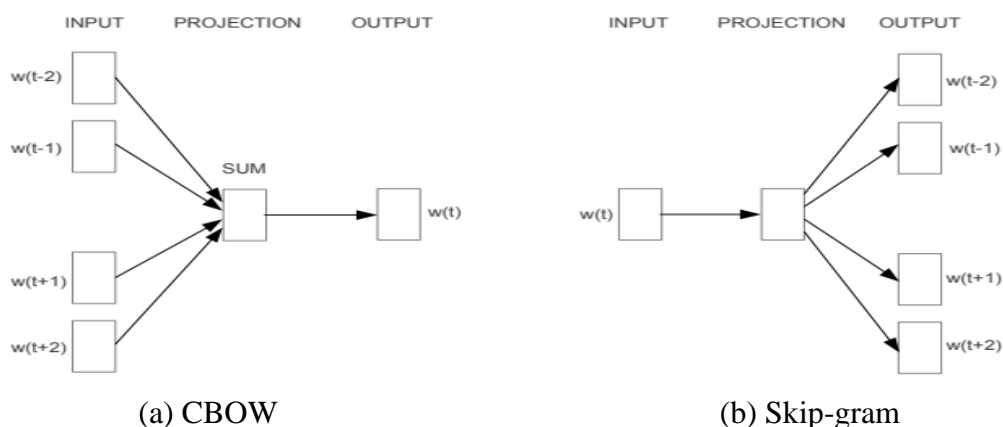


Figure 2. 1 The structure of CBOW and Skip-gram model [60]

2.6.1.5. GloVe

GloVe is one of the newest methods for calculating the vector representation of words [74]. It is referred to as global vectors because the global corpus statistics were captured directly by the model. It finds great performance in world analogy and named entity recognition problems. It enables word vectors to contain as much semantic and grammatical information as possible. The construction method of word vector is firstly, the co-occurrence matrix of words is constructed based on the corpus, and then the word vector is learned based on the co-occurrence matrix and GloVe model. Finally, the represented text is fed into the classifier according to selected features. Word2Vec only captures the local context of words. During training, it only considers neighboring words to capture the context. GloVe considers the entire corpus and creates a large matrix that can capture the co-occurrence of words within the corpus.

2.6.2. Classification Techniques

There are several statistical and non-statistical classification techniques for the LID problem, the followings being the most common.

2.6.2.1. Grammatical-based Approach

The grammatical approach is based on the presence of some grammatical features of words related to the language such as prepositions, conjunctions, determiners, names, adverbs, auxiliaries, etc. representing approximately 50% of sentences and texts in most languages [75]. For example, in English we can find words like (the, they, are, he, she, them, of, and, a, to, in, is, you, that etc.). If an input text is mostly tokenized by the word list of a certain language, then this language is taken as the result. This approach is faster and more efficient than stopwords approach, but it also embodies several weaknesses, such as the texts should be segmented into words which is difficult for some languages, grammatical words are often removed during a preprocessing performed on the text, the approach gives poor results in the case of short texts because of the absence of these grammatical words.

2.6.2.2. Stopwords-based Approach

This method uses dictionaries that contain stopwords [76]. The stopword dictionaries are created before the algorithm is applied. The algorithm selects the text, finds the most common words and compares them to stopwords. The language with the most stopwords is selected as the identified language.

The algorithm counts how many unique stopwords are seen in the analyzed text for inclusion in the language-relationship dictionary and recognizes the language based on the ratio in the language-relationship dictionary. Stopwords prove to be very effective for automatic LID. Although they have different semantical meanings, stopwords can be very similar or even the same for related languages [76].

2.6.2.3. *Lexicon-based Approach*

The lexical-based approach is based on the use of a lexicon or dictionary for every language to identify [14] [77]. This method identifies the language of a target text by comparing the words of the new text with a fixed list of words for every language (a language lexicon). The language whose lexicon contains all or most words of the text is the effective language of the text [78] [79]. Several problems can arise here, does not handle variations in the spellings of the words or misspellings, absence of scientific lexicon and typing mistakes can disrupt results gotten by this approach.

2.6.2.4. *Artificial Neural Networks*

Artificial neural networks, also known as neural networks, rely on machine learning techniques that mimic the structure of the human brain [80]. The basic idea behind a neural network is that it consists of many neurons that receive knowledge about a task through training, just like the human brain is trained to learn new things throughout life [80] [81]. A biological neuron receives its input signals from other neurons via dendrites, and these input signals are represented as numerical values in the perceptron. At the synapses between dendrites and axons, electrical signals are modulated to different degrees. Similarly, this is modeled in the perceptron by multiplying each input value by a value called a weight, and they measure the importance level of each input.

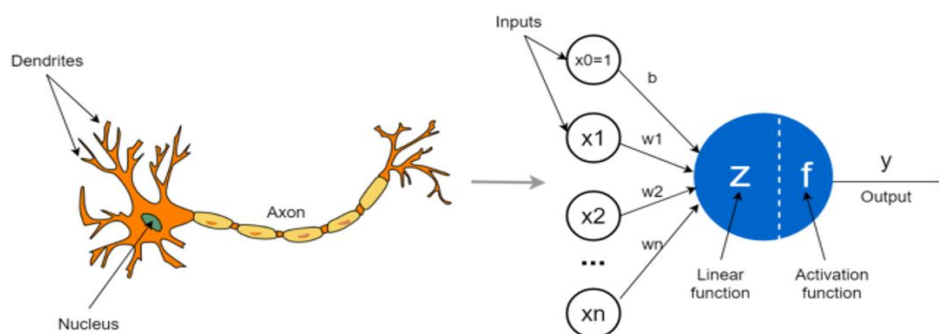


Figure 2. 2 Structural similarity of biological and artificial neurons [82]

In a biological neuron, the nucleus produces an output signal based on the signals provided by dendrites. Similarly, a perceptron performs some calculations based on the input values and produces an output. In a biological neuron, the output signal is carried by the axon. Likewise, the axon in a perceptron is the output that will be the input for the next perceptrons [80]. The similarity of their structure and functionality is shown in Figure 2.2, where the left side of a figure represents a biological neuron with its nucleus, dendrites and axon, while the right side of the figure represents an artificial neuron with its inputs, weights, bias, weighted sum, activation functions and output.

2.6.2.5. Deep Neural Networks

Deep learning is when the NN becomes deeper and when more so called hidden layers are added to the network [83]. Deep learning models are trained using large amounts of labeled data. Due to their complex nature, DNNs usually require long periods of time to train the network on the input data and powerful computers with specialized processing units [81]. DNNs are powerful algorithms used to handle complex computational tasks such as text or image classification, machine translation and self-driving cars. There are two basic types of neural network architectures based on how the information is propagated through the network, namely feedforward neural networks and feedback neural networks [83].

Feedforward neural networks: The flow of information is unidirectional through input layer; this information continues to be processed in this one direction until it reaches the output layer, this means that the computational model represents an acyclic graph. A feed forward NN can be used for all three types of machine learning; supervised, unsupervised and reinforcement learning [83]. The goal of feedforward neural networks (FFNN) is to approximate the function f . For example, the function $y = f(x)$ maps the input x to the value y . FFNN defines the mapping $y = f(x; \theta)$ and finds the value of the parameters θ , which leads to the best approximation of the function.

Feedback neural networks: The flow of information occurs in both directions by introducing network loops, propagating values backward to earlier layers from the hidden and output layers. Multilayer Feed-forward Neural Networks (MLNNs) or Multi-layer Perceptrons (MLPs), Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNNs) are the most common algorithms of deep neural network [83].

2.6.2.6. Multi-layer Perceptrons

Multi-layer Perceptrons (MLPs) are composed of artificial neurons called perceptrons [84]. Therefore, before explaining the general structure of MLPs, the general structure of a perceptron will be explained. Perceptron, also called artificial neuron, is a single-layer supervised machine-learning algorithm that solves the problem of binary classification [84]. As shown in Figure 2.2, a perceptron receives n features as input x_1, x_2, \dots, x_n , each of these features is assigned a weight w_1, w_2, \dots, w_n and adds the bias term b , then computes the linear function, z on which an activation function, f is applied to get the output y .

The linear or summation function of the perceptron is denoted by z . Its output is the weighted sum of the inputs plus bias unit and can be computed as:

$$z = (x_1 * w_1 + x_2 * w_2 + \dots + x_n * w_n) + b \quad (2.4)$$

Where, x_1, x_2, \dots, x_n are input that take numerical values for learning process. They can be raw input data or outputs of the other perceptrons. The parameters w_1, w_2, \dots, w_n are weights that take numerical values appended to each input and control the level of importance of each input. The higher the value, the more important the input and $x_1 * w_1 + x_2 * w_2 + \dots + x_n * w_n$ is called the weighted sum of inputs. The parameter b is called the bias unit that also takes a numerical value. In some contexts, the bias b is denoted by w_0 . The input x_0 always takes the value 1. Therefore, $b * 1 = b$. It is added to the weighted sum of inputs. The purpose of including a bias is to avoid the zero value of the activation function of each perceptron. In other words, if all x_1, x_2, \dots, x_n inputs are 0, the z is equal to the value of bias.

Obviously linear problems can be solved in a single perceptron, but it is not well suited to non-linear cases. To solve these complex problems, MLPs can be considered. MLPs known as the deep feedforward neural networks, belong to the class of feedforward neural networks (FFNNs) with at least three layers [84]. MLPs models are the most basic deep neural network, which is composed of a series of fully connected layers, meaning that all neurons in a layer are connected to all neurons in the next layer.

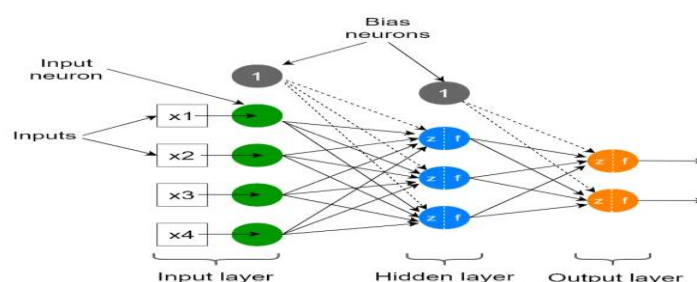


Figure 2. 3 A simple multilayer neural network

The basic structure of multi-layer neural network consists of input layer, weights, biases, hidden layer, activation functions and output layer [85]. The input layer is the initial layer of the network that receives the input signal to be processed. The hidden layer performs all kinds of computation on the features entered through the input layer and transfers the result to the output layer. The output layer takes input from preceding hidden layers and comes to a final prediction based on the model's learnings.

Using an activation function an output signal is calculated and feed forward towards the next layer of nodes in the network, where the neurons collect and compute the next signal given different weights to the neurons, until it reaches the output layer, which represents the possible classes, where the output is determined [85]. Neural networks are nonlinear models, designed to describe and handle nonlinear relationships. In the non-linear functions of the neural network an activation function f is applied to the weighted sum z to get the final output y as shown in equation 2.5.

$$y = f(z) = f\left(\sum_{i=1}^n w_{ij}x_i + b\right) \quad (2.5)$$

where, w_{ij} , x_i , b and y are the weights, input values, biases and output values respectively and $f(.)$ is non-linear function.

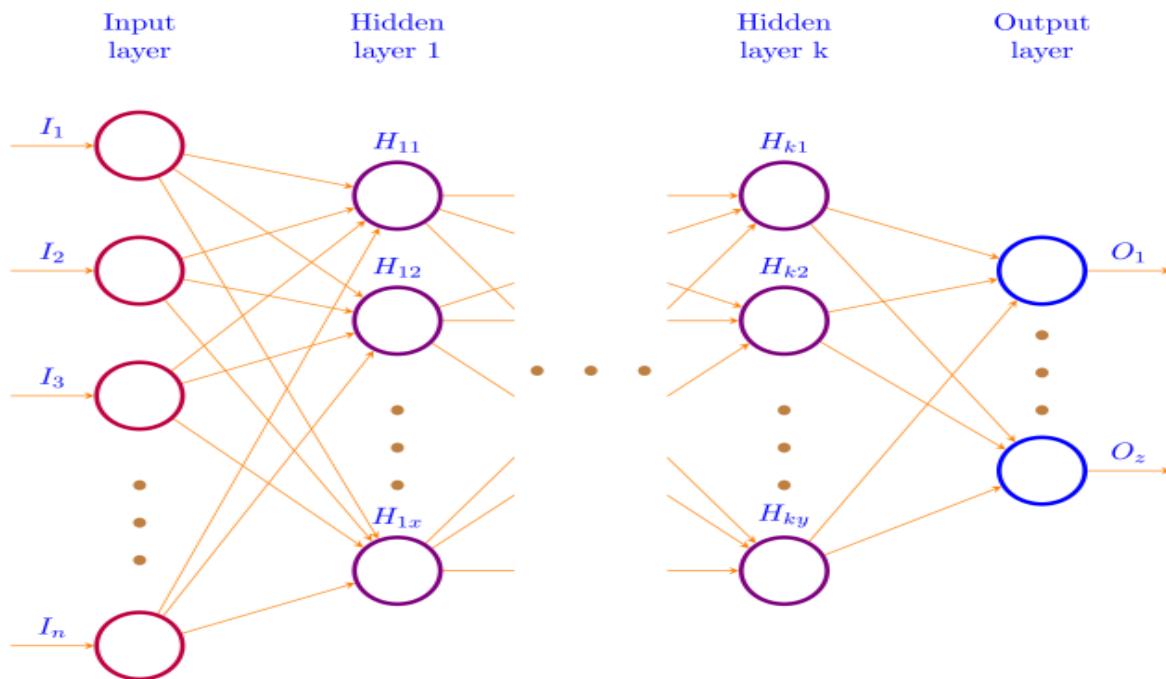


Figure 2. 4 Deep neural network with multiple number of hidden layers [86]

A DNN contains an input and output layer, separated by l layers of hidden units. Given an input sample clamped to the input layer, the other units of the network compute their values according to the activity of the units that they are connected to in the layers below. We will consider a particular sort of topology here, where the input layer is fully connected to the first hidden layer, which is fully connected to the second layer and so on up to the output layer.

Given an input \mathbf{x} , the value of the j -th unit (node) in the i -th layer is denoted $h_j^i(\mathbf{x})$, with $i = 0$ referring to the input layer, $i = l + 1$ referring to the output layer. The default activation level is determined by the internal bias b_j^i of that unit. The set of weights w_{jk}^i between $h_k^{i-1}(\mathbf{x})$ in layer $i - 1$ and unit $h_j^i(\mathbf{x})$ in layer i determines the activation function of unit $h_j^i(\mathbf{x})$ as follow:

$$h_j^i(\mathbf{x}) = \text{sig}(a_j^i) \text{ where } a_j^i(\mathbf{x}) = b_j^i + \sum_k w_{jk}^i h_k^{i-1}(\mathbf{x}) \forall i \in \{1, \dots, l\}, \text{ with } h^0(\mathbf{x}) = \mathbf{x} \quad (2.6)$$

where $\text{sig}(\cdot)$ is the sigmoid function. Given the last hidden layer, the output is computed similarly by:

$$\mathbf{o}(\mathbf{x}) = \mathbf{h}^{l+1}(\mathbf{x}) = f(\mathbf{a}^{l+1}(\mathbf{x})) \text{ where } \mathbf{a}^{l+1}(\mathbf{x}) = \mathbf{b}^{l+1} + \mathbf{w}^{l+1} \mathbf{h}^l(\mathbf{x}) \quad (2.7)$$

where the activation function $f(\cdot)$ depends on the supervised task the network must achieve. Typically, to find a distribution over K classes, it becomes a Softmax function for a classification problem.

2.6.2.7. Convolutional Neural Networks

Convolutional Neural Networks (CNN) is one of the most popular deep neural network models in use today. This neural network computational model uses a variation of multi-layer perceptrons and contains one or more convolutional layers that can be either fully connected or pooled. CNN were first proposed by [87]. CNNs are originally developed for computer vision tasks, but later on made their way in various AI applications, including facial recognition, text digitization and natural language processing [88]. The CNN architecture consists of three types of layers, also called multi-building blocks [88]. Each layer in the CNN architecture, including its function, is described in detail below.

Convolutional Layer: It consists of a collection of convolutional filters (so called kernels). The input texts, expressed as N -dimensional metrics, is convolved with these filters to generate the output feature map. A grid of discrete numbers or values describes the kernel. Each value is called the kernel weight. Random numbers are assigned to act as the weights of the kernel at the beginning of the CNN training process.

Convolution consists of shifting the convolution kernel over the whole set of values. These weights are adjusted at each training era; thus, the kernel learns to extract significant features [88].

Pooling Layer: A pooling layer receives the result from a convolutional layer and compresses it. The main task of the pooling layer is the sub sampling of the feature maps. Concurrently, it maintains the majority of the dominant information in every step of the pooling stage. Several types of pooling methods are available for utilization in various pooling layers. These methods include tree pooling, gated pooling, average pooling, min pooling, max pooling, global average pooling, and global max pooling [68].

Fully Connected Layer: Commonly, this layer is located at the end of each CNN architecture. Inside this layer, each neuron is connected to all neurons of the previous layer, the so called fully or densely connected approach. It is utilized as the CNN classifier [89].

Character-level CNNs have been studied for text classification [68] [67]. As illustrated in Figure 2.5, the model takes as input the characters in a fixed-sized, encoded as one-hot vectors, passes them through a deep CNN model that consists of six convolutional layers with pooling operations and three fully connected layers. This approach scales well with alphabet size, allowing to preserve more information from the original text to enhance classification performance.

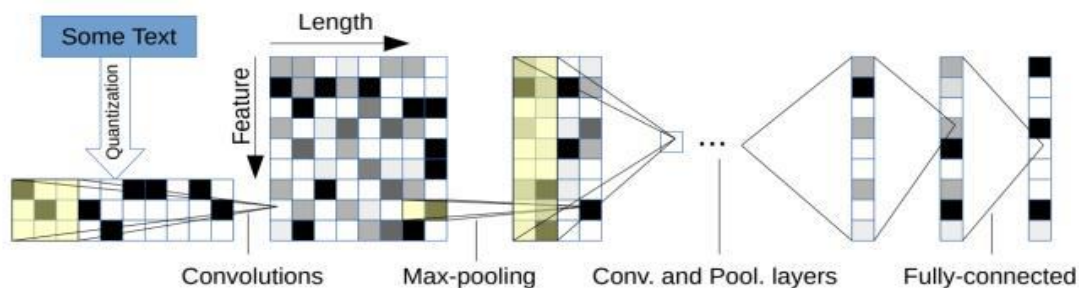


Figure 2. 5 The architecture of a character-level CNN model [68]

2.6.2.8. Recurrent Neural Networks

Recurrent neural networks (RNN) are another class of deep neural networks designed to handle sequential samples of data. The input of RNN consists of the current input and the previous samples. RNNs take the output of a processing node and transmit the information back into the network. This results in theoretical "learning" and improvement of the network. Each neuron in an RNN owns an internal memory that keeps the information of the computation from the previous samples, and these historical processes are reused in the future during processing [90].

Recurrent neural networks have certain problems when handling time serial data samples where a vanishing gradient problem can occur. The gradients can blow up and give an unreliable model. Compared to a regular FFNN, the RNN has a feedback connection within the hidden layer, as shown in Figure 2.6. This is the reason for recurrent neural networks to in theory, handle sequential input data better than feed forward neural networks.

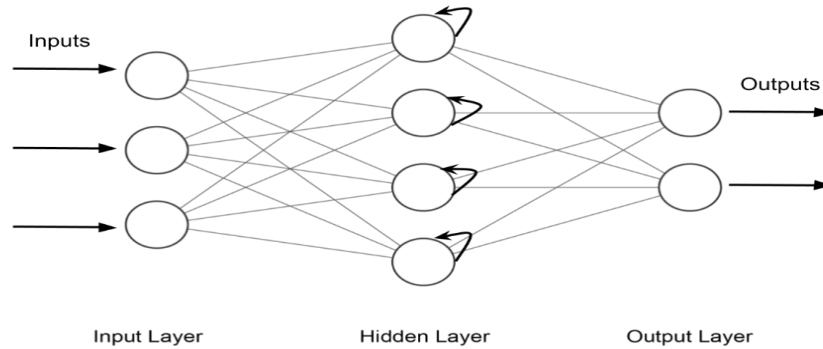


Figure 2. 6 A standard RNN with a feedback connection inside the hidden layer

Given an input sequence $x = (x_1, \dots, x_T)$, a standard recurrent neural network (RNN) computes the hidden vector sequence $h = (h_1, \dots, h_T)$ and output vector sequence $y = (y_1, \dots, y_T)$ by iterating the following equations from $t = 1$ to T :

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (2.8)$$

$$y_t = W_{hy}h_t + b_y \quad (2.9)$$

where the W terms denote weight matrices (e.g. W_{xh} is the input-hidden weight matrix), the b terms denote bias vectors (e.g. b_h is hidden bias vector) and f is the hidden layer function.

The key aspects (hyperparameters) in building the deep neural network infrastructure are discussed in detail as follows:

Activation Functions

The output of an MLP network is determined using a variety of activation functions, also known as transfer functions [91] [92]. The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. The idea of an activation function is to mimic, like mentioned earlier, the functioning of the human brain and act like a "switch", where an input contribution either gives a value of 1 or is kept to 0. The purpose of the activation function is to introduce non-linearity into the output of a neuron. The following types of activation functions are most commonly used in deep neural networks [85].

Sigmoid: The sigmoid function usually used in output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, output can be predicted easily to be 1 if value is greater than 0.5 and 0 otherwise. The function is defined as:

$$f(z) = \frac{1}{1 + e^{-z}} \quad (2.10)$$

When using the Sigmoid function for hidden layers, it is a good practice to use a “Xavier Normal” or “Xavier Uniform” weight initialization (also referred to Glorot initialization, named for Xavier Glorot) prior to training [93]. When we are working with deep neural networks, initializing the network with the right weights can be the hard to deal with because deep neural networks suffer from problems called vanishing or exploding gradients.

Weight initialization is used to define the initial values for the parameters in neural network models prior to training the models on a dataset. The Xavier initialization method is calculated as a random number with a uniform probability distribution U between the range $-\frac{1}{\sqrt{n}}$ and $\frac{1}{\sqrt{n}}$, where n is the number of inputs to the node [93]. Biases are initialized be 0 and the weights W_{ij} at each layer are initialized as:

$$W_{ij} = U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right] \quad (2.11)$$

Rectified Linear Unit (ReLU): It belongs to one of the most frequently used activation functions applied in deep networks. It gives an output z if z is positive and 0 otherwise. It is defined as:

$$f(z) = m(0, z) \quad (2.12)$$

A couple of downsides with the ReLU is that negative neurons will be kept at zero and have a hard time recovering, if the learning rate is too high the model can stop updating the weights. It is only used for the hidden layers of neural network [85].

Softmax: Softmax is a last layer activation function that returns the probability of a data point belonging to each individual class in a multiclass classification problem. This could give a hint towards how certain the model is on predicting the correct class, or how confused the model is. The range of the output for each class is between 0 and 1.

The sum of all the classes probabilities is 1. It can be mathematically expressed as:

$$f(z)_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (2.13)$$

Here, e^{z_i} represents the non-normalized output from the preceding layer, K represents the number of classes in the output layer, z represents input vector and e^{z_j} represents standard exponential function for output vector.

Loss Function

The other key aspect in setting up the deep neural network infrastructure is selecting the right loss functions. Loss function is a function that computes the distance between the current output of the algorithm and the expected output. Loss functions are utilized in the output layer to calculate the predicted error created across the training samples in the neural network model. This error reveals the difference between the actual output and the predicted one. Using the error backpropagation [92], the neuron weights in particular layer are updated in such manner, that the error rate decreases several types of loss functions are employed in various problem types. The following concisely explains some of the loss function types used for classification problems.

Cross-Entropy Loss Function: It is an optimization function which is used in case of training a classification model which classifies the data by predicting the probability of whether the data belongs to one class or the other class [92].

Categorical Cross-Entropy: is a loss function used for multi-class classification tasks. The output label is assigned one-hot category encoding value in form of 0s and 1. The outputted loss is the negative of the sum of the true values p multiplied by the log of the predicted values $\log y$. If the number of classes $M > 2$, the categorical cross-entropy loss function can be computed as:

$$Loss(p, y) = - \sum_{i=1}^M \hat{y}_i \cdot \log(\hat{p}_i) \quad (2.14)$$

where, M represent number of classes, \hat{y} is true output and \hat{p} is the classifier's predicted probability distributions.

Optimization Algorithms

Optimization refers to a procedure for finding the input parameters to a function that result in the minimum or maximum output of the function. Optimizers are algorithms used to change the attributes of the neural network such as weights and learning rate to reduce the losses. MLPs are trained through a method called backpropagation [91]. Backpropagation is an algorithm that back propagates the errors from output nodes to the input nodes. Therefore, it is simply referred to as backward propagation of errors. The point of backpropagation is to fine-tune the weights of a network based on the errors obtained in the previous epochs [91].

Gradient Descent or **Gradient-based learning algorithm**: To minimize the training error, this algorithm repetitively updates the network parameters through every training epoch. It is highly used in supervised learning to minimize the error function and find the optimal values for the parameters. The backpropagation algorithm looks for the minimum value of the error function in weight space using the method of descent-based learning algorithms such as AdaGrad, Root Mean Squared Propagation (RMSprop) and Adaptive Moment Estimation (Adam) [91] [92] [94]. Generally, each training iteration of neural network has three main stages: feedforward of the input training pattern, backward propagation of the associated error and modification of weights.

Adaptive Moment Estimation (Adam): Adam is an optimization function, used to minimize the error rate of the model in the prediction, which computes the learning estimation for each parameter [94]. Adam represents the latest trends in deep learning optimization that only requires first-order gradients with little memory requirement and it is the most suitable optimization method in deep neural network [94].

Adam optimization algorithm incorporates the momentum method and RMSprop, along with bias correction. Adam keeps an exponentially decaying average of past squared gradients $v(t)$ and past gradients $m(t)$. $v(t)$ and $m(t)$ are values of the first moment which is (the mean) and the second moment which is the un-centered variance of the gradients respectively [94] [95]. Both moving averages are initialized to 0, which leads to the moments' estimation biased towards zero. Such situation occurs mostly during the initial phases when decay parameters (β_1, β_2) have values close to 1.

