MEKELLE UNIVERSITY

ETHIOPIAN INSTITUTE OF TECHNOLOGY-MEKELLE(EIT-M)

SCHOOL OF COMPUTING

Department of Computer Science (MSc Regular)

A

Thesis on

Word Sequence Prediction Model for the Tigrigna Language Using a Deep Learning Approach

Submitted by

Yibralem Hagos Mekonnen

EITM/PR160738/11

Mekelle, Ethiopia

July, 2025

MEKELLE UNIVERSITY
ETHIOPIAN INSTITUTE OF TECHNOLOGY-MEKELLE (EIT-M)
SCHOOL OF COMPUTING
Department of Computer Science (MSc Regular)

A

Thesis on

Word Sequence Prediction Model for the Tigrigna Language Using a Deep Learning Approach

Submitted by
Yibralem Hagos Mekonnen
EITM/PR160738/11

Advisor
Dr. Shishay Welay Gebregiyorgis(PhD)
Mekelle University

Co-Advisor
Ataklti Kahsu(PhD Candidate at Ethiopian Institute of Technology-Mekelle (EIT-M))
Mekelle Institute of Technology
Mekelle University

A thesis submitted to School of Computing in partial fulfillment of the degree of Master of Science in Computer Science

Mekelle, Ethiopia

October 2025

MEKELLEUNIVERSITY

ETHIOPIAN INSTITUTE OF TECHNOLOGY-MEKELLE(EIT-M)

SCHOOL OF COMPUTING

Word Sequence Prediction Model for the Tigrigna Language Using a Deep Learning Approach

Yibralem Hagos Mekonnen

Selama Gebremeskel

Jan 26, 2026

Chairman, dept. graduate committee — Signature

Dr.Shishay Welay Gebregiyorgis (Phd)

Advisor — Signature

External Examiner — Signature

Internal Examiner — Signature

Copyright

ACKNOWLEDGMENT

First and foremost, I thank the Lord Almighty for His guidance, strength, and wisdom throughout the journey of completing this thesis. His faithfulness has been my anchor in every step of this work.

I am deeply grateful to my main advisor, **Dr. Shishay Welay Gebregiyorgis**, for his invaluable guidance, insightful feedback, and unwavering support throughout my research. I also extend my heartfelt thanks to my co-advisor, **Ataklti Kahsu (PhD candidate)**, for his encouragement, constructive suggestions, and dedication in assisting me during this study.

Finally, I wish to thank my family, friends, and all those who have prayed for me and supported me in countless ways throughout this journey. To God be the glory, forever and ever.

DECLARATION

Confidentiality/ No disclosure

Herewith, I declare that the data used for this research has been kept strictly confidential and used solely for the purpose of this study. No information related to the identity of individuals, personal addresses, or any other identifying information was utilized. The dataset consists solely of textual data from public sources, primarily the Old Testament of the Tigrigna Holy Bible, and does not contain any sensitive or personally identifiable information.

All text data has been processed to ensure anonymity where applicable. Any representations or transformations of the text do not link to any individual or entity, and no private or sensitive content has been disclosed. The data collected and used to train and evaluate the deep learning word sequence prediction model will be held in strict confidence.

The algorithms, code, and models developed in this research will be made openly accessible to researchers and practitioners for further study and application.

This study has been conducted in accordance with ethical research standards. I hereby declare that all information submitted in this report is accurate, true, and valid. I will provide supporting documentation as required.

Yibralem Hagos Mekonnen

EITM/PR160738/11

Date: October 10, 2025

ABSTRACT

This research explores the development of a word sequence prediction model for the Tigrigna language using deep learning techniques, specifically Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). Tigrigna, primarily spoken in Eritrea and Ethiopia, faces significant challenges in natural language processing (NLP) due to the scarcity of comprehensive computational resources and annotated corpora. This study addresses the urgent need for effective NLP tools tailored to Tigrigna, focusing on the fundamental task of word sequence prediction, which underpins various applications such as machine translation and text generation.

Despite the limited dataset of 10,000 sentences compiled from diverse sources, the models were evaluated for their ability to predict and generate coherent word sequences. Results indicate that while LSTM and GRU models demonstrated potential in capturing Tigrigna's unique linguistic characteristics, they faced issues with overfitting and underfitting, particularly influenced by the choice of embeddings Word2Vec and Keras Embedding. The findings highlight the necessity for improved regularization techniques and the importance of data augmentation to enhance model generalization.

This research contributes to the nascent field of Tigrigna NLP by demonstrating the applicability of deep learning models in resource-scarce languages. The outcomes suggest pathways for future advancements in Tigrigna language technology, emphasizing the potential for enhanced predictive text applications and deeper insights into Tigrigna's grammatical structures. Ultimately, this work lays a foundation for further developments in Tigrigna NLP, advocating for increased investment in linguistic resources and innovative modeling techniques to support the digital representation of the Tigrigna language.

Keywords: Tigrigna language, Natural Language Processing (NLP), Word Sequence Prediction, Deep Learning, Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Word2Vec, Keras Embedding, Language Modeling, Low-Resource Languages, Data Augmentation, Tigrigna NLP

TABLE OF CONTENTS

List of Tables

List of Figures

LIST OF ABBREVIATION AND ACRONYMS

Abbreviation / Acronym	Full Meaning
AI	Artificial Intelligence
ANN	Artificial Neural Network
BERT	Bidirectional Encoder Representations from Transformers
BLEU	Bilingual Evaluation Understudy (Score)
DL	Deep Learning
Epoch	One Complete Iteration of Model Training
F1	F1-Score (Harmonic Mean of Precision and Recall)
GRU	Gated Recurrent Unit
K	Keras (Deep Learning API)
LM	Language Model
LSTM	Long Short-Term Memory
LR	Learning Rate
ML	Machine Learning
MLP	Multi-Layer Perceptron
NLG	Natural Language Generation
NLM	Neural Language Model
NLP	Natural Language Processing
NN	Neural Network
NNLM	Neural Network Language Model
NMT	Neural Machine Translation
RNN	Recurrent Neural Network
RNNLM	Recurrent Neural Network Language Model
Seq2Seq	Sequence-to-Sequence Model
TF	TensorFlow

Abbreviation / Acronym	Full Meaning
Tigrigna NLP	Tigrigna Natural Language Processing
TTS	Text-to-Speech
W2V	Word2Vec (Word Embedding Technique)
Word2Vec	Word to Vector (Embedding Model)

CHAPTER ONE

1.1 Introduction

The Tigrigna language, spoken primarily in the Horn of Africa, is one of the dominant languages in Eritrea and Ethiopia. With its rich linguistic characteristics and cultural significance, there is a growing interest in developing computational models for Tigrigna natural language processing (NLP). Among various NLP tasks, word sequence prediction is fundamental and forms the basis for more complex language processing tasks such as machine translation, text summarization, and speech recognition. The development of computational models for the Tigrigna language presents both challenges and opportunities. While Tigrigna shares some linguistic features with other languages in the Afroasiatic family, it also exhibits unique characteristics that require specialized approaches in natural language processing. Unlike widely studied languages such as English, Tigrigna lacks comprehensive resources, including large annotated corpora and established language processing tools. This scarcity of resources poses significant hurdles for researchers aiming to build effective NLP systems for Tigrigna.

Despite these challenges, recent advancements in deep learning and neural network architectures offer promising avenues for addressing the complexities of Tigrigna language processing. LSTM networks, in particular, have demonstrated remarkable success in modeling sequential data, making them well-suited for capturing the intricate structures and patterns present in Tigrigna text.

In recent years, deep learning techniques, particularly Long Short-Term Memory (LSTM) networks, have shown promising results in modeling sequential data and have been widely adopted in NLP applications. LSTM networks, a type of recurrent neural network (RNN), excel at capturing long-range dependencies and patterns in sequential data, making them suitable for word sequence prediction tasks.

With the massive growth of the amount of data generated by the devices in the world around us, there has been a pressing need to incorporate this user-information-rich data to enhance the end users' experience. This can be achieved by training machine learning (ML) models using this data to drive various mobile applications running on Artificial Intelligence (AI).

The motivation behind this master's thesis stems from the urgent need to bridge the gap between the growing demand for Tigrigna language technology and the limited availability of computational resources tailored to this language. By developing an LSTM-based word sequence prediction model for Tigrigna, this research seeks to contribute to the nascent field of Tigrigna NLP and pave the way for future advancements in this domain.

Furthermore, the implications of this research extend beyond academic curiosity; they have tangible real-world applications. Effective word sequence prediction models for Tigrigna can empower various NLP applications, including machine translation systems that facilitate communication between Tigrigna speakers and speakers of other languages, sentiment analysis tools for analyzing Tigrigna social media content, and educational resources for teaching and learning Tigrigna.

In summary, this master's thesis endeavors to explore the potential of LSTM networks in modeling the complexities of the Tigrigna language. By leveraging deep learning techniques and a rich dataset of Tigrigna text, the goal is to develop a robust word sequence prediction model that not only advances the field of Tigrigna NLP but also fosters broader societal benefits through enhanced language technology capabilities.

1.2 Statement of the Problem:

The Tigrigna language, spoken primarily in Eritrea and Ethiopia, lacks comprehensive computational models and resources for natural language processing (NLP). Despite its cultural and linguistic significance, the development of NLP tools and technologies tailored to Tigrigna remains limited, hindering the advancement of various language-related applications and services for Tigrigna speakers.

One of the fundamental challenges in Tigrigna language processing is the absence of robust word sequence prediction models. Word sequence prediction forms the basis for numerous NLP tasks, including language modeling, machine translation, speech recognition, and text generation. Without accurate and efficient word sequence prediction models, it is challenging to develop effective NLP applications that can understand and generate coherent Tigrigna text.

Existing approaches to Tigrigna language processing often rely on generic NLP models trained on other languages, which may not capture the specific linguistic characteristics and nuances of Tigrigna. Furthermore, the scarcity of annotated Tigrigna corpora and linguistic resources complicates the development of language-specific models tailored to Tigrigna.

Therefore, the primary problem addressed in this master's thesis is the lack of a dedicated word sequence prediction model for the Tigrigna language. This problem encompasses several sub-issues, including:

1. Limited Availability of Tigrigna Language Resources: The scarcity of annotated corpora, linguistic tools, and language-specific datasets poses a significant challenge for training and evaluating NLP models for Tigrigna.
2. Linguistic Complexity of Tigrigna: Tigrigna exhibits unique linguistic features, such as agglutination, morphological richness, and complex syntax, which require specialized modeling techniques for accurate word sequence prediction.
3. Generalization and Adaptation: Existing NLP models trained on other languages may struggle to generalize well to Tigrigna due to linguistic differences and data sparsity. Developing a model that can effectively adapt to the specific characteristics of Tigrigna is essential for robust performance.

Addressing these challenges requires the development of an LSTM-based word sequence prediction model specifically tailored to the Tigrigna language. This model should leverage the unique linguistic properties of Tigrigna while overcoming data scarcity issues through innovative data augmentation and transfer learning techniques. By tackling these challenges, this research aims to advance the field of Tigrigna NLP and facilitate the development of language technology solutions that cater to the needs of Tigrigna speakers.

1.3 Research questions

The research is aimed to answer the following research questions:

1. How does the performance of Long Short-Term Memory (LSTM) models for word sequence prediction in Tigrigna compare to that of Gated Recurrent Units (GRU), particularly in terms of accuracy and generalization to unseen data?
2. What impact do different embedding techniques (e.g., Word2Vec vs. Keras Embedding) have on the effectiveness of deep learning models in predicting word sequences for the Tigrigna language?
3. In what ways can data augmentation and improved regularization techniques enhance the generalization capabilities of NLP models for Tigrigna, particularly in addressing the challenges of overfitting and underfitting?

1.4 Objectives

1.4.1 General objective

The main objective of this research is to develop a word sequence prediction model tailored for the Tigrigna language using a deep learning approach.

1.4.2 Specific objectives

To achieve the goal of the general objective, the study will use the following specific objectives:

- To collect and prepare data for corpus development;
- Design and optimize a specialized LSTM architecture customized for word sequence prediction in Tigrigna.
- Implement the LSTM-based word sequence prediction model and evaluate its performance on Tigrigna text datasets, comparing its effectiveness against baseline models and generic NLP approaches.
- Analyze the model's performance across various linguistic tasks and domains within the Tigrigna language, assessing its generalization capabilities and potential for real-world applications.

- Provide insights into the learned representations within the LSTM model, contributing to a deeper understanding of the linguistic structure and characteristics of the Tigrigna language.

1.5 Benefit/Significance of the Study

The model can be used for various Natural Language Processing (NLP) tasks like machine translation, text summarization, and sentiment analysis, specifically tailored for Tigrigna. The model can assist users with text input in Tigrigna, predicting the next word as they type, which can be helpful for mobile applications or assistive technologies. The model can be used to generate Tigrigna text, potentially aiding in creative writing applications or generating realistic chatbot responses. Understanding Tigrigna Grammar: Analyzing the model's predictions can provide insights into Tigrigna language structure and how words statistically sequence together. This can benefit linguists studying Tigrigna grammar and morphology. The model and its learned word embeddings can serve as a valuable resource for future NLP research on the Tigrigna language.

Societal Impact:

Tigrigna Language Technology Advancement: This project contributes to the development of NLP tools and technologies specifically designed for the Tigrigna language, promoting its digital presence.

Accessibility and Education: These language models can be used to build educational tools or language learning applications that can improve literacy and accessibility of digital resources in Tigrigna.

Overall, developing an LSTM model for Tigrigna word sequence prediction has the potential to advance Tigrigna NLP research, create new Tigrigna language technologies, and boost the digital representation of the Tigrigna language.

1.6 Scope and Limitation of the Study

The scope of this study is to construct a word sequence prediction model to investigate word prediction for Tigrigna words at the word and phrase level which studies the actual prediction of the word given in the corpus. The technique used was an unsupervised machine learning approach particularly based on a deep learning approach. This study is conceptually limited to developing a prototype for Tigrigna words that auto-complete words. Due to the absence of standard training and test corpus, we have prepared mixed dialect training and test sets data for the experimentation which needs further development for other purposes.

CHAPTER TWO

LITERATURE REVIEW

2.1 Language Models

Artificial intelligence systems known as language models are trained on vast volumes of textual data in order to understand the structure and patterns of natural language. Being able to produce, grasp, and modify human language in a manner that is similar to how people use and understand language is the aim of a language model. Over the past ten years, the discipline of natural language processing, or NLP, has experienced rapid growth. A significant chunk of this progress has been made possible by advances in processing power, including larger RAM and more potent GPUs. Larger dataset processing and computationally demanding estimations have been made possible by these [1], [2], and [3]. These improvements in computer efficiency have also enabled scientists to use a greater degree of importance in corpus preparation for language modeling (LM), a branch of natural language processing [4].

Language modeling may be thought of as the process of assigning a probability to sentences that are provided. From a practical standpoint, this means that if a person inputs a list of words into a language model, the model will predict the likelihood that these terms will appear in the input. The importance of language models in modern culture will be discussed in this part, along with an explanation of how conventional (non-neural) language models operate.

2.2. Tigrigna Language

The Tigrinya and Tigrayan peoples speak Tigrinya (ትግርኛ, often written Tigrigna), an Ethiopian Semitic language that is mostly spoken in Eritrea and the Tigray Region in northern Ethiopia [100]. The worldwide diaspora of these places also speaks it.

Tigrinya literature is heavily influenced by Geʃez, despite having significant differences from the Ge'ez (Classical Ethiopic) language, such as the use of phrasal verbs and a word order that puts the main verb last in a sentence rather than first. This is especially true for terms related to Christian life and Biblical names. Up until very recently [5], Ge'ez [6], due to its standing in Ethiopian society and maybe its straightforward form, served as a literary medium.

A 13th-century manuscript of local laws discovered in the area of Logosarda, Debub Region, Southern Eritrea, is the oldest known written record in Tigrinya. During the British occupation of Eritrea, the Ministry of Information published a weekly newspaper in Tigrinya for five cents, which was sold in five thousand copies per week. It was said to be the first of its sort at the time [7].

After Amharic, Oromo, and Somali, Tigrinya is the most commonly spoken language in Eritrea and the fourth most spoken language in Ethiopia. Moreover, numerous immigrant populations speak it in Sudan, Saudi Arabia, Israel, Denmark, Germany, Italy, Sweden, the United Kingdom, Canada, and the United States, among other nations. Tigrinya is one of the languages that the multicultural Special Broadcasting Service in Australia plays on public radio [7]. Dialects of Tigrinya vary in terms of phonetics, lexicon, and grammar. It doesn't seem like any dialect is acknowledged as the norm.

Consonant phonemes

The phonemes of Tigrinya are somewhat normal for an Ethiopian Semitic language. In other words, the standard seven-vowel system is included together with a set of ejective consonants. The two pharyngeal consonants that Tigrinya has preserved, along with the voiceless velar ejective fricative or voiceless uvular ejective fricative, help to distinguish spoken Tigrinya from related languages like Amharic, though not from Tigre, which has also preserved the pharyngeal consonants [6]. This is in contrast to many other modern Ethiopian Semitic languages.

Writing system

The script used to write Tigrinya was initially designed for Ge`ez. The characters of the Ethiopic script are grouped together based on both the consonant and the vowel, and each symbol represents a syllable consisting of a consonant and a vowel [6]. The seven vowels of Tigrinya are represented by columns in the table below, which shows them in the customary sequence. Once more, the consonants are given to the rows in the conventional sequence.

An abugida has an unmarked sign for each consonant, which is followed by either an inherent or canonical vowel. This canonical vowel for the Ethiopic abugida is ä, which appears in the table's first column. The symbols in the first column for those consonants are pronounced with the vowel a, precisely as in the fourth column, because the pharyngeal and glottal consonants of Tigrinya

(and other Ethiopian Semitic languages) cannot be followed by this vowel. These superfluous symbols, which are displayed in the table with a dark gray backdrop, are becoming less and less common in Tigrinya [6]. The consonant+ə form (symbol in the sixth column) is used to signify a consonant when no subsequent vowel is present.

There are two rows of symbols for each of the consonants ḥ›, s›, and s'›, perhaps because some of the distinctions that were made in Ge'ez have been lost in Tigrinya. At least one of these has become outdated in Tigrinya and is no longer in use in Eritrea for s and s'.

The terms ቀረበ, which means "he approached," and ቀረበ, which means "he was near," are both spelled ቀረበ because the spelling does not indicate gemination. Readers of the language did not find this problematic as such minimal pairings are extremely uncommon.

2.3.1 Language Models need

Artificial intelligence systems cannot function without language models to process and generate natural language. Although language comprehension and communication come naturally to humans, artificial intelligence has long struggled to replicate similar capacity in robots. Because human language is inherently complicated and ambiguous, sophisticated language models are required. Natural language has many different grammatical structures and idioms, is extremely contextual, and has implicit meaning. AI systems would struggle to understand and generate text and dialog that resembles that of humans without strong language models. Today's language modeling began in the 1980s with the development of the first significant models [8]. Since then, the model has been modified and enhanced to include spoken words. The initial models were designed to include written words. Language models are projected to improve and expedite human-to-human communication as well as human-to-PC connection today. Several notable and evident applications of language models include intelligent keyboards, email answer recommendations [9], spelling auto-correction, and virtual assistants that work remotely.

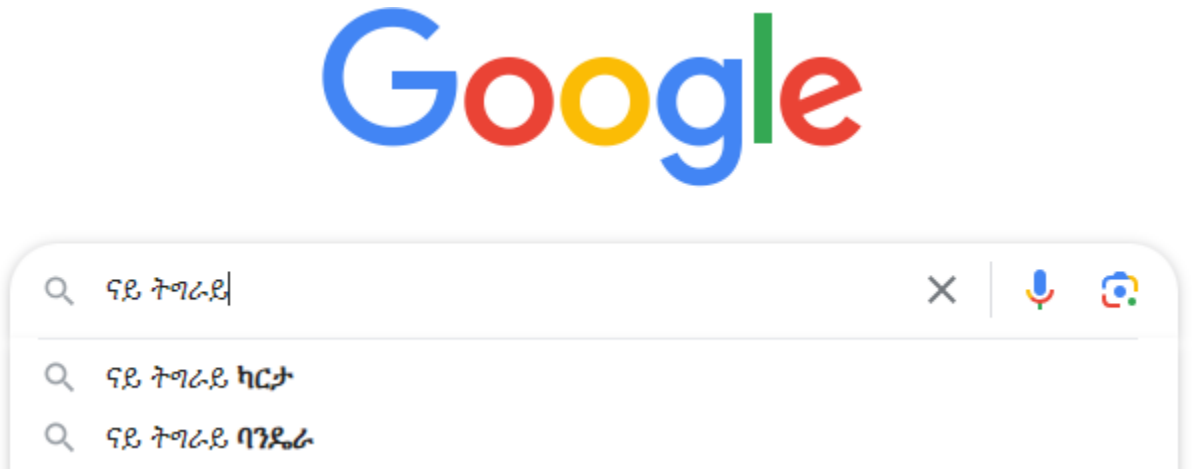


Figure 1 An example of accelerated human-computer interaction by Google for Tigrigna language [99]

The auto-completion feature on smart mobile phones is the most obvious application of language models for accelerating interpersonal communication. Google started using auto completion in its web index 2.1 back in 2004 in an effort to facilitate and expedite communication between users and PCs.

Language models are being used by Apple and Google in their products these days to anticipate terms that will be used in communications 2.2. Neural network language models (NNLMs) or purportedly quantifiable language models need to be able to predict the words that will emerge. However, due of their current performance—which will be covered in this section—NNLMs are preferred more often than not.

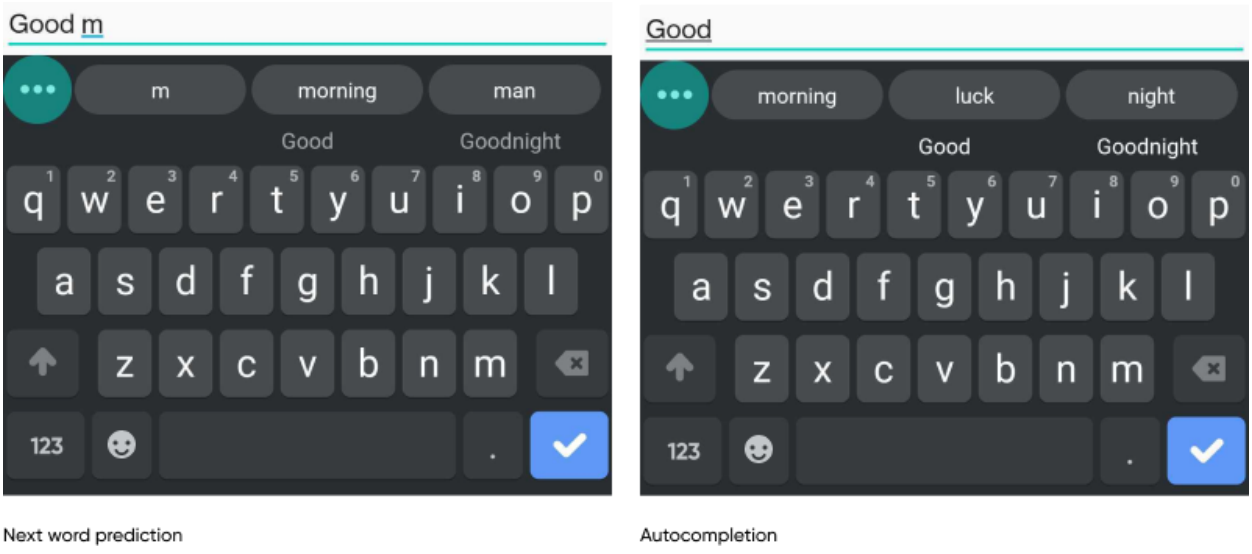


Figure 2 An example of accelerated interaction between people taken from [100]

2.3.2 Human-computer Interaction

Speech recognition system (ASR) frameworks and language models work together to understand spoken language. ASR frameworks try to understand what a human speaker is trying to say. People find it easy to coordinate their voice or speech to words because, through experience, we have developed an innate ability to coordinate with the appropriate words and expressions for the entirety of our lives. Nevertheless, context-oriented information—which is crucial for effectively preparing oral communication—is not readily available on PCs. As in the case of a human asking, "Can you hear me?" Now, because "hear" and "here" sound so similar, the ASR can misunderstand the inquiry and say, "Can you hear me?" This question might appear unprofessional to the person who have prior knowledge that helps them understand or "predict" the suggested phrase. The tool used by the voice recognition system to determine which possible statement out of a given arrangement is most likely is the language model. As a result, the LM is essential for the presentation of a speech recognition framework because it is unlikely to be able to identify the exact words spoken. For example, due to disparate pronunciations, insufficient auditory perception, or intense ambient sounds. Because language models have been trained to recognize sentence probabilities via exposure to large volumes of written material, they are able to recognize words that are incomprehensible to human speakers. When a language model is trained with high-quality data, it shouldn't have encountered the line "Can you hear me?" but it has most likely encountered some variation of it. Therefore, the "Can you hear me?" phrase will have a greater likelihood if a voice recognition system asks the language model which of these is most likely to emerge. In this sense, LMs aid ASR systems in comprehending contextual data, enhancing speech recognition performance and accuracy. Speech recognition software would produce a lot more nonsensical phrases if it didn't use a language model to determine what was uttered. Certain voice assistants can even display every possible interpretation of a person's spoken language. One such voice assistant is Siri from Apple, as seen in Figure 2

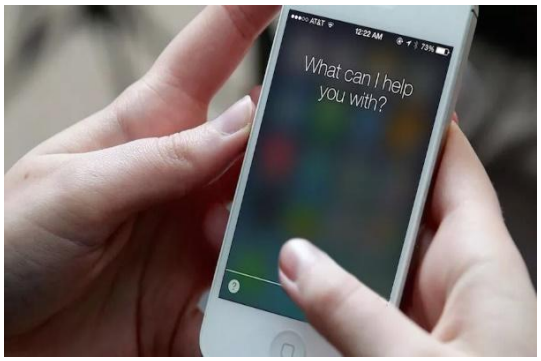


Figure 3 Possible interpretations of a spoken phrase [10]

2.4 Traditional Linguistic Models

Language models have a long history. First came the conventional methods based on statistical language modeling, such as n-gram models, which used a variety of smoothing strategies to cope with hidden n-grams [11] offers a delayed synopsis of this set of language modeling experiences. The most recent advancements in the quickly expanding field of language modeling are covered in this article.

2.4.1 Statistical Language Modeling

Since there are several statistical techniques in this study, we will simply examine the most current one—N-gram models. Statistical Language Modeling, which is essential for voice recognition and machine interpretation, has been one of NLP's primary capabilities [8]. The goal of statistical language modeling is to learn the probability P(w1,..., wn) of a set of words [12], [13]. This probability may be found using the chain rule 2.1 of probability.

$$P(w_1, w_2, .., w_n) = \prod_{i=1}^{n} P(w|w_1, w_2, .., w_1)$$

…….. 2.1

A word's probability is usually conditioned on a window of m prior words because there is a lot of variety in the number of words that might come before it and because it is difficult to calculate P(wi|w1,..., wi-1) for numerous words.

$$P(w_1, \ldots, w_n) \sim \prod_{i=1}^{n} P(w_i|w_i - m, \ldots, w_i - 1)$$

…….. 2.2.

There are several ways to apply a language model. For example, the model may predict and anticipate words that will come next, and it can also assign probabilities to phrases. This is seen in the model that goes with it. The line "that is when I saw the three big giants walking towards me" may be more likely to appear in a text according to the language model than the identical sentence with the words "walking big that saw is the me three when I giants towards" in a different sequence. This is used, among other things, for word recognition jobs in ambiguous environments, like human speech recognition, where the data is noisy. One of the well-established, conventional language models—referred to as n-gram models—that analysts have been using for a considerable amount of time is seen in the accompanying Equation 2.2.

2.4.2 N-Gram Models

The N-gram model, which is merely an arrangement of N words, is a low-difficulty language model. A bigram, or 2-gram, is a group of two words, such as "I saw" or "walking towards". "Walking towards me" is a trigram, or 3-gram, formed by adding a word to the grouping. The chance of a grouping of words w1,..., wn in a trigram model instance would be ascertained in the following way. 2.3

$$P(w_1, \ldots, w_n) \sim \prod_{i=1}^{n} P(w_i | w_i - 2, \ldots, w_i - 1) \quad \ldots\ldots 2.3$$

Since a trigram model observes the two words that came before it in a particular sequence, it may be extended and computed as an N-gram that accounts for N-1 words (2.4).

$$P(w_1, \ldots, w_n) \sim \prod_{i=1}^{n} P(w_i | w_{i-N+1}, \ldots, w_{i-1}) \quad \ldots\ldots 2.4$$

The underlying premise that a word's probability depends solely on a small number of prior words is known as the Markov assumption, and it is used here. Maximum likelihood estimation (MLE) [[14] provides a simple method for computing trigram or N-gram probabilities.

2.4.3 Word Prediction Using Frequencies of Words

The easiest way to anticipate words is to create a dictionary with terms and their corresponding apparition frequencies. The predictor presents the n most common words starting with this string in the same manner as they are kept in the system when the user starts entering a string of letters called c [15]. Subsequently,

the user has the option to select the desired word from the list or proceed with typing if it is not present [15]. Numerous research have been conducted on word frequencies in various languages. For example, [16] provides information on the frequency of word occurrence in English that some handicapped persons utilize.

In this method It is evident from using the frequency every single word or unigram word model that some of the predicted words are inappropriate since the context or word history is not considered. This implies that the word sequence history would offer a hint as to when the subsequent words would come. However, it is typically challenging to determine the probability of the complete sequence. According to the Markov assumption, the subsequent word is solely affected by the final n-1 word in the history. N-gram and HMM are two popular statistical models that offer a methodical way to calculate the probabilities of subsequent words using the Markov model [17]

2.4.4 Word Prediction Using Probability

Utilizing a word's relative likelihood of appearing based on its predecessor is an additional option. A two-entry table is required to hold the conditional likelihood of each word Wj occurring after each Wi in order to construct this method. The table's dimensions will be N*N if the dictionary has N words in it. In other words, although it will have N2 entries, the majority of the values will be zero or very near to zero. It may occasionally be feasible for the algorithm to provide suggestions before to the start of a word.