Such biased can be removed using modified estimations \hat{m}_t and \tilde{v}_t :

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \tilde{v}_t = \frac{v_t}{1 - \beta_1^t} \quad (2.15)$$

The parameters are updated according to the formula:

$$\theta_{t+1} = \theta_{t-1} - \frac{\alpha}{\sqrt{\tilde{v}_t} + \epsilon} \cdot \hat{m}_t \quad (2.16)$$

Adaptive Gradient (AdaGrad): AdaGrad works on the learning rate component by dividing the learning rate by the square root of S is initialized to 0 which is the cumulative sum of current and past squared gradients (i.e. up to time t) [95] [96].

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}, \text{ where } S_t = S_{t-1} + \left[\frac{\partial L}{\partial w_t}\right]^2 \text{ and } S \text{ is initialized to } 0. \quad (2.17)$$

Root Mean Square prop (RMSprop): It is another adaptive learning rate that is an improvement of AdaGrad. Instead of taking cumulative sum of squared gradients like in AdaGrad, it takes the exponential moving average of these gradients [95].

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t}, \text{ where } S_t = S_{t-1} + (1 - \beta) \left[\frac{\partial L}{\partial w_t}\right]^2 \text{ and } S \text{ is initialized to } 0. \quad (2.18)$$

Regularization Techniques

There are two main problems when training neural networks: overfitting and underfitting.

Overfitting: Overfitting refers to the phenomenon where a neural network models the training data very well, but fails when it sees new data from the same problem domain. Less complex neural networks are less susceptible to overfitting [97].

Underfitting: Underfitting is the opposite of overfitting. It occurs when the model is not sensitive enough to the training data and as a result, the model fails to learn the most important patterns in the training data i.e., it only performs well on training data but performs poorly on testing data. Underfitting is often not a real problem because we can prevent it by simply making the model deeper i.e. (add more layers or neurons to the model) or train for a few more epochs [97].

Dropout: Dropout is one of the most effective and commonly used techniques to prevent overfitting in neural networks [97]. The term “dropout” refers to dropping out the nodes in a neural network as seen in Figure 2.7. During each training epoch, neurons are randomly dropped.

Dropout is easily implemented by randomly selecting nodes to be dropped out with a given probability in each weight update epoch. For example, on the left side of Figure 2.7, suppose we have a feedforward neural network with no dropout. Using dropout with an assumed probability of $P = 0.5$ that a random neuron will be turned off during training would result in a neural network on the right hand side.

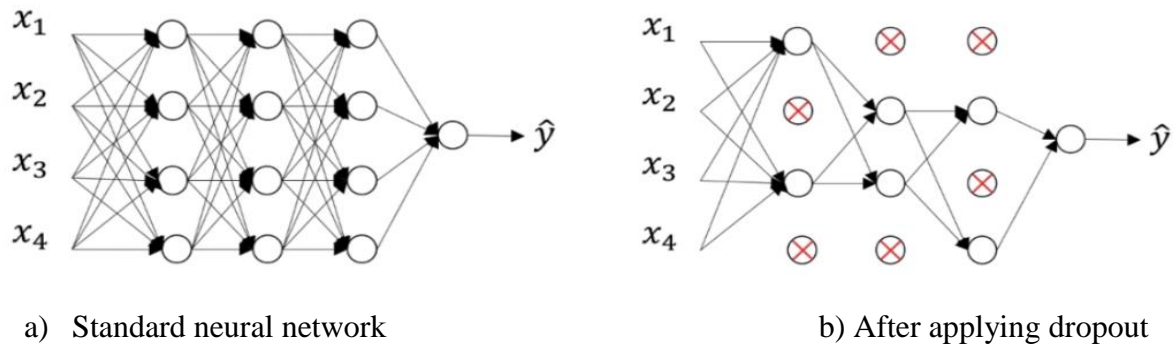


Figure 2. 7 Neural network with dropout and without dropout [97]

Batch Size: The batch size is responsible for how many samples we want to use in one epoch, which means how many samples are used in one forward/backward pass. This increases the speed of the computation as it need fewer epochs to run, but it also needs more memory, and the model may degrade with larger batch sizes.

2.6.3. Evaluation Techniques

Evaluation metrics adopted within deep learning tasks play a crucial role in achieving the optimized classifier [98]. Confusion matrix is the summary of prediction results on a classification problem that records the number of occurrences between two raters the actual and the predicted classification.

Table 2. 2 Confusion Matrix

Actual Values	Predicted Values	
	<i>Positive (1)</i>	<i>Negative (0)</i>
<i>Positive (1)</i>	TP	FN
<i>Negative (0)</i>	FP	TN

The performance of a deep neural network can be evaluated by several performance measures, such as precision, recall, accuracy and F-score [99]. Such metrics can be computed as:

Accuracy: The accuracy is the number of correctly classified data samples, out of all the data samples in each dataset.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + FP}, \text{ where} \quad (2.19)$$

True Positive (TP): is the number of correct predictions when the actual class is positive.

True Negative (TN): is the number of correct predictions when the actual class is negative.

False Positive (FP): is the number of incorrect predictions when the actual class is positive.

False Negative (FN): is the number of incorrect predictions when the actual class is negative.

Precision: Utilized to calculate the positive patterns that are correctly predicted by all predicted patterns in a positive class and it tells us how good the model is at predicting a specific class.

$$Precision = \frac{TP}{TP+FP} \quad (2.20)$$

Sensitivity or Recall: Utilized to calculate the fraction of positive patterns that are correctly classified and tells us how many times the model was able to detect a specific class. It is also called the True Positive Rate (TPR).

$$Recall = \frac{TP}{TP+FN} \quad (2.21)$$

F1-Score: Calculates the harmonic average between recall and precision rates.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.22)$$

To apply such metrics in the multi-class classification, those metrics could be computed for each class. Usually, we need to compute the confusion matrix as illustrated in Table 2.2 for each class $c_i \in \mathcal{C} = \{1, \dots, K\}$. For each class c_i , the i -th class is considered as positive, while the rest of other classes as a negative class.

Then, to summarize the performance of the classifier on all classes, metrics can be micro or macro averaged [99]. The use of micro or macro averaging is dependent on the particular use case. In the following formulas, we will use TP_i , FP_i , and FN_i as the true positive, false positive, and false negative rates associated with the class i . Specifically, there are 3 averaging techniques applicable to multiclass classification namely macro, micro and weighted averaging.

Micro-average: It is the same as accuracy. Micro-averaging is found by dividing the sum of the diagonal cells of the matrix by the sum of all the cells. Micro-averaging is preferred in case of class imbalance present in the data.

Micro-averaged precision, recall and F1-score metrics are computed as:

$$Precision_{\text{micro}} = \frac{\sum_{i=1}^C TP_i}{\sum_{i=1}^C TP_i + FP_i} \quad (2.23)$$

$$Recall_{\text{micro}} = \frac{\sum_{i=1}^C TP_i}{\sum_{i=1}^C TP_i + FN_i} \quad (2.24)$$

$$F1_Score_{\text{micro}} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \quad (2.25)$$

Macro-average: Macro-averaging is based on the computation of precision, recall and F1-score for each class and then averaging the overall metrics:

$$Precision_{\text{macro}} = \frac{\sum_{i=1}^C Precision_i}{C} \quad (2.26)$$

$$Recall_{\text{macro}} = \frac{\sum_{i=1}^C Recall_i}{C} \quad (2.27)$$

$$F1_Score_{\text{macro}} = \frac{\sum_{i=1}^C F1_score_i}{C}, \text{ where } C \text{ represents the number of classes} \quad (2.28)$$

Weighted-Average Precision: The weighted average precision is the sum of the number of support multiplied by the precision of individual class divided by the total number of samples. The total number of samples will be the sum of all the individual samples.

$$Weighted_{\text{average precision}} = \frac{\sum_{i=1}^C w_i \times Precision_i}{\sum_{i=1}^C w_i} \quad (2.29)$$

where w_i is represents sample (support) weight of each class.

Weighted Average Recall: The weighted average recall is computed by multiplying the precision of each class and multiply them with their sample size and divide it by the total number of samples. The total number of samples will be the sum of all the individual samples.

$$Weighted_{\text{average recall}} = \frac{\sum_{i=1}^C w_i \times Recall_i}{\sum_{i=1}^C w_i} \quad (2.30)$$

Weighted Average F1-score: It is calculated by taking the mean of all per-class F1-scores while considering each class's support. Support is the number of actual occurrences of the class in the test dataset. Imbalanced support in the training data may indicate the need for stratified sampling or rebalancing.

$$Weighted_{\text{average F1-score}} = \frac{\sum_{i=1}^C w_i \times F1_score_i}{\sum_{i=1}^C w_i} \quad (2.31)$$

2.7. Related Works

Automatic text-based language identification has been researched since the 1960s [100]. Several researchers have attempted to solve LID problem using different modeling and classification techniques for different languages. Previous work that is related to this study; applied Cumulative Frequency Addition (CFA) and Naïve Bayes Classifier (NBC) to explore a comparative study of automatic LID on four Ethio-Semitic languages, namely Amharic, Tigrigna, Geez and Guragigna [21]. This work done to solve the factors that determine classification accuracy in short size of the textual string using N-gram as feature set. This research work utilizes characters-based N-gram of maximum length of 5 characters features to develop the language detection models.

In their system, authors employed an experimental study to measure the performance CFA and NBC classifiers. F1-score used as an optimal measure of performance for comparing classifiers performances. The average F1-score across the different length test strings 97.71% and 97.67 for CFA and NBC respectively. The researcher suggests that CFA classifier performs a better classifier both theoretically and practically therefore, this research can be extended to include other Ethio-Semitic languages that use the Ethiopic characters as their system of writing [21].

Another prior study which is conducted by Kidist Ergetie, presented a general purpose LID for four Ethio-Semitic languages (Amharic, Geez, Guragigna and Tigrigna) using hybrid approach that combines rule based with machine learning both on short and long documents written using Ethiopic [22]. For training and testing purpose 27 MB data from different sources (news, bible and books) were used. In order to observe the capability of the proposed language identifier they used a character N-gram model in two different forms: fixed character N-gram size (optimal maximal length N-grams empirically from 2 up to length 5) and infinity or all character N-gram size (depends on a word length, maximum at $N = \text{word length}$).

Besides, Naïve Bayes as classification method and Precision, Recall and F-score used as performance measures. The highest F-score performance obtained for the Naive Bayes classifier on infinity N-gram language model was 88.75% considering location feature and 86.67% without considering the location feature. The experimental result show that the infinity N-gram based approach is outperforming than fixed size character N-gram approach for language identification task at both monolingual as well as multilingual document setting [22].

In addition to this, Biruk Tadesse has provided a text-based language identification for seven Ethiopian languages namely Afar, Amharic, Nuer, Oromo, Sidamo, Somali and Tigrigna [19]. To solve the factors that affecting accuracy such as the size and variety of training data and the size of the string has been investigated. The corpus data collected from various sources such as newspapers, TV news, Religious books, academic books and dictionaries. The researcher utilized N-gram language modeling and identification algorithm Naïve Bayes, SVM classifier and Dictionary Method.

Naïve Bayes and SVM classifiers trained by using character N-gram of size 3 as a feature set. The dictionary method uses stop-words [19]. The 3-gram Naïve Bayes classifier, 3-gram SVM classifier and the dictionary method showed an average classification accuracy of 98.37%, 99.53% and 90.53% respectively. Higher error rates were noticed on the smallest character window for all classifiers. Finally, they recommended the use of a classification approach combined with linguistic initiative features such as POS and morphological information to provide concrete evidence of language types in terms of lexicon, orthography, morphology, and syntax.

In another work, Legesse Wedajo focused on developing a textual-based language identification model for five Ethiopian Cushitic languages namely Afaan Oromo, Afar, Sidama and Somali by using character N-gram as a feature set and Naïve Bayes and Frequency rank order classification algorithms [20]. The corpus for the study was collected from sources such as TV news websites, Bible, news bulletins, government documents, and documents from ministry of education to insure the corpus spans various domains. For test string of size 15 characters an accuracy of 99.55% on character N-gram feature set and 99.78% on character N-gram and its location in a word feature set was achieved for Naïve Bayes classifier.

The results showed that Naïve Bayes classifier achieved highest accuracy for short, medium and long string test documents. The identification accuracy of frequency rank order is low but showed an improvement for short text test documents. When using character N-gram and its location frequency as feature set, the accuracy of the both models showed an improvement. The result of the research can be used by anyone who has interest to develop spell checker for these languages and the researcher recommends that the performance of the models can be improved by using parallel corpus [20].

In a more recent work, language identification model is developed for five typologically and phylogenetically related low-resourced East African languages that use the Ge'ez script as a writing system; namely Amharic, Blin, Ge'ez, Tigre, and Tigrinya [4]. The datasets compiled from news sources of diverse websites, Bibles of the Ethiopian and Eritrean Orthodox Churches and the existing datasets. They integrated the dataset into an existing open-source language-identification tool, langdetect by creating profiles for the five languages and then evaluate the performance of the underlying Naive Bayes model that uses character N -grams (the frequency counts of unigrams, bigrams, and trigrams of characters) as features.

Furthermore, they also fine-tune five pre-trained language models (PLMs) of varying sizes (by the number of parameters), architectures, and training data. Overall, the evaluated models perform very competitively on the task, with F1-scores ranging between 98.28% and 99.90%. The experimental results show that the integrated langdetect can reach 99.90% in F1-score. In comparison, the approach of langdetect with Geez-Switch was able to outperform all the large language models, though not by a large boundary. Finally, the researchers suggest that as future work, the dataset can be extended to include other languages written in the Ge'ez script that was left out in the current version due to the lack of known data sources.

A number of studies are being conducted on European and Asian languages to test the effectiveness of statistical approaches to language identification, the researchers presented a language identification model to discriminate similar languages [101]. They applied a technique that combines word vectors representation and Long Short-Term Memory (LSTM) recurrent neural networks. For the evaluation of the implemented system, they used the Wikipedia text corpus for Serbian and Croatian languages, Voice of America website for Persian and Dari languages, discriminating between similar language (DSL) shared task datasets for Bulgarian and Macedonian languages and TweetID dataset that contains tweets written in Catalan and Spanish languages. The results show that enough effective and accurate to discriminate similar languages, even when it is applied to short texts. For the experimental evaluation on public and well-known datasets has shown that the proposed method improves accuracy and precision of language identification tasks [101]. Finally, they suggest several modifications that could be tested to improve the proposed method, such as other feature extraction techniques, other neural network-based classifiers, or more datasets that could be used. The summary of related works is shown in Table 2.3 underneath.

2.7.1. Summary of Related Works

Table 2. 3 Summary of related works

<i>Author, Year</i>	<i>Title</i>	<i>Techniques</i>	<i>Result</i>	<i>Recommendations</i>
Rediet Bekele (2018)	Comparative Study of Automatic Language Identification of Ethio-Semitic Languages, namely Amharic, Tigrigna, Geez and Guragigna	CFA and Naïve Bayes classifier algorithm using characters N-gram of maximum length of 5 characters.	CFA scores 97.71% and NBC scores 97.67% of an average F-score, CFA classifier performs better as compared to NBC and the study shows that higher performance is achieved as test phrase length increases.	The researcher recommends to include other Ethio-Semitic languages that use the Ethiopic script as their writing system and other classification approaches such as ANN.
Kidist Ergetie (2017)	General Purpose Language Identification for Ethiopia Semitic Language using Hybrid Approach, namely Amharic, Geez, Guragigna and Tigrigna	Fixed character N-gram size (N=2 up to 5) and infinity N-gram size (N= 2 up to maximal length of word) as a feature set, to construct the models and Naïve Bayes for classifier.	The infinity N-gram based approach is outperforming than fixed size character N-gram approach.	The author indorses that to adopt the infinity N-gram-based approach for other classification tasks and performing a comparative study with other different machine learning classifiers.
Biruk Tadesse (2018)	Automatic Identification of Major Ethiopian Languages, namely Afar, Amharic, Nuer, Oromo, Sidamo,	Naïve Bayes, SVM classifier using character N-gram of size 3 and (15, 100 and 300-character window) and	The Naïve Bayes, SVM and dictionary methods (stopwords) achieved an average accuracy of 98.37%,	They offers that a way of classification algorithms combined with linguistically motivated features such as

	Somali and Tigrigna	dictionary method using stop-words.	99.53% and 90.53% respectively.	POS tags and morphological information.
Legesse Wedajo (2014)	Modeling Text Language Identification for Ethiopian Cushitic Languages, namely Afaan Oromo, Afar, Sidama and Somali	Character N-gram as a feature set (N=2 up to 5) and (15, 100, and 300 characters' window). Naïve Bayes and Frequency rank order as a classifier algorithm.	Naïve Bayes classifier achieved highest accuracy for short, medium and long string test documents.	The researcher advises that to improve the performance of the models by using a parallel corpus.
Fitsum Gaim et al. (2022)	Language Identification in Typologically Related Low-resourced East African Languages, namely Amharic, Blin, Ge'ez, Tigre and Tigrigna	Integrating the dataset into the langdetect tool and then evaluating it using Character N-grams (N=3) as features and Naive Bayes as a classifier algorithm.	The experimental results show that the integrated langdetect can reach 99.90% in F1-score.	They suggest that as future work, to extended the dataset to include other languages written in the Ge'ez script that was left out in the current version due to the lack of known data sources.
Ermelinda Oro et al. (2018)	Language Identification of Similar Languages using Recurrent Neural Networks, namely Serbian, Croatian, Persian, Dari, Bulgarian, Macedonian, Catalan and Spanish	Word vectors representation and Long Short-Term Memory (LSTM) recurrent neural networks	Proposed method obtains better results when compared with prior works considering same languages.	They suggest several modifications that could be tested to improve the proposed method, such as: Other feature extraction techniques, other neural network-based classifiers, or more datasets that could be used.

2.8. Research Gaps

In this subsection, the researchers present the research gaps identified in related works and literature reviews. As reported in previous related works, considering Ethiopian languages, most investigators have applied related classical machine learning classifiers and data representation techniques to solve language identification problems [4] [19] [20] [21] [22]. Naive Bayes classifiers prove to be quite popular and successful with high orders of N for the N-grams used and the use of SVMs also appears to be a popular choice for achieving good performance. However, the classical machine learning approaches reliance on the hand-crafted features requires tedious feature engineering and analysis to obtain good performance [65]. This is a well-recognized issue that necessitates a better approach. Deep learning approaches are proposed to address the limitations due to the use of hand-craft features [65].

To the best of our knowledge, there is no previous research work on LID using a deep learning approach with a character-level embedding technique for closely related Ethiopian languages that use the Ethiopic script as their writing system. However, nowadays, deep learning algorithms are the state-of-the-art for many natural language processing applications like LID and deep learning algorithms can achieved highest accuracy [83].

Although some attempts have been made to address LID problems, a series of recent studies have indicated that there is still a need to improve the performance of language detection for short texts and the previous research works focused on Ethio-Semitic language families [4] [21] [22], however, there are more than 15 languages that use the Ge'ez script in their writing system, including the Ethio-Semitic, Cushitic, and Omotic language families [5].

As previously described in the related works, most of the researchers applied word-based N-gram model [4] [19] [20] [21] [22], although, this approaches handle OOV words badly. The word-based language models suffer from the problem of sparsity and the word distribution has a long tail and many parameters are required to capture all the words in a corpus. For example, an embedding size of 300 with a vocabulary of 10k words, 3 million parameters are needed. It becomes complex to train machine learning models when the dataset has a large number of features. Although character N-grams should work better than word N-grams for LID, especially for languages with rich morphology, regular morphological features like suffixes and prefixes repeat much more often than whole words and can be representative of a language [23]. These are more likely to be learned during model training and be indicators when predicting the language for a new unseen text.

CHAPTER THREE

METHODOLOGY

3.1. Overview

This chapter extensively explores an overview of the selected research design, proposed system architecture, data sources, data collection techniques, data cleaning, tokenization, text representation technique, standardization technique, the proposed deep neural network model architecture, training algorithm, hyperparameters and performance evaluation techniques.

3.2. The Proposed Research Design

Each field of study has its own unique features that requires specific methods of investigation. Understanding these features helps researchers to use the most appropriate methodologies for research study. In most studies, the primary issue is choosing a research methodology based on the research problem and stated research questions. Accordingly, this proposed research follows an experimental research design which is one of the most common approaches in social science, computer science, information systems, software engineering and psychology [102]. Experimental research design is a framework of procedures created to conduct experimental research with a scientific approach using two sets of variables: independent and dependent variables. An experimental research is scientifically driven, quantitative research to find out the relationship between these variables [102][103].

The general structure for a quantitative research design is based on the scientific method and the results are presented numerically and statistically [103]. In experimental research design, the researcher actively attempts to change the datasets, scenarios, algorithms, or parameters. The experimental research design provides conclusions about cause and effect between the set of variables that make up a study. Therefore, in our study, at the end of the experiments, the researchers drew a conclusion about the performance of the proposed language identification model on short and long texts. After a research topic is selected and the problem is clearly formulated, an experimental design includes preparing the dataset, choosing applicable tools for the experiments, designing an experimental setup, conducting various experiments, evaluating the performances, analyzing the result of the experiment, report the findings and finally draw the conclusions [103].

During the experiment, the researcher can manipulate the independent variable to check its influence on dependent variables. In short, experimental research could be viewed as a test or series of tests that makes deliberate changes to the input variable of a process or system in order to observe and understand the effects of those changes on the output variables.

Independent variables are known as predictors, causes or input variables that are used to predict the value or outcome of dependent variables [103]. The dependent variables are called responses, effects, output variables or classes where the output of that variable depends on other variables [103]. Scientists are generally agreed that the most effective means of testing a prediction is deliberately to manipulate the independent variable and then to observe the consequential changes in the dependent variable [102] [103].

In machine learning, independent variables are called features and dependent variables are called class names. Therefore, in the case of this study, the textual data were marked as independent variables, while language names like Amharic, Awngi, Geez, Guragigna, Tigrigna and Xamtanga were categorized as dependent variables. The relationship between the various independent and dependent variables within a given system can be expressed mathematically as follows:

$$y = f(x) \tag{3.1}$$

where y denotes the dependent variable and x indicates the independent variables that affect the output when their values change.

3.3. The Proposed System Architecture

The general architecture of the proposed language identification model consists of corpus collection from various sources, data cleaning, tokenization, text representation for dependent and independent variables, data scaling, dataset splitting, building a deep neural network model, training the constructed model, and testing the model. The general architecture of the proposed language identification model is shown in Figure 3.1.

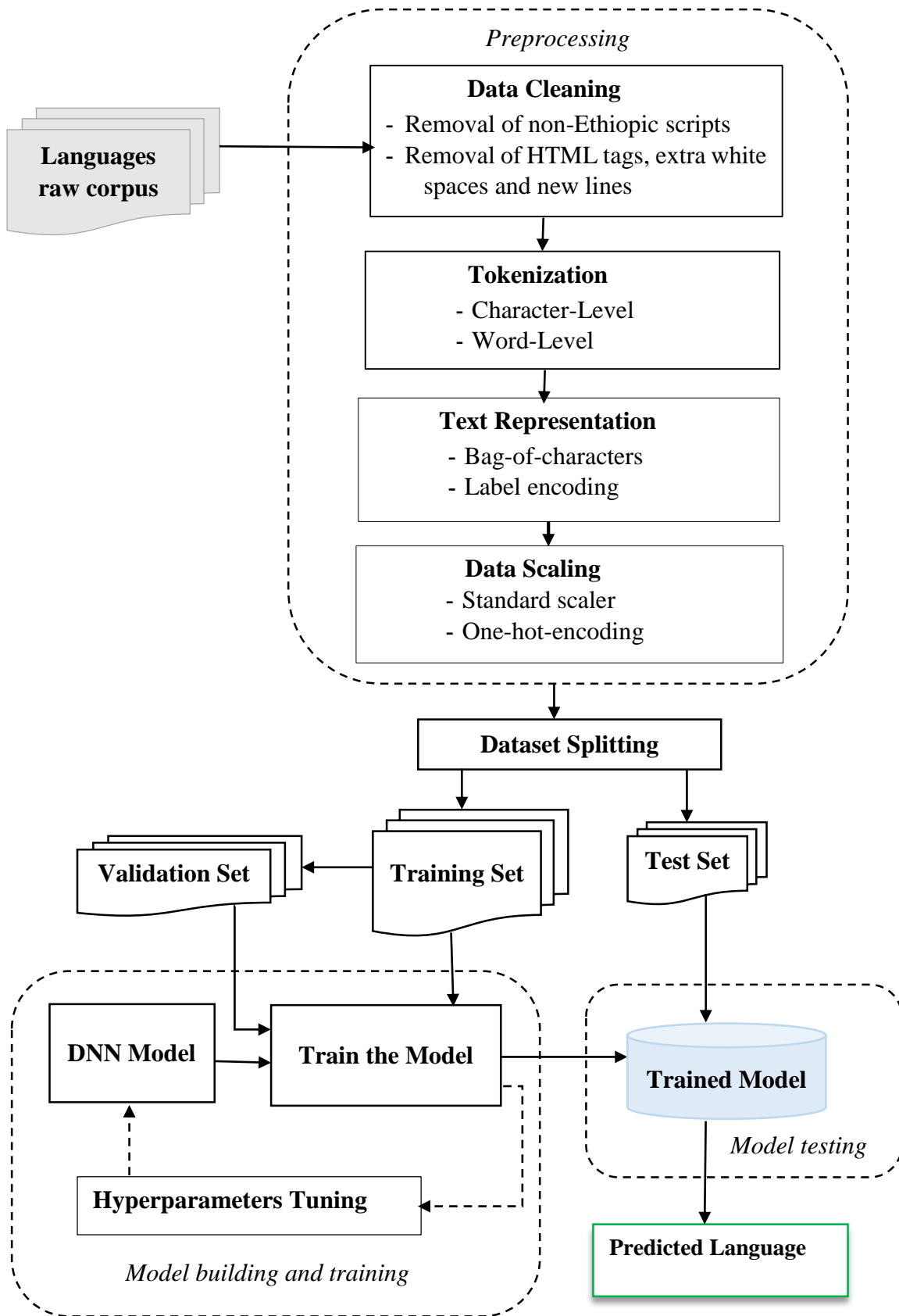


Figure 3. 1 General architecture of the proposed language identification model

As shown in Figure 3.1, the languages raw corpus is a set of noisy text files composed of the six Geez-based Ethiopian languages, namely Amharic, Awnji, Geez, Guragigna, Tigrigna, and Xamtanga. Each language corpus contains monolingual text, which means that each document file has only one language.