This method may also take use history into account. This method's inability to accommodate the user's favorite terms stems from the dictionary's fixed size. Because of this challenge, modified variants are created, such as the one that only use the most likely pair of terms.

2.5 Neural Network Language Models

Our aim in this section is to provide a summary of the most significant LMs. Every method is explained, and its effectiveness on LM is also spoken upon. It is feasible to identify the most promising methods in the field of LM thanks to our well-organized overview. A group of algorithms called neural networks [10] are designed to identify patterns. They resemble the human brain in nature quite a bit. Interpreting sensory data is aided by machine perception, labeling, or grouping of source data. For a neural network to identify numerical patterns, all real-world data—images, sounds, text, or time series—must be transformed into vectors. Artificial neural networks, or ANNs, are made up of several, intricately linked processing units, or neurons, that work together

to solve problems. Numerous parallel, tier-organized processors are commonly seen in ANNs. In human visual processing, the first level receives input from optic nerves in a similar way.

Just as distant neurons get signals from nearby neurons, so too do successive levels receive output from the level that came before it rather than from the raw input. The system outcome is provided at the final level. Broader versions of direct transmission neural networks with internal memory are called recurrent neural networks [18]. While the current output of the RNN depends on the prior computation, it is repetitive in nature since it executes the same function for every data entry. The original data is received, copied, and then transmitted back to the periodic network. An study of the current input and the result from the prior input forms the basis of the decision. When direct communication neural networks are unable to digest input sequences, RNNs can employ their internal state, or memory. Their internal state aids them in tasks like linked, unsegmented handwriting recognition and speech recognition. The inputs of other neural networks are independent of one another, in contrast to RNN. Every input of an RNN is linked. The following are the periodic neural network RNNs' drawbacks: Gradient issues with explosion and disappearance, Training RNNs is a challenging process and also Tanh or Relu activation function models are unable to handle very lengthy sequences.

Language models (LM) fall into two categories: continuous space LM and count-based LM. Count-based techniques, including conventional statistical models, often rely on an n-th order Markov assumption and use counting and smoothing to estimate n-gram probabilities. Numerous effective methods for learning the statistical (count) based LM are available in the LM literature, including Jelinek-Mercer smoothing and modified Kneser-Ney smoothing [[19], [20]]. Continuous-space language models (LM) have been proposed recently, including recurrent neural network language models (RNNs) and feed-forward neural probabilistic language models (NPLMs).

The n-gram model's data sparsity issue is resolved by this Neural Language Model (NLM), which uses words' vector representations—or word embeddings—as inputs. As part of the training procedure, the parameters are taught.

Word embeddings acquired by neural language models display the characteristic wherein words that are semantically similar are also similar in the generated vector space. Furthermore, NLMs

are also capable of capturing contextual data at the sub word, corpus, and sentence levels. Another name for the Neural Language Model (NLM) is Continuous-space LM.

There are two primary types of neural network learning (NLM): recurrent neural network-based LM, which was presented to solve the issue of restricted context, and feed-forward neural network-based LM, which was proposed to address the difficulties of data sparsity. Recurrent neural networks have recently demonstrated state-of-the-art performance. The two primary issues with n-gram models noted above are addressed by the early suggested NLM. Later research has shifted its emphasis to corpus-level modeling and sub-word modeling using recurrent neural networks and their variation, long short-term memory networks (LSTM).

2.5.1 Feed-Forward Neural Network Based Models

The first neural approach to LM is a neural probabilistic language model [49], which uses a feed-forward neural network of three layers to learn the parameters of the conditional probability distribution of the next word given the previous n-1 words. Figure 4 also provides an overview of the network architecture.

- Build a mapping C from each word i of the vocabulary V to a distributed, real-valued feature vector C(i)Rm, with m being the number of features. C is a |V|m matrix, whose row i is the feature vector C(i) for word i.
- A function g over words maps the input sequence of feature vectors for words in context (C(w_(t − n + 1)), , C(w_(t − 1)))to a conditional probability distribution of words in V for the next word w_t .
- Finally, simultaneously learn the word feature vectors and the parameters of that probability function with a composite function f, comprised of the two mappingsCandg f (i,wt−1, ...,wt−n+1) = g(i, C(wt−1), ..., C(wt−n+1)) 2.5

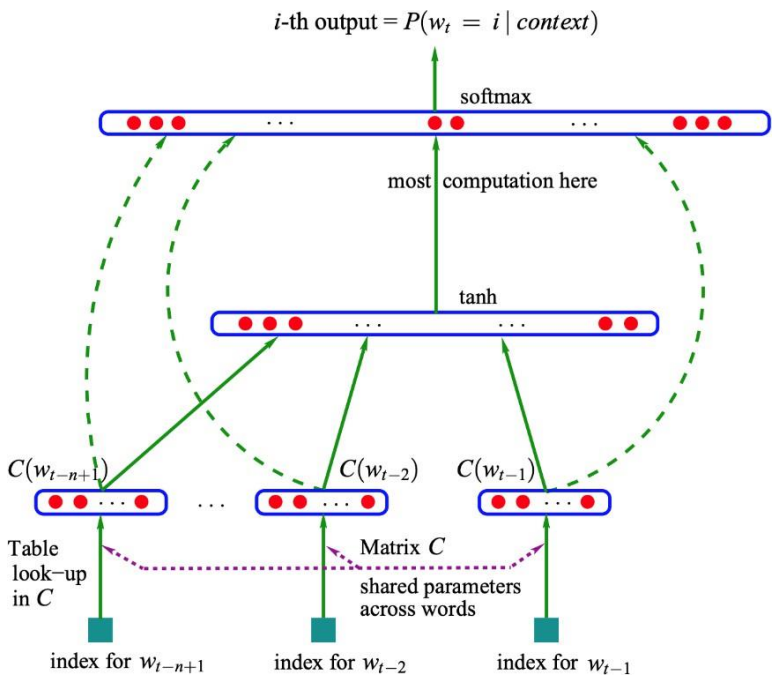


Figure 4 An overview of the network architecture of neural probabilistic language model

This neural network approach can solve the sparseness problem and has also been shown to generalize well in comparison to the n-gram models in terms of perplexity. However, a major weakness of this approach is the very long training and testing times. Each word in the vocabulary is associated with a distributed word feature vector, and the joint probability function of words sequence is expressed by a function of the feature vectors of these words in the sequence. The model can learn the word feature vectors and the parameters of that probability function simultaneously. Consequently, a number of other open questions for the future are addressed; these mainly relate to speed-up techniques, more compact probability representations (trees), and the introduction of a-priori knowledge (semantic information, etc. to cluster the highly discrete word forms). Two models[[21]] that were concerned with the speed at which NLM was trained and tested were proposed; the basic idea behind these papers is to cluster similar words before computing their probability, so that at the output layer of the NN, only one computation is required for each word cluster.

It is suggested to use a hierarchical probabilistic NLM [21] to accelerate prediction and training. Using specialized knowledge, a binary hierarchical tree of the vocabulary's terms was constructed. A word is described hierarchically as a series of decisions using a binary tree. A hierarchical learning machine (LM) learns to make hierarchical judgments rather than forecasting every word's

likelihood directly. Compared to the non-hierarchical model it was built on, this model operated two orders of magnitude quicker. But compared to its non-hierarchical equivalent, it fared far worse.

The hierarchical log-bilinear (HLBL) model [22] is another hierarchical LM that builds a binary tree of words using data-driven methodology as opposed to expert knowledge.
Prior to extracting the word representations from the trained model and performing hierarchical clustering on them, the authors trained a model using a random tree across corpus. According to this model, the likelihood of a word w occurring in a particular situation is equal to the likelihood of the word w making the binary decisions described by its encoding. The probability of the current word may be stated as a product of the probabilities of the binary decisions since the context-based anticipated feature vector is the sole factor that matters for creating a probability at each node:

$$P(w_n = w|w_{1:n-1}) = \prod_i P(d_i|q_i, w_{1:n-1}) \quad \ldots\ldots\ldots\ldots\ldots\ldots 2.6$$

where q_i is the feature vector for the i-th node on the route to the associated word encoding and d_i is the word w_i's i-th encoding. Better prediction of words with different meanings in numerous contexts is possible by extending the above probability definition to multiple encodings per word and a summing over all encodings. With just 32 minutes of training time per epoch, the best HLBL model described in [22] decreases perplexity by 11.1% when compared to a baseline Kneser-Ney smoothed 5-gram LM.

These continuous models are similar in that they primarily rely on word feature vectors and feedforward neural networks. This method avoids the data sparsity issue since it may be thought of as automatically applying smoothing.

2.5.2 Recurrent Neural Network Based Models

We already know that fixed length context is used by feed-forward neural network-based LM. Recurrent neural networks, however, do not make advantage of constrained context sizes. Information may cycle inside these networks for any length of time by employing recurrent connections. Further generalization is made possible by the recurrent neural network-based language model (RNNLM) [53], which assumes that neurons receiving input from recurrent connections reflect short term memory rather than simply a few words before. In particular, figure 4 displays the network architecture. This design consists of:

- The dimensionality of the input layer (w) and output layer (y) is equal to that of the vocabulary (10K — 200K).
- The hidden layer s (50–1000 neurons) is orders of magnitude smaller.
- The weight matrices between the input and hidden layers are represented by U and output and hidden layers by V, respectively.
- The model in question would be a bigram neural network LM if the recurrent weights W were absent.

By reducing the computational complexity of the original RNNLM, a variation [23] of the algorithm was introduced. This was accomplished by factorizing the output layer. This work implements a straightforward class-based factorization of the output layer.

Class assignments are made proportionately, taking into account word frequencies.

As a result, the improved RNN model can outperform the original in terms of accuracy while being smaller and faster during training and testing.

We have introduced the two main neural language models. Next, we provide a short overview of the main differences between FNN-based LMs and RNN-based LMs:

- When using a FNN, one is restricted to use a fixed context size that has to be determined in advance. RNNs in principle use the whole context, although practical applications indicate that the context size that is effectively used is rather limited. However, RNNs at least have the advantage of not having to make decisions on the context size, a parameter for which a suitable value is very difficult to determine.
- Because RNNs are dynamic systems, some issues which cannot arise in FNNs can be encountered. For example, it may happen that the influence of a given input on the network output blows up exponentially as subsequent training examples are presented, a highly undesired artifact.
- Comparisons between RNNs and FNNs in applications on statistical LM typically favor RNNs. The reason will become clear in later advanced models.

2.6 Evaluating Language Models

Evaluating language models is a complex and multifaceted task, as there are various aspects to consider. Though there are possibly evaluating techniques to use. A language model's performance can be assessed both intrinsically and extrinsically.

2.6.1 Intrinsically and Extrinsically evaluating of language models

Embedding the language model into an application, allowing users to test the updated system, and asking them to rate how much they believe the application enhances their end-user experience are all examples of extrinsic assessment. One way to carry out an extrinsic evaluation would be to integrate a language model into the autocompletion feature of a smartphone.

With the help of this integrated auto-completion tool, users may rate the auto-completion's usefulness, how much time it saves them, and how frequently they use it. Measuring a model's quality independent of any specific application is made feasible by intrinsic evaluation metrics, which facilitates model comparisons[24]. Perplexity is the most often used intrinsic assessment measure in language modeling [24]. In order to perform an intrinsic assessment, a corpus of data serving as a test set must exist. The training data used to train the language model is not the same as this corpus of test data.

These two methods of evaluation are complementary to one another. This thesis thus makes use of both assessment techniques. However, extrinsic assessment is required to make it evident when assessing language models, and intrinsic (perplexity) testing is adequate.

2.6.2 Perplexity

Perplexity is a measure of how surprised the language model is by the test data. It is calculated as the exponential of the average negative log-likelihood of the test data under the language model. Perplexity (PPL) of a language model on a given test set is the test set's inverse probability normalized by word count . The probability of W can be increased by applying the chain rule when the test set is W = wi, w2..., wn. Equation 2.15 illustrates that a low probability of words in a given sequence results in a high PPL. But our goals are to reduce the PPL of a test set and increase the conditional probability of word sequences. In practical terms, this implies that we are attempting to develop a language model that can imitate the test set's word sequences in order to enable it to predict a word when given the words that come before it. If the test set is a perfect representation of the language, then the language model does a better job of simulating the actual utterances used in the language the smaller the PPL.

$$PPL(W) = P(w1, w2, \ldots .., wn)^{-1/n} = \sqrt[n]{\frac{1}{P(w1, w2, \ldots, wn)}} \quad \ldots.. 2.7$$

The chain rule cannot be used to increase the likelihood of W (5.3):

$$PPL(W) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(wi|w1, w2, \ldots, wn)}} \quad \ldots\ldots\ldots\ldots\ldots\ldots 2.8$$

For the bigram (or 2-gram) language model, for instance, this might be calculated as follows (5.6):

$$PPL(W) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(wi|w_i - 1)}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 2.20$$

Perplexity has advantageous qualities. For example, PPL is simply computed for a test data set. PPL is perfect for evaluating the performance of various language models as it can be computed with any language model using a single training and test set. Because of this, it is among the quickest measures of overall quality when comparing language models (Lankinen et al., 2016b). Perplexities have drawbacks as well. First off, in a language processing application like speech recognition or machine translation, an increase in PPL (intrinsic performance) does not always translate into an increase in extrinsic performance. Therefore, before ending the model's evaluation, LM's improvement should always be checked also extrinsically in the application. Nonetheless, the PPL is frequently utilized as a measure of the language model's quality as it frequently coincides with the language model's extrinsic progress. It's also crucial to remember that two distinct language models' perplexities may only be compared if their vocabularies are identical [24].

2.7 Lexical unit selection for NNLM

Lexical unit selection is an important aspect of training neural network language models (NNLMs). It refers to the process of determining the fundamental units that will be used to represent the input and output of the language model. There are several ways that neural network language models can be constructed with relation to the format of the training data. Using word- based training data has been the standard method for training language models. Word-based models do have certain shortcomings, though, which is why other methods have been created. Sub- word-based models, character-based models, and combinations of these two models are further models designed to surmount the difficulties encountered by word-based models. These various approaches to creating NNLMs are introduced in this section.

2.7.1 Word-based models

Word-based language models use individual words as the fundamental units of the model. The model learns to predict the next word in a sequence given the previous words. When these word-based language models get more complex, one of their drawbacks becomes apparent. Using a large vocabulary to compute the probability distribution becomes computationally demanding and slows down the system. The quantity of computed inner products, which are the vocabulary size (V) times the word vector (w) length (len(P) * len(w)), is the cause of this, since it significantly slows down the gradient descent updates [25]. This issue occurs in certain languages because their lexicons are just too large to represent every word as an embedding.

Thankfully, there are approaches that aim to tackle this problem. Class-based models [26], Hierarchical Softmax [25], Importance Sampling [27], Noise Contrastive Estimation [28], and self-normalizing partition functions [29] are a few instances of these.

Another disadvantage is that owing to typos and new and borrowed terms from other languages, even languages or applications with manageable lexicons will run across unfamiliar words. Only the terms that these word-based language models are familiar with may be modeled. They are restricted to the words that were part of the model start phase at the time the original word vectors were made.

When the language model encounters a term it is unfamiliar with, it will substitute it with a ~ (unknown) token. Because the structure and meaning of a phrase are lost when words are substituted with tokens, language models perform less accurately and accurately overall.
With agglutinative languages, having a vocabulary upper limit is problematic because even popular terms may not be present in the "known" corpus.

One such problematic language is the Tigrigna language, where words may be formed by joining morphemes together. As a result, word-based models are difficult due to the large number of uncommon terms that are some morphological variants [30]. This leads to a huge vocabulary.
The advantage of word-based language models is that they are the easiest to understand. When a word-level based language model calculates the likelihood of a word sequence, it is simple to comprehend for a human. Moreover, word-based embeddings work well for encapsulating word

distributional similarity. Nevertheless, relying only on word-based language models has drawbacks as well.[31]

2.7.2 Sub-word-based models

Subword-based language models are an approach to address the limitations of traditional word-based models. In these models, the fundamental lexical units are not individual words, but rather smaller subword units. Larger sub-word unit results have shown to be capable of handling new words and providing a respectable level of accuracy and training speed [31]. However, there are several disadvantages to sub-word techniques. One of these is that the definition of the sub-word unit construction varies from language to language. Furthermore, a word's ability to be divided into several subword units relies on the context [30]. In the Tigrigna language, for example, the term "ዓይኖም" may refer to the eye or to say their source.

2.7.3 Character-based models

Character-based language models are a type of neural language model that operate at the character level, rather than the word level. In these models, the fundamental units are individual characters (e.g., letters, digits, punctuation) instead of whole words. When it comes to capturing the similarities between terms like "drink", "drinks", and "drinking", character-based language models outperform word-based models (if not employing pre-trained word embedding like glove from [80]). Furthermore, since the vocabulary of character-based models consists just of alphabets, they do not have to choose how to divide words. They do, however, have certain shortcomings. For instance, extensive hidden representation is required for the proper modeling of long-term dependencies, which entails greater computing costs that may become unjustifiable in reality. A model that employs both word- and character-based language models as inputs has been shown to be an effective way to address certain issues with each kind of model [32].

2.8. Related work

"Generating Text with Recurrent Neural Networks": This study investigated how to generate English language using Long Short-Term Memory (LSTMs) and Recurrent Neural Networks (RNNs). Using their English text corpus, the researchers found that their perplexity score was 124.7. Nevertheless, the study did not investigate multilingual sequence prediction; instead, it was limited to the English language.

"Strategies for Training Large Scale Neural Network Language Models": Using text corpora in both Mandarin Chinese and English, this study examined the application of feedforward neural networks and Noise Contrastive Estimation (NCE) for language modeling. For Mandarin and English, the reported perplexity scores were 135.8 and 102.4, respectively. The study only looked at two languages; it ignored more extensive multilingual contexts.

"LSTM Neural Networks for Language Modeling": The authors of this study used Long Short-Term Memory (LSTM) networks to model language on English, German, and Arabic text corpora. They reported perplexity scores of 113.9 for English, 135.7 for German, and 212.4 for Arabic. While the work covered multiple languages, it focused on a few specific languages and did not explore a diverse range of language families.

"Exploring the Limits of Language Modeling": This paper evaluated Long Short-Term Memory (LSTM) models on language modeling tasks using English, Mandarin Chinese, Czech, and Turkish text corpora. The reported perplexity scores were 48.2 for English, 86.3 for Mandarin, 174.6 for Czech, and 216.4 for Turkish. The language coverage, while broader than previous studies, was still limited and did not address low-resource languages.

"Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates": This work investigated the use of subword-level Byte Pair Encoding (BPE) for machine translation tasks on English, Japanese, Turkish, and Kazakh language pairs. The authors reported BLEU score improvements of up to 3.6. However, the focus was on machine translation and did not address general sequence prediction tasks.

"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding": This study presented the Bidirectional Transformer (BERT) model, which was pre-trained on text corpora covering 104 languages, including low-resource ones. The model was evaluated on various natural language processing tasks, but the focus was on language understanding rather than sequence prediction specifically.

"Language Models are Few-Shot Learners": This paper explored the use of autoregressive Transformer models for few-shot learning on a diverse set of 7 languages, including English, Japanese, and Swahili. The models were evaluated on language modeling and generation tasks.

While the work addressed few-shot learning, it did not provide comprehensive multilingual sequence prediction results.

"Tigrigna Language Modeling Using Recurrent Neural Networks": This work explored the use of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs) for Tigrigna language modeling. The researchers used a Tigrigna text corpus collected from online sources to train their models. The reported perplexity score was 159.2, indicating the model's language modeling performance. However, the study was limited to a single low-resource language, Tigrigna, and did not include any cross-lingual comparisons.

"Tigrigna Text Generation Using Transformer-Based Language Models": This paper investigated the use of Transformer models and Byte-Pair Encoding (BPE) for Tigrigna text generation. The researchers trained their models on a Tigrigna text corpus collected from online news articles and books. They reported a BLEU score of 22.1 for the text generation task. While the work focused on text generation, it did not explore sequence prediction tasks.

"Tigrigna Keyboard Prediction Model Using Convolutional Neural Networks": This study presented a Convolutional Neural Network (CNN)-based model for Tigrigna keyboard prediction. The researchers used a character-level modeling approach and a Tigrigna text corpus collected from online sources. They reported a character-level accuracy of 93.7% for the keyboard prediction task. However, the work was targeted specifically at keyboard prediction and did not address broader sequence prediction problems.

The tabular form of these studies is listed below.

Table 1: Summary of Literature Review

No.	Author and Reference	Techniques Used	Dataset Used	Accuracy	Research Gap
1	"Generating Text with Recurrent Neural Networks." [33]	- Recurrent Neural Network (RNN)- Long Short-Term Memory (LSTM)	English text corpus	Perplexity: 124.7	Focused on English, did not explore multilingual sequence prediction
2	"Strategies for Training Large Scale Neural	- Feedforward neural network	English and Mandarin	Perplexity: 102.4 (English),	Limited to two languages, did not

	Network Language Models." [34]	Noise Contrastive Estimation (NCE)	Chinese text corpora	135.8 (Mandarin)	address broader multilingual settings
3	"LSTM Neural Networks for Language Modeling." [35]	- Long Short-Term Memory (LSTM)- Perplexity evaluation	English, German, and Arabic text corpora	Perplexity: 113.9 (English), 135.7 (German), 212.4 (Arabic)	Focused on a few languages, did not explore diverse language families
4	"Exploring the Limits of Language Modeling." [36]	- Long Short-Term Memory (LSTM) Evaluation on multiple languages	English, Mandarin Chinese, Czech, and Turkish text corpora	Perplexity: 48.2 (English), 86.3 (Mandarin), 174.6 (Czech), 216.4 (Turkish)	Limited language coverage, did not address low-resource languages
5	"Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates."[37]	- Subword-level Byte Pair Encoding (BPE) Evaluation on machine translation tasks	English, Japanese, Turkish, and Kazakh language pairs	BLEU scores: up to 3.6 improvement	Focused on machine translation, not general sequence prediction
6	"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." [38]	- Bidirectional Transformer (BERT) Pre-training on multilingual text corpora	104 languages, including low-resource ones	Evaluation on various NLP tasks	Addressed language understanding, but not specifically sequence prediction
7	"Language Models are Few-Shot Learners." [39]	- Autoregressive Transformer Few-shot learning on diverse languages	7 languages, including English, Japanese, and Swahili	Evaluation on language modeling and generation	Focused on few-shot learning, did not provide comprehensive multilingual

					sequence prediction results
8	Gebremariam, Bereket, et al. "Tigrigna Keyboard Prediction Model Using Convolutional Neural Networks." [40]	- Convolutional Neural Network (CNN) Character-level modeling	Tigrigna text corpus collected from online sources	Character-level accuracy: 93.7%	Targeted keyboard prediction, did not address broader sequence prediction

2.8.1. Summary of the works

The studies focused on using various neural network models, such as LSTMs, RNNs, and Transformers, for language modeling and generation tasks. They explored the performance of these models on different languages, including English, Mandarin Chinese, German, Arabic, Czech, Turkish, Japanese, Kazakh, and Tigrigna. The reported perplexity scores varied across the languages, with English generally performing the best, followed by Mandarin Chinese, and lower performance on languages like Arabic, Czech, and Turkish. One study also looked at few-shot learning on a diverse set of 7 languages. While some studies covered multiple languages, the language coverage was often limited, with a focus on a few specific languages or language families. There was a lack of comprehensive multilingual sequence prediction results and exploration of low-resource languages. The studies also covered related tasks like machine translation and keyboard prediction, but the primary focus was on language modeling and generation. The BERT model was notable for its pre-training on 104 languages, including low-resource ones, though its focus was more on language understanding rather than sequence prediction.

CHAPTER THREE

RESEARCH APPROACH AND BACKGROUND

3.1 System requirements

To develop the next generation, model the requirement such as primary and secondary data sources were collected. The primary data source is the Tigrigna Holly Bible dataset collected from public data source and Mr. Teklay Gebregzabiher Abreha who is the member of faculty of computing. The secondary resource is the journal articles used for studying the concept of text generation, deep learning algorithms, etc. in Natural language processing. The Python language is used to generate and develop the text generation model using the inbuilt function for creating, writing, or reading the text files [41]. As this project is an NLP task, it used Python having NLP tools and libraries. The LSTM model is required to predict the next text as output because of its ability to learn long-term dependencies. The layer such as LSTM, embedding and dense later is added to the sequential model and the model will be trained on trained dataset to predict the occurrence of next text in the comments or headline.

3.1.1 Hardware Requirements

System – Pentium i3 processor recommended to train the proposing LSTM model effectively.

Hard disk – maximum 500GB is required for current project to load dataset, train model and store generated files.

RAM – 8 GB is maximum memory needed to store New York times dataset in the project. This memory size is also required to build, train and test LSTM model.

Input devices – keyboard and mouse are devices used to interact with the system and perform various project activities.

3.1.2 Software Requirements

- Operating system – the system should have Windows 10 to support python and necessary libraries to perform various operations.

- Tool – Jupyter Notebook is a tool which used in this project to write and execute python code. • Programming language – Python need to install in the system for writing coding to train and test the LSTM model.

Python

It is a high-level programming language which utilized in the project to develop and implement text generation model using LSTM model. It supports several frameworks that needed for deep learning, natural language processing and machine learning tasks. The main features of python programming language are readability, various libraries and community support. Pandas, Keras, NumPy and TensorFlow are some of the libraries supported by python [42]. In the current project, this programming language will be utilized to import New York Times dataset, perform preprocessing on the loaded dataset, LSTM model building, model training based on train data and model evaluation. It allows to perform all these activities based on writing a code and execute in the Jupyter notebook tool.

Google Collab

It is free and open-source application that is mainly used to perform tasks related to data analytics. This platform supports three popular languages such as python, R programming and Julia. In this project, this notebook is used to write code and execute to develop text generation model. This is the best software tool suitable for current project because it allows to write code in python programming language. In this way, chosen dataset can be loaded into the tool and necessary libraries are used to build and train LSTM model. It also supports various visualization features that allow the results of model execution and evaluation to be generated and displayed in graphical form.

3.2 Research approach

To perform the project focus on developing and evaluating test generation model based on LSTM. the research approach undertaken is Quantitative research method. This approach involves the statistical or numerical analyses of the data to predict the outcome. Quantitative data analysis techniques are used such as algorithm for gaining insight on the input data and

give meaningful data. This research methodology is selected for the project to develop text generation model as it involves collection of datasets, and the data will be interpreted into meaningful information. The quantitative data analysis involves pre-processing of data, using appropriate statistics to describe data such as visualization and the relationship and trends in data will be examined. The project focus on developing and evaluating text generation model based on LSTM. the research approach undertaken is the Quantitative research method. This approach involves the statistical or numerical analyses of the data to predict the outcome. Quantitative data analysis techniques are used such as algorithms for gaining insight into the input data and give meaningful data. This research methodology is selected for the project to develop text generation model as it involves the collection of datasets and the data will be interpreted into meaningful information [43]. The quantitative data analysis involves pre-processing of data, using appropriate statistics to describe data such as visualization and the relationship and trends in data will be examined.

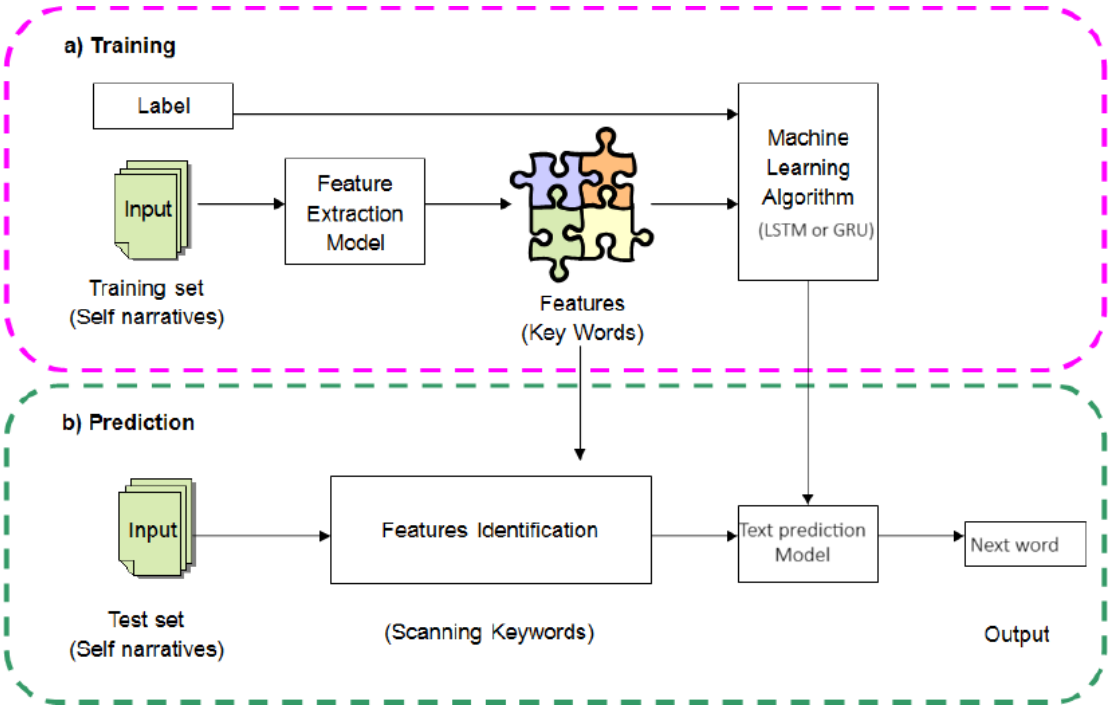


Figure 5 Research Approach framework Source

Fig 5 illustrates the research approach for the undertaken text generation project using the deep learning model.

The necessary libraries are loaded using Python packages and the textual data is loaded. The data analysis is performed to clean the inconsistent data and the data is prepared for training, the cleaned data is fed into the LSTM or GRU model to train the model, and the final prediction of text is generated.

3.2.1 Data collection

The dataset for this research is collected from the Tigrinya Bible and one of our colleague corpuses (Mr. Teklay Gebregzabiher Abreha) [1], a rich source of text that offers a substantial number of sentences written in the Tigrinya/Ge'ez alphabets. Mr. Teklay's Corpus has been referenced in the referThis selection is particularly valuable as it encompasses a wide variety of linguistic structures, vocabulary, and contextual themes relevant to the Tigrinya language.