The preprocessing phase is an important step in preparing the text corpus in a machine learning or deep learning understandable format for model training and testing. In general, the preprocessing phase involves removing unhelpful parts of the corpora, converting the texts into a stream of words or characters, and converting texts into numbers to allow machines to understand and decode patterns within a language. In addition, this phase includes the standardization of the real-valued input and output variables to ensure that all features contribute equally to the model and to prevent features with larger values from dominating the model. Finally, the pre-processed text corpus is divided into training, validation and test sets.

The model building and training phase includes designing and building the proposed deep neural network (DNN) model architecture, such as the number of input layers, hidden layers, output layers, the number of nodes in each layer, and other hyperparameters. Once built, the model is trained on the specified training dataset with several hyperparameters tuning.

In the model testing phase, the performance of each trained model to correctly identify the languages is evaluated on the test dataset. Finally, using the best trained model, the unknown text language is identified and displayed during the prediction phase.

3.3.1. Data Collection

The WWW is a source of textual data written in different languages on webpages and social media platforms, hence the corpus for this study was collected from various sources. The Amharic¹⁰ and Tigrigna¹¹ corpus were obtained from Amhara Media Corporation (AMC), Ethiopian broadcasting corporation, Ethiopian press agency, Fana broadcasting corporation, Tigrai TV and Walta TV using their own websites, Facebook pages and Telegram channel.

¹⁰ Amharic data sources: <http://amharaweb.com/>, <https://t.me/EBCNEWSNOW>, <https://bit.ly/2wdOpiZ>, <https://t.me/WALTATVEth>, <https://t.me/fanatelevision>

¹¹ Tigrigna data sources: <https://t.me/tigrignafana>, <https://t.me/EPATigrigna>, <https://tigraitv.com/>, <https://www.facebook.com/AmharaMediaCorporationTigrigna>

The corpus for Awngi¹² and Xamtanga¹³ was collected from AMC websites, Facebook page and Telegram channel. The corpus for the Ge'ez¹⁴ and Guragigna¹⁵ language was collected from the digital Bibles and various Ethiopian Orthodox Tewahedo softcopy books and PDF files, hence the corpus spans multiple domains. The corpus was collected using an automated data collection tool called Beautiful Soup, but we also did a manual assessment to ensure the quality of the data. Beautiful Soup is a Python library web scraping tool used to collect data from various websites. In addition, we applied data collection technique such as document analysis and online interviews with the domain expert. The corpus we collected is monolingual text, meaning that each document file contains only one language.

3.3.2. Text Preprocessing

After the corpus has been collected, the next step is text preprocessing. Textual data derived from various sources is unstructured and noisy. A text preprocessing step transforms this raw corpus into a clean and coherent format before feeding it into a neural network for learning and further analysis. We have performed various text pre-processing procedures such as data cleaning, tokenization, text representation and feature standardization. To preprocess the data, we used NLTK and regular expressions python libraries.

3.3.2.1. Data Cleaning

The first step is to identify some reliable sources of content for each language. Many of the languages written with the Ethiopic script are small and not available online, which is the main reason we focus on the selected six languages. In this step, we performed text cleaning procedures such as removing non-Ethiopic alphanumeric scripts, punctuation marks, HTML tags, URLs, special characters, emoticons and extra newlines were removed, except for hyphen (-) that are commonly used to indicate multi-word expressions. For example, to write the word constitution “ህገ መንግሥት” in Amharic, a hyphen is commonly used between the two words like “ህገ-መንግሥት”.

¹² Awngi data sources: <https://bit.ly/2XEVIHq>, <https://bit.ly/3CudESE>, <https://bit.ly/3Av5GZ0>

¹³ Xamtanga data sources: <https://www.amharaweb.com/category/ኸምጠ-ዊከ/>, <https://t.me/Himtagne>, <https://t.me/AMCHimtagne>, <https://www.facebook.com/AmharaMediaCorporationHimtagne>

¹⁴ Ge'ez data sources: <https://www.ethiopicbible.com/>, <https://myorthodoxbooks.org/geez/>, <https://www.tau.ac.il/~hacohen/Biblia.html>, <http://mermru.com/bibles/>, <https://github.com/geezorg/data>

¹⁵ Guragigna data sources: http://gospelgo.com/f/gurage_chaha_nt.htm, <https://www.bible.com/en-GB/bible/3203/GEN.INTRO1.ጠቅጥ>, <https://allaboutethio.com/books/library63a3.pdf>, <https://live.bible.is/bible/SGWBSE/MAT/1>

Besides, the researchers removed extra spaces and replaced them with single space because we need to preserve single spaces between words as this is a useful feature set for training the neural network.

Algorithm 3. 1: Algorithm for data cleaning

Input: original corpora file

Output: cleaned file

- 1: **Open** and **read** original corpora file;
 - 2: Define *regular expression* for each noisy;
 - 3: **repeat:**
 - 4: **for each** lines **in** original corpora file **do:**
 - 5: cleaned file \leftarrow *remove* (non-Ethiopic alphanumeric characters, extra white space, extra newlines, URLs, HTML tags);
 - 6: increment lines;
 - 7: **end for;**
 - 8: **until** successfully cleaned the corpus;
 - 9: **return** cleaned file;
-

Each language document is represented in both machine-readable and human-interpretable form in text files with UTF-8 encoding. In general, as shown in Table 3.1, the corpus consists of different sizes for each language after the data cleaning.

Table 3. 1 The corpus size of each language

<i>Languages</i>	<i>Corpus size in megabytes</i>
<i>Amharic</i>	27.58 MB
<i>Awngi</i>	3.94 MB
<i>Geez</i>	10.68 MB
<i>Guragigna</i>	5.77 MB
<i>Tigrigna</i>	24.58 MB
<i>Xamtanga</i>	6.31 MB

3.3.2.2. *Tokenization*

Tokenization is a way of breaking a piece of text into smaller units called tokens. With tokenization, the documents are broken down into words, sub-words (N-grams) or characters. Finally, these token occurrences in a document can be used directly as a vector representing that document. When tokenizing text into words, spaces within a string are used as delimiters of words, and when tokenizing text into characters, spaces are treated as characters.

Algorithm 3. 2: Algorithm for word tokenization

Input: cleaned_corpus_file

Output: tokenized_words

- 1: Open and read the cleaned_corpora_file;
 - 2: **for each** words **in** cleaned_corpus_file **do**:
 - 3: tokenized_words \leftarrow length (cleaned_corpus_file) and split by (space);
 - 4: word_token += total length of (tokenized_words);
 - 5: word_type += unique words of (tokenized_words);
 - 6: increment words;
 - 7: **end for**;
 - 8: **return** word_token, word_type;
-

In this study, we implemented character tokenization to allow the tokenization process to retain information about out of vocabulary words that word tokenization cannot. Instead of breaking text into words, it completely separates text into characters. For example, the character tokenization vocabulary for Amharic would have about 283 characters as shown in Table 3.2.

Algorithm 3. 3: Algorithm for character tokenization

Input: cleaned_corpus_file

Output: tokenized_characters

- 1: Open and read the cleaned_corpora_file;
 - 2: **for each** chars **in** cleaned_corpus_file **do**:
 - 3: tokenized_characters \leftarrow length (cleaned_corpus_file);
 - 4: total_characters += total length of (tokenized_characters);
 - 5: unique_characters += unique words of (tokenized_characters);
 - 6: increment chars;
 - 7: **end for**;
 - 8: **return** total_characters, unique_characters;
-

Table 3.2 presents the total number of individual words (word tokens), number of unique words (word types), total number of characters and average length of the words in the corpus. In addition, Table 3.3 shows total number of predefined alpha-syllabic, special characters, numerals and unique symbols of the six Geez-based languages that are we used for training the neural network.

Table 3. 2 Word and character distributions of each language in the corpus

Languages	Word tokens	Word types	Number of characters	Average word length
<i>Amharic</i>	2053939	186769	10562730	5.14 characters
<i>Awngi</i>	294687	40479	1508861	5.12 characters
<i>Geez</i>	844574	105508	4122801	4.88 characters
<i>Guragigna</i>	489104	70070	2248174	4.6 characters
<i>Tigrigna</i>	2001818	124432	9527454	4.76 characters
<i>Xamtanga</i>	486621	68537	2426550	4.99 characters

Table 3. 3 The alphanumeric and special characters distribution of the six languages

Languages	Numbers of predefined alpha-syllabics	Total number of special characters and numerals	Total number of unique symbols
<i>Amharic</i>	283	22 symbols (base Ge'ez numbers, single space and hyphen)	405
<i>Awngi</i>	200		
<i>Geez</i>	202		
<i>Guragigna</i>	357		
<i>Tigrigna</i>	276		
<i>Xamtanga</i>	259		

The alpha-syllabic of each language is depicted in Appendix II, V, VI, VII, VIII and IX. There are several different ways of lexical similarity metrics that can be used for quantifying the similarity between text units (words or characters) such as Jaccard, Cosine and Euclidian similarity for this study, we used Jaccard similarity measures. The Jaccard measure works quite well in practice, especially for sparse data [104]. The Jaccard similarity is defined as an intersection of two documents divided by the union of that two documents. The value of Jaccard similarity of two documents ranges from 0 to 1, where 0 indicates no similarity and 1 signifies complete overlap. The mathematical representation of the Jaccard similarity is shown below:

$$J(doc_1, doc_2) = \frac{|doc_1 \cap doc_2|}{|doc_1 \cup doc_2|} \quad (3.2)$$

For example, consider the following two languages to calculate the similarity between them at the word-level.

Amharic = “የዓለም ሁሉ መድሃኒት ዛሬ ተወለደ”

Tigrigna = “ናይ ዓለም ኩሎም መድሃኒት ሎሚ ተወለዱ”

The intersection word between the languages: {'መድሃኒት'}, so one word is common. The union words in the languages: {'የዓለም', 'ሁሉ', 'መድሃኒት', 'ተወሊዱ', 'ኩሎም', 'ናይ', 'ዛሬ', 'ዓለም', 'ተወለደ', 'ሎሚ'}. Totally, there are 10 union words. Hence, the Jaccard similarity is $1/10 = 0.1$

As shown in Figure 3.2, the dataset analysis between the six closely related Ethiopian languages, namely Amharic (AMH), Awngi (AWN), Geez (GEZ), Guragigna (SGW), Tigrigna (TIR) and Xamtanga (XAN). The report revealed that Awngi and Xamtanga are very similar at the character level because they are grouped in a similar language family. There are some similarities between Amharic and Tigrigna at the word level. In general, the analysis showed more similarity at the character-level than at the word-level. This is because the all languages use Geez script as a writing system. Texts are more similar the more their words or characters overlap.

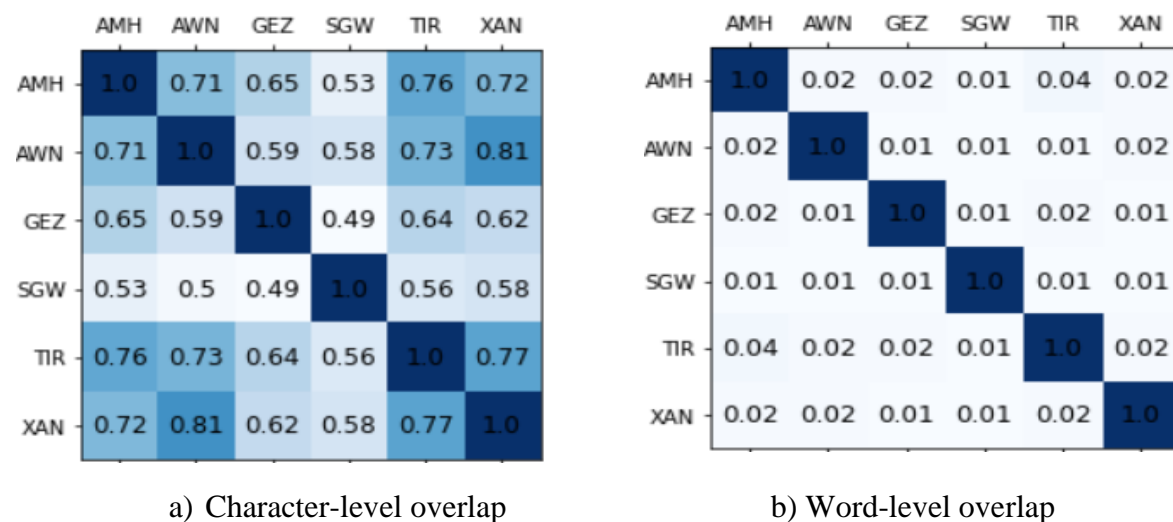


Figure 3. 2 Lexical similarity between the Geez-based languages at character and word-level

3.3.3. Text Representation

As stated in the literature review in subsection 2.6.1, there are various vectorization techniques. In this study, we applied the BoC technique to convert raw text into a numeric representation by mapping each document character to a feature vector. BoC use the letter frequency approach to represent texts as vectors of characters. The fact that character level models have proven to be the correct method for a lot of NLP tasks because of their effectiveness [105] [106]. We chose this method for the following three reasons: The first reason, with the character embedding, each individual word's to be formed as a vector, even if they are OOV words that is capable of generating novel words [23].

After training, they can predict a vector for any word, not just words that they have seen before. The second reason is that it handles infrequent words better than word-based and N-gram based, since one later suffers from a lack of sufficient training facility for these rare words. The third reason is that the complexity of the model is reduced due to the small amount of dictionaries. It becomes complex to train machine learning models when the dataset has a greater number of features. Less the features, the better the performance of the model. The following algorithm describes the proposed chars2vec model.

Algorithm 3. 4: Algorithm for bag-of-characters model

Input: Cleaned corpus file

Output: Vectorized file

```

1: Define each language alphabets: alphabet
2: Define maximum length of characters: MAX_LEN
3: with open (Cleaned corpus file) as file:
4:     content = read (Cleaned corpus file)
5:     random_index = random-randrange (0, length (content))
6:     define get_sample_text (content, start_index, MAX_LEN):
7:         sample_text = get_sample_text (content, random_index, MAX_LEN)
8:         return sample_text;
9:     define count_characters (sample_text, alphabet):
10:        predefined_alphabet_counts = []
11:        for alphabet in sample_text do:
12:            count = sample_text-count(alphabet)
13:            predefined_alphabet_counts-append(count)
14:        return predefined_alphabet_counts;
15:    define get_input_row (content, start_index, MAX_LEN, alphabet):
16:        sample_texts = get_input_row (content, random_index, MAX_LEN, alphabet)
17:        bag_of_characters = count_characters (sample_texts, alphabet)
18:        return bag_of_characters;

```

Bag-of-characters contain two elements to construct the vectors: unique alpha-syllables of the languages and a measure of the occurrence of predefined alphabets in a given corpus. The list of unique symbols is the union of the six Geez-based languages without repeating letters. Only the Ethiopic alphanumeric characters and some special characters are reserved to train the neural network. Other special characters such as full stops and semicolons are not considered as they are the same in all the languages and hence, they do not provide any additional information for the classification task. We have constructed a total of 405 unique symbols that will be used as input data to train the neural network model.

The machine learning or deep learning models cannot work on categorical variables in the form of strings, therefore we need to change it into numerical form. Therefore, we have converted each language name to an integer representation as shown in Table 3.4.

Table 3. 4 Label encoding of each language

<i>Class Name</i>	<i>Numeric Value</i>
Amharic	0
Awngi	1
Geez	2
Guragigna	3
Tigrigna	4
Xamtanga	5

3.3.4. Data Scaling

Although the dataset is converted to numeric form, it is unscaled. This is a significant factor given the use of small weights to train the neural network model. Differences in scales between input variables can increase the difficulty of training the model. A neural network model with large vector values can result in extremely slow training [107]. Therefore, data scaling is a recommended preprocessing step when working with neural networks. There are several data or feature scaling techniques such as min-max scaling, standard scaling, max-absolute scaling and robust scaling [107].

For this study, we used the standard or z-score data scaling technique because the dataset should follow a normal distribution and is recommended for optimization algorithms such as gradient descent [108]. With standard scaling, the shape of the dataset distribution is fixed, so in this case the vector values are not restricted to a certain range, while with min-max scaling, the shape of the dataset distribution can be changed. Standardization is the transformation of the feature value by subtracting it from the mean and dividing it by the standard deviation. The equation is presented as follows:

$$X_{new} = \frac{X - mean(X)}{standard\ deviation(X)} \quad (3.3)$$

where $mean(X)$ is the mean of the vector values and $standard\ deviation(X)$ is the standard deviation of the vector values.

The researchers also implement a one-hot-encoding word embedding technique to convert the integer representation of each label into the machine-understandable format it is explained in detail in Chapter two. This technique transforms each categorical integer codes to one new feature of integers (0 to N_categories – 1) as presented in Table 3.5.

Table 3. 5 One-hot-encoding of each language

<i>Integer Values</i>	<i>One-hot-encoding Values</i>
0	[1, 0, 0, 0, 0, 0]
1	[0, 1, 0, 0, 0, 0]
2	[0, 0, 1, 0, 0, 0]
3	[0, 0, 0, 1, 0, 0]
4	[0, 0, 0, 0, 1, 0]
5	[0, 0, 0, 0, 0, 1]

The following snippets show the sample text before data scaling (Figure 3.5) and after data scaling (Figure 3.6) where X indicates the chars2vec representation of the text and Y represents the one-hot-encoding values of classes.

```
X :
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  2.  0.  1.  0.  1.  0.  0.  0.  0.  0.
 0.  0.  0.  4.  0.  0.  2.  1.  5.  0.  0.  3.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  3.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  3.  0.  1.
 0.  0.  3.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  3.  0.  1.  1.
 6.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  2.  0.  1.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  2.
 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  2.  0.  0.  1.
 2.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  2.  0.  5.  0.
 0.  0.  0.  0.  0.  2.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  2.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  2.  0.  0.  1.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  23. 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]

Y :
3.0
```

Figure 3. 5 Sample data before data scaling

```

X :
[-2.41089091e-01 -3.25914115e-01 -1.52497917e-01 -3.33205104e-01
-1.88196376e-01 -4.12551790e-01 -2.89046228e-01 -7.75824606e-01
-4.33525354e-01 -5.08921981e-01 -7.35993803e-01 -4.05893326e-01
-1.01411879e+00 -4.51962113e-01 -7.60871172e-02 -2.66283572e-01
-1.51985466e-01 -1.18451528e-01 -3.24163735e-01 -2.69409508e-01
-4.57972348e-01 -1.23884961e-01 -2.58199754e-03 -9.46200728e-01
-3.92300665e-01 -5.69173872e-01 1.07724655e+00 -4.04790819e-01
-7.73923397e-01 -3.19123805e-01 5.81164074e+00 -1.83934495e-01
-6.04246855e-02 -4.17418554e-02 -1.19735755e-01 -5.50995097e-02
-2.71541059e-01 -5.58359176e-02 0.00000000e+00 2.33579016e+00
-4.30055976e-01 -5.50386131e-01 1.35644090e+00 1.41551566e+00
1.47739124e+00 -4.80583340e-01 -6.51853979e-02 1.86649644e+00
-2.95178175e-01 -3.78668278e-01 -5.82718372e-01 -3.36939484e-01
-1.15102100e+00 -2.50942200e-01 -4.46412303e-02 -3.84764284e-01
-1.32477716e-01 -2.11083412e-01 -2.51512885e-01 -1.55601844e-01
-4.90230411e-01 -1.63311094e-01 -1.26178991e-02 -5.08785844e-01
-2.98827231e-01 -1.92602962e-01 -3.87387395e-01 -1.04977086e-01
3.46121287e+00 -2.21399188e-01 -2.00947270e-01 -1.66372538e-01
-1.49840400e-01 -4.57111076e-02 -7.27500096e-02 -2.74385870e-01
-1.78027779e-01 -1.58469781e-01 -2.10224494e-01 -6.49472624e-02
-3.31426948e-01 -8.94089639e-02 -8.44748840e-02 -4.38672081e-02
-6.53250068e-02 -6.32468145e-03 -1.80784259e-02 1.06323922e+00
-3.53520840e-01 1.30574965e+00 -7.46034324e-01 -4.54778016e-01
8.17254841e-01 -3.77265543e-01 -1.28076702e-01 -2.56442666e-01
-4.07869704e-02 -1.55160740e-01 -1.57270148e-01 -6.61063567e-02
-2.82750160e-01 -2.73254625e-02 -1.18167205e-02 9.98716652e-01
-5.06028771e-01 5.03258348e-01 -6.38433024e-02 -4.44074690e-01
1.73171151e+00 -4.46614176e-01 -9.31873098e-02 -3.43081892e-01
0.00000000e+00 0.00000000e+00 0.00000000e+00 2.14683580e+00
-1.75557751e-02 -4.86400872e-02 -2.98956241e-02 -3.71002257e-02
-3.13333534e-02 -3.15700695e-02 -1.99654233e-02 -3.21150050e-02
-1.98201686e-02 -2.10146978e-02 -3.59988548e-02 -2.93054860e-02
-2.49696895e-02 -1.90048739e-02 -2.17458792e-02 -1.49505697e-02
-1.78681482e-02 -9.31441039e-03 -8.41295999e-03 -3.10373437e-02
-2.09291764e-02]

Y :
[0. 0. 0. 1. 0. 0.]

```

Figure 3. 6 Sample data after applying data scaling technique

3.3.5. Dataset Splitting

Imbalanced dataset typically refers to a problem with classification problems where the classes are not represented equally. Before partitioning the dataset, we performing dataset balancing. As previously shown in Table 3.1, we have unbalanced dataset sizes for the six Geez-switch languages, so we randomly selected 100,000 samples text from each language to solve this problem. Therefore, each of the six languages contributes an equal number of samples, which are then combined and a grand total of $600k \times 406$ samples have been prepared for training, validation, and testing.

Table 3. 6 Distribution of sample text dataset

Languages	Sample text per language	Total sample text
<i>Amharic</i>	100,000	600,000
<i>Awngi</i>	100,000	
<i>Geez</i>	100,000	
<i>Guragigna</i>	100,000	
<i>Tigrigna</i>	100,000	
<i>Xamtanga</i>	100,000	

To ensure that the test and train splits have the same ratio of class ratio for training classification models, we have applied the stratification technique. Stratified sampling for splitting a dataset alleviates the problem of random sampling in datasets with an imbalanced class distribution.

In addition, we have randomly shuffled the sample dataset which serves the purpose of preventing any bias during training, making sure that models remain general, preventing the model from learning the order of the training, and ensuring the model is not overfitting to certain pattern duo sort order.

Once we have prepared the sample dataset, the dataset is first divided into two halves, training and the test set. The validation set is then separate from the training set, which is used to validate the model performance during training and provide an unbiased evaluation of a model fit to the training dataset while tuning the model hyperparameters. The training set is a dataset that we employ to train the parameters of the neural network model. In each epoch, the same training data is repeatedly fed into the neural network architecture and the model continues to learn the features of the data. The test set is unseen dataset that play an important role for in the unbiased evaluation of the final model in terms of accuracy, precision, recall and F1-score. As shown in Table 3.7, we used 70% of the sample data for training, 10% for validation, and 20% for testing.

Table 3. 7 The distribution of dataset splitting

<i>Training</i>	432,000 (432k)
<i>Validation</i>	48,000 (48k)
<i>Testing</i>	120,000 (120k)
<i>Total</i>	600,000 (600k)

3.3.6. The Proposed Deep Neural Network Model

In this study, we have used state-of-the-art technology, which is a deep learning approach, to develop a language identification model. We chose this approach for the following basic reasons: Deep learning approaches extract the features by itself without much human intervention, so this hand-off approach allows algorithms to quickly adapt to the data. Deep learning models have achieved unprecedented success, approaching human-level performances when trained on large amount of labeled data [109]. Therefore, one of the deep learning models, a deep multilayer perceptron, also known as a deep neural network with multiple hyperparameter tuning applied for the proposed language identification model.

The proposed deep neural network model architecture consists of an input layer that receives input data, an output layer that makes a prediction about the input text, and three hidden layers between these two with different number of nodes, so we can call it a three-layer deep neural network model architecture, as shown in Figure 3.7.

```

Model: "sequential"

```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	207872
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 6)	774

```

=====
Total params: 372,870
Trainable params: 372,870
Non-trainable params: 0

```

Figure 3. 7 Summary of the proposed deep neural network model architecture

From the previous subsection, we see that the chars2vec embedding gets 406 features (input dimensions), including class names. We have prepared $480,000 \times 406$ training dataset including 10% of validation set. A large number of nodes in each layer can increase the capacity of the network to learn complex patterns in the data, resulting in better performance on the training set [110]. The researchers set the number of layers and nodes depends on the size of the dataset and the computational resources available.

Therefore, for the proposed neural network model, the input layer has 406 input dimensions that connected to 512 the first hidden layer nodes. All three layers are fully connected or dense layer. Next, we used 256 for the second hidden layer nodes. Then another 128 for the third hidden layer nodes and 6 neurons in the output layer for each of the six languages. As shown summary report of the proposed deep neural network model architecture in Figure 3.7, there are a total of 372,870 trainable parameters and zero non-trainable parameters out of 372,870.

Mathematically, we can calculate the total number of parameters as follows: The number of weights between the input and the first hidden layer is 207,872 parameters, which means multiplying the input size and the number of nodes for the first hidden layer (406×512). Number of weights between hidden layer one and two plus the number of biases for hidden layer two: $512 \times 256 + 256 = 131,328$ number of parameters. Number of weights between the second hidden layer and the third layer plus the number of biases for the third hidden layer: $256 \times 128 + 128 = 32,896$ number of parameters. Finally, the number of weights between the hidden layer three and the output layer plus the number of biases for the output layer 128 has $6 + 6 = 774$ number of parameters. Therefore, a total of 372,870 ($207,872 + 131,328 + 32,896 + 774$) parameters are available to train the proposed neural network model.

3.3.6.1. Hyperparameter Setting

Hyperparameter tuning is the process of determining the best set of hyperparameters [110]. Hyperparameters are those parameters that are explicitly defined by the user to control the learning process. The basic hyperparameters used in this paper are dropout ratio, optimization algorithms, regularization mechanism, activation functions, loss function, number of epochs and batch size. We tried several experiments by setting up these hyperparameters to different values. Finally, the combination of hyperparameters that yields highest accuracy is selected for the final LID model.