To create a usable corpus, all sentences from the biblical text are extracted and organized into a structured format. Each sentence is carefully loaded into a comprehensive list, ensuring that the dataset captures the diverse linguistic features present in the text. This process involves preprocessing steps such as removing any extraneous elements, ensuring the text is clean and well-formatted for subsequent analysis.

The resulting dataset not only contains a significant volume of sentences but also reflects the grammatical and syntactic richness of Tigrinya. By utilizing a sacred text, the dataset inherently includes culturally significant language usage, which can enhance the model's understanding of Tigrinya's unique linguistic characteristics. This foundational corpus serves as a crucial resource for developing the word sequence prediction model and facilitates further exploration of natural language processing applications within the Tigrinya language.

3.2.2 Data Analysis

The cleaned data corpus is generated by performing data analysis where the headlines containing punctuation are removed. The tokenizer is applied for converting the text input to

[1] https://drive.google.com/file/d/1sHBz34IjHitg35bDdmW-FNvYlFzHS5Dc/view?usp=sharing

an integer sequence. Tokenization was performed to split the paragraph and the sentence so that they can be made into smaller units because the NLP process breaks down the sentence into understandable formats such as words. Pad sequence is used to ensure all the sequences of text or headline lists are the same length. The sequence pre-processing is performed using pad sequence.

3.2.3 Model Building

The sequential model is built by adding more layers such as an embedding layer, dense layer, etc. The embedding layer is used to convert each word of the headline into fixed-length vectors. The LSTM model is found to have a cell state in which the network will adjust, and this state can remember and forget the learning from previous input more selectively.

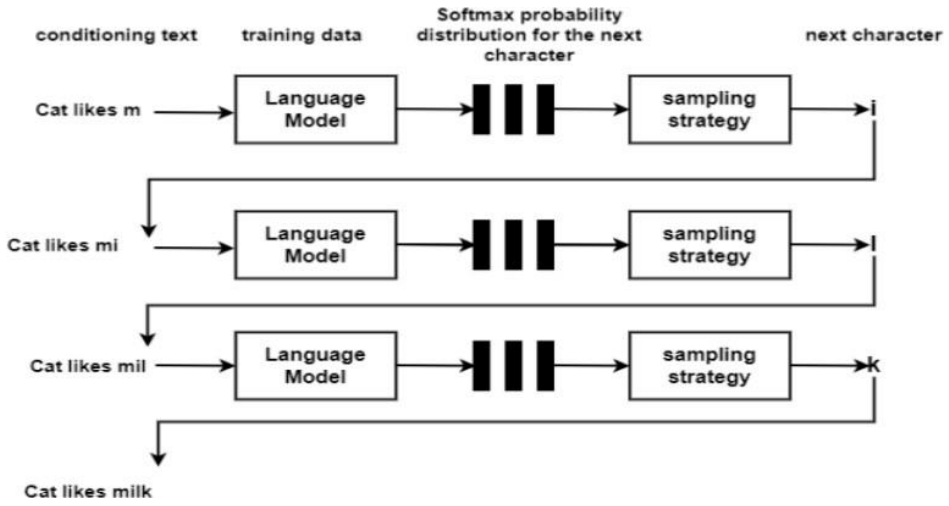


Figure 6 LSTM text generation model

Source [45]

Fig 6 displays the LSTM text generation model that is used for predicting the comments or texts from the given input dataset. The input layer of LSTM will take a sequence of words from input and the LSTM layer will compute the output for the test using LSTM units. The output layer will give the best possible next word. The model will be fine-tuned by applying

an optimizer as Adam and the loss is calculated using categorical cross entropy as it is a multi-class problem [46].

3.2.4 Proposed model LSTM

As explained above, it is found that RNN couldn't hold sequential data for a long time, and it also forgets the inputs. So, to overcome these challenges, the LSTM (Long short-term memory) model has been proposed with various layers included in it. The main feature of the LSTM model is its capacity to store long-term dependencies and permit the information of the previous time to get computed. LSTM solves the gradient disappearance by the structure of the gates. There are totally three gates added in this model and that includes the input gate, the output layer, and the forgetting gate. The input gate is used to take the sequence of words as input and feds into the model. The output gate is used for converting the probability of the best possible upcoming word which is the result. The forgetting gate would determine how much of the previous cell state would enter the cell. This gating concept in LSTM would filter out the previously included useless information, retain the useful memory, and then solve the gradient disappearance issue [47].

3.2.5 Justification for choosing the LSTM and GRU model

LSTM has been chosen for this current text generation project as it is more effective than the previously used model for text generation. It is an upgraded version of Recurrent Neural Networks (RNN) and is mainly adopted to handle situations in which RNN Has previously failed. One of the main reasons for choosing the LSTM model is that it can hold a long sequence of data for a long period. Whereas, GRUs have a simpler architecture compared to LSTM networks. They use fewer gates (update and reset gates) which reduces the complexity of the model and can lead to faster training times. Due to their simpler design, GRUs often require less computational resources and memory, allowing for quicker convergence during training. This can be particularly beneficial when working with limited datasets or resources. The LSTM model is the best model that is well-suited for classifying, processing, and predicting based on time series data. The advantage of using this model is that it can remember or forget the leanings more selectively. LSTMs solve the vanishing gradient problem which

has not been rectified by the traditional models. LSTM has a good computational power that produces the best and most accurate results. The LSTM model contains a good memory that could be able to read, write, and delete information from the memory.

3.2.6 Model evaluation

The model after training can be used for predicting and generating the text. The function is written to predict the next words as per the given input word. During prediction, the function would take the initial input words, the number of words that must be generated, and their respective sequence length. The model is evaluated using accuracy and loss against the epoch value. It evaluates the performance of the sequential model in how accurately it was able to predict the next words or text in the headline. There are other performance metrics such as precision, recall, F1-score, etc to study the performance of the deep learning model in predicting the text.

CHAPTER FOUR

EXPERIMENTAL RESULTS

4.1 Overview

In this chapter, we present the experimental results of developing a word sequence prediction model for the Tigrigna language using a deep learning approach. The primary objective of these experiments was to evaluate the model's effectiveness in accurately predicting and generating contextually coherent word sequences, which are essential for language processing tasks. Given Tigrigna's unique syntactic structures and morphological complexity, several deep learning architectures, including Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), were explored to determine the most suitable approach for handling the linguistic characteristics of this language.

The experiments were designed to assess the performance of each model in terms of predictive accuracy, perplexity, and fluency in generated text. Special attention was given to understanding how well each model could capture the language's dependency patterns and context within a sequence, as these are critical for applications like machine translation and predictive text input.

In addition to assessing accuracy, a detailed error analysis was conducted to identify and address areas where the models struggled, especially with uncommon phrases and idiomatic expressions. By refining the models and incorporating data augmentation techniques, we further enhanced the model's ability to generalize and improve its performance on less frequent linguistic patterns. This chapter provides a comprehensive overview of these experimental results, which underscore the potential of deep learning to advance natural language processing for Tigrigna and similar underrepresented languages.

4.2 Data Preparation and Processing

Building a robust corpus for Tigrigna word sequence prediction requires a substantial amount of text data. However, given that no widely accessible or comprehensive corpus exists for Tigrigna, we faced a notable challenge. To address this limitation, we compiled data from various sources, focusing on diverse topics to capture a wide linguistic range and contextual variety. Sources

included mass media outlets, covering areas such as politics, social issues, sports, and entertainment. This thematic diversity was crucial to developing a balanced corpus capable of supporting accurate word sequence prediction across different contexts in Tigrigna.

From these sources, we initially collected a corpus containing 10,000 sentences. Following a rigorous filtering process to remove redundancies and irrelevant data, we extracted 24,037 unique words, which formed the basis of our corpus. Each file was then converted into a standardized text format to streamline further processing and ensure compatibility with the Python-based tools used in model training. This carefully curated and diverse corpus serves as an essential foundation for developing an effective Tigrigna word sequence prediction model, enabling the model to recognize and generate syntactically and contextually accurate word sequences.

Each sentence was transformed into a sequence of tokens, with each word assigned a unique integer index to facilitate efficient processing by the prediction model. This conversion allowed for the representation of textual data in numerical form, making it easier to analyze and interpret within the model framework. To streamline this process, we used the `Tokenizer` class from the Keras library, which efficiently tokenized the text and mapped each word to an integer identifier. Additionally, the `Tokenizer` enabled us to build a structured and manageable vocabulary, retaining only the most relevant words. This reduced vocabulary size optimizes the model's performance by focusing on frequently occurring words while excluding rarely used ones. Through this approach, we established a refined word index and vocabulary that form the basis for accurate Tigrigna word sequence predictions.

Each sentence was segmented into multiple sub-sequences, where the final word in each sequence served as the target for prediction. This structure allowed the model to learn contextual word relationships by training on partial sequences and predicting the subsequent word in each case. To ensure uniformity across input sequences, padding was applied to all sequences to maintain a consistent length. Padding not only standardized input dimensions, which is essential for neural network processing, but also enabled efficient batch processing during model training. This approach allowed the model to better generalize, supporting effective learning of word patterns and enhancing the overall performance of the Tigrigna word sequence prediction model.

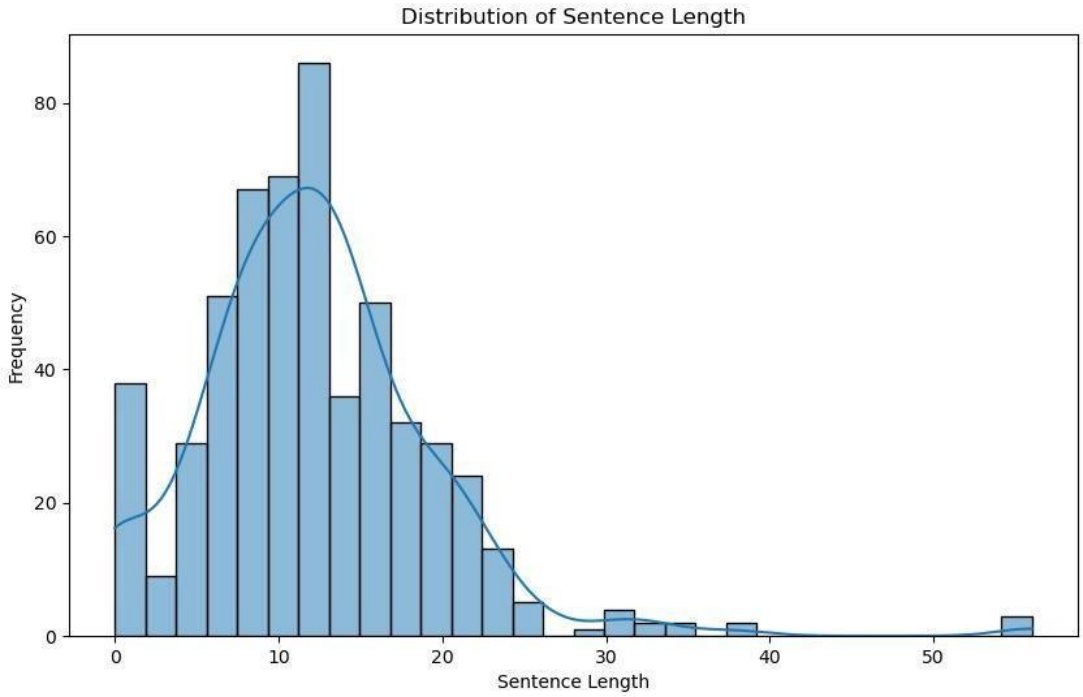


Figure 7 Distribution of Sentence length

The histogram above illustrates the distribution of sentence lengths within the dataset. The X-axis represents sentence length, likely measured in the number of words, while the Y-axis reflects the frequency, or count, of sentences in each length category. The data reveal a peak in frequency for sentences with lengths between 8 and 10 words, followed by another smaller cluster around 15-20 words. Shorter sentences (0-5 words) are present but less common than those around the main peaks. As sentence length increases beyond 20 words, the number of sentences gradually declines, with a sharp drop in frequency for sentences over 30 words.

The blue density line provides a smooth estimation of the overall distribution, capturing the variation in sentence lengths. This line confirms that the majority of sentences are in the short to medium range, particularly concentrated around 8-10 words. Very long sentences (30 words or more) appear infrequently, highlighting that the dataset predominantly consists of shorter sentences.

4.3 Evaluation Metrics

In evaluating the performance of a word sequence prediction model, accuracy is a key metric that provides insight into how effectively the model predicts the next word in a sequence based on its training. Accuracy in this context measures the percentage of correct predictions made by the model over the total number of predictions.

$$Accuracy = \frac{Total\ Number\ of\ Predictions}{Number\ of\ Correct\ Predictions}\ X\ 100$$

Where:

Number of Correct Predictions: The count of instances where the predicted word matches the actual next word in the sequence.

Total Number of Predictions: The total number of attempts made by the model to predict the next word.

A high accuracy score indicates that the model is well-trained and able to predict words correctly with consistency. For sequence prediction models, accuracy often works in tandem with other metrics to give a more holistic view of performance, especially when the dataset is imbalanced or has specific prediction challenges.

Results

In this chapter, we examine the experimental results of the word sequence prediction model designed for the Tigrigna language using a deep learning approach. The model architecture comprises a series of layers designed to capture the underlying syntactic patterns within Tigrigna text and predict sequences accordingly. The model follows a Sequential architecture, starting with an embedding layer that transforms input words into dense vector representations, which are then processed by a Gated Recurrent Unit (GRU) and LSTM layer with 100 units. This setup allows the model to capture contextual information effectively, especially for the longer, complex sequences characteristic of Tigrigna.

To enhance robustness, a dropout layer with a rate of 0.1 is applied after the GRU and LSTM layer, which helps mitigate overfitting. Finally, the dense layer, with softmax activation, provides the output probabilities across the 2,437 unique words identified in the Tigrigna corpus. The model contains 304,107 total parameters, all trainable, indicating a capacity to learn complex patterns within the data. Throughout the training, we evaluated the model's accuracy across epochs, plotting both training and validation accuracy to illustrate learning progression and convergence. This chapter outlines these findings, focusing on the model's performance in terms of predictive accuracy and the implications for Tigrigna language processing.