To control the overfitting problem, we used a regularization mechanism called dropout with different ratios. Dropout is one of the most effective and commonly used techniques to prevent overfitting in neural networks [97]. While building and training neural networks, it is crucial to initialize the weights appropriately to ensure a model with high accuracy. If the weights are not correctly initialized, it may give rise to the Vanishing Gradient problem or the Exploding Gradient problem [93]. Therefore, to address this issue, we used Xavier Glorot prior to training the model, because it is a good practice to use this method when using the sigmoid activation function in hidden layers [93].

The researchers also used sigmoid activation functions in the hidden layers and Softmax activation function in the output layer to produce the estimated result of target variables. When training a neural network, the main goal is to minimize the loss function, which describes the distance between the correct label of the language text and the predicted label given by the neural network model. The loss function we used was related to the Softmax activation function, which is recommended for multi-class classification problems with so-called categorical cross-entropy.

Another key aspect in designing a neural network infrastructure is selecting the right optimization algorithm, hence instead of traditional optimization techniques (random and grid search). The researchers used three automated gradient-based hyperparameter optimization through reversible learning [94][110]. These are Adam, RMSprop and AdaGrad with their default values. The optimization algorithm is one of the main hyperparameter to train the model faster and get a good result [94]. The default parameter values of Adam optimizer is: $\beta_1 = 0.9$, $\beta_2 = 0.999$, learning rate = 0.001 and 10^{-8} for the epsilon [94] [95]. The optimizer controls the learning rate. When comparing Adam to other methods, its advantage is faster convergence and training speed of the model is quiet fast and gives better performance. Algorithm 3. 5 shows a detailed description of the Adam optimization algorithm.

Algorithm 3. 5: Algorithm for Adam optimizer

Require: α : Step size (Leaning rate)

Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates

Require: $f(\theta)$: Stochastic objective function with parameters θ

Require: θ_0 : Initial parameter vector

- 1: Initialize 1st moment vector: $m_0 \leftarrow 0$
 - 2: Initialize 2nd moment vector: $v_0 \leftarrow 0$
 - 3: Initialize time step: $t \leftarrow 0$
 - 4: **while** θ_t not covered **do:**
 - 5: $t \leftarrow t + 1$
 - 6: Get gradients with respect to stochastic objective at time step t : $g_t \leftarrow \Delta_{\theta} f_t(\theta_{t-1})$
 - 7: Update biased first moment estimate: $m_t \leftarrow \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$
 - 8: Update biased second raw moment estimate: $v_t \leftarrow \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$
 - 9: Compute bias-corrected first moment estimate: $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
 - 10: Compute bias-corrected second raw moment estimate: $\tilde{v}_t \leftarrow v_t / (1 - \beta_2^t)$
 - 11: Update parameters: $\theta_t \leftarrow \theta_{t-1} - \alpha * \hat{m}_t / (\sqrt{\tilde{v}_t} + \epsilon)$
 - 12: **end while;**
 - 13: **return** θ_t (Resulting parameters)
-

where g_t^2 indicates the elementwise square $g_t * g_t$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t .

Adaptive gradient or AdaGrad, works on the learning rate component by dividing the learning rate by the square root, which is the cumulative sum of current and past squared gradients [95], [96]. The second optimization algorithm we used was AdaGrad with default parameter values: learning rate = 0.01 and 10^{-7} for the epsilon. The following algorithm describes the AdaGrad optimizer.

Algorithm 3. 6: Algorithm for AdaGrad optimizer

Require: Global learning rate η

Require: Initial parameter θ

- 1: Initialize gradient accumulation variable $\gamma = 0$
 - 2: **while** stopping criterion not met **do:**
 - 3: Sample a minibatch of m examples from the training set $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ with corresponding targets $\mathbf{y}^{(i)}$.
 - 4: Set $g = 0$
 - 5: **for** $i = 1$ to m **do:**
 - 6: Compute gradient: $g \leftarrow g + \nabla_{\theta} L(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)}; \theta)$.
 - 7: **end for**
 - 8: Accumulate squared gradient: $\gamma \leftarrow \gamma + \mathbf{g}^2$ (square is applied element-wise)
 - 9: Compute update: $\Delta\theta \leftarrow -\frac{\eta}{\sqrt{\gamma}} g$ (Division and square root applied element-wise)
 - 10: Apply update: $\theta \leftarrow \theta + \Delta\theta$
 - 11: **end while;**
-

RMSprop is another adaptive learning rate that is an improvement of AdaGrad [95] [111]. Instead of taking cumulative sum of squared gradients like in AdaGrad, we take the exponential moving average of these gradients. RMSprop solves the vanishing gradient problem by using a moving average of squared gradients to normalize the gradient. This normalization balances the step size (momentum) by decreasing the step for large gradients to avoid exploding and increasing the step for small gradients to avoid vanishing. The third optimization algorithm we used was RMSprop with default parameter values: $\beta = 0.9$, learning rate = 0.001 and 10^{-6} for the epsilon. The following algorithm defines the RMSprop optimizer.

Algorithm 3. 7: Algorithm for RMSprop optimizer

Require: Global learning rate η , decay rate p

Require: Initial parameter θ

- 1: Initialize gradient accumulation variable $\gamma = 0$
 - 2: **while** stopping criterion not met **do:**
 - 3: Sample a mini-batch of m examples from the training set $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ with corresponding targets $\mathbf{y}^{(i)}$.
 - 4: Set $g = 0$
 - 5: **for** $i = 1$ to m **do:**
 - 6: Compute gradient: $g \leftarrow g + \nabla_{\theta} L(f(\mathbf{x}^{(i)}; \theta), \mathbf{y}^{(i)}; \theta)$.
 - 7: **end for**
 - 8: Accumulate squared gradient: $\gamma \leftarrow p\gamma + (1 - p)\mathbf{g}^2$
 - 9: Compute parameter update: $\Delta\theta \leftarrow -\frac{\eta}{\sqrt{\gamma}} g$ (square root is applied element-wise)
 - 10: Apply update: $\theta \leftarrow \theta + \Delta\theta$
 - 11: **end while;**
-

3.3.7. Model Training

Once we constructed the deep neural network model architecture, the training process is done using a labeled training set with different hyperparameters setting. Multilayer Perceptrons are trained through a method called backpropagation [110]. Backpropagation is an algorithm that back propagates the errors from output nodes to the input nodes [112].

Gradient descent is an optimization algorithm used for minimizing the cost function in various machine learning algorithms. Stochastic gradient descent is an optimization algorithm for finding the model parameters that correspond to the best fit between predicted and actual outputs [94].

Backpropagation is a special case of automatic differentiation applied to neural networks because reversing learning saves memory and can optimize thousands of hyperparameters [110] [112]. The point of backpropagation is to fine-tune the weights of a network based on the errors obtained in the previous epochs. The backpropagation algorithm looks for the minimum value of the error function in weight space using the method of gradient descent [91] [110].

Generally, each training iteration of neural network has three main stages: feed forward of the input training data, backward propagation of the associated error and modification of weights.

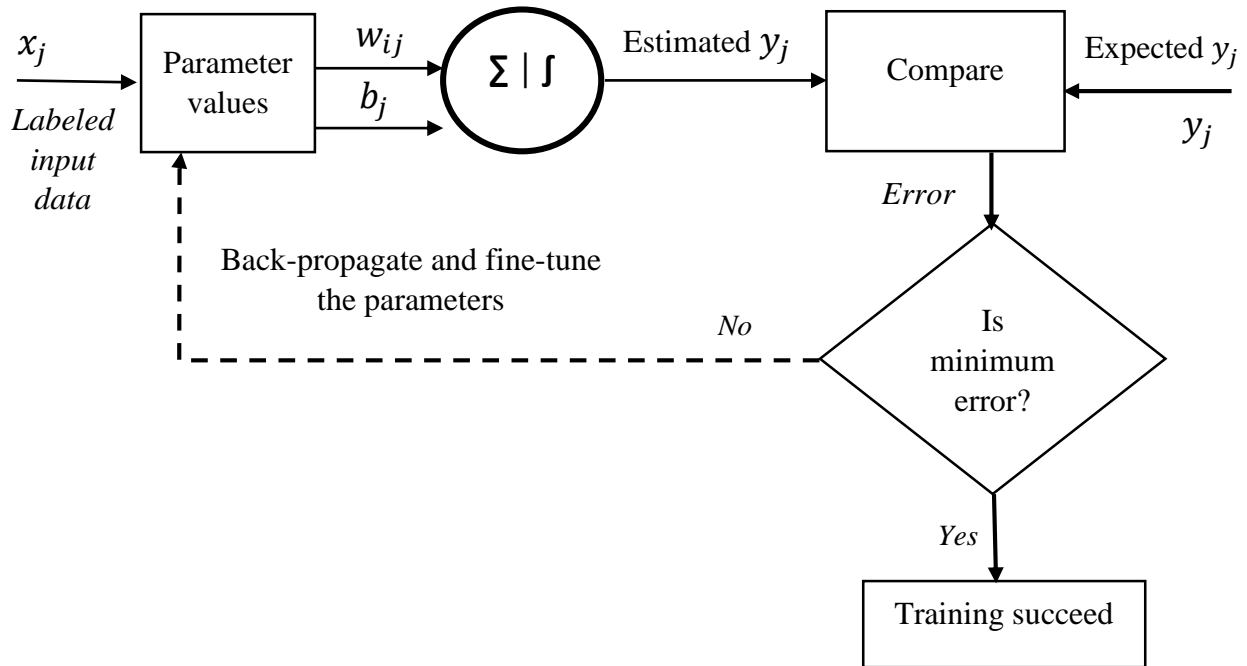


Figure 3. 8 The proposed DNN model training process with backpropagation algorithm

The backpropagation training algorithm procedures are summarized as follows:

- 1) Initialize with values for the network parameters (w_{ij} weights and b_j biases). Where, i indicates the value for neurons in the preceding layer, and j is the values for neurons in the next layer. Weights and biases values are usually chosen randomly. Present the input and desired output.
- 2) Take samples of input data and propagate them through the network.
- 3) Compute the output of each neuron from the input layer to the hidden layer to the output layer.
- 4) Compare these predictions obtained with the values of expected labels and calculate the error between the expected and the estimated output.

$$\text{Error} = \text{Expected Output} - \text{Desired Output}$$

- 5) If the error is huge then, perform backpropagation with the gradient descent and update all parameters in order to minimize this error. After that, check the error again.
- 6) Keep repeating the previous steps until the error is minimized and the desired output is achieved.

The following pseudo-code describes the backpropagation algorithm for training the model.

Algorithm 3. 8: Backpropagation algorithm for training the proposed DNN model

```

1: function Back-Propagation-Learning(examples, network)
2:   inputs examples: a set of examples, each with input vector  $\mathbf{x}$  and output vector  $\mathbf{y}$ 
3:     network: a multilayer network with  $L$  layers, weights  $w_{ij}$ , biases  $b_j$ , activation function  $g$ 
4:   local variables:  $\Delta$  a vector of errors indexed by network node
5:   repeat
6:     for each weight  $w_{ij}$  in network do
7:        $w_{ij} \leftarrow$  a small random number, typically between -1 and 1
8:     end for;
9:     for each example  $(\mathbf{x}, \mathbf{y})$  in examples do
10:      // Propagate the inputs forward through the network to compute the outputs
11:      for each node  $i$  in the input layer do
12:         $a_i \leftarrow x_i$ 
13:      end for
14:      for  $l = 2$  to  $L$  do
15:        for each node  $j$  in layer  $l$  do
16:           $in_j \leftarrow \sum_i w_{ij} a_i + b_j$ 
17:           $a_j \leftarrow g(in_j)$ 
18:        end for
19:      end for
20:      // Propagate deltas(errors) backward from output layer to input layer
21:      for each node  $j$  in the output layer do
22:         $\Delta[j] \leftarrow g'(in_j) \times (y_j - a_j)$ 
23:      end for
24:      for  $l = L - 1$  to  $1$  do
25:        for each node  $i$  in layer  $l$  do
26:           $\Delta[i] \leftarrow g'(in_i) \sum_j w_{ij} \Delta[j]$ 
27:        end for
28:      end for
29:      // Update every weight in network using deltas
30:      for each weight  $w_{ij}$  in network do
31:         $w_{ij} \leftarrow w_{ij} + b_j \times a_i \times \Delta[j]$ 
32:      end for
33:    end for
34:  until minimum error is reached to the specified value
35:  return network

```

3.3.8. Model Evaluation

During the training process, we evaluated the model using the two most well-known metrics, accuracy and loss. Accuracy is a method to measure the performance of classification models, usually expressed as a percentage [98] [99]. A loss function, also known as a cost function, takes into account the probabilities or uncertainty of a prediction based on how much the prediction varies from the true value. Unlike accuracy, loss is not a percentage, it is a summation of the errors made for each sample in training or validation sets.

Loss is often used in the training process to find the best parameter values for the model because during the training process the goal is to minimize this value [98] [99]. One of the most widely used combinations of metrics is training and validation loss over time. The training loss is a metric used to evaluate how a deep learning model fits the training data. On the contrary, validation loss is a metric used to assess the performance of a deep learning model on the validation set.

After the candidate models have been trained with the available corpus, the classification performance of the trained model is evaluated on the test set using various evaluation metrics such as accuracy, precision, recall, F1-score, micro average of precision, micro average of recall, micro average of F1-score, macro-average of precision, macro-average of recall, macro-average of F1-score, weighted-average of precision, weighted-average of recall and weighted-average of F1-score which are explained in detail in Chapter Two, subsection 2.6.3. In the model training process, the model may be biased against the training set as well as the validation set. Consequently, we have evaluated the trained model on 120K unseen test data.

CHAPTER FOUR

EXPERIMENTAL RESULTS AND DISCUSSION

4.1. Overview

This chapter outlines the experimental analysis of the study. The experimental setup used to train and test the proposed model are all discussed. Several experiments have been conducted to evaluate the performance of our proposed method and the discussion of each experimental result is briefly clarified, as well as the main findings of the study.

4.2. Experimental Setup

To train and test the proposed language identification model, we first setup the experiment environments such as hardware and software tools, datasets, and hyperparameters.

4.2.1. Hardware and Software Tools

We have used the two main tools hardware and software to successfully carry out this study. Software tools such as Microsoft Windows 11 Pro with 64-bit operating system and Python 3.8 with Jupyter Notebook 6.3.0 and Anaconda Navigator version 3. Python is a programming language that suits well for machine learning tasks, and has many well developed libraries for that specific purpose. Hardware devices like HP desktop computer with Intel(R) Core (TM) i7-8700 @ CPU 3.20 GHz, 8 GB physical memory and 1 TB hard disk for corpus preparation, pre-processing, to conduct the experiments and for writing the thesis report.

Additionally, when building machine learning models, especially deep neural networks, it is essential to use a Python library that handles specific computations. Python programming is chosen for experimentation because it is easy to use, has an abundant community support, offers a variety of visualization options, and has many built-in packages for specific tasks. There are several free, open-source Python libraries available today. The most common libraries we used in this study include Keras and TensorFlow to build deep neural network models that can run on the CPU [113]. NumPy to perform mathematical operations on multidimensional arrays and matrices. Scikit-learn to compare, validate, select parameters and models in predictive data analysis. Pandas for data manipulation and analysis, Matplotlib for experimental result visualization in the form of graphs and bar charts and Ipywidgets to develop a simple user interface for the proposed model and make a prediction [113].

4.2.2. Dataset

The dataset is the main one to train the proposed deep neural network model. As described in Table 3.7, the researchers have set a total of 600k sample texts with 406 features for training, validation, and testing. Of this, 70% of the dataset is used for training the model, 10% for validation when tuning model hyperparameters to find and optimize the best model, and 20% for testing the trained model (i.e. 432k texts for training, 48k texts for validation and 120k for testing the model).

4.3. Experiments and Discussion of Results to Select Best Hyperparameter Set

The proposed model consists of several hyperparameters that affect the model's behavior and performance in different way. There are several ways to select the best hyperparameters set, a common approach is to try different hyperparameters and see what happens. A better approach is to objectively look for different values for model hyperparameters and choose a subset that results in a model that performs best for a given dataset. We have also applied this approach to the proposed study. In this subsection, the researchers conducted three main different experiments with different hyperparameter settings to select the optimal hyperparameter set. Depending on each test performed, other parameters changed. In order to select the best hyperparameter sets all the experiments were conducted with the same text length of 100 characters. A comparison of all the different test results is also presented in this subsection. Finally, the hyperparameter sets with the highest score were selected for further experiments.

4.3.1. Experiment One Using AdaGrad Optimizer

The first experiment done using deep neural network model with chars2vec embedding, AdaGrad optimization algorithm and other parameters as shown in Table 4.1.

Table 4. 1 Hyperparameters for experiment one

Hyperparameters	Values
Weight initializer	glorot_uniform
Activation function for hidden layers	Sigmoid
Epoch	5, 10 and 15
Batch size	16, 32 and 64
Dropout ratio	0.3, 0.4 and 0.5
Activation function for output layer	Softmax
Optimization algorithm	AdaGrad with default values
Loss function	Categorical cross-entropy

The proposed DNN model is trained with the specified training dataset and finally, the identification performance of the trained model is evaluated with the test set by applying the hyperparameters described in Table 4.1. As shown below in Table 4.2, the test accuracy results are 99.61%, 99.62%, and 99.55% at trial 1, trial 2, and trial 3, respectively, using different batch sizes, epochs and dropout values, and the test accuracy result decreased in the third trial. Loss is measured as cross-entropy normalized by its maximum value, while training accuracy is measured by the number of correctly identified training samples at the end of each epoch. The test loss had a loss score of 0.0286, 0.0230, and 0.0289 in trial 1, trial 2, and trial 3 respectively, and the loss result increased in the third trial. In these experiments with three hyperparameter value changes and tests, the higher test accuracy of 99.62% and less test loss result of 0.0230 is achieved with a hyperparameter set of AdaGrad optimizer, a batch size of 32, a dropout rate of 0.4 and an epoch of 10.

```

Epoch 1/10
13500/13500 - 19s - loss: 1.7685 - accuracy: 0.2426 - val_loss: 1.5152 - val_accuracy: 0.9762
Epoch 2/10
13500/13500 - 18s - loss: 1.1710 - accuracy: 0.6357 - val_loss: 0.5380 - val_accuracy: 0.9907
Epoch 3/10
13500/13500 - 21s - loss: 0.4486 - accuracy: 0.9219 - val_loss: 0.1519 - val_accuracy: 0.9936
Epoch 4/10
13500/13500 - 20s - loss: 0.1997 - accuracy: 0.9755 - val_loss: 0.0700 - val_accuracy: 0.9949
Epoch 5/10
13500/13500 - 18s - loss: 0.1219 - accuracy: 0.9851 - val_loss: 0.0460 - val_accuracy: 0.9954
Epoch 6/10
13500/13500 - 17s - loss: 0.0890 - accuracy: 0.9888 - val_loss: 0.0362 - val_accuracy: 0.9955
Epoch 7/10
13500/13500 - 16s - loss: 0.0719 - accuracy: 0.9903 - val_loss: 0.0311 - val_accuracy: 0.9957
Epoch 8/10
13500/13500 - 18s - loss: 0.0613 - accuracy: 0.9912 - val_loss: 0.0281 - val_accuracy: 0.9960
Epoch 9/10
13500/13500 - 18s - loss: 0.0546 - accuracy: 0.9921 - val_loss: 0.0261 - val_accuracy: 0.9961
Epoch 10/10
13500/13500 - 18s - loss: 0.0492 - accuracy: 0.9926 - val_loss: 0.0246 - val_accuracy: 0.9962

```

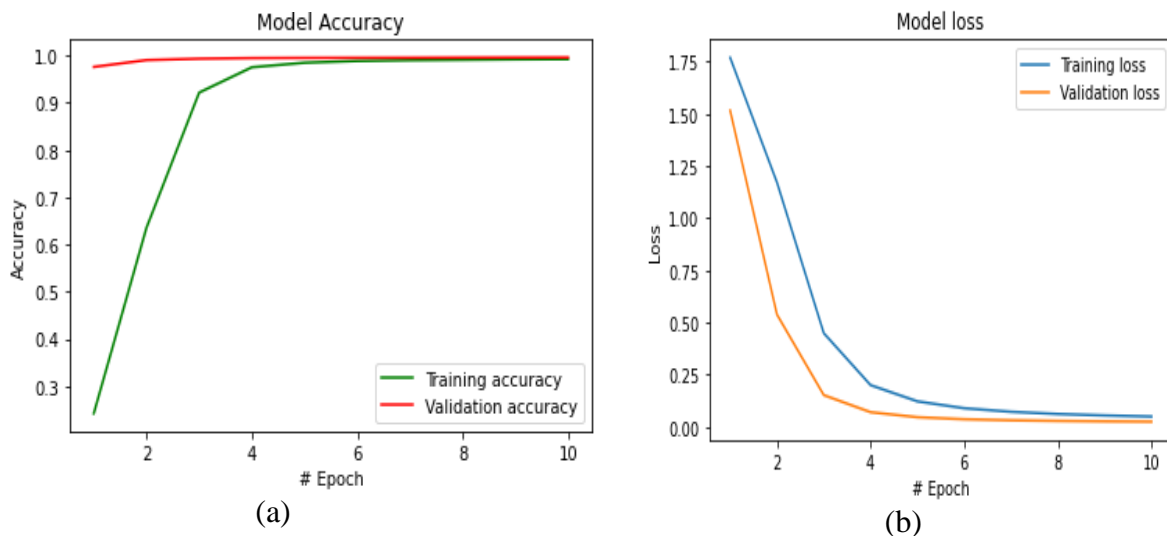


Figure 4. 1 Learning curves of accuracy and loss over 10 epochs with the AdaGrad optimizer

The learning curve of training accuracy and validation accuracy (a), training loss and validation loss (b) of the best hyperparameter sets with AdaGrad optimizer are shown in Figure 4.1. The plot shows that both training and validation accuracy increase almost linearly and stabilize after epoch 4. In addition, the training loss and validation loss both decrease slowly and stabilize at the last training iteration. Table 4.2 describes the experimental results of the model that we have done for experiment one.

Table 4. 2 A summary of experimental results using AdaGrad optimizer

<i>Exp.</i>	<i>Batch Size</i>	<i>Epoch</i>	<i>Dropout Ratio</i>	<i>Optimizer</i>	<i>Test Accuracy</i>	<i>Test Loss</i>
Run1	16	5	0.3	AdaGrad	99.61%	0.0286
Run2	32	10	0.4	AdaGrad	99.62%	0.0230
Run3	64	15	0.5	AdaGrad	99.55%	0.0289

4.3.2. Experiment Two Using Adam Optimizer

This experiment is conducted by considering the proposed DNN model with chars2vec embedding, Adam optimizer and other parameters. The various parameters and values for this experiment are shown in Table 4.3. These are the ones that were chosen to be tested.

Table 4. 3 Hyperparameters for experiment two

<i>Hyperparameters</i>	<i>Values</i>
Weight initializer	glorot_uniform
Activation function for hidden layers	Sigmoid
Epoch	5, 10 and 15
Batch size	16, 32 and 64
Dropout ratio	0.3, 0.4 and 0.5
Activation function for output layer	Softmax
Optimization algorithm	Adam with default values
Loss function	Categorical cross-entropy

In this scenario, the model was trained and evaluated with three different epochs: 5, 10, and 15. As a result, the model with epoch 15 performs well. Second, we tried batch sizes 16, 32 and 64 and found promising results with batch size of 64 and finally experimenting with dropout ratios of 0.3, 0.4 and 0.5; and we got a promising result with a dropout rate of 0.5.

The model proposed in this experiment is trained on the specified training dataset by applying the hyperparameters described in Table 4.3, and finally the language identification performance of this trained model is evaluated on test data. The test accuracy results are 99.89%, 99.91%, and 99.91% at trial 1, trial 2, and trial 3, respectively using different batch sizes, epochs and

dropout values and the test accuracy result is stable at second experiment and third experiment. Besides, the test loss achieved a loss score of 0.0053, 0.0051, and 0.0048 in trial 1, trial 2, and trial 3 respectively, and the loss result decreased slowly at all tests. This yields a more trustworthy result. After three hyperparameters value changes and tests, the highest test accuracy of 99.91% which means it does predict a large part of new data samples correctly and less test loss result of 0.0048 is scored with a hyperparameters set of Adam optimizer, a batch size of 64, a dropout ratio of 0.5 and an epoch of 15 as shown below in Table 4.4. We have shown that the training accuracy of the model improves as the number of epochs increases.

```

Epoch 1/15
6750/6750 - 16s - loss: 0.0431 - accuracy: 0.9885 - val_loss: 0.0093 - val_accuracy: 0.9979
Epoch 2/15
6750/6750 - 16s - loss: 0.0132 - accuracy: 0.9970 - val_loss: 0.0070 - val_accuracy: 0.9981
Epoch 3/15
6750/6750 - 15s - loss: 0.0100 - accuracy: 0.9976 - val_loss: 0.0062 - val_accuracy: 0.9984
Epoch 4/15
6750/6750 - 14s - loss: 0.0086 - accuracy: 0.9979 - val_loss: 0.0053 - val_accuracy: 0.9986
Epoch 5/15
6750/6750 - 14s - loss: 0.0070 - accuracy: 0.9983 - val_loss: 0.0046 - val_accuracy: 0.9989
Epoch 6/15
6750/6750 - 15s - loss: 0.0062 - accuracy: 0.9985 - val_loss: 0.0045 - val_accuracy: 0.9990
Epoch 7/15
6750/6750 - 15s - loss: 0.0056 - accuracy: 0.9986 - val_loss: 0.0039 - val_accuracy: 0.9991
Epoch 8/15
6750/6750 - 16s - loss: 0.0053 - accuracy: 0.9988 - val_loss: 0.0041 - val_accuracy: 0.9991
Epoch 9/15
6750/6750 - 17s - loss: 0.0047 - accuracy: 0.9990 - val_loss: 0.0041 - val_accuracy: 0.9991
Epoch 10/15
6750/6750 - 15s - loss: 0.0045 - accuracy: 0.9990 - val_loss: 0.0040 - val_accuracy: 0.9990
Epoch 11/15
6750/6750 - 15s - loss: 0.0041 - accuracy: 0.9991 - val_loss: 0.0041 - val_accuracy: 0.9993
Epoch 12/15
6750/6750 - 15s - loss: 0.0040 - accuracy: 0.9992 - val_loss: 0.0039 - val_accuracy: 0.9993
Epoch 13/15
6750/6750 - 16s - loss: 0.0039 - accuracy: 0.9992 - val_loss: 0.0039 - val_accuracy: 0.9992
Epoch 14/15
6750/6750 - 16s - loss: 0.0035 - accuracy: 0.9992 - val_loss: 0.0038 - val_accuracy: 0.9994
Epoch 15/15
6750/6750 - 14s - loss: 0.0035 - accuracy: 0.9993 - val_loss: 0.0039 - val_accuracy: 0.9992

```

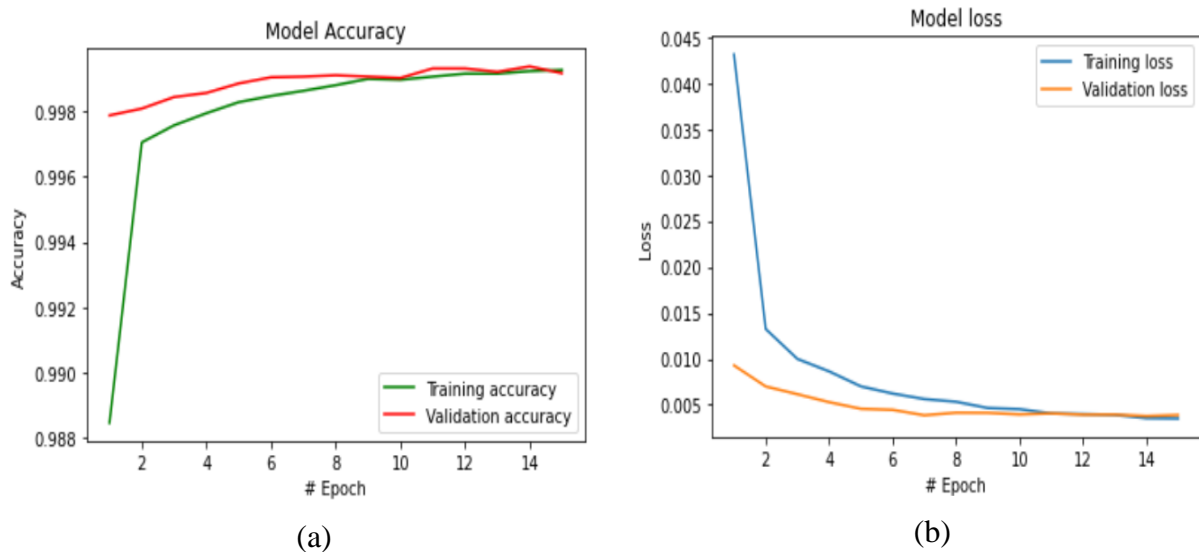


Figure 4. 2 Learning curves of accuracy and loss over 15 epochs with the Adam optimizer

The plot of training accuracy and validation accuracy (a), training loss and validation loss (b) of the best hyperparameter sets with Adam optimizer are shown in Figure 4.2. The learning curve shows that the training and validation accuracy increases almost linearly and stabilizes throughout the curve. The loss after the 10th epoch is not reduced much because the model has already learned enough parameters to classify. The training history of the model and it looks like the training and validation both accuracy and loss lie close to each other, meaning the model is not overfitted. This yields a more trustworthy result.

In addition, the training loss and validation loss both decreases slowly and constantly throughout the curve. As we can see in the diagram, the loss on the training set decreases rapidly for the first two epochs. For the test set, the loss does not decrease at the same rate as the training set, but remains almost flat for multiple epochs. This means our model is generalizing well to unseen data. The learning curves shows that the model is good fitted. Because the plot of validation loss decreases to a point of stability and has a small gap with the training loss. Table 4.4 describes the experimental results of the model that we have done for experiment two.

Table 4. 4 A summary of experimental results using Adam optimizer

Exp.	Batch Size	Epoch	Dropout Ratio	Optimizer	Test Accuracy	Test Loss
Run1	16	5	0.3	Adam	99.89%	0.0053
Run2	32	10	0.4	Adam	99.91%	0.0051
Run3	64	15	0.5	Adam	99.91%	0.0048

4.3.3. Experiment Three Using RMSprop Optimizer

The third experiment is conducted the same way using the proposed deep MLP model architecture and chars2vec embedding, but uses the RMSprop optimization algorithm and other different parameters as shown in Table 4.5.

Table 4. 5 Hyperparameters for experiment three

<i>Hyperparameters</i>	<i>Values</i>
Weight initializer	glorot_uniform
Activation function for hidden layers	Sigmoid
Epoch	5, 10 and 15
Batch size	16, 32 and 64
Dropout ratio	0.3, 0.4 and 0.5
Activation function for output layer	Softmax
Optimization algorithm	RMSprop with default values
Loss function	Categorical cross-entropy

In this experiment, the model is trained on the training dataset by applying the hyperparameters described in Table 4.5, and finally the language identification performance of this trained model is evaluated on new unseen data. The test accuracy results are 99.72%, 99.73%, and 99.76% in test 1, test 2, and test 3, respectively, and the accuracy increases as the number of epochs increases as we progress from experiment 1 to experiment 3 go through. In addition, the test loss was a loss score of 0.0331, 0.0270, and 0.0235 in trial 1, trial 2, and trial 3 respectively, and the loss result gradually decreased in all tests. After three hyperparameter value changes and experiments, the highest test accuracy of 99.76% and the result with less test loss of 0.0235 using RMSprop optimizer, a batch size of 64, a dropout rate of 0.5, and an epoch of 15 as shown in the summary of results of Table 4.6.

```

Epoch 1/15
6750/6750 - 16s - loss: 0.0431 - accuracy: 0.9885 - val_loss: 0.0093 - val_accuracy: 0.9979
Epoch 2/15
6750/6750 - 16s - loss: 0.0132 - accuracy: 0.9970 - val_loss: 0.0070 - val_accuracy: 0.9981
Epoch 3/15
6750/6750 - 15s - loss: 0.0100 - accuracy: 0.9976 - val_loss: 0.0062 - val_accuracy: 0.9984
Epoch 4/15
6750/6750 - 14s - loss: 0.0086 - accuracy: 0.9979 - val_loss: 0.0053 - val_accuracy: 0.9986
Epoch 5/15
6750/6750 - 14s - loss: 0.0070 - accuracy: 0.9983 - val_loss: 0.0046 - val_accuracy: 0.9989
Epoch 6/15
6750/6750 - 15s - loss: 0.0062 - accuracy: 0.9985 - val_loss: 0.0045 - val_accuracy: 0.9990
Epoch 7/15
6750/6750 - 15s - loss: 0.0056 - accuracy: 0.9986 - val_loss: 0.0039 - val_accuracy: 0.9991
Epoch 8/15
6750/6750 - 16s - loss: 0.0053 - accuracy: 0.9988 - val_loss: 0.0041 - val_accuracy: 0.9991
Epoch 9/15
6750/6750 - 17s - loss: 0.0047 - accuracy: 0.9990 - val_loss: 0.0041 - val_accuracy: 0.9991
Epoch 10/15
6750/6750 - 15s - loss: 0.0045 - accuracy: 0.9990 - val_loss: 0.0040 - val_accuracy: 0.9990
Epoch 11/15
6750/6750 - 15s - loss: 0.0041 - accuracy: 0.9991 - val_loss: 0.0041 - val_accuracy: 0.9993
Epoch 12/15
6750/6750 - 15s - loss: 0.0040 - accuracy: 0.9992 - val_loss: 0.0039 - val_accuracy: 0.9993
Epoch 13/15
6750/6750 - 16s - loss: 0.0039 - accuracy: 0.9992 - val_loss: 0.0039 - val_accuracy: 0.9992
Epoch 14/15
6750/6750 - 16s - loss: 0.0035 - accuracy: 0.9992 - val_loss: 0.0038 - val_accuracy: 0.9994
Epoch 15/15
6750/6750 - 14s - loss: 0.0035 - accuracy: 0.9993 - val_loss: 0.0039 - val_accuracy: 0.9992

```

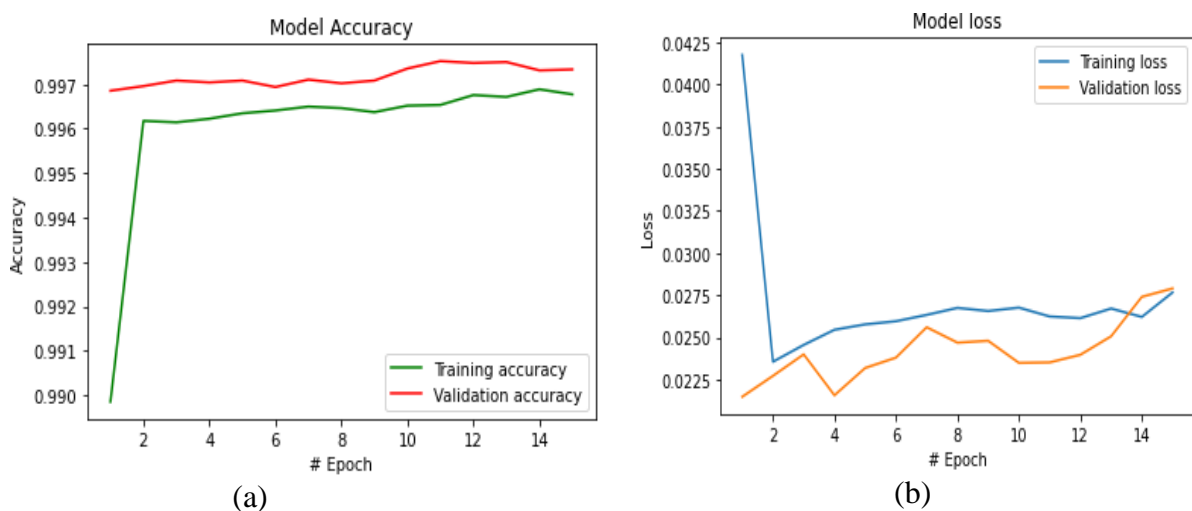


Figure 4. 3 Learning curves of accuracy and loss over 15 epochs with the RMSprop optimizer

The learning curve of training accuracy and validation accuracy (a), training loss and validation loss (b) of the best hyperparameter sets with RMSprop optimizer are shown in Figure 4.3. The learning curve shows that the training and validation accuracy increases almost linearly and stabilizes throughout the curve. Both training and validation loss oscillations occur across the curve and are not stable. Table 4.6 describes the experimental results of the model that we have done for experiment three.

Table 4. 6 A summary of experimental results using RMSprop optimizer

Exp.	Batch Size	Epoch	Dropout Ratio	Optimizer	Test Accuracy	Test Loss
Run1	16	5	0.3	RMSprop	99.72%	0.0331
Run2	32	10	0.4	RMSprop	99.73%	0.0270
Run3	64	15	0.5	RMSprop	99.76%	0.0235

4.3.4. Summary of the Experiments

The above experiments aim to select the best hyperparameters set which are used to evaluate the performance of the proposed language detection model with varying sample text lengths. All experiments were performed with 432K training datasets, 48K validation datasets and 120K test datasets. The experiments were done with different hyperparameter settings. The first experiment was implemented using the AdaGrad optimizer, the second using Adam optimizer, and the third using RMSprop optimizer with similar hyperparameter values.

We used a maximum of 15 epochs for all experiments to train the models as this can show stable trends in accuracy. Although higher performance can be obtained by increasing the number of epochs, this consumes more computation time. To select the well-fitted model, we evaluated each trained model against an unseen dataset and the experimental results show that the Adam optimizer achieved a better accuracy of 99.91% and a lower test error of 0.0048 compared to the other optimizers with a batch size of 64, a dropout ratio of 0.5, and an epoch of 15.

A good fit is characterized by a minimal gap between the training and validation loss values that decrease to a stable level, so it is true in our Experiment Two. The model in Experiment Two has achieved great accuracy with a low loss value, so the result indicates that the goal of reducing loss has been achieved. We can conclude that the model fits well since the plot of training loss and validation loss decreases to a point of stability and the generalization gap is minimal, and lower loss indicates higher model performance.

Therefore, the hyperparameter sets shown in Table 4.7 are more suitable and selected for the proposed final language identification model.

Table 4. 7 The chosen hyperparameter settings

<i>Hyperparameters</i>	<i>Values</i>
Weight initializer	glorot_uniform
Activation function for hidden layers	Sigmoid
Number of epochs	15
Batch size	64
Dropout ratio	0.5
Activation function for output layer	Softmax
Optimizer	Adam with default values
Loss function	Categorical cross-entropy

4.4. Model Evaluation Using Different Text Lengths

As discussed before in the experimental results, the Adam optimizer achieved better classification accuracy than the AdaGrad and RMSprop optimizers with the same hyperparameter values for the proposed model. Accordingly, the following four experiments are conducted to evaluate the performance of the proposed model for the given short and long text by applying the best hyperparameter setting described in Table 4.7.

The length of the text of the corpus varies according to the field and source we have collected. The corpus we collected is a mixture of short, medium and long texts or sentences. Therefore, it is necessary to analyze the performance of the model on short and long texts by conducting different experiments by varying the text length.

We can measure sentence length in units of words or characters, for this study the researchers used character length. In this study, the corpus distribution report revealed that the approximate average word length of all languages is about 5 characters, as shown in Table 3.2. Accordingly, to assess the performance of the proposed model for the given short and long texts, we started testing the model with a sample text length of 5 characters per row and then randomly tested it with 10, 50 and 100 characters of the sample text. The experimental result and discussion of each experiment are detailed below in subsections 4.4.1, 4.4.2, 4.4.3, 4.4.4 and 4.4.5.

4.4.1. Experiment One Using Text Length of 5 Characters

In accordance with the corpus distribution of the average word length of the six Geez-based languages, we first evaluated the proposed deep neural network model by training it on a sample text length of 5 characters, and the model achieved an accuracy of 77.68% and a loss of 0.5862 for the test set. The experimental result showed that the model obtained many incorrect language classifications for short texts because all languages use the same writing system and have many words in common.

Confusion matrix is a very popular measure used in solving multi-class classification problems, which represents the number of actual and predicted values to perfectly analyze the potential of a classifier. The evaluation scores for Amharic, Awngi, Geez, Guragigna, Tigrigna and Xamtanga languages are reported in the form of incorrect and correct classification as shown in Figure 4.4.

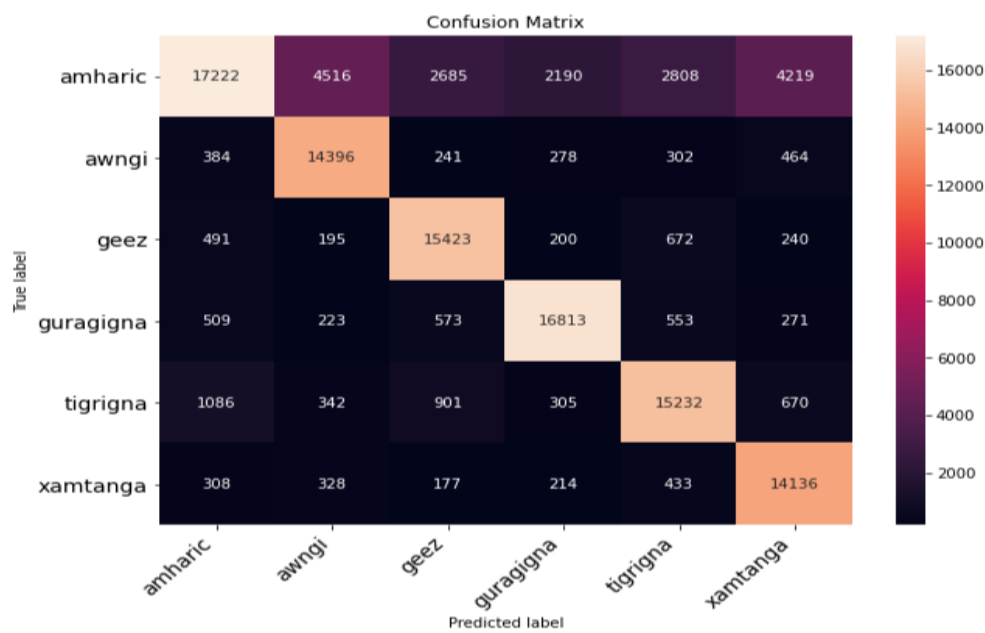


Figure 4. 4 Confusion matrix of the model evaluated on sample text length 5 characters

As shown in Figure 4.4 the confusion matrix has six possible classes. The columns represent predicted values and the rows represent actual values. We have evaluated the model using a total of 120k samples data. All the diagonal elements denote correctly classified outcomes and the off-diagonal numbers give the number of times a language was incorrectly predicted as another. The classification result in the first class reported that the evaluated model correctly classified 17,222 samples as Amharic and misclassified a total of 16,418 samples as other languages, while those should have been Amharic language.

Most wrong predictions are in the first row, which has the highest misclassification among all the classes. In the second class, the evaluated model correctly classified 14,396 samples as Awngi and misclassified a total of 1,669 samples as other languages, while those should have been Awngi language. In the third class, the evaluated model correctly classified 15,423 samples as Geez and misclassified a total of 1,798 samples as other languages, while those should have been Geez language. In the fourth class, the evaluated model correctly classified 16,813 samples as Guragigna and misclassified a total of 2,129 samples as other languages, while those should have been Guragigna language. In the fifth class, the evaluated model correctly classified 15,232 samples as Tigrigna and misclassified a total of 3,304 samples as other languages, while those should have been Tigrigna language. In the sixth class, the evaluated model correctly classified 14,136 samples as Xamtanga and a total of 1,460 samples of text were wrongly predicted as other languages, while those should have been Xamtanga language. Figure 4.5 illustrated that the classification report of the six Geez-based Ethiopian languages when the model is evaluated with sample text of 5 characters long.

	precision	recall	f1-score	support
amharic	0.51	0.86	0.64	20000
awngi	0.90	0.72	0.80	20000
geez	0.90	0.77	0.83	20000
guragigna	0.89	0.84	0.86	20000
tigrigna	0.82	0.76	0.79	20000
xamtanga	0.91	0.71	0.79	20000
micro avg	0.78	0.78	0.78	120000
macro avg	0.82	0.78	0.79	120000
weighted avg	0.82	0.78	0.79	120000

Figure 4. 5 Classification report of the model evaluated with 5 chars of sample text length

For example, from the confusion matrix, we can calculate the precision, recall, and F1-score of the Amharic language by using Equations 2.20, 2.21 and 2.22, respectively. Precision of Amharic is equal to $17,222 / (17,222 + 16,418) = 0.51$. Recall (Amharic) = $17,222 / (17,222 + 384 + 491 + 509 + 1086 + 308) = 0.86$. F1-score (Amharic) = $2 \times ((0.51 \times 0.86) / (0.51 + 0.86)) = 0.64$ and soon.

As usual in a confusion matrix, the diagonal elements are the correctly predicted samples. A total of 93,222 (where total = $17,222 + 14,396 + 15,423 + 16,813 + 15,232 + 14,136$) samples were correctly predicted out of the total of 120,000 samples. Thus, the overall accuracy of the model performance is 0.78 or 77.68% in percent (where accuracy = $93,222 / 120,000$) using Equation 2.19. The overall micro-average of precision result is also $0.78 = 93,222 / (93,222 + 26,778)$ using Equation 2.23.

The overall micro-average of recall score is also $0.78 = 93,222 / (93,222 + 26,778)$ using Equation 2.24. The overall micro-averaged F1-score of the model performance is $0.78 = 93,222 / (93,222 + (0.5 \times (26,778+26,778)))$ using Equation 2.25. Therefore, we can conclude that the accuracy, micro-average of precision, micro-average of recall and micro-average of F1-score have the same result since we used a balanced sample dataset.

The macro-average precision of the model is 0.82, (where $0.51 + 0.90 + 0.90 + 0.89 + 0.82 + 0.91$)/6 using Equation 2.26. The macro-average recall of the model gives 0.78, (where $0.86 + 0.72 + 0.77 + 0.84 + 0.76 + 0.71$)/6 using Equation 2.27 and the macro-average F1-score of the model is 0.79, (where $0.64 + 0.80 + 0.83 + 0.86 + 0.79 + 0.79$)/6 using Equation 2.28.

Furthermore, the weighted average of the trained model precision, recall, and F1-score are 0.82, 0.78, and 0.79, respectively, as shown in Figure 4.5, the results are calculated as follows:

$$Weighted_{average_precision} = \frac{(0.51 \times 20k) + (0.90 \times 20k) + (0.90 \times 20k) + (0.89 \times 20k) + (0.82 \times 20k) + (0.91 \times 20k)}{120k} = 0.82$$

$$Weighted_{average_recall} = \frac{(0.86 \times 20k) + (0.72 \times 20k) + (0.77 \times 20k) + (0.84 \times 20k) + (0.76 \times 20k) + (0.71 \times 20k)}{120k} = 0.78$$

$$Weighted_{average_F1score} = \frac{(0.64 \times 20k) + (0.80 \times 20k) + (0.83 \times 20k) + (0.86 \times 20k) + (0.79 \times 20k) + (0.79 \times 20k)}{120k} = 0.79$$

4.4.2. Experiment Two Using Text Length of 10 Characters

In the second scenario, we have train and evaluated the proposed DNN model with a sample text length of 10 characters per row and the model achieved a test accuracy of 91.10% and a test loss of 0.2626. The loss result of this experiment halved the loss of the first experiment, but indicates that the model still contains many incorrect predictions. Besides, the improvement in accuracy as the character length increases is noticeable. Hence, we can conclude that the accuracy of the model increased when using larger training data. Figure 4.6 shows the performance of our deep neural network model in each language variant when the model is trained and evaluated with a text length of 10 characters, while Figure 4.7 shows the corresponding classification report.

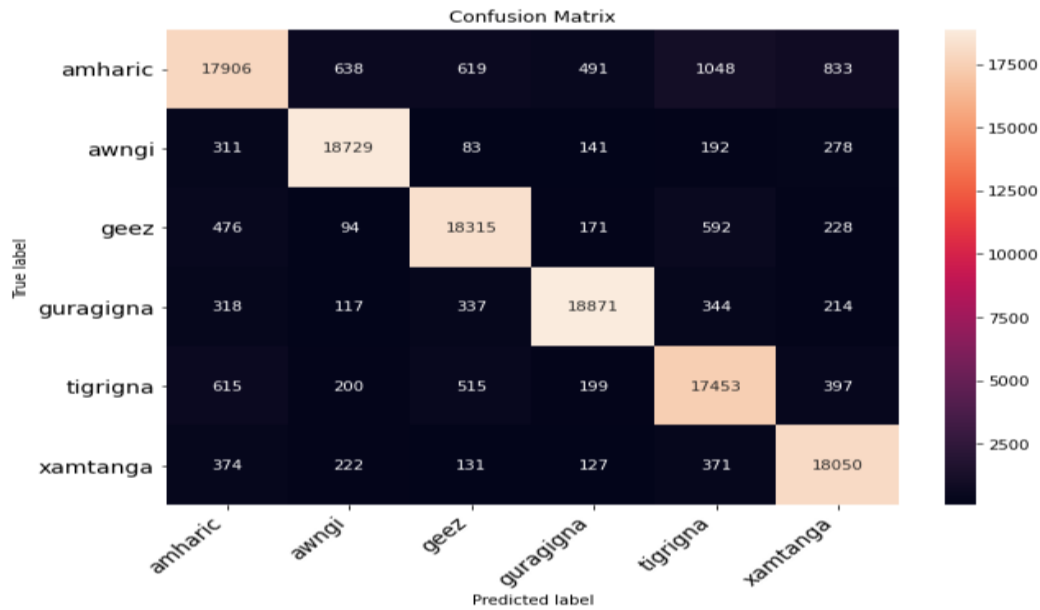


Figure 4. 6 Confusion matrix of the model evaluated on sample text length 10 characters

In the confusion matrix of the model, we can see the correct and incorrect classification values of the six languages, which are trained and evaluated on a sample text length of 10 characters. In this experiment, similarly, we used a total of 120k sample data to evaluate the model. A total of 109,324 samples out of 120k sample texts were correctly classified, therefore we noticed some improvement compared to the first experiment.

In the first class, 638 Amharic sample texts were incorrectly classified as Awngi, 619 Amharic sample texts were incorrectly classified as Geez, 491 Amharic sample texts were incorrectly classified as Guragigna, 1,048 Amharic sample texts were incorrectly classified as Tigrigna and 833 Amharic sample texts were incorrectly classified as Xamtanga. In total, the model correctly classified 17,906 samples as Amharic and incorrectly classified 3,629 samples as other languages. The classification result in the second class reported that the evaluated model correctly classified 18,729 samples as Amharic and 1,005 samples of text were wrongly predicted as other languages, while those should have been Amharic language. The classification result in the third class reported that the evaluated model correctly classified 18,315 samples as Amharic and 1,561 samples of text were wrongly predicted as other languages, while those should have been Amharic language.

The classification result in the fourth class reported that the evaluated model correctly classified 18,871 samples as Amharic and 1,330 samples of text were wrongly predicted as other languages, while those should have been Amharic language. The classification result in the fifth class reported that the evaluated model correctly classified 17,453 samples as Amharic and 1,926 samples of text were wrongly predicted as other languages, while those should have been Amharic language. The classification result in the sixth class reported that the evaluated model correctly classified 18,050 samples as Amharic and 1,225 samples of text were wrongly predicted as other languages, while those should have been Amharic language.

	precision	recall	f1-score	support
amharic	0.83	0.90	0.86	20000
awngi	0.95	0.94	0.94	20000
geez	0.92	0.92	0.92	20000
guragigna	0.93	0.94	0.94	20000
tigrigna	0.90	0.87	0.89	20000
xamtanga	0.94	0.90	0.92	20000
micro avg	0.91	0.91	0.91	120000
macro avg	0.91	0.91	0.91	120000
weighted avg	0.91	0.91	0.91	120000

Figure 4. 7 Classification report of the model evaluated with 10 chars of sample text length

The classification report in Figure 4.7 shows how the classification performance differs between the different languages. From the classification report Amharic, Awngi, Geez, Guragigna, Tigrigna and Xamtanga achieves a precision of 0.83, 0.95, 0.92, 0.93, 0.90 and 0.94 respectively. The proposed model performs recall results of 0.90, 0.94, 0.92, 0.94, 0.87 and 0.90 for Amharic, Awngi, Geez, Guragigna, Tigrigna and Xamtanga, respectively. In addition, the F1 score were 0.86, 0.94, 0.92, 0.94, 0.89, and 0.92 for Amharic, Awngi, Geez, Guragigna, Tigrigna, and Xamtanga respectively. In general, the classifier model achieved a micro-average, macro-average, and weighted-average F1-score of 0.91 on the test dataset.