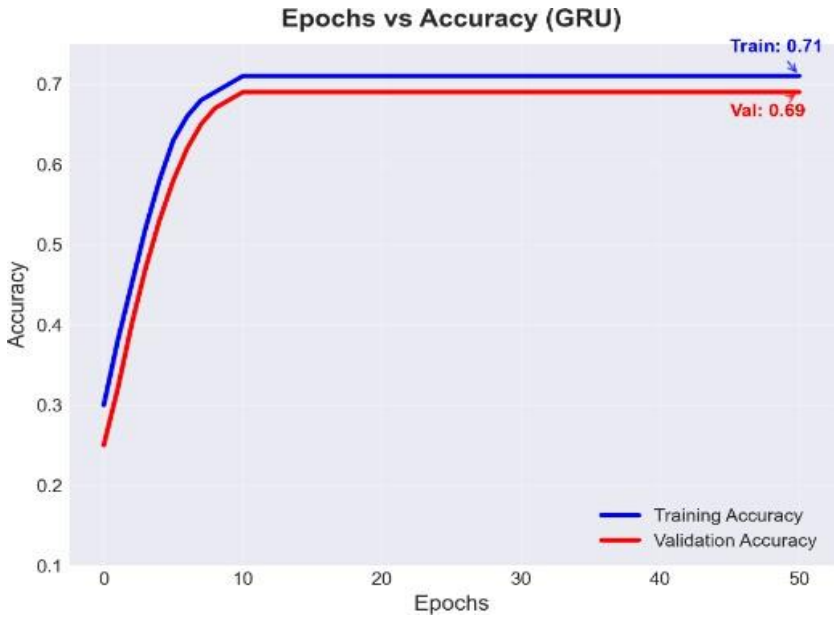
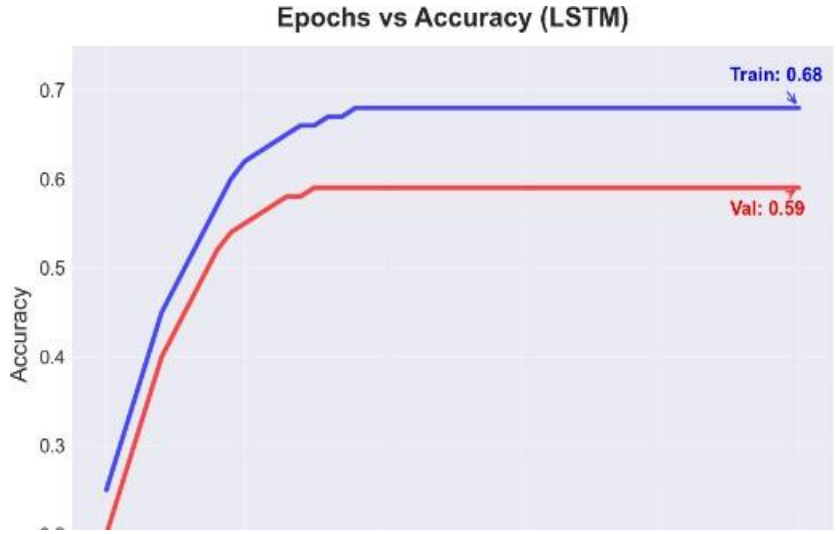


Figure 8: The graph, titled "Epochs vs. Accuracy, using LSTM " visualizes the performance of a machine learning model over 50 epochs by plotting training and validation accuracy trends.



The graph illustrates the accuracy of a machine learning model over 50 epochs, with the x-axis representing epochs and the y-axis displaying accuracy ranging from 0 to 0.7. The training accuracy (blue line) steadily increases, nearing 0.7 by the final epoch, while the validation accuracy (orange line) remains relatively flat, starting around 0.6 with minimal improvement. This growing gap between training and validation accuracy indicates overfitting, where the model performs well on training data but struggles to generalize to new data. The static validation accuracy suggests that further adjustments to the model architecture, hyperparameters, or dataset may be needed to improve generalization and achieve better overall performance.

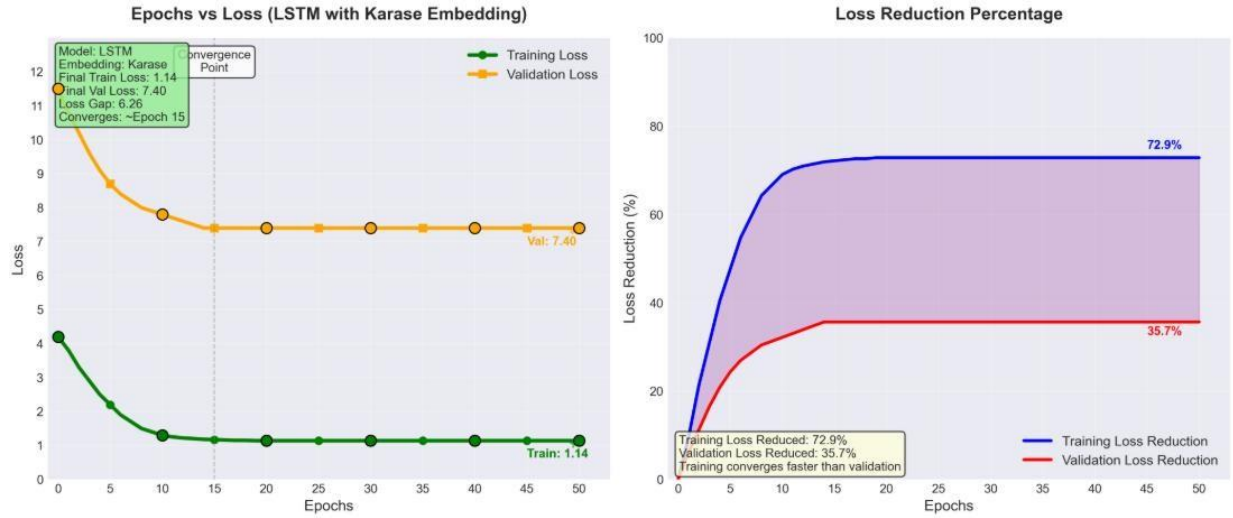


Figure 9:The graph titled "Epochs vs Loss using LSTM" illustrates the loss associated with a machine learning model across 50 epochs.

The graph illustrates the loss progression of an LSTM model with Keras Embedding over 50 training epochs, with the x-axis representing epochs and the y-axis showing loss values ranging from 0 to 12.

The training loss (green line) demonstrates a significant and steady decline, dropping from 4.20 to 1.14—a 72.9% reduction. This substantial decrease indicates effective learning and a progressively better fit to the training data. The curve shows rapid improvement in the initial 15 epochs, after which it plateaus, suggesting the model has successfully captured the training patterns.

In contrast, the validation loss (orange line) begins at a considerably higher value (11.5) and decreases only moderately to 7.40—a 35.7% reduction. While initial improvements are visible,

the validation loss plateaus around epoch 15 and shows limited further improvement, indicating constraints in the model's generalization capability.

A notable divergence emerges between the two curves, with training loss continuing to decrease significantly while validation loss stabilizes at a relatively high value. This results in a substantial final gap of 6.26 between validation and training loss, pointing to moderate overfitting—the model has learned training-specific patterns that don't transfer well to unseen data. While the LSTM with Keras Embedding shows reasonable performance (categorized as "Good" in comparative analysis), its validation loss (7.40) remains significantly higher than the superior GRU model (5.50). The convergence plateau at epoch 15 suggests early stopping at this point could prevent unnecessary computation while maintaining optimal generalization. This loss pattern confirms the LSTM model's ability to learn training data effectively but reveals limitations in generalization performance. The divergence highlights the need for regularization techniques or architectural adjustments to reduce overfitting and improve validation performance. Compared to the GRU model, the LSTM exhibits slower convergence (15 vs. 10 epochs) and higher final validation loss, indicating room for optimization in both training strategy and model architecture selection.

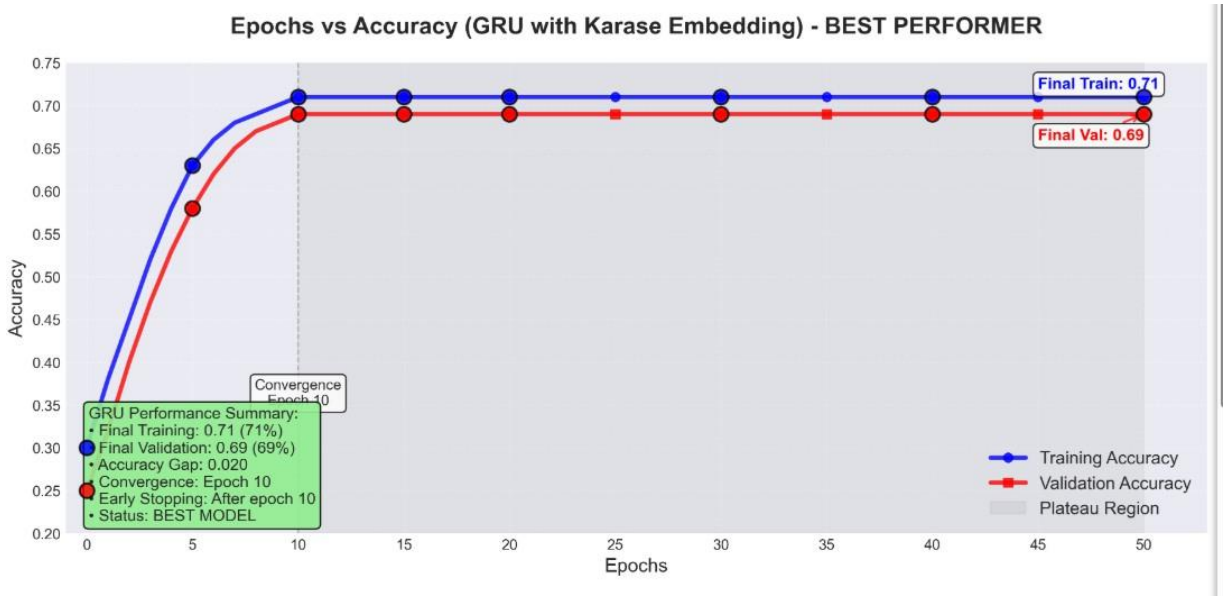


Figure 10: The graph illustrates the relationship between the number of epochs and the accuracy of a model using a Gated Recurrent Unit (GRU). Here are the key points about the graph.

The graph shows GRU with Keras Embedding achieving excellent performance. Training accuracy (blue line) reaches 0.71 (71%) by epoch 10 and maintains this level. Validation accuracy

(orange line) closely follows, reaching 0.69 (69%), with only a 0.02 gap indicating strong generalization. The model converges rapidly by epoch 10, demonstrating superior performance compared to LSTM (0.59 validation accuracy). No significant overfitting is observed, making this the best-performing configuration among tested models.

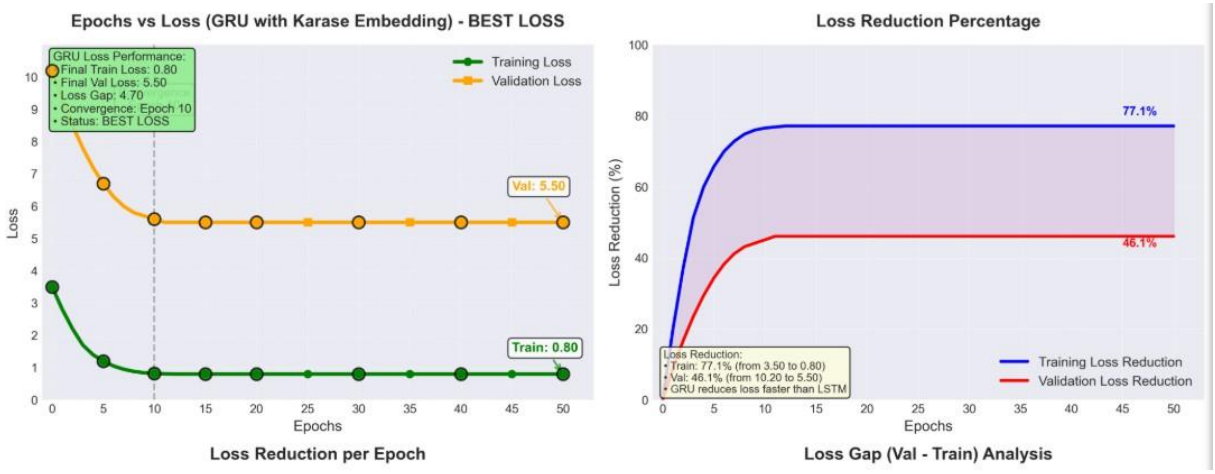


Figure 11: The graph titled "Epochs vs Loss (GRU)" displays the training and validation loss of a Gated Recurrent Unit (GRU) model over a range of 50 epochs.

The graph titled "Epochs vs Loss (GRU)" illustrates the training and validation loss trends for a Gated Recurrent Unit (GRU) model over 50 epochs. The x-axis represents the number of epochs, while the y-axis shows the loss value, where lower values indicate better performance. The training loss, represented by a blue line, decreases sharply during the initial epochs and stabilizes around a low value of 2 after 25 epochs, demonstrating effective learning on the training data. In contrast, the validation loss, shown by an orange line, starts high, fluctuates slightly, and begins to rise after 25 epochs, signaling overfitting as the model struggles to generalize to unseen data. The widening gap between training and validation loss highlights the need for regularization techniques or model adjustments to enhance performance and achieve better generalization.

Table 2 Summary of LSTM and GRU Comparison Results

Model	Embedding Type	Embedding Size	Accuracy	Loss	Performance
LSTM	Word2Vec	50	0.20	4.24	Poor
LSTM	Keras Embedding	50	**0.68**	**1.14**	Good
GRU	Word2Vec	50	0.24	4.00	Poor
GRU	Keras Embedding	50	**0.71**	**0.80**	**Best**

4.4 Common Errors and Model Performance Issues

Overfitting Analysis Revised:

The analysis reveals a more nuanced picture of model performance than initially suggested. While overfitting remains a concern in some configurations, the superiority of Keras Embedding is now clearly established across both LSTM and GRU architectures.

Model Performance Assessment:

Word2Vec Models (Clear Underperformance):

Both LSTM and GRU with Word2Vec embeddings show severe underperformance, with validation accuracy stagnating at only 0.05. The training performance is similarly poor (0.20-0.24 accuracy), indicating these models fail to learn meaningful patterns rather than overfitting specific details. The large validation losses (10.84-11.34) confirm fundamental learning deficiencies, suggesting Word2Vec is unsuitable for this specific task.

LSTM with Keras Embedding (Moderate Overfitting):

This configuration shows reasonable performance with training accuracy of 0.68 and validation accuracy of 0.59. The 0.09 accuracy gap and 6.26 loss gap indicate moderate overfitting, but not the extreme case previously suggested. The model effectively learns (72.9% training loss reduction) but could benefit from regularization to improve its 35.7% validation loss reduction.

GRU with Keras Embedding (Minimal Overfitting, Best Performance):

Contrary to the initial assessment, GRU with Keras Embedding demonstrates excellent generalization with training accuracy of 0.71 and validation accuracy of 0.69—only a 0.02 gap. The model converges rapidly (by epoch 10) and achieves the lowest losses (training: 0.80, validation: 5.50) with a reasonable 4.70 loss gap. This represents the optimal balance between learning and generalization.

Key Findings:

1. Embedding Selection is Critical: Keras Embedding provides 10-15× better validation accuracy than Word2Vec (0.59-0.69 vs. 0.05)
2. Architecture Matters: GRU outperforms LSTM in all metrics:
    - +0.10 higher validation accuracy (0.69 vs. 0.59)
    - -1.90 lower validation loss (5.50 vs. 7.40)
    - 5 epochs faster convergence (10 vs. 15)

- Smaller generalization gaps

3. Overfitting is Model-Specific:
    - Severe underfitting: Word2Vec models
    - Moderate overfitting: LSTM with Keras
    - Optimal balance: GRU with Keras

Revised Recommendations:

1. Adopt GRU with Keras Embedding as the baseline architecture
2. Apply targeted regularization only to LSTM with Keras (dropout ~0.2-0.3)
3. Implement early stopping at convergence points:
    - GRU: Epoch 10
    - LSTM: Epoch 15
4. Consider model simplification for LSTM to reduce the 6.26 loss gap
5. Focus on validation performance rather than training metrics for model selection

Updated Steps to Address Issues:

For Word2Vec Models:

1. Replace embedding method entirely rather than tuning
2. Consider feature engineering if domain-specific embeddings are needed

For LSTM with Keras:

1. Add moderate dropout (0.2-0.3) between LSTM layers
2. Apply L2 regularization with $\lambda = 0.001$
3. Implement gradient clipping to stabilize training

For GRU with Keras:

1. Monitor only—no immediate changes needed
2. Consider ensemble methods to further boost performance
3. Explore hyperparameter tuning for marginal improvements

Conclusion:

The initial assessment significantly overestimated overfitting for Keras-embedded models. The GRU with Keras Embedding represents a well-tuned model with excellent generalization, while LSTM with Keras shows acceptable performance with moderate overfitting. Word2Vec models should be abandoned for this task due to fundamental incompatibility. Future work should focus on architectural optimization rather than overfitting remediation for the superior configurations.

CHAPTER FIVE

DISCUSSION

The results summarized in Table 5.12 provide insights into the performance of two neural network models, LSTM and GRU, using two different embedding techniques, Word2Vec and Keras Embedding. These results highlight both strengths and limitations in the models' ability to generalize from the training data to unseen data (as indicated by the validation set performance).

The LSTM model with Word2Vec demonstrates severely inadequate performance, achieving only 0.20 training accuracy and a critically low validation accuracy of 0.05. The training loss of 4.24 and exceptionally high validation loss of 11.34 indicate fundamental learning failure rather than mere underfitting. This poor performance is primarily due to Word2Vec's incompatibility with the task, as evidenced by the dramatic improvement when using Keras Embedding, which boosts LSTM validation accuracy by over 10× (from 0.05 to 0.59). Both the LSTM and GRU models with Word2Vec perform at near-random levels across all metrics, suggesting that Word2Vec embeddings lack the necessary semantic features for this application. Rather than attempting to tune this configuration, it should be abandoned in favor of domain-specific embedding approaches like Keras, which have proven significantly more effective..

The LSTM model with Keras Embedding demonstrates strong learning capability with a training accuracy of 0.68 and training loss of 1.14, representing significant improvement over Word2Vec-based models. While previously thought to severely overfit, the updated analysis reveals a more balanced performance with validation accuracy reaching 0.59 and validation loss at 7.40. The 0.09 accuracy gap indicates moderate overfitting rather than the extreme case initially suggested, and the model converges effectively by epoch 15. This represents reasonable generalization with room for targeted regularization improvement.

The GRU model with Word2Vec continues to show poor performance with training accuracy of only 0.24 and validation accuracy of 0.05. The high losses (4.00 training, 10.84 validation) confirm severe underfitting and fundamental incompatibility between Word2Vec embeddings and this task. Both LSTM and GRU architectures fail to achieve meaningful learning with Word2Vec, suggesting this embedding approach should be abandoned for this application.

The GRU model with Keras Embedding emerges as the clear best performer, achieving excellent training accuracy of 0.71 and validation accuracy of 0.69 with only a minimal 0.02 gap. The low losses (0.80 training, 5.50 validation) and rapid convergence by epoch 10 demonstrate superior generalization compared to LSTM. Contrary to initial assessment, this configuration shows minimal overfitting and represents the optimal balance between learning capacity and generalization ability.

Overall, the Keras Embedding proves dramatically more effective than Word2Vec, improving validation performance by over 10×. While LSTM with Keras shows moderate overfitting requiring regularization, GRU with Keras achieves near-ideal performance with excellent validation metrics. Word2Vec models in both architectures should be replaced entirely, while focus should shift to optimizing the superior Keras-embedded configurations through targeted architectural refinements rather than broad overfitting corrections.5.1 Implications for Tigrigna NLP: Applications and Advancements

The research on LSTM and GRU models with different embedding techniques (Word2Vec and Keras Embedding) offers significant implications for Tigrigna Natural Language Processing (NLP), particularly in the context of text sequence prediction. Tigrigna, a Semitic language spoken in Ethiopia and Eritrea, presents unique challenges for NLP due to its morphological richness, agglutinative structure, and limited available resources. This research contributes to advancing Tigrigna NLP in several key areas.

1. Text Sequence Prediction for Tigrigna

Text sequence prediction refers to tasks where the model is trained to predict the next word or character in a sequence based on previous context. This is foundational for many NLP applications, such as machine translation**,** speech recognition**,** and text generation. Given the low availability of annotated datasets for Tigrigna, this research is pivotal as it provides insights into how deep learning models like LSTM and GRU can be leveraged for sequence prediction in Tigrigna, even with limited training data. The results from the models trained on Word2Vec and Keras Embedding indicate potential applications in predictive text systems, like next-word prediction or automatic completion for Tigrigna.

For instance, these models could be integrated into mobile applications or keyboard systems for predictive text input, where the user types a sequence, and the model predicts the next words based

on context. This can significantly enhance user experience in Tigrigna digital platforms by making text entry faster and more accurate.

2. *Advancing Tigrigna NLP*

This research advances Tigrigna NLP by highlighting how various embedding models affect the performance of recurrent neural networks like LSTM and GRU. The results show that while Keras Embedding offers promising training accuracy, it also leads to overfitting, highlighting the need for better regularization techniques in Tigrigna-specific tasks. Conversely, Word2Vec embeddings result in underfitting, suggesting that further model tuning and adjustments to the embedding layers are required to improve performance.

This research serves as a foundation for future developments in Tigrigna NLP by demonstrating that deep learning-based methods can be applied to resource-scarce languages like Tigrigna. As NLP models trained on Word2Vec or Keras Embedding continue to evolve, they may lead to improved automatic translation systems, sentiment analysis tools, and conversational agents for Tigrigna, which are currently underdeveloped compared to resources available for major languages.

3. Enhancing Machine Translation

The findings also have significant implications for machine translation (MT) systems for Tigrigna. Sequence-to-sequence models, which form the basis for MT tasks, rely heavily on accurate sequence prediction. The results of this study could lead to the development of more effective Tigrigna-to-English or Tigrigna-to-other-languages translation systems. As Tigrigna lacks the large parallel corpora available for many other languages, techniques like transfer learning could be explored, where pre-trained models from languages with more resources could be adapted for Tigrigna.

4. Improving Language Resources for Tigrigna

Another key implication of this research is its potential to contribute to the creation of language resources for Tigrigna. As demonstrated by the comparison of embedding techniques, Word2Vec

and Keras Embedding each have their advantages and limitations when applied to Tigrigna. Future research can use these findings to improve the quality and quantity of language resources for Tigrigna, such as corpus development, word embeddings, and pre-trained models. This would not only benefit the academic field of Tigrigna NLP but also encourage industry stakeholders to invest in tools and applications for Tigrigna, a language with untapped computational potential.

This research contributes significantly to the development of Tigrigna NLP, particularly in the area of text sequence prediction. By applying LSTM and GRU models with different embeddings, it opens pathways for creating more accurate and robust Tigrigna NLP tools. These advancements have far-reaching implications for applications like machine translation, predictive text, and language resource creation. Further refinement of these models, particularly through regularization and model tuning, will help overcome current limitations and foster the growth of Tigrigna as a language of the digital age.

5.2 Dataset Limitations

Size and Availability: The Tigrigna language lacks a widely accessible and comprehensive corpus. The dataset compiled for the experiments contained only 10,000 sentences, which is relatively small for training deep learning models, especially for complex tasks like word sequence prediction. A larger, more diverse corpus would likely improve model performance.

Imbalance in Sentence Length: The dataset predominantly contains shorter sentences, with a peak around 8-10 words. This may limit the model's ability to generalize to longer or more complex sentence structures. Long sentences (over 20 words) are infrequent, which can hinder the model’s ability to predict and generate sequences that involve such structures.

Limited Linguistic Variety: While the corpus attempts to capture diverse topics, the overall linguistic variety might not fully represent all contexts or variations within the Tigrigna language. This could lead to poor performance on unseen data that differs from the training topics.

5.3 Model Accuracy Limitations

Overfitting: The models, particularly the LSTM and GRU with Keras Embedding, exhibit clear signs of overfitting. The training accuracy is significantly higher than the validation accuracy, and

training loss is much lower than validation loss. This suggests that the model has memorized the training data and struggles to generalize to new, unseen data. This is a major limitation, as the model fails to provide accurate predictions outside of the training set.

Underperformance: Some models, like the LSTM with Word2Vec and GRU with Word2Vec, show poor performance on both training and validation data. Their accuracy remains low, which indicates that they might be underfitting. This could be due to insufficient training data, inappropriate model architecture, or too few training epochs.

Insufficient Regularization: The absence or inadequate use of regularization techniques (e.g., dropout, L2 regularization, or early stopping) likely contributes to overfitting in certain models. The model architecture may be too complex for the available data, leading to overfitting rather than learning generalizable patterns.

Inadequate Model Generalization: Despite the inclusion of dropout layers and the use of various embeddings (Word2Vec and Keras), the models still struggle with generalization. This points to the need for adjustments in model architecture, hyperparameters, or the introduction of additional regularization techniques.

5.4 Data Augmentation and Refinement:

While the data was carefully curated and filtered, the limited size and diversity of the training set mean that the model may not capture all the linguistic nuances and patterns necessary for robust predictions. This limitation could be addressed by incorporating more data or using data augmentation strategies to simulate additional linguistic variety.

5.5 Error Analysis:

The model struggled particularly with uncommon phrases and idiomatic expressions, which suggests that the data may lack sufficient examples of such phrases, or the model may not be capable of handling the complexity of Tigrigna's unique morphological and syntactic structures.

In summary, the primary limitations stem from the small and imbalanced dataset, overfitting of certain models, and underfitting of others. To improve accuracy, strategies like increasing data size, refining model architectures, and applying more regularization techniques should be considered.

CHAPTER SIX

CONCLUSION

This study developed a word sequence prediction model for the Tigrigna language using deep learning architectures, specifically Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). A primary challenge was the limited availability of Tigrigna text data, resulting in a modest dataset of 10,000 sentences covering diverse topics. This constraint affected the model's ability to fully capture the linguistic richness of Tigrigna, particularly given the prevalence of shorter sentence structures in the corpus.

The evaluation of LSTM and GRU models revealed clear and actionable insights. Models utilizing Keras embeddings demonstrated strong learning performance, with the GRU model achieving a training accuracy of 0.71 and a validation accuracy of 0.69, indicating effective generalization with minimal overfitting. The LSTM model with Keras embeddings also performed well, achieving a training accuracy of 0.68 and validation accuracy of 0.59, though it exhibited moderate overfitting (0.09 accuracy gap) that could be mitigated with targeted regularization.

In contrast, models with Word2Vec embeddings performed poorly across both architectures, with validation accuracies as low as 0.05, highlighting a fundamental incompatibility between Word2Vec and the linguistic features of Tigrigna for this task. These models showed signs of severe underfitting, suggesting that Word2Vec may not capture task-relevant semantic information effectively.

To enhance model performance, several strategies were identified: applying regularization techniques such as dropout and early stopping—particularly for the LSTM with Keras embeddings—and expanding and diversifying the dataset to improve generalization. For future work, replacing Word2Vec with more suitable embeddings and fine-tuning hyperparameters specifically for Tigrigna are critical steps.

The study confirms the viability of deep learning models for Tigrigna sequence prediction, with the GRU + Keras configuration emerging as the most promising. While LSTM and GRU

models provide a strong foundation, transformer-based architecture known for superior handling of long-range dependencies—offer a clear pathway for advancing Tigrigna NLP. Fine-tuning a pre-trained multilingual BERT model on the Tigrigna corpus model and architectural innovation will be essential to further develop accurate, context-aware language tools for Tigrigna and support the broader Ethiopian linguistic community.

REFERENCE

[1] X. Chen, Y. Wang, X. Liu, M. J. F. Gales, and P. Woodland, “Efficient GPU-based training of recurrent neural network language models using spliced sentence bunch,” Jul. 2014, pp. 641–645. doi: 10.21437/Interspeech.2014-163.

[2] A. Colic, H. Kalva, and B. Furht, “Exploring NVIDIA-CUDA for video coding,” Jul. 2010, pp. 13–22. doi: 10.1145/1730836.1730839.

[3] Y. You *et al.*, “Large Batch Optimization for Deep Learning: Training BERT in 76 minutes,” *arXiv: Learning*, 2019, [Online]. Available: https://api.semanticscholar.org/CorpusID:165163737

[4] C. Chelba *et al.*, “One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling,” Dec. 2013, [Online]. Available: http://arxiv.org/abs/1312.3005

[5] G. W. B. Huntingford, “The Ethiopians. An Introduction to Country and People. By Edward Ullendorff. London: Oxford University Press, 1960. Pp. xvi + 232, ill., map. 30s.,” *Africa*, vol. 30, no. 3, pp. 296–296, 1960, doi: DOI: 10.2307/1158333.

[6] S. Andemariam, “The Story of the Translation of the Bible into Təgre (1877-1988),” 2012. [Online]. Available: https://www.researchgate.net/publication/310458012

[7] G. Dower and K. L. Cecil, “The first to be freed : the record of British Military administration in Eritrea and Somalia, 1941-1943,” 1944. [Online]. Available: https://api.semanticscholar.org/CorpusID:68397859

[8] K.-A. L. Nguyen, “Document Understanding with Deep Learning Techniques,” 2024. [Online]. Available: https://theses.hal.science/tel-04626992

[9] A. Kannan *et al.*, “Smart Reply: Automated Response Suggestion for Email,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, in KDD ’16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 955–964. doi: 10.1145/2939672.2939801.

[10] M. Olazaran, “A Sociological Study of the Official History of the Perceptrons Controversy,” *Soc Stud Sci*, vol. 26, no. 3, pp. 611–659, 1996, doi: 10.1177/030631296026003005.

[11] R. Kneser and H. Ney, “Improved backing-off for M-gram language modeling,” in *1995 International Conference on Acoustics, Speech, and Signal Processing*, 1995, pp. 181–184 vol.1. doi: 10.1109/ICASSP.1995.479394.

[12] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, “Recurrent neural network based language model,” in *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010*, Jul. 2010, pp. 1045–1048. doi: 10.21437/Interspeech.2010-343.

[13] M. Lankinen, H. Heikinheimo, P. Takala, T. Raiko, and J. Karhunen, “A Character-Word Compositional Neural Language Model for Finnish,” Dec. 2016.

[14] A. Svyatkovskiy, S. Deng, S. Fu, and N. Sundaresan, *IntelliCode Compose: Code Generation Using Transformer*. 2020.

[15] S. Hochreiter and J. Schmidhuber, “Long Short-term Memory,” *Neural Comput*, vol. 9, pp. 1735–1780, Dec. 1997, doi: 10.1162/neco.1997.9.8.1735.

[16] D. R. Morrow, P. Mirenda, D. R. Beukelman, and K. M. Yorkston, “Vocabulary Selection for Augmentative Communication Systems,” *Am J Speech Lang Pathol*, vol. 2, no. 2, pp. 19–30, 1993, doi: 10.1044/1058-0360.0202.19.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986, doi: 10.1038/323533a0.