4.4.3. Experiment Three Using Text Length of 50 Characters

For this experiment, we increased the sample text length to 50 characters. The model trained and evaluated using this sample data and achieved a test accuracy of 99.53% and a test loss of 0.0204. Comparing the performance of this experiment with the previous two experiments, we notice that the test accuracy of the model increased slightly while the test loss decreased significantly. For a sample text length of 50 characters per row, the model trained with chars2vec embedding showed better accuracy than 10 characters on the same 120K test dataset.

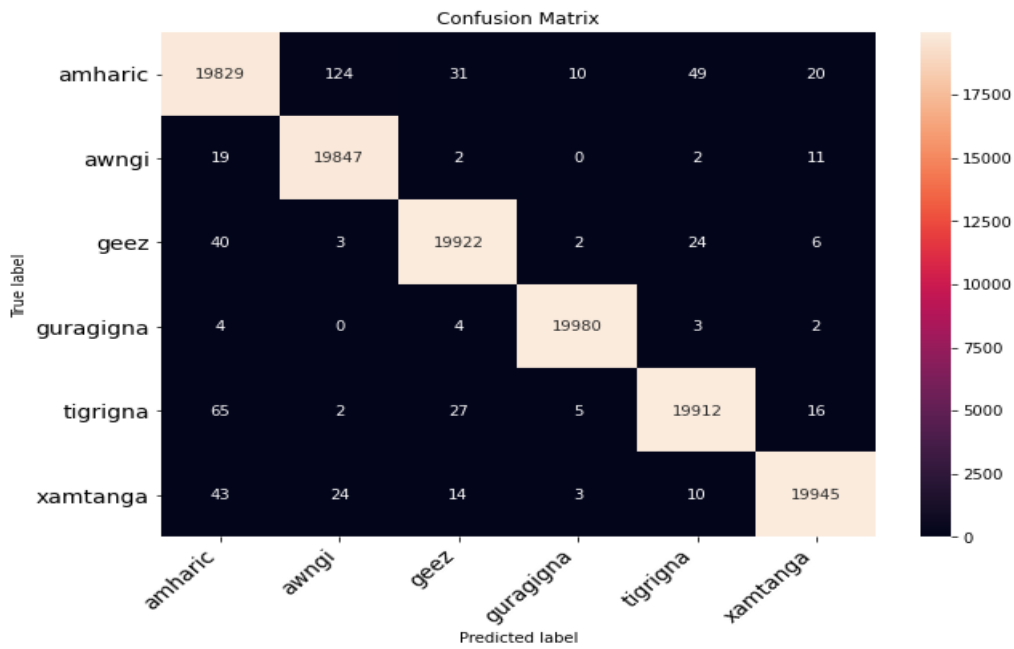


Figure 4. 8 Confusion matrix of the model evaluated on sample text length 50 characters

The confusion matrix shown in Figure 4.8 gives an overview of how well the classifier works for each label language. The white diagonal indicates the number of correct predictions for each language. The off-diagonal numbers indicate the number of times a language was incorrectly predicted as another. A total of 119,435 samples out of 120k sample texts were correctly identified. The classification result in the first class reported that the model correctly classified 19,829 samples as Amharic and 234 samples of text were wrongly predicted as other languages, while those should have been Amharic language. For example, Awngi is incorrectly predicted as Amharic 124 times.

The prediction result in the second class show that the model correctly classified 19,847 samples as Awngi and 34 samples of text were wrongly predicted as other languages, while those should have been Awngi language. The prediction result in the third class show that the model correctly classified 19,922 samples as Geez and 75 samples of text were wrongly predicted as other languages, while those should have been Geez language. The prediction result in the fourth class show that the model correctly classified 19,980 samples as Guragigna and 13 texts were wrongly predicted as other languages, while those should have been Guragigna language.

The prediction result in the fifth class show that the model correctly classified 19,912 samples as Tigrigna and 115 samples of text were wrongly predicted as other languages, while those should have been Tigrigna language.

The prediction result in the six class show that the model correctly classified 19,945 samples as Xamtanga and 94 samples of text were wrongly predicted as other languages, while those should have been Xamtanga language.

	precision	recall	f1-score	support
amharic	0.99	0.99	0.99	20000
awngi	1.00	0.99	1.00	20000
geez	1.00	1.00	1.00	20000
guragigna	1.00	1.00	1.00	20000
tigrigna	0.99	1.00	0.99	20000
xamtanga	1.00	1.00	1.00	20000
micro avg	1.00	1.00	1.00	120000
macro avg	1.00	1.00	1.00	120000
weighted avg	1.00	1.00	1.00	120000

Figure 4. 9 Classification report of the model evaluated with 50 chars of sample text length

Looking at the classification report of the six Ethiopic-based languages, these were trained and evaluated using a sample text length of 50 characters per line. The classifier performed correct predictions for most languages. The model with a sample text length of 50 characters per line achieved 100% micro-average, macro-average, and weighted-average F1-score on the test dataset.

4.4.4. Experiment Four Using Text Length of 100 Characters

For the last experiment, we used 100 characters long per row and the model achieved a test accuracy of 99.92% and a test loss of 0.0045. Similarly, we used a total of 120k samples of text to evaluate the model to conduct this experiment. Some of the off-diagonal numbers are zero, indicating that the model achieved very good predictions. Figure 4.10 shows the performance of our model in each language variant when the model is trained and evaluated with a text length of 100 characters per row, while Figure 4.11 shows the corresponding classification report.

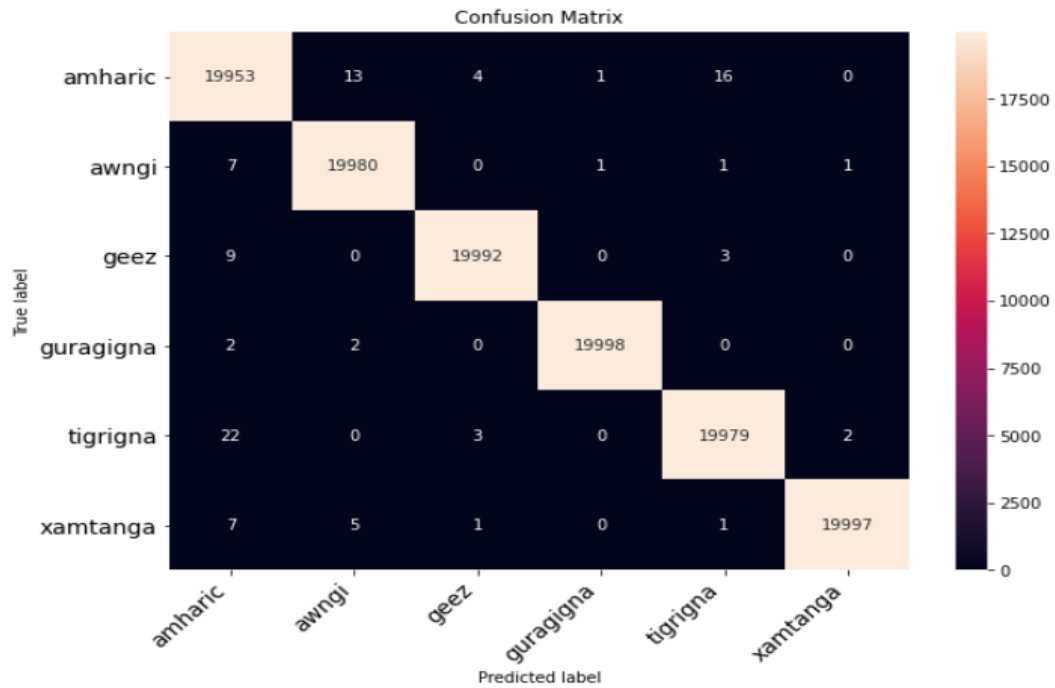


Figure 4.10 Confusion matrix of the model evaluated on sample text length 100 characters

As shown in Figure 4.10 the confusion matrix has six possible classes. Out of 120k sample texts, a total of 119,899 samples were correctly classified and only 101 sample of texts were classified incorrectly. The prediction result in the first class reported that the model correctly classified 19,953 samples as Amharic and 34 samples of text were wrongly predicted as other languages, while those should have been the Amharic language. The prediction result in the second class show that the model correctly classified 19,980 samples as Awngi and 10 samples of text were wrongly predicted as other languages, while those should have been Awngi language. The prediction result in the third class show that the model correctly classified 19,992 samples as Geez and 12 samples of text were wrongly predicted as other languages, while those should have been Geez language.

The prediction result in the fourth class show that the model correctly classified 19,998 samples as Guragigna and 4 texts were wrongly predicted as other languages, while those should have been Guragigna language. The prediction result in the fifth class show that the model correctly classified 19,979 samples as Tigrigna and 27 samples of text were wrongly predicted as other languages, while those should have been Tigrigna language. The prediction result in the six class show that the model correctly classified 19,997 samples as Xamtanga and 14 samples of text were wrongly predicted as other languages, while those should have been Xamtanga language.

	precision	recall	f1-score	support
amharic	1.00	1.00	1.00	20000
awngi	1.00	1.00	1.00	20000
geez	1.00	1.00	1.00	20000
guragigna	1.00	1.00	1.00	20000
tigrigna	1.00	1.00	1.00	20000
xamtanga	1.00	1.00	1.00	20000
micro avg	1.00	1.00	1.00	120000
macro avg	1.00	1.00	1.00	120000
weighted avg	1.00	1.00	1.00	120000

Figure 4. 11 Classification report of the model evaluated with 100 chars of sample text length

As shown in Figure 4.11, the model performed very good predictions for most languages. The model with a sample text length of 100 characters per line achieved 100% micro-average, macro-average, and weighted-average F1-score on the test dataset.

4.4.5. Summary of the Experiments

In this subsection, we have been done four basic experiments to evaluate the performance of the proposed model at different sample text lengths. All experiments were performed with the best hyperparameter settings. We have been trained and evaluated the model with a sample text length of 5, 10, 50 and 100 characters per line. In general, the results of the experiment show that high accuracy is achieved for long texts and low accuracy for short texts. In the first experiment, we observed that many misclassifications occurred, because of all the languages were closely related at the word level. In addition, from confusion matrix, we found that Amharic language has a higher relationship with other languages and in all experiments, the most incorrect predictions are occurred in Amharic language. The summary of all experimental results is described in Table 4.8 and Figure 4.12.

Table 4. 8 Test accuracy and loss results of the proposed model with variety of sample text lengths

<i>Length of sample text in characters</i>	<i>Test Accuracy</i>	<i>Test Loss</i>
5	77.68%	0.5862
10	91.10%	0.2626
50	99.53%	0.0204
100	99.92%	0.0045

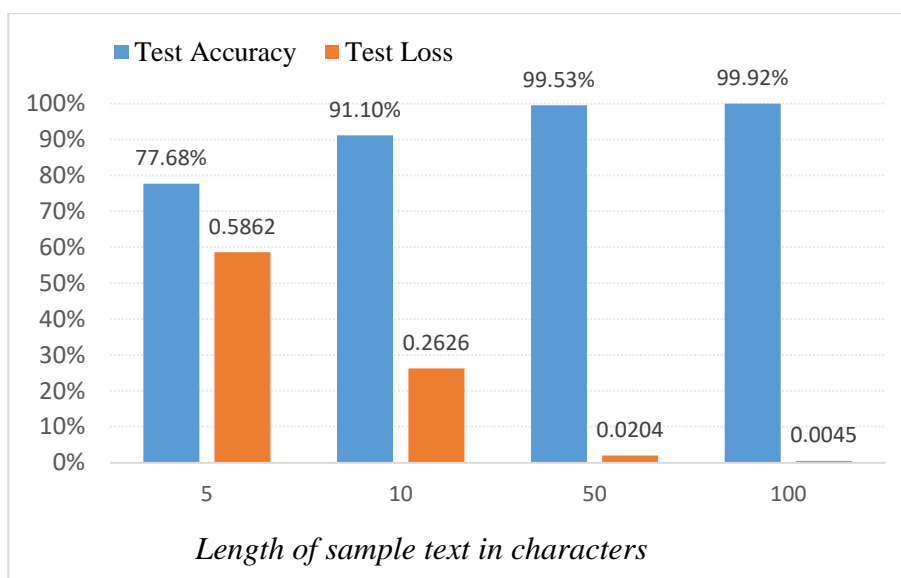


Figure 4. 12 Accuracy and loss of the proposed model with different character length of texts

In the summary of all experimental results, as shown in Figure 4.12, the proposed model obtained a test accuracy of 77.68%, 91.10%, 99.53% and 99.92% for sample text lengths of 5, 10, 50 and 100 characters respectively. With sample text lengths of 5, 10, 50, and 100 characters per row, the proposed model achieved a test loss of 0.5862, 0.2626, 0.0204 and 0.0045, respectively.

In all experiments, the Amharic sample texts were mostly incorrectly classified as other languages. Generally, the experimental result shows that the model identification accuracy increases when the length of sample text increases and language identification for very short texts still needs improvement.

4.5. Prototype and Predictions

In the final phase, the researcher builds a simple user interface to make language predictions for the given pieces of text. We evaluated the performance of the final developed model on out-of-vocabulary texts. The prediction result of the developed language identification model across different texts is shown in Figures 4.13 to 4.18.

A) Prediction result for the given short and long Amharic text

The prediction result shows that the model correctly identified the language for the given Amharic short and long texts as shown in Figure 4.13.

Enter text =>: ቆይተዋል

Identify Language

This text is AMHARIC language.

Enter text =>: ተከስቷል ካለፈው ጥቅምት ወር ጀምሮ በክልሉ ከተሞች የቤንዚን እጥረት ተከስቷል በዚህም የባጃጅ አሽከርካሪዎችና ሌሎችም ተጠቃሚዎች ቅሬታቸውን ሲያሰሙ ቆይተዋል የአማራ ብዙኃን መገናኛ ድርጅት በባሕር ዳር ከተማ በመዘዋወር ያነጋገራቸው የባጃጅ አሽከርካሪዎች በተከሰተው የቤንዚን እጥረት የተነሳ ስራ ለማቆም መገደዳቸውን ተናግረዋል የቤንዚን ነዳጅ ለማግኘትም እስከ ሁለት ቀናት ድረስ ወረፋ ይጠብቃሉ

Identify Language

This text is AMHARIC language.

Figure 4. 13 Prediction result for the given short and long Amharic text

B) Prediction result for the given short and long Awngi text

The prediction result shows that the model correctly identified the language for the given Awngi short and long texts as shown in the screenshot below.

Enter text =>: ዲግስሻስ ሙሱቡ ቢሪኪቲስ

Identify Language

This text is AWNGI language.

Enter text =>: ዲግስሻስ ሙሱቡ ቢሪኪቲስ ማን አቼሳ ምግቡሳ ዲግስሻ ጅምራትሻጅሻጅ ደስ ፈያማ ዙራሙራው እምጥልታቸስ ዲግስሻስ ካሉንኩ ጅትካዋ ጎቤሻ ዶክተር አብይ አህመድ ዙሙና ባህር ዳር አሚኮ ኬቴማኻዳ ዝኩኽ ልቓ ብቲዳ ቴጋኑሳ ትክልትካዋ ዶርካዋስታ እንሴስካዋ እሩብጅሻ ቡዚስሻስ ፅኻራዋ ሱኽጅሻ እንቲታኪላ እሊኩ ሙሱቡ ቢሪኪቲስ ማን አቼሳ ምግቡሳ ዲግስሻ ጅምራትሻጅሻጅ ደስ ፈያማ ዙራሙራው

Identify Language

This text is AWNGI language.

Figure 4. 14 Prediction result for the given short and long Awngi text

C) Prediction result for the given short and long Geez text

In this case, the prediction result shows that the model incorrectly classified the short texts into the Amharic language when it should have been the Geez language. However, the model correctly identified the given long texts.

Enter text =>: መንገሥ ሰሜን አውደ የአውድ

Identify Language

This text is AMHARIC language.

Enter text =>: ቃለ መክብብ ወልደ ዳዊት ንጉሠ እስራኤል በኢየሩሳሌም ከንቱ ከንቱ ይቤ መክብብ ከንቱ ከንቱ ክሉ ከንቱ ምንት ፍድፋዱ ለሰብእ በክሉ ዳግሁ ዘይዳሙ እምታሕተ ፀሓይ ትውልድ የሐልፍ ወትውልድ ይመጽእ ወምድርስ ለዓለም ትቀውም ይሠርቅ ፀሓይ ወየዐርብ ፀሓይ ወውስተ መክኑ ይገብእ እንዘ ይሠርቅ ውእቱ ህየ ይገብእ በይምን ወየአውድ መንገሥ ሰሜን አውደ የአውድ ወየሐውር መንፈስ ወበኡደቱ ይገብእ መንፈስ ክሉ ወሓይዝት የሐውራ ውስተ ባሕር ወባሕር ኢኮነት እንተ ትመልእ ውስተ መካን ውእደ የሐውሩ አፍላግ ህየ ካዕበ ይትመየጡ

Identify Language

This text is GEEZ language.

Figure 4. 15 Prediction result for the given short and long Geez text

D) Prediction result for the given short and long Guragigna text

The prediction result shows that the model correctly classified the given short and long texts as shown in the snapshot below.

Enter text =>: ይትኻሸሪ

Identify Language

This text is GURAGIGNA language.

Enter text =>: ተምሳይት ይኸር ጉራጌ ትንግሊዚና ድቦያ ድምጥ ያነጌ ድቦያ ኸኖ ይጠጭፍ ደንብ አውጦት ነረብንደ የተምሳይት አኸር እንግሊዚና ጩ ኤነን የኸሬ ይትቅራኑብ ድምጥ ቲኸር ንቅ አመነውም ይትኻሸሪ ቲጠፎ እንጨረብየ ይኸር ትሸክቶይ ይንግሊዚና የጡፍ ይወደረ ቁልፍ ንግዶት ነረብኸ ትጠፎይ የሸኸ አት ኤነት ፊደር ቁልፍ ትትደርጎ አንጠፍ በባሪናኹ የዛም ፊደር ንቅ

Identify Language

This text is GURAGIGNA language.

Figure 4. 16 Prediction result for the given short and long Guragigna text

E) Prediction result for the given short and long Tigrigna text

As shown in Figure 4.17, the prediction result of the model correctly identified the given word and long texts.

Enter text =>:

Identify Language

This text is TIGRIGNA language.

Enter text =>:

Identify Language

This text is TIGRIGNA language.

Figure 4. 17 Prediction result for the given short and long Tigrigna text

F) Prediction result for the given short and long Xamtanga text

For the first trial, the developed LID model correctly identified the given word and long texts as shown in Figure 4.18.

Enter text =>:

Identify Language

This text is XAMTANGA language.

Enter text =>:

Identify Language

This text is XAMTANGA language.

Figure 4. 18 Prediction result for the given short and long Xamtanga text

4.5.1. Language Predictions on Out-of-Vocabulary Texts

As we mentioned in the literature review, the character-level embedding technique is mainly important for solving out-of-vocabulary words problem. Therefore, to verify this, we have evaluated the performance of our model against out-of-vocabulary words for the Amharic and Guragigna languages. The Amharic¹⁶ OOV texts were collected from a Facebook page and Guragigna¹⁷ language OOV sample texts were collected from the web page.

I. Model evaluation for the given Amharic out-of-vocabulary texts

As shown in the snapshots below, the developed LID model is correctly identified for the given Amharic short and long out-of-vocabulary texts.

Enter text => ጀለሳምች ሹፈ.ው አርፈ.ው

Identify Language

SCORE: 52.309203147888184
This text is AMHARIC language.

Enter text => አጨህልኝ ደልተው ፎረመተው አልገብቶኝም

Identify Language

SCORE: 69.78404521942139
This text is AMHARIC language.

Figure 4. 19 Prediction results for the given Amharic out-of-vocabulary texts

¹⁶ Amharic Texts: <https://www.facebook.com/aradaslang>

¹⁷ Guragigna Texts: <https://www.unicode.org/L2/L2021/21037-gurage-adds.pdf>

II. Model evaluation for the given Guragigna out-of-vocabulary texts

As shown in the snapshots below, the developed LID model is correctly identified for the given Guragigna short and long out-of-vocabulary words or texts. Additionally, we can notice that language identification performance increases as the length of texts increases.

Enter text =>: ዘረሕና

Identify Language

SCORE: 54.6021044254303

This text is GURAGIGNA language.

Enter text =>: የብጃርነተሕና ዘረሕና ቦነሕሕና

Identify Language

SCORE: 99.90543723106384

This text is GURAGIGNA language.

Enter text =>: በጣፊጥ ይሕርጥ ያሕጥ ይሕርጥ ያተዋነጥ

Identify Language

SCORE: 99.91068243980408

This text is GURAGIGNA language.

Figure 4. 20 Prediction results for the given Guragigna out-of-vocabulary texts

We used the texts written in the newly added Guragigna characters to evaluate the performance of the model in solving out-of-vocabulary problems.

4.6. Answering Research Questions

At the beginning of this work, the researchers were asked two main research questions that should be answered after the experiment. Therefore, this subsection contains the discussions to ensure that the question is answered as follows:

The first research question was, “*To what extent does the proposed language identification model correctly identify Geez-based Ethiopian languages?*”. This is the basic question of the study, and the aim was to assess the overall performance of our proposed study in terms of correctly identifying languages for the given short and long texts. Therefore, the researchers first conducted more than nine experiments using AdaGrad, Adam, and RMSprop optimizers to select the optimal hyperparameter sets for the proposed language identification model, and the experimental result showed that the Adam optimizer achieved better accuracy with a batch size of 64, an epoch of 15, and a dropout ratio of 64 as we mentioned earlier in Table 4.7. Using the selected hyperparameter sets, the performance of the proposed LID model is evaluated on different sample texts. The researchers conducted four experiments with sample texts of different lengths for six typologically related Ethiopian languages, namely Amharic, Awngi, Geez, Guragigna, Tigrigna and Xamtanga. Finally, the developed model achieved an accuracy of 77.68%, 91.10% and 99.53% when trained and evaluated with sample text lengths of 5, 10 and 50 characters, respectively. Also, the proposed model showed a better accuracy of 99.92% when trained and evaluated on sample texts with a length of 100 characters per line. Generally, for text longer than 50 characters, the proposed model identified the language correctly with more than 99% accuracy.

The second question was, “*How robust is the language identifier when tested against a prototype for different domain texts?*”. To answer this question, we designed a simple user interface and evaluated the performance of our final trained model on an unseen text by varying text lengths. Furthermore, we also tested our model on OOV texts and as a result, the proposed model correctly identified the text of the language.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1. Overview

This chapter presents the conclusions of the research described in the thesis. The aim and objectives of the research outlined in Chapter one are reviewed and their achievement is addressed. The contributions of this study and the recommendations for future work indicated by the research are suggested.

5.2. Conclusion

Human language can be developed naturally or constructed on purpose, but in all cases the defining feature is that it is used to communicate between people through speaking, signing, or writing. In this research paper, the researchers focused on written languages. Language identification is an application of NLP that automatically identifies the language in which the contents of the text are written, which is generally considered a special case of text categorization. LID is a preprocessing task to develop multilingual-based NLP applications.

The aim of this research was to develop an automatic LID model for typologically related Ethiopian languages namely Amharic, Awngi, Geez, Guragigna, Tigrigna and Xamtanga by applying a deep neural network. To achieve the research aim, an experimental research methodology was designed. In this study, various approaches and applications of language identification studied by various researchers are reviewed. Most of the researchers are done using N-gram model and machine learning classifiers. However, not many studies have been conducted regarding the deep learning. Deep learning is a state-of-the-art approach and has recently been identified as a major advance in text classification. Due to the fact that the researchers used a deep neural network algorithm with bag-of-character embedding techniques for the proposed study.

We have collected the dataset from various sources and performed appropriate preprocessing tasks. We applied chars2vec for text representation, one-hot-encoding for class labels and standard scalar standardization technique. Finally, we randomly selected 100k samples text for each language and a total of 600k samples text were allocated for training and testing the proposed DNN model. In all experiments, the researchers allocated 432k for training the model, 48k for validation and 120k for evaluating the trained model.

The researchers constructed a three-layer DNN model architecture that consists 406 input size, 512 nodes for the first hidden layer, 256 nodes for the second hidden layer, 128 nodes for the third hidden layer and 6 neurons in the output layer and total of 372,870 trainable parameters. The hyperparameters used in this paper were dropout ratio, optimization algorithms, sample text length, number of epochs, activation functions, batch size, sigmoid function, Softmax, categorical-cross entropy loss function. Finally, the combination of hyperparameters that yields the highest accuracy in the test set were selected for the final LID model. To select the best hyperparameter setting, we tried more than nine experiments and the experimental results showed that the Adam optimizer attains better accuracy and a lower test error compared to the other optimizers with a batch size of 64, a dropout ratio of 0.5, and an epoch of 15. The researchers then conducted four experiments to train the proposed model and evaluate the performance of the final trained model with different sample text lengths using the selected best hyperparameter setting.

From the experimental results, we discovered that the proposed DNN model with the bag-of-character embedding approach achieved 77.68% accuracy and a loss result of 0.5862 for samples text length of 5 characters. While, for a text length of 100 characters per line, the proposed DNN model with a chars2vec embedding approach achieved an accuracy of 99.92 and a loss of 0.0045 for the test set. In other words, the trained model performed a micro-average F1-score of 100% for sample text lengths of 100 characters while, for test sample text lengths of 5 characters, the model achieved a micro-average F1-score of 0.78% for all supported Geez-based Ethiopian languages.

One of the key findings of this research study is that the model accuracy increased as the length of the sample texts increased, and vice versa. Another finding of this study is that our model found promising results with a small number of trainable parameters and that the model also worked well for the OOV texts. This study differs from previous related works in various parameters like the algorithms applied, the dataset, the vectorization technique, and the language selected. It is more difficult to discriminate languages within language families than those across families. Languages written with the Geez script are very closely related. In such a scenario, distinguishing the word language was the main challenge in the LID task because of the words from one language are adopted from another language. This inheritance of words makes some words available in many languages, which makes language detection a difficult task.

5.3. Contributions

The main contributions of the presented work are as follows:

- © This work presents a new corpus for six closely related low-resourced Ethiopian languages, namely Amharic, Awnigi, Geez, Guragigna, Tigrinya, and Xamtanga, so that corpus can be applied to other similar cases. We have also compiled the alphabet of each language.
- © The study contributed to the accessibility of two Ethiopian Cushitic languages that were not included in previous related studies. To the best of our knowledge, this is the first language identification model for the Awnigi and Xamtanga languages.
- © In this study, researchers presented a state-of-the-art deep neural network approach and character vector representation (chars2vec) embedding technique for the proposed language identification. To the best of our knowledge, this is the first investigation on language identification using a deep neural network approach with the chars2vec model for Geez-based Ethiopian languages.
- © In addition, the study contributes to solving the problem of out-of-vocabulary words through the use of the chars2vec text representation technique.

5.4. Recommendations

The experimental result shows that the Adam optimizer is a better hyperparameter choice than others, therefore, we recommend future researchers to use it as the default optimizer for most applications. Based on the obtained results, the researcher concludes that the proposed character-level embedding method is effective in textual based language identification task with minimal parameters. With this in mind, the researcher recommends adopting this new approach for other text classification tasks as well. The developed model accurately identifies the language of texts longer than 10 characters. This developed model accurately predicts languages at the phrase, sentence and paragraphs level. Therefore, we recommend that future researchers adopt this all-in-one model as a preprocessing task to implement phrase, sentence and paragraph level multilingual natural language processing applications like information retrieval, machine translation, fact-checking applications, sentiment analysis and plagiarism detection.

5.5. Future works

To improve the current state of the study, the researchers recommend the following points for further research directions.

- In this research work, we applied a bag-of-characters (chars2vec) embedding technique to represent the features of the sample dataset. For future work, it would be better to use more sophisticated embedding techniques like Bag-of-Words, GloVe and Word2vec.
- In this study, the researchers present a language identification model for only six typologically and phylogenetically related low-resourced Ethiopian languages that use the Geez script as a writing system. Therefore, we recommend that future researchers consider other related languages to span the coverage of Ethiopian languages.
- In this investigation, we used a deep-feedforward neural network architecture for the proposed language identification model. For future work, it is recommended to conduct a comparative study with different deep learning and machine learning classifiers to enhance the performance and determine the best language identifier.
- In all experiments, the Amharic texts were mostly incorrectly classified as other languages. Future researchers can investigate on such factors.
- In multilingual language identification, it is mandatory first to check whether the text document is written with monolingual or multilingual language before language identification applies. For multilingual identifier, knowing language switching is a big challenge. This specifies how frequently or where a shift from one language to another can occur in a document. Therefore, another possible future work could be to investigate language identification when the documents contain code-switching languages.
- In this research work, we have obtained very good results in identifying the language of sample texts longer than 10 characters. However, it is still an open research area to improve the accuracy of language identifiers for very short texts under 10 characters long and closely related Ethiopic-based languages.

REFERENCES

- [1] Solomon T. Abate, M. Yifiru Tachbelie, and T. Schultz, “Deep Neural Networks Based Automatic Speech Recognition for Four Ethiopian Languages”, *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, Barcelona, Spain, May 2020, pp. 8274–8278. doi: 10.1109/ICASSP40776.2020.9053883.
- [2] “Ethiopian Languages - Semitic, Cushitic, Omotic and Nilo-Saharan.” <http://www.ethiopiantreasures.co.uk/pages/language.htm> (accessed May 16, 2023).
- [3] B. Piper and A. J. van Ginkel, “Reading the script: How the scripts and writing systems of Ethiopian languages relate to letter and word identification” *Witting System Research*, vol. 9, no. 1, pp. 36–59, September 2016.
- [4] F. Gaim, W. Yang, and J. C. Park, “GeezSwitch: Language Identification in Typologically Related Low-resourced East African Languages,” in *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, Marseille, France: European Language Resources Association, Jun. 2022, pp. 6578–6584.
- [5] Eberhard, David M., Gary F. Simons, and Charles D. Fennig, “*Ethnologue: languages of the world. Dallas, Texas, SIL International,*” vol. 26, (eds.). 2023: online version <http://www.ethnologue.com>.
- [6] C. Gobinda G., “Natural language processing,” *Fundamental of Artificial. Intelligence. Univ. Strathclyde UK*, pp. 603–649, 2020, doi: https://doi.org/10.1007/978-81-322-3972-7_19.
- [7] J. Hirschberg and C. D. Manning, “Advances in natural language processing,” *Science*, vol. 349, no. 6245, pp. 261–266, Jul. 2015, doi: 10.1126/science.aaa8685.
- [8] T. Gottron and N. Lipka, “A comparison of language identification approaches on short, query-style texts,” in *Advances in Information Retrieval: 32nd European Conference on IR Research, ECIR 2010, Milton Keynes, UK, March 28-31, 2010. Proceedings 32*, Springer, 2010, pp. 611–614.
- [9] A. Babhulgaonkar and S. Sonavane, “Language identification for multilingual machine translation,” in *2020 International Conference on Communication and Signal Processing (ICCSP)*, Chennai, India: IEEE, 2020. doi: 10.1109/ICCSP48568.2020.9182184.
- [10] D. Khurana, A. Koli, K. Khatter, and S. Singh, “Natural language processing: State of the art, current trends and challenges,” *Multimedia Tools Application*, pp. 1–32, 2022, doi: <https://doi.org/10.1007/s11042-022-13428-4>.
- [11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” *Journal of machine learning research*, vol. 12, no. ARTICLE, pp. 2493–2537, 2011.
- [12] M. Lui, J. H. Lau, and T. Baldwin, “Automatic detection and language identification of multilingual documents,” *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 27–40, Dec. 2014, doi: https://doi.org/10.1162/tacl_a_00163.

- [13] A. Garg, V. Gupta, and M. Jindal, “A Survey of Language Identification Techniques and Applications.” *Journal of Artificial Intelligence Research* 65 (2019): 675-782
- [14] D. Londhe, A. Kumari, and M. Emmanuel, “Language identification for multilingual sentiment examination,” *International Journal of Recent Technology and Engineering*, vol. 8, no. 2/11, pp. 3571–3576, 2019.
- [15] T. Jauhiainen, M. Lui, M. Zampieri, T. Baldwin, and K. Lindén, “Automatic language identification in texts: A survey,” *Journal of Artificial Intelligence Research*, vol. 65, pp. 675–782, Aug. 2019.
- [16] Y. Liu and M. Zhang, “Neural Network Methods for Natural Language Processing,” *Computational Linguistic*, vol.44, no.1, pp.193–195, MIT Press, Cambridge, MA. March 2018, doi: 10.1162/COLI_r_00312.
- [17] Lin, Xiaotian, Nankai Lin, Kanoksak Wattanachote, Shengyi Jiang, and Lianxi Wang, “Multilingual Text Classification for Dravidian Languages,” *ArXiv Prepr. ArXiv211201705*, December 3 2021.
- [18] M. Lui and T. Baldwin, “Langid.py: An off-the-shelf language identification tool,” *Proceedings of the ACL system demonstrations*. July. 2012, pp. 25–30.
- [19] Biruk Tadesse, “Automatic Identification of Major Ethiopian Languages,” MSc. Thesis, Bahir Dar University, Ethiopia, February 2018
- [20] L. Wedajo, “Modeling Text Language Identification for Ethiopian Cushitic Languages”, MSc. Higher Learning Center of Excellence College, Addis Ababa, Ethiopia, July 2014
- [21] R. Bekele, “A Comparative Study of Automatic Language Identification of Ethio-Semitic Languages,” MSc.Thesis, Addis Ababa University, Ethiopia, June 2018.
- [22] K. Erigete, “General Purpose Language Identification for Ethiopia Semitic Language using Hybrid Approach,” MSc. Thesis, Jimma University, Ethiopia, 2017.
- [23] S. Lahiri and R. Mihalcea, “Using N-gram and Word Network Features for Native Language Identification,” in *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, Atlanta, Georgia: Association for Computational Linguistics, Jun. 2013, pp. 251–259.
- [24] R. Meyer, “The Ethiopic Script: Linguistic Features and Socio-cultural Connotations,” *Oslo Stud. Lang.*, vol. 8, no. 1, Art. no. 1, 2016, doi: 10.5617/osla.4422.
- [25] F. Coulmas, *Writing Systems: An Introduction to Their Linguistic Analysis*, Cambridge University Press, 2003.
- [26] A. De Voogt and J. F. Quack, *The idea of writing: Writing across borders*, vol. 2. Brill, 2011.
- [27] I. Taylor and D. R. Olson, *Scripts and literacy: Reading and learning to read alphabets, syllabaries, and characters*, vol. 7. Springer Science & Business Media, 1995.
- [28] መምህር ኢያሱ ድረስ, “መዝገብ አጎምሮ ግዕዝ ዐምዳ ወድዳ ለኢትዮጵያ, ኅንደር, ከኑ ማተሚያ ቤት, 2012.

- [29] “The Ethiopian Orthodox Tewahedo Church” <https://www.ethiopianorthodox.org/> (accessed Feb. 21, 2023).
- [30] Central Statistical Agency, “*The 2007 population and housing census of Ethiopia, Results at Country Level, Analytical Report*”, CSA Addis Ababa, May 2007.
- [31] M. Midega, “*Official Language Choice in Ethiopia: Means of Inclusion or Exclusion?*,” *Open Access Libr. J.*, vol. 1, no. 7, pp. 1–13, 2014.
- [32] ኪዳነ ወልድ ክፍሌ, “*ሰዋሰው ወግስ ወመዝገብ ቃላት ሐዲሶ*”, ኦዲስ አበባ, አርቲስቲክ ማተሚያ ቤት, 1948.
- [33] Tesema Habte Mikael Gisew, *YeAmarigna Mezgebe Kalat የአማርኛ መዝገብ ቃላት*, Addis Ababa, 1958.
- [34] Joswig, Andreas, “*The phonology of Awngi*”, *Summer Institute of Linguistics Electronic Working Papers*, Ethiopia, vol.87, January 2010.
- [35] D. Amsalu, “*An ethnographic introduction to the Kumpal Agaw*,” *Journal of Ethiopian Studies*, vol. 49, pp. 35–56, Dec. 2016.
- [36] Tsegaye Miskir, “*Developing a Stemming Algorithm for Awngi Text: A Longest match approach*,” MSc. Thesis, Addis Ababa University, Ethiopia, 2013.
- [37] D. M. Eberhard, Gary F. Simons, and Charles D. Fennig, *Ethnologue: Languages of the World.*, Twenty-Fifth edition. Dallas, Texas: SIL International., 2022.
- [38] Leyew, Zelealem, “*First report on a survey of the Shinasha and Agew dialects and languages*,” *Summer Institute of Linguistics International.*, p. 11, 2002.
- [39] A. Haileysus, “*Offline Handwritten Awngi Character Recognition Using Deep Learning Technique*,” MSc. Thesis, Bahir Dar University, Ethiopia, 2021.
- [40] “*ልሳነ ግእዝ ዘ ኢትዮጵያ / Lisane geez.*” <http://www.lisanegeez.com/> (accessed Feb. 21, 2023).
- [41] መምህር ዕንባቆም ገብረ ጻድቅ, *የግእዝ ቋንቋ የሰዋሰው መጽሐፍ ደብረ ብርሃን, ፋር ኢስት ትሬዲንግ ኃላፊ.የተ.የግ. ማኅበር*, 2010.
- [42] F. Menuta and D. Yacob, “*A Review of Shifts in Gurage Orthography*,” Ge’ez Frontier Foundation, *Unicode Technology Notes Hawassa University. Ethiopia*, September 2022.
- [43] A. Shumneka, “*Levels of Language Shift and Language Endangerment in The Gurage Varieties of Muher and Ezha*,” *ZENA-LISSAN (Journal of Academy of Ethiopian Languages and Cultures)*, vol. 26, no. 2, Art. no. 2, 2017.
- [44] “*Ethiopian Languages - Semitic, Cushitic, Omotic and Nilo-Saharan.*” <http://www.ethiopiantreasures.co.uk/pages/language.htm> (accessed Jan. 02, 2023).
- [45] Daniel Yacob, Fekede Menuta, and Feidu Akmel Gobena, “*The Ge’ez Frontier Foundation Keyboard for Ge’ez Language.*” Jan. 11, 2021. Accessed: Mar. 03, 2023. [Online]. Available: https://help.keyman.com/keyboard/gff_gurage/0.7/gff_gurage

- [46] Tsigie, Asteraye, Berhanu Beyene, Daniel Aberra, and Daniel Yacob, “A Roadmap to the Extension of the Ethiopic Writing System Standard Under Unicode and ISO-10646,” *15th Int. Unicode Conf.*, vol. 4, San Jose, California. September 1999.
- [47] በሀሩ ሊላጋ, “ቋንቋ ከጠፋ አጽሙ እንኳን አይገኝልትም,” *Gurage Zone Government Communication Affairs Department*, December 12, 2019. (accessed Mar. 08, 2023) <https://www.facebook.com/gurage1Zone/posts/2493544404231607>.
- [48] Daniel Yacob, Fekede Menuta, and Feidu Akmel Gobena, “*The Ge’ez Frontier Foundation Keyboard for Ge’ez Language*.” Jan. 11, 2021. Accessed: Mar. 03, 2023. [Online]. Available: https://help.keyman.com/keyboard/gff_gurage/0.9.1/gff_gurage
- [49] Tekabe Legesse Feleke, “The similarity and mutual intelligibility between Amharic and Tigrigna varieties,” in *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, Verona, Italy, Apr. 2017, pp. 47–54.
- [50] Tsegay Woldemariam, “A Morph-syntactic tag set for the Annotation of Texts in Tigrinya,” *Addis Ababa Univ. Addis Ababa Ethiop Master’s Degree Thesis*, Jun. 2013.
- [51] Abraham Negash, “The Origin and Development of Tigrinya Language Publications (1886-1991),” *Santa Clara University*, vol. 1, 2016.
- [52] Maria. Bulakh, “Tigrinya 1,” in *The Semitic Languages*, 2nd edition. Routledge, 2019.
- [53] D. L. Appleyard, “The internal classification of the Agaw languages: a comparative and historical phonology,” in *Current Progress in Afro-Asiatic Linguistics: Papers of the Third International Hamito-Semitic Congress*, Amsterdam, 1984, pp. 33–67.
- [54] Berhanu A. Agajie, “Operation labeling algorithm within Xamtanga sentences,” *Journal of Applied Studies in Language. Injibara College. Teaching. Education. Ethiopia*, vol. 4, no. 1, pp. 115–127, Jun. 2020.
- [55] David L. Appleyard, “A grammatical sketch of Khamtanga—II,” *Bulletin of the School of Oriental and African Studies*, vol. 50, no. 3, pp. 470–507, Oct. 1987.
- [56] C. Wedekind and K. Wedekind, *Sociolinguistic survey of the Awngi language of Ethiopia. SIL International (Société Internat. de Linguistique)*, 2002.
- [57] Li, Qian, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He., “A survey on text classification: From traditional to deep learning,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 2, pp. 1–41, Apr. 2022, doi: <https://doi.org/10.1145/3495162>.
- [58] A. Garg, V. Gupta, and M. Jindal, “A survey of language identification techniques and applications,” *Journal of Emerging Technologies in Web Intelligence*, vol. 6, no. 4, pp. 388–400, Nov. 2014.
- [59] M. Padró and L. Padró, “Comparing methods for language identification,” *Procesamiento del lenguaje natural. Spain*, vol. 33, 2004.
- [60] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, “Efficient estimation of word representations in vector space,” *ArXiv Prepr. ArXiv13013781*, January 2013.

- [61] K. Shaalan, "Rule-based approach in Arabic natural language processing," *The International Journal on Information and Communication Technologies (IJICT)*, University Edinburgh. UK, vol. 3, no. 3, pp. 11–19, Jun. 2010.
- [62] M. Rahimi, M. Youhanaee, and H. Barati, "A Short Analysis of Rule-based Linguistic Knowledge," *Theory & Practice in Language Studies*, vol. 4, no. 2, February 2014.
- [63] L. Grothe, E. W. De Luca, and A. Nürnberger, "A Comparative Study on Language Identification Methods," *In Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco., May 2008.
- [64] Abney, Steven, "Statistical methods and linguistics," *The balancing act: Combining symbolic and statistical approaches to language*, MIT press, Cambridge, pp. 1–26, 1996.
- [65] J. D. Prusa and T. M. Khoshgoftaar, "Designing a Better Data Representation for Deep Neural Networks and Text Classification," *In 2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, July 2016, pp. 411–416.
- [66] Qader, Wisam A., Musa M. Ameen, and Bilal I. Ahmed, "An Overview of Bag of Words; Importance, Implementation, Applications, and Challenges," in *2019 International Engineering Conference (IEC)*, January 2019, pp. 200–204.
- [67] Kim, Y., Jernite, Y., Sontag, D. and Rush, A., "Character-Aware Neural Language Models," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, Art. no. 1, Mar. 2016, doi: 10.1609/aaai.v30i1.10362.
- [68] X. Zhang, J. Zhao, and Y. LeCun, "Character-level Convolutional Networks for Text Classification," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2015.
- [69] Seger, Cedric, *An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing*. 2018.
- [70] A. Aizawa, "An information-theoretic perspective of TF–IDF measures," *Information Processing & Management*, vol. 39, no. 1, pp. 45–65, Jan. 2003, doi: 10.1016/S0306-4573(02)00021-3.
- [71] T. Vatanen, J. J. Väyrynen, and S. Virpioja, "Language Identification of Short Text Segments with N-gram Models," *Aalto, Finland: in Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC)*, May 2010.
- [72] W. B. Cavnar and J. M. Trenkle, "N-gram-based text categorization," in *the Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, Las Vegas, NV, Apr. 1994.
- [73] B. Ahmed, S.-H. Cha, and C. Tappert, "Language identification from text using n-gram based cumulative frequency addition," *Proceedings of Student/Faculty Research Day, CSIS, Pace University*, vol. 12, no. 12.8, May 2004.
- [74] J. Pennington, R. Socher, and C. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. doi: 10.3115/v1/D14-1162.

- [75] G. Grefenstette, “Comparing two language identification schemes,” in *Proceedings of JADT, 3rd International conference on Statistical Analysis of Textual Data*. Rome., December 1995.
- [76] C.O. Truica, J. Velcin, and A. Boicea, “Automatic language identification for romance languages using stop words and diacritics,” in *2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Bucharest, Romania: IEEE, Jun. 2015, pp. 243–246.
- [77] Souter, Clive, “Natural language identification using corpus-based models,” *HERMES-Journal of Language and Communication in Business*, no. 13, pp. 183–203, 1994.
- [78] A. Selamat and N. Akosu, “A Word-length algorithm for language identification of under-resourced languages,” *Journal of King Saud University-Computer and Information Sciences*, vol. 28, no. 4, pp. 457–469, Oct. 2016, doi: 10.1016/j.jksuci.2014.12.004.
- [79] S. Gadri, A. Moussaoui, and L. Belabdelouahab-Fernini, “Language identification: A new fast algorithm to identify the language of a text in a multilingual corpus,” in *2014 ICMCS*, Apr. 2014, pp. 321–326. doi: 10.1109/ICMCS.2014.6911338.
- [80] Y. Goldberg, “Neural network methods for natural language processing” *Synthesis lectures on human language technologies*, vol. 10, no. 1, pp. 1–309, Apr. 2017.
- [81] T. Young, D. Hazarika, S. Poria, and E. Cambria, “Recent Trends in Deep Learning Based Natural Language Processing”, *IEEE Computational intelligence magazine*, vol. 13, no. 3, pp. 55–75, Aug. 2018, doi: 10.1109/MCI.2018.2840738.
- [82] S. Agatonovic-Kustrin and R. Beresford, “Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research,” *Journal of pharmaceutical and biomedical analysis*, vol. 22, no. 5, pp. 717–727, 2000.
- [83] J. Cai, J. Li, W. Li, and J. Wang, “Deep learning Model Used in Text Classification,” in *2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, December 2018, pp. 123–126. doi: 10.1109/iccwamtip.2018.8632592.
- [84] Hounmenou, Castro Gbememali, Kossi Essona Gneyou, and Romain Lucas GLELE KAKAI, “A Formalism of the General Mathematical Expression of Multilayer Perceptron Neural Networks,” 2021, doi: 10.20944/preprints202105.0412.v1.
- [85] S. Vieira, W. H. L. Pinaya, and A. Mechelli, “Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications,” *Neuroscience & Biobehavioral Reviews*, vol. 74, pp. 58–75, Mar. 2017, doi: 10.1016/j.neubiorev.2017.01.002.
- [86] M. Aqib, R. Mehmood, A. Alzahrani, I. Katib, A. Albeshri, and S. M. Altowaijri, “Smarter Traffic Prediction Using Big Data, In-Memory Computing, Deep Learning and GPUs,” *Sensors*, vol. 19, no. 9, Art. no. 9, Jan. 2019, doi: 10.3390/s19092206.

- [87] Yoon Kim, “Convolutional Neural Network for Sentence Classification” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. doi: 10.3115/v1/D14-1181.
- [88] Jacovi, Alon, Oren Sar Shalom, and Yoav Goldberg, “Understanding Convolutional Neural Networks for Text Classification.” arXiv preprint, April 27, 2020. doi: 10.48550/arXiv.1809.08037.
- [89] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, “Very Deep Convolutional Networks for Text Classification.” arXiv, Jan. 27, 2017. doi: 10.48550/arXiv.1606.01781.
- [90] A. Sherstinsky, “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network,” *Physica D: Nonlinear Phenomena.*, vol. 404, p. 132306, Mar. 2020, doi: 10.1016/j.physd.2019.132306.
- [91] R David E., Geoffrey E. Hinton, and Ronald J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, Art. no. 6088, Oct. 1986, doi: 10.1038/323533a0.
- [92] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, “Backpropagation and the brain,” *Nature Reviews Neuroscience*, vol. 21, no. 6, Art. no. 6, Jun. 2020, doi: 10.1038/s41583-020-0277-3.
- [93] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings*, Mar. 2010, pp. 249–256.
- [94] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *ArXiv Prepr. ArXiv14126980*, December, 2014.
- [95] J. Zhang, “Gradient descent based optimization algorithms for deep learning models training” *ArXiv Prepr. ArXiv190303614*, Mar 11, 2019.
- [96] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization”, *Journal of machine learning research*, vol. 12, no. 7, July 1, 2011.
- [97] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
- [98] Hossin, Mohammad, and Md Nasir Sulaiman. Review on Evaluation Metrics for Data Classification Evaluations,” *International journal of data mining & knowledge management process*, vol. 5, no. 2, pp. 01–11, Mar. 2015, doi: 10.5121/ijdkp.2015.5201.
- [99] M. Grandini, E. Bagli, and G. Visani, “Metrics for Multi-Class Classification: an Overview”, *arXiv preprint arXiv*, August 13, 2020. doi: 10.48550/arXiv.2008.05756.
- [100] T. Jauhiainen, K. Lindén, and H. Jauhiainen, “Language model adaptation for language and dialect identification of text,” *Natural Language Engineering*, vol. 25, no. 5, pp. 561–583, Sep. 2019, doi: 10.1017/S135132491900038X.

- [101] E. Oro, M. Ruffolo, and M. Sheikhalishahi, "Language Identification of Similar Languages using Recurrent Neural Networks," *In International Conference on Agents and Artificial Intelligence*, 2018. doi: 10.5220/0006678606350640.
- [102] A. R. Dennis and J. S. Valacich, "Conducting Experimental Research in Information Systems," *Communications of the association for information systems*, vol. 7, Jul. 2001, doi: 10.17705/1CAIS.00705.
- [103] Yogesh Kumar Singh, "*Fundamentals of Research Methodology and Statistics*", New Age International (P) Limited, Publishers, New Delhi, 2006.
- [104] Fletcher, Sam, and Md Zahidul Islam, "Comparing sets of patterns with the Jaccard index," *Australasian Journal of Information Systems*, vol. 22, Mar. 2018.
- [105] W. Ling *et al.*, "Finding function in form: Compositional character models for open vocabulary word representation," *ArXiv Prepr. ArXiv150802096*, Aug. 2015.
- [106] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, 2015.
- [107] Y. Xu and R. Goodacre, "On splitting training and validation set: a comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning," *Journal of analysis and testing*, vol. 2, no. 3, pp. 249–262, 2018.
- [108] V. G. Raju, K. P. Lakshmi, V. M. Jain, A. Kalidindi, and V. Padma, "Study the influence of normalization/transformation process on the accuracy of supervised classification", in *2020 Third ICSSIT*, IEEE, Aug. 2020, pp. 729–735.
- [109] V. Dogra, S. Verma, P. Chatterjee, J. Shafi, J. Choi, and M. F. Ijaz, "A complete process of text classification system using state-of-the-art NLP models," *Comput. Intell. Neurosci.*, vol. 2022, 2022.
- [110] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based Hyperparameter Optimization through Reversible Learning," in *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, Jun. 2015, pp. 2113–2122.
- [111] Ruder, Sebastian, "An overview of gradient descent optimization algorithms," *ArXiv*, September 2016.
- [112] Lillicrap, Timothy P., Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton, "Backpropagation and the brain," *Nature Reviews Neuroscience*, vol. 21, no. 6, pp. 335–346, Jun. 2020.
- [113] Jason Brownlee, *Deep Learning with Python: Develop Deep Learning Models on Theano and TensorFlow Using Keras*, *Machine Learning Mastery*, vol. 19. May 13 2016.

APPENDIXES

Appendix I: Closely Related Ethiopic and South Arabian Abjad Scripts

Ancient South Arabian script		Ethiopic script	
<i>Letter</i>	<i>Name</i>	<i>Letter</i>	<i>Name</i>
𐩦	he	ሀ	ha
𐩧	lamedh	ለ	le
𐩨	heth	ሐ	ha
𐩩	mem	መ	me
𐩪	sat	ሰ	se
𐩫	resh	ረ	re
𐩬	shin	ሠ	se
𐩭	qoph	ቀ	qe
𐩮	beth	በ	be
𐩯	taw	ተ	te
𐩰	kheth	ኀ	ha
𐩱	nun	ነ	ne
𐩲	alef	አ	a
𐩳	kaph	ከ	ke
𐩴	waw	ወ	we
◦	ayn	ዐ	a
𐩶	zayn	ዘ	ze
𐩷	yodh	የ	ye
𐩸	daleth	ደ	de
𐩹	gimel	ገ	ge
𐩺	teth	ጠ	te
𐩻	sadhe	ጸ	tse
𐩼	dhadhe	ፀ	tse
𐩽	fe	ፈ	fe
𐩾	samekh	-	
𐩿	ghayn	-	
𐊀	dhaleth	-	
𐊁	thaw	-	
𐊂	theth	-	

Appendix II: The Current (HaLeHaMe “ሀለሐመ”) Arrangement of the Ge'ez Alphabet

C	Vowel Orders						
	1 st (ግእዝ)	2 nd (ካዕብ)	3 rd (ሣልስ)	4 th (ራብዕ)	5 th (ኅምስ)	6 th (ሳድስ)	7 th (ሳብዕ)
	C+ä (አ)	C+u (ኡ)	C+i (ኢ)	C+a (ኣ)	C+e (ኤ)	C+ə/i (ኦ)	C+o (ኦ)
H	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
L	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
H	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሐ
M	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
S	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
R	ረ	ሩ	ሪ	ራ	ራ	ር	ሮ
S	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
Q	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
B	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
T	ተ	ቱ	ቲ	ታ	ቲ	ት	ቶ
H	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
N	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
A	አ	ኡ	ኢ	ኣ	ኤ	ኦ	ኦ
K	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
W	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
A	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
Z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
Y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
D	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
G	ገ	ገ	ጊ	ጋ	ጌ	ግ	ጊ
T'	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
P'	ጳ	ጴ	ጵ	ጶ	ጷ	ጸ	ጹ
TS	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
TS	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
F	ፈ	ፉ	ፊ	ፋ	ፅ	ፍ	ፎ
P	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
Labialized variant (Labiovelars)							
QW	ቄ		ቄ	ቄ	ቄ	ቄ	
HW	ኀ		ኀ	ኃ	ኃ	ኀ	
KW	ከ		ከ	ካ	ካ	ከ	
GW	ገ		ገ	ጋ	ጋ	ገ	

Letter “C” represents consonants.

Appendix III: The Earlier (ABeGeDe “አበገደ”) Arrangement of the Ge'ez Alphabet

Similar to the current arrangement of Geez alphabet (Fidel) the former has 26 base characters and 4 labiovelars. The order of the base Ge'ez characters is as follows: አ በ ገ ደ ሀ ወ ዘ ሐ ጎ ጠ የ ከ ለ መ ነ ሠ ዐ ፈ ጸ ፀ ቀ ረ ሰ ተ ጰ ፐ and labiovelar consonants: ቈ ኅ ኈ ነ. The construction of the alpha-syllabary of አበገደ order is similar to the current Geez Fidel as shown in Appendix II.

Appendix IV: The Ge'ez Numbers and Punctuation Marks

A) Ge'ez (Ethiopic) Numbers

፩	፪	፫	፬	፭	፮	፯	፰	፱	፲
1	2	3	4	5	6	7	8	9	10
፳	፴	፵	፶	፷	፸	፹	፺	፻	፷፱
20	30	40	50	60	70	80	90	100	10000

B) Ge'ez (Ethiopic) Punctuation Marks

Punctuations Marks	St. Yared Zema Marks		
:	Ethiopic word space	፡	ይዘት “yizet”
::	Ethiopic full stop	፡፡	ኅጻር ርክርክ “hitsir rikrk”
፤	Ethiopic semi-colon	፤	ርክርክ “rikrk”
፣	Ethiopic a comma	፣	ደረት “deret”
፥	Ethiopic colon	፥	ድፋት “difat”
:-	Ethiopic preface colon	፥-	ቅረት “chiret”
፥	Ethiopic question mark	፥	ቅናት “qinat”
፥፥	Ethiopic paragraph separator	፥፥	ሐደት “hidet”
※	Ethiopic a section mark	፥፥	ደረት ሐደት “deret hidet”
		፡	ቁርጥ “qurt”

Appendix V: The Amharic Alpha-syllabic (Fidel)

C	Vowel Orders						
	1 st (ግእዝ)	2 nd (ካዕብ)	3 rd (ሣልስ)	4 th (ራብዕ)	5 th (ጎምስ)	6 th (ሰድስ)	7 th (ሰብዕ)
	C+ä (አ)	C+u (ኡ)	C+i (ኢ)	C+a (ኣ)	C+e (ኤ)	C+ə/ï (እ)	C+o (ኦ)
H	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
L	ለ	ሉ	ሊ	ላ	ሌ	ሎ	ሎ
H	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሐ
M	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
S	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
R	ረ	ሩ	ሪ	ራ	ሪ	ሪ	ሪ
S	ሰ	ሱ	ሲ	ሳ	ሴ	ሶ	ሰ
SH	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሸ
Q	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
B	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
T	ተ	ቱ	ቲ	ታ	ቲ	ቲ	ቲ
CH	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቸ
H	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
N	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
ጐ	ኘ	ኙ	ኚ	ና	ኔ	ን	ኖ
A	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
K	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
H'	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኸ
W	ወ	ዉ	ዊ	ዋ	ዌ	ወ	ዐ
A	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
Z	ዘ	ዙ	ዚ	ዛ	ዞ	ዟ	ዠ
ZH	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
Y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
D	ደ	ዱ	ዲ	ዳ	ዤ	ዶ	ደ
J	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
G	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ገ
T'	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጠ
CH'	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጠ
P'	ቆ	ቆ	ቆ	ቆ	ቆ	ቆ	ቆ
TS	ጸ	ጸ	ጸ	ጸ	ጸ	ጸ	ጸ
TS	ፀ	ፀ	ፀ	ፀ	ፀ	ፀ	ፀ
F	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፍ
P	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
V	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ
Labialized variant (Labiovelars)							
QW	ቁ		ቁ	ቁ	ቁ	ቁ	
HW	ኀ		ኀ	ኀ	ኀ	ኀ	
KW	ከ		ከ	ከ	ከ	ከ	
GW	ገ		ገ	ገ	ገ	ገ	
Other Labiovelars							
	ቃ	ቃ	ቃ	ቃ	ቃ	ቃ	ቃ
	ቃ	ቃ	ቃ	ቃ	ቃ	ቃ	ቃ
	ቃ	ቃ	ቃ	ቃ	ቃ	ቃ	ቃ

Appendix VI: The Awngi Alpha-syllabic (Fidel)

C	Vowel Orders						
	1 st (ግእዝ)	2 nd (ካዕብ)	3 rd (ሣልስ)	4 th (ራብዕ)	5 th (ኅምስ)	6 th (ሳድስ)	7 th (ሳብዕ)
	C+ä (አ)	C+u (ኡ)	C+i (ኢ)	C+a (ኣ)	C+e (ኤ)	C+ə/ī (ኦ)	C+o (ኦ)
H	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
L	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
M	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
R	ረ	ሩ	ሪ	ራ	ራ	ር	ሮ
S	ሰ	ሱ	ሲ	ሳ	ሴ	ሶ	ሶ
SH	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
Q'	ቆ	ቅ	ቂ	ቃ	ቄ	ቅ	ቆ
B	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
T	ተ	ቱ	ቲ	ታ	ቲ	ት	ቶ
CH	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቸ
N	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
A	አ	ኡ	ኢ	ኣ	ኤ	ኦ	ኦ
K	ካ	ኡ	ኢ	ኣ	ኤ	ኦ	ኦ
H'	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
W	ወ	ዉ	ዊ	ዋ	ዌ	ወ	ዐ
Z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
Y	የ	ዩ	ዪ	ያ	ይ	ይ	ዮ
D	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
J	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
G	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ገ
ገ	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
TS	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
F	ፈ	ፉ	ፊ	ፋ	ፅ	ፍ	ፎ
P	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
V	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ
Labialized variant (Labiovelars)							
KW	ኰ		ኲ	ኳ	ኴ	ኵ	኶
GW	ጎ		጑	ጒ	ጓ	ጔ	ጕ
ገW	ኸ		ኺ	ኻ	ኼ	ኽ	ኾ
H'W	ኸ		ኺ	ኻ	ኼ	ኽ	ኾ
Q'W	ቆ		ቂ	ቃ	ቄ	ቅ	ቆ

Appendix VII: The Guragigna Orthography (Fidel)

The 1966 Gurage syllabary alphabetical order, palatalized velars and labialized velars.

ኸ	ኸ'	ለ	መ	ረ	ሰ	ሸ	ቀ	ቀ'	በ	ተ	ቸ	ነ	ኘ	አ	ከ	ኸ	ወ	ዘ	ዠ	የ	ደ	ጀ	ገ	ገ'	ጠ	ጨ	ጰ	ፀ	ፈ	ፐ
h	h'	l	m	r	s	she	q	q'	b	t	che	n	gn	a'	k	K'	w	z	zhe	y	d	J'	g	g'	t'	che'	p'	tse	f	p

Palatized Velars							
Base	ə	u	i	a	e	ɨ	o
ቀ (q')	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
ከ (k')	ከ	ኩ	ኪ	ካ	ኼ	ኽ	ኾ
ኸ (h')	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
ገ (g')	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ግ'

Added Labialized Velars				
Base	wə	wi	wɨ	wi
መ (m)	መ	ማ	ሜ	ሞ
በ (b)	ቡ	ባ	ቤ	ቦ
ፈ (f)	ፈ	ፉ	ፊ	ፋ
ፐ (p)	ፐ	ፑ	ፒ	ፓ

Regular Labialized Velars				
Base	wə	wi	wɨ	wi
ኸ (h)	ኸ	ኹ	ኺ	ኻ
ቀ (q)	ቀ	ቁ	ቂ	ቃ
ከ (k)	ከ	ኩ	ኪ	ካ
ገ (g)	ገ	ጉ	ጊ	ጋ

The 1977 and 1998 Gurage orthography alphabetical order, palatalized velars and labialized velars. As shown in the tables below, both Gurage alphabets are almost similar except for the added labialized velar.

ኸ	ኸ'	ለ	መ	ረ	ሰ	ሸ	ቀ	ቀ'	በ	ተ	ቸ	ነ	ኘ	አ	ከ	ኸ	ወ	ዘ	ዠ	የ	ደ	ጀ	ገ	ገ'	ጠ	ጨ	ጰ	ፀ	ፈ	ፐ
h	h'	l	m	r	s	she	q	q'	b	t	che	n	gn	a'	k	K'	w	z	zhe	y	d	J'	g	g'	t'	che'	p'	tse	f	p

Palatized Velars							
Base	ə	u	i	a	e	ɨ	o
ቀ (q')	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
ከ (k')	ከ	ኩ	ኪ	ካ	ኼ	ኽ	ኾ
ኸ (h')	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
ገ (g')	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ግ'

1977 Added Labialized Velars				
Base	wə	wi	wɨ	wi
መ (m)	መ	ማ	ሜ	ሞ
በ (b)	ቡ	ባ	ቤ	ቦ
ፈ (f)	ፈ	ፉ	ፊ	ፋ
ፐ (p)	ፐ	ፑ	ፒ	ፓ

Regular Labialized Velars				
Base	wə	wi	wɨ	wi
ኸ (h)	ኸ	ኹ	ኺ	ኻ
ቀ (q)	ቀ	ቁ	ቂ	ቃ
ከ (k)	ከ	ኩ	ኪ	ካ
ገ (g)	ገ	ጉ	ጊ	ጋ

1998 Added Labialized Velars				
Base	wə	wi	wɨ	wi
መ (m)	መ	ማ	ሜ	ሞ
በ (b)	ቡ	ባ	ቤ	ቦ
ፈ (f)	ፈ	ፉ	ፊ	ፋ
ፐ (p)	ፐ	ፑ	ፒ	ፓ

The modern (2013) Gurage orthography alphabetical order, palatalized velars and labialized velars.



ደቡብ ብሔራዊ/ሕዝብ/መንግሥት
የጉራጌ ዞን ዋና አስተዳዳሪ ጽ/ቤት

ወልቲጤ

SNNP Regional state Gurage Zone Chief Admin. Office
Wolkite

Reference No ዘአ/1877/ዓ-6

Date 29/04/2013

To: The Unicode Consortium

From: Gurage Zone Administration

Subject: Unicode for Gurage Orthography

For over half a century the Gurage language has used the Amharic Fidel with some additional letters developed for sounds which are not found in Amharic. It is not clearly known who shaped these additional letters. The writing system was used to publish some fictional works in Gurage language as well as a New Testament, called *Geder Gurda*, and the complete Bible called *Met'af Qidus*. As Gurage language was not in the past used in institutions such as education, media and administration bureaus, the orthography was not officially recognized. Recently, the need for introducing the Gurage language for education necessitated improvement of the pre-existing orthography in order to make it simple and pattern based so that students can learn it quickly and write without difficulty. The new orthography was evaluated continuously by linguists and educators through several workshops since 2020. It was finally officially endorsed by Gurage Zone Administration Council in 29/01/2013. The orthography has also been introduced on a limited basis in the school system since September 2020. This letter is therefore to request the Unicode Consortium help us with our primary technical challenge by adopting the letters in the Unicode Standard so that we may then enjoy the benefit of using our language in all modern digital devises. A table of the Gurage Fider ("Alphabet") is presented on the following page.

Sincerely yours,

JLS

መሃመድ ጅማል ሸሪፍ
Mohammed Jamal Sherif

ወራሪ ዞን አስተዳዳሪ ዋና
አስተዳዳሪ
Gurage Zone President
Chief Administrator



የጉራጊና ፊደር
(Gurage Alphabet)

ሐ	ሐ	ሐ	ሐ	ሐ	ሐ	ሐ
ሸ	ሸ	ሸ	ሸ	ሸ	ሸ	ሸ
ለ	ለ	ለ	ለ	ለ	ለ	ለ
መ	መ	መ	መ	መ	መ	መ
ረ	ረ	ረ	ረ	ረ	ረ	ረ
ሰ	ሰ	ሰ	ሰ	ሰ	ሰ	ሰ
ሸ	ሸ	ሸ	ሸ	ሸ	ሸ	ሸ
ቀ	ቀ	ቀ	ቀ	ቀ	ቀ	ቀ
ቆ	ቆ	ቆ	ቆ	ቆ	ቆ	ቆ
በ	በ	በ	በ	በ	በ	በ
ሸ	ሸ	ሸ	ሸ	ሸ	ሸ	ሸ
ተ	ተ	ተ	ተ	ተ	ተ	ተ
ቸ	ቸ	ቸ	ቸ	ቸ	ቸ	ቸ
ኃ	ኃ	ኃ	ኃ	ኃ	ኃ	ኃ
ሃ	ሃ	ሃ	ሃ	ሃ	ሃ	ሃ
አ	አ	አ	አ	አ	አ	አ
ከ	ከ	ከ	ከ	ከ	ከ	ከ
ኸ	ኸ	ኸ	ኸ	ኸ	ኸ	ኸ
ወ	ወ	ወ	ወ	ወ	ወ	ወ
ዐ	ዐ	ዐ	ዐ	ዐ	ዐ	ዐ
ዘ	ዘ	ዘ	ዘ	ዘ	ዘ	ዘ
ዠ	ዠ	ዠ	ዠ	ዠ	ዠ	ዠ
የ	የ	የ	የ	የ	የ	የ
ደ	ደ	ደ	ደ	ደ	ደ	ደ
ጀ	ጀ	ጀ	ጀ	ጀ	ጀ	ጀ
ገ	ገ	ገ	ገ	ገ	ገ	ገ
ገ	ገ	ገ	ገ	ገ	ገ	ገ
ጠ	ጠ	ጠ	ጠ	ጠ	ጠ	ጠ
ጨ	ጨ	ጨ	ጨ	ጨ	ጨ	ጨ
ጳ	ጳ	ጳ	ጳ	ጳ	ጳ	ጳ
ጸ	ጸ	ጸ	ጸ	ጸ	ጸ	ጸ
ረ	ረ	ረ	ረ	ረ	ረ	ረ
ፕ	ፕ	ፕ	ፕ	ፕ	ፕ	ፕ

ሐ	ሐ	ሐ	ሐ	ሐ
መ	መ	መ	መ	መ
ቀ	ቀ	ቀ	ቀ	ቀ
በ	በ	በ	በ	በ
ከ	ከ	ከ	ከ	ከ
ገ	ገ	ገ	ገ	ገ
ደ	ደ	ደ	ደ	ደ
ፕ	ፕ	ፕ	ፕ	ፕ



Appendix VIII: The Tigrinya Alpha-syllabic (Fidel)

C	Vowel Orders						
	1 st (ግእዝ)	2 nd (ካዕብ)	3 rd (ሣልስ)	4 th (ራብዕ)	5 th (ጎምስ)	6 th (ሳድስ)	7 th (ሳብዕ)
	C+ä (አ)	C+u (ኡ)	C+i (ኢ)	C+a (ኣ)	C+e (ኤ)	C+ə/ī (ኦ)	C+o (ኦ)
H	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
L	ለ	ሉ	ሊ	ላ	ሌ	ሎ	ሎ
H	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሐ
M	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
S	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
R	ረ	ሩ	ሪ	ራ	ራ	ሮ	ሮ
S'	ሰ	ሱ	ሲ	ሳ	ሴ	ሶ	ሶ
SH	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሸ
Q	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
Q'	ቐ	ቑ	ቒ	ቓ	ቄ	ቅ	ቆ
B	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
T	ተ	ቱ	ቲ	ታ	ቲ	ቲ	ቲ
CH	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቸ
N	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
ጐ	ኘ	ኙ	ኚ	ኝ	ኞ	ኟ	ኞ
A	አ	ኡ	ኢ	ኣ	ኤ	ኦ	ኦ
K	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
H'	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኸ
W	ወ	ዉ	ዊ	ዋ	ዌ	ወ	ዐ
A	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዐ
Z	ዘ	ዙ	ዚ	ዛ	ዛ	ዘ	ዘ
ZH	ዘ	ዙ	ዚ	ዛ	ዛ	ዘ	ዘ
Y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
D	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
J	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
G	ገ	጑	ጒ	ጓ	ጔ	ጕ	጖
T'	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
CH'	ጨ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
P'	ቆ	ቆ	ቆ	ቆ	ቆ	ቆ	ቆ
TS	ቆ	ቆ	ቆ	ቆ	ቆ	ቆ	ቆ
TS'	ቆ	ቆ	ቆ	ቆ	ቆ	ቆ	ቆ
F	ፈ	ፉ	ፊ	ፋ	ፈ	ፍ	ፎ
P	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
V	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ

Labialized variant (Labiovelars)						
QW	ቁ		ቁ	ቁ	ቁ	ቁ
KW	ከ		ከ	ከ	ከ	ከ
GW	ጐ		ጐ	ጐ	ጐ	ጐ
H'W	ኸ		ኸ	ኸ	ኸ	ኸ
Q'W	ቐ		ቐ	ቐ	ቐ	ቐ

Other Labiovelars						
	ቧ	ቧ	ቧ	ቧ	ቧ	ቧ
	ቧ	ቧ	ቧ	ቧ	ቧ	ቧ

Appendix IX: The Xamtanga Alpha-syllabic (Fidel)

C	Vowel Orders						
	1 st (ግእዝ)	2 nd (ካዕብ)	3 rd (ሣልስ)	4 th (ራብዕ)	5 th (ጎምስ)	6 th (ሰድስ)	7 th (ሰብዕ)
	C+ä (አ)	C+u (ኡ)	C+i (ኢ)	C+a (ኣ)	C+e (ኤ)	C+ə/ī (እ)	C+o (ኦ)
H	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
L	ለ	ሉ	ሊ	ላ	ሌ	ሎ	ሎ
H	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሐ
M	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
R	ረ	ሩ	ሪ	ራ	ራ	ር	ሮ
S	ሰ	ሱ	ሲ	ሳ	ሴ	ሶ	ሶ
SH	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሸ
Q	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
Q'	ቐ	ቑ	ቒ	ቃ	ቄ	ቅ	ቆ
B	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
T	ተ	ቱ	ቲ	ታ	ቲ	ቲ	ቲ
CH	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቸ
H	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
N	ነ	ኑ	ኒ	ና	ኑ	ን	ኖ
A	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
K	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
H	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
W	ወ	ዉ	ዊ	ዋ	ዌ	ወ	ወ
Z	ዘ	ዙ	ዚ	ዛ	ዜ	ዘ	ዘ
Y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
D	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
J	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
G	ገ	጑	ጒ	ጓ	ጔ	ጕ	጖
ገ	ገ	጑	ጒ	ጓ	ጔ	ጕ	጖
T'	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
CH'	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
P'	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
TS	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
TS	ፀ	ፁ	ፒ	ፓ	ፔ	ፅ	ፆ
F	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
P	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
V	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ

Labialized variant (Labiovelars)

KW	ከጐ	ከ጑	ከጒ	ከጓ	ከጔ	ከጕ
GW	ገጐ	ገ጑	ገጒ	ገጓ	ገጔ	ገጕ
ገW	ገጐ	ገ጑	ገጒ	ገጓ	ገጔ	ገጕ
HW	ከጐ	ከ጑	ከጒ	ከጓ	ከጔ	ከጕ
H'W	ከጐ	ከ጑	ከጒ	ከጓ	ከጔ	ከጕ
QW	ቀጐ	ቀ጑	ቀጒ	ቀጓ	ቀጔ	ቀጕ
Q'W	ቀጐ	ቀ጑	ቀጒ	ቀጓ	ቀጔ	ቀጕ

Appendix X: Sample Python Source Code

Importing essential Python packages for preprocessing, training, and testing

```
#Importing packages
import os
import random
import numpy as np
import tensorflow as tf
import time

from sklearn import preprocessing
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split

import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
import keras.optimizers
from keras.layers import LayerNormalization

# This code was tested with TensorFlow v1.8
print("You have TensorFlow version", tf.__version__)
```

Python program to encode the label for each language, to configure the length of sample texts, and the number of language samples per language.

```
# Configuration

# dictionary of languages that our classifier will cover
LANGUAGES_DICT = {'amharic':0, 'awngi':1, 'geez':2, 'guragigna':3, 'tigrigna':4, 'xamtanga':5}

# Length of cleaned text used for training and prediction - 100 chars
MAX_LEN = 100 # Number of character

# number of language samples per language that we will extract from source files (# rows)
NUM_SAMPLES = 100000

# For reproducibility
SEED = 42

from support import define_alphabet
# Load the unique Alphabet
alphabet = define_alphabet()
print('ALPHABET:')
print(alphabet[1])

VOCAB_SIZE = len(alphabet[1])
print('ALPHABET LEN(VOCAB SIZE):', VOCAB_SIZE)
```

Sample python program to analysis the average word length of Tigrigna language

```
f=open("All_Cleaned_Data/TigrignaCleaned.txt", 'r', encoding="utf-8")
# read the file
data = f.read()

# calculate average word length

words = data.split()
total_words = 0

for word in words:
    total_words += len(word)

average_word_length = total_words / len(words)

print("Total words: ", total_words)
print("Number of words: ", len(words))
print(f"Average word length: {average_word_length}")
print("Truncate:", float(f'{average_word_length:.2f}'))
```

Python code for data scaling of independent and dependent variables

```
# X PREPROCESSING
# Feature Standardization - Standar scaler will be useful later during DNN prediction
standard_scaler = preprocessing.StandardScaler().fit(X)
X = standard_scaler.transform(X)
print ("X preprocessed shape :", X.shape)

# Y PREPROCESSING
# One-hot encoding
Y = keras.utils.to_categorical(Y, num_classes=len(LANGUAGES_DICT))
```

Python program to show the training and test set

```
# Load train data first from file
path_tt = os.path.join(train_test_directory, "train_test_data_"+str(VOCAB_SIZE)+".npz")
train_test_data = np.load(path_tt)

# Train Set
X_train = train_test_data['X_train']
print ("X_train: ",X_train.shape)
Y_train = train_test_data['Y_train']
print ("Y_train: ",Y_train.shape)

# Test Set
X_test = train_test_data['X_test']
print ("X_test: ",X_test.shape)
Y_test = train_test_data['Y_test']
print ("Y_test: ",Y_test.shape)

del train_test_data

X_train: (480000, 405)
Y_train: (480000, 6)
X_test: (120000, 405)
Y_test: (120000, 6)
```

Python program to build the DNN model

```
model = Sequential()
model.add(Dense(512, input_dim=input_size, kernel_initializer="glorot_uniform", activation="sigmoid"))
# Note: glorot_uniform is the Xavier uniform initializer.
model.add(Dropout(drop_ratio))
model.add(Dense(256, kernel_initializer="glorot_uniform", activation="sigmoid"))
model.add(Dropout(drop_ratio))
model.add(Dense(128, kernel_initializer="glorot_uniform", activation="sigmoid"))
model.add(Dropout(drop_ratio))
model.add(Dense(len(LANGUAGES_DICT), kernel_initializer="glorot_uniform", activation="softmax"))
#model_optimizer = keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08,
model.compile(loss='categorical_crossentropy', optimizer="Adam", metrics=['accuracy'])
model.summary()
```

Python program to train the model on train set

```
from keras.callbacks import TensorBoard

# Tensorboard
tensorboard = TensorBoard(log_dir='logs', update_freq='epoch', write_graph=True, profile_batch=0)
tf.profiler.experimental.stop

history = model.fit(X_train, Y_train,
                    epochs=epochs,
                    validation_split=0.1,
                    batch_size=batch_size,
                    callbacks = [tensorboard],
                    shuffle=True,
                    verbose=2)
```

Python program to evaluate the model on test set

```
# Evaluate the accuracy of our trained model on test set
scores = model.evaluate(X_test, Y_test, batch_size=batch_size, verbose=2)
print("Test %s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

Python program to view the confusion matrix

```
# Plot confusion matrix
from sklearn.metrics import confusion_matrix
from support import print_confusion_matrix

cnf_matrix = confusion_matrix(np.argmax(Y_pred,axis=1), np.argmax(Y_test,axis=1))
_ = print_confusion_matrix(cnf_matrix, LABELS)
```

Python program to show the classification result

```
# Classification Report
print(classification_report(Y_test, Y_pred, target_names=LABELS))
```