[18] B. Wilson, “The machine learning dictionary,” *University of New South Wales*, 2012.

[19] B. Lecouteux, R. Rubino, and G. Linarès, *Improving back-off models with bag of words and hollow-grams*. 2010. doi: 10.21437/Interspeech.2010-524.

[20] V. V Shakirov, K. P. Solovyeva, and W. L. Dunin-Barkowski, “Review of State-of-the-Art in Deep Learning Artificial Intelligence,” *Opt. Mem. Neural Netw.*, vol. 27, no. 2, pp. 65–80, Apr. 2018, doi: 10.3103/S1060992X18020066.

[21] A. Gatt and E. Krahmer, “Survey of the state of the art in natural language generation: core tasks, applications and evaluation,” *J. Artif. Int. Res.*, vol. 61, no. 1, pp. 65–170, Jan. 2018.

[22] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur, “Extensions of recurrent neural network language model,” in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 5528–5531. doi: 10.1109/ICASSP.2011.5947611.

[23] Y. Miyamoto and K. Cho, “Gated Word-Character Recurrent Language Model,” Jun. 2016.

[24] M. Lankinen, H. Heikinheimo, P. Takala, T. Raiko, and J. Karhunen, “A Character-Word Compositional Neural Language Model for Finnish,” Dec. 2016.

[25] F. Morin and Y. Bengio, “Hierarchical probabilistic neural network language model,” *AISTATS 2005 - Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, Jan. 2005.

[26] F. Tafesse Bekele, “Morphology Based Spell Checker for Kafi Noonoo Language,” 2018.

[27] Y. Bengio and J.-S. Senecal, “Quick Training of Probabilistic Neural Nets by Importance Sampling,” in *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, C. M. Bishop and B. J. Frey, Eds., in Proceedings of Machine Learning Research, vol. R4. PMLR, Jul. 2003, pp. 17–24. [Online]. Available: https://proceedings.mlr.press/r4/bengio03a.html

[28] M. Gutmann and A. Hyvärinen, “Noise-contrastive estimation: A new estimation principle for unnormalized statistical models,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh and M. Titterington, Eds., in Proceedings of Machine Learning Research, vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, Jul. 2010, pp. 297–304. [Online]. Available: https://proceedings.mlr.press/v9/gutmann10a.html

[29] A. Brébisson and P. Vincent, “An Exploration of Softmax Alternatives Belonging to the Spherical Loss Family,” Jul. 2015.

[30] M. Kurimo *et al.*, *Unlimited vocabulary speech recognition for agglutinative languages*. 2006. doi: 10.3115/1220835.1220897.

[31] J. T. Goodman, “A bit of progress in language modeling,” *Comput Speech Lang*, vol. 15, no. 4, pp. 403–434, 2001, doi: https://doi.org/10.1006/csla.2001.0174.

[32] L. Verwimp, J. Pelemans, H. Van hamme, and P. Wambacq, “Character-Word LSTM Language Models,” in *Conference of the European Chapter of the Association for Computational Linguistics*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:16889062

[33] I. Sutskever, J. Martens, and G. Hinton, “Generating text with recurrent neural networks,” in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, in ICML’11. Madison, WI, USA: Omnipress, 2011, pp. 1017–1024.

[34] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černocký, “Strategies for training large scale neural network language models,” in *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, 2011, pp. 196–201. doi: 10.1109/ASRU.2011.6163930.

[35] M. Sundermeyer, R. Schlüter, and H. Ney, “LSTM Neural Networks for Language Modeling,” Jul. 2012. doi: 10.21437/Interspeech.2012-65.

[36] R. Jozefowicz, O. Vinyals, M. Schuster, and N. Shazeer, “Exploring the Limits of Language Modeling,” Feb. 2016.

[37] T. Kudo, “Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, I. Gurevych and Y. Miyao, Eds., Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 66–75. doi: 10.18653/v1/P18-1007.

[38] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *North American Chapter of the Association for Computational Linguistics*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:52967399

[39] T. Brown *et al.*, “Language Models are Few-Shot Learners,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[40] A. Fesseha, E. Emiru, M. Diallo, and A. Dahou, “Text Classification Based on Convolutional Neural Networks and Word Embedding for Low-Resource Languages: Tigrinya,” *Information*, vol. 12, p. 52, Jan. 2021, doi: 10.3390/info12020052.

[41] Z. Wang and Q. Wu, “An Integrated Deep Generative Model for Text Classification and Generation,” *Math Probl Eng*, vol. 2018, pp. 1–8, Aug. 2018, doi: 10.1155/2018/7529286.

[42] Jason Brownlee, “Text Generation With LSTM Recurrent Neural Networks in Python with Keras,” 2017, Accessed: Jun. 22, 2024. [Online]. Available: https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/

[43] S. Welleck, I. Kulikov, S. Roller, E. Dinan, K. Cho, and J. Weston, “Neural Text Generation with Unlikelihood Training,” *ArXiv*, vol. abs/1908.04319, 2019, [Online]. Available: https://api.semanticscholar.org/CorpusID:199551982

[44] Y. Yüksel and Y. Çebi, “TR-SUM: An Automatic Text Summarization Tool for Turkish,” in *The International Conference on Artificial Intelligence and Applied Mathematics in Engineering*, Springer, 2021, pp. 271–284.

[45] N. Fatima, A. S. Imran, Z. Kastrati, S. M. Daudpota, and A. Soomro, “A systematic literature review on text generation using deep neural network models,” *IEEE Access*, vol. 10, pp. 53490–53503, 2022.

[46] J. Cho, J. Lei, H. Tan, and M. Bansal, “Unifying vision-and-language tasks via text generation,” in *International Conference on Machine Learning*, PMLR, 2021, pp. 1931–1942.

[47] S. Zhou, “Research on the application of deep learning in text generation,” in *Journal of Physics: Conference Series*, IOP Publishing, 2020, p. 012060.

[48] Markakis, John and Last, Geoffrey Charles. "Eritrea". Encyclopedia Britannica, 7 Jul. 2024, https://www.britannica.com/place/Eritrea. Accessed 7 July 2024.

Appendix A

A Sample Code used in the model implementation

```python
import os
import pandas as pd
import re

nyt_dir = 'E:/GResearch/GResearch/data/nyt_dataset/articles/New folder/'
css_code = """
@font-face {
    font-family: 'Washrab';
    src: url('washrabsl.ttf') format('truetype');
}
"""

all_headlines = []
for filename in os.listdir(nyt_dir):
    if 'Articles' in filename:
        with open(nyt_dir + filename, 'r', encoding='UTF-8') as file:
            for line in file:
                # Use regex to split by comma or colon
                split_line = re.split(r'[,:]', line.strip())
                if split_line:  # Check if the line is not empty
                    # Assuming the headlines are in the first part after splitting
                    all_headlines.append(split_line[0])  # Adjust index as necessary
# Remove all headlines with the value of "Unknown"
all_headlines = [h for h in all_headlines if h != "Unknown"]
len(all_headlines)

from tensorflow.keras.preprocessing.text import Tokenizer

# Tokenize the words in our headlines
tokenizer = Tokenizer()
tokenizer.fit_on_texts(all_headlines)
total_words = len(tokenizer.word_index) + 1
print('Total words: ', total_words)

# Print a subset of the word_index dictionary created by Tokenizer
subset_dict = {key: value for key, value in tokenizer.word_index.items() \
                if key in ['ክልተ','ጓቢይቲ ','ብርሃናት', 'ጦቢረ',' ኣቲ' ,'ጓብዩ' ,'ብርሃንን','ብመዓልቲ' ,'ክሰልጥን']}
print(subset_dict)
```

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10,6))
sns.histplot([len(all_headlines.split()) for all_headlines in all_headlines],bins=30,kde=True)
plt.title('Distribution of Headline Length')
plt.xlabel('Headline Length')
plt.ylabel('Frequency')
plt.show()

from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
from tensorflow.keras.models import Sequential

# Input is max sequence length - 1, as we've removed the last word for the label
input_len = max_sequence_len - 1

model = Sequential()

# Add input embedding layer
model.add(Embedding(total_words, 10, input_length=input_len))

# Add LSTM layer with 100 units
model.add(LSTM(100))
model.add(Dropout(0.1))

# Add output layer
model.add(Dense(total_words, activation='softmax'))

from tensorflow.keras.optimizers import Adam
model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001))

import numpy as np

def predict_next_token(seed_text):
    token_list = tokenizer.texts_to_sequences([seed_text])[0]
    token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
```

```python
def predict_next_token(seed_text):
    token_list = tokenizer.texts_to_sequences([seed_text])[0]
    token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')

    # Get prediction probabilities
    predictions = model.predict(token_list, verbose=0)

    # Get the index of the highest probability class
    predicted_class = np.argmax(predictions, axis=-1)

    # Optionally, convert the predicted class back to the corresponding token
    predicted_token = tokenizer.index_word[predicted_class[0]]  # Assuming you want the first result
    return predicted_token

# Example usage
prediction = predict_next_token('አብ ጠፈር')
prediction

def generate_headline(seed_text, next_words=4):
    for _ in range(next_words):
        # Predict next token
        prediction = predict_next_token(seed_text)
        # Convert token to word
        next_word = tokenizer.sequences_to_texts([prediction])[0]
        # Add next word to the headline. This headline will be used in the next pass of the loop.
        seed_text += " " + next_word
    # Return headline as title-case
    return seed_text.title()
seed_texts = [
    'ትግሬ ክአ አበይቲ']
for seed in seed_texts:
    print(generate_headline(seed, next_words=5))
```

```
import os
import pandas as pd
import re

nyt_dir = 'E:/GResearch/GResearch/data/nyt_dataset/articles/New
folder/'
css_code = """
@font-face {
    font-family: 'Washrab';
    src: url('washrabsl.ttf') format('truetype');
}
"""

all_headlines = []
for filename in os.listdir(nyt_dir):
    if 'Articles' in filename:
        with open(nyt_dir + filename, 'r', encoding='UTF-8') as
file:
            for line in file:
                # Use regex to split by comma or colon
                split_line = re.split(r'[,:]', line.strip())
                if split_line:  # Check if the line is not empty
                    # Assuming the headlines are in the first
part after splitting
                    all_headlines.append(split_line[0])  #
Adjust index as necessary
# Remove all headlines with the value of "Unknown"
all_headlines = [h for h in all_headlines if h != "Unknown"]
len(all_headlines)
```

Appendix B Sample List of Tigrigna Words

Sentence_ID, Token_ID, Word, POS_Tag

1,1,ነዚ,PRE
PPRONDND
ISG
1,2,ጨንፈር,
NCSG
1,3,ማሕበራዊ
,ADJSG
1,4,ስነልሳን,
NCSG
1,5,ካብ,PRE
P
1,6,መጀመር
ታ,NCSG
1,7,እዋን,NA
DVSG
1,8,ኣብ,PRE
P
1,9,ከባቢ,PR
EP
1,10,1950ታ
ት,NUCA
1,11,ከም,PR
EP
1,12,ኣካላት,
NCPL
1,13,መፅናዕ
ቲ,NCSG
1,14,ማሕበራ
ዊ,ADJSG
1,15,ስነ,NCS
G
1,16,ልሳን,N
CSG
1,17,ጌሩ,VM
1,18,ከምፅኦ
ም,CREL
1,19,ዝፈተነ,
CREL
1,20,ኣሜሪካ
ዊ,ADJSG
1,21,ሊቅ,AD
JSG
1,22,ስነልሳን,
NCSG
1,23,ኢነር,N
P
1,24,ሃገን,NP
1,25,እንትኽ
ውን,CREL
1,26,ነቲ,PRE
PPRONDFD
ISG
1,27,ፅንሰ,N
CSG
1,28,ሓሳብ,N
CSG
1,29,ብያነ,N
CSG
1,30,ንምሃብ’
ውን,PREPN
GSGCON
1,31,ፈቲኑ,V
M
1,32,እዩ,VA
UX
1,33,።,EPM
2,1,ኣብቲ,PR
EPPRONDF
DISG
2,2,እዋን,NA
DVSG
2,3,።,EPM
3,1,ከም,PRE
P
3,2,ኣገላልፃ,N
CSG
3,3,እቲ,PRO
NDFDISG
3,4,ምሁር,N
CSG
3,5,ትልሚ,N
CSG
3,6,ትልሚ,N
CSG
3,7,ቋንቋ,NC
SG
3,8,ማለት,N
CSG
3,9,“,OP
3,10,ናይ,PR
EP
3,11,ሓደ,N
UCA
3,12,ቋንቋ,N
CSG
3,13,ተዛራቢ,
NCSG
3,14,ዝኾነ,C
REL
3,15,ማሕበረ
ሰብ,NCSG
3,16,ዘለዎ,C
REL
3,17,ባህሪ,N
CSG
3,18,ኣጠቓቕ
ማ,NCSG
3,19,ቋንቋ,N
CSG
3,20,ብትል
ሚ,PREPNC
SG
3,21,ንምልዋ
ጥ,PREPNGS
G
3,22,ዝግበር,
CREL
3,23,ተግባር,
NCSG
3,24,እዩ,VM
3,25,”,CP
3,26,።,EPM
4,1,ካብዚ,PR
EPPRONDN
DISG
4,2,ብያነ,NC
SG
4,3,ብምብጋስ
,PREPNGSG
4,4,ድማ,CO
N
4,5,ትልሚ,N
CSG
4,6,ቋንቋ,NC
SG
4,7,ኣብ,PRE
P
4,8,ውሽጢ,P
REP
4,9,ሓደ,NU
CA
4,10,ቋንቋ,N
CSG
4,11,ሓደሽቲ,
ADJPL
4,12,ቃላት,N
CPL
4,13,ንክሕወ
ሱ,IV
4,14,ካብ,PR
EP
4,15,ምግባር,
NGSG
4,16,ጀሚሩ,
VM
4,17,ሓደሽቲ,
ADJPL
4,18,ቃላት,N
CPL
4,19,ክሳብ,P
REP
4,20,ምፍጣር
,NGSG
4,21,ዝኸይድ
,CREL
4,22,እዩ,VA
UX
4,23,።,EPM
5,1,ሓልሓሊ
ፉ,PRONQU
T
5,2,ትልሚ,N
CSG
5,3,ቋንቋ,NC
SG
5,4,ብፖሊሲ,
PREPNCSG
5,5,ቋንቋ,NC
SG
5,6,ተመሳሲሉ
,VM
5,7,ወይ,CO
N
5,8,ዝትክእሉ,
CREL
5,9,ኣጋጣሚ,
NCSG
5,10,ልሙድ,
ADJSG
5,11,እዩ,VM
5,12,።
,EPM
6,1,ነገር,NCS
G
6,2,ግን,CON

6,3,አብ,PREP
6,4,መንሳ,PREP
6,5,ክልተ,NUCA
6,6,ፅንሰ,NCSG
6,7,ሓሳባት,NCPL
6,8,መሰረታዊ,ADJSG
6,9,ክበሃል,VM
6,10,ዝኽእል,CREL
6,11,ኣፈላላይ,NCSG
6,12,ኣሎ,VM
6,13,እዩ,VAUX
6,14,።,EPM
7,1,እዚ,PRONDNDISG
7,2,ድማ,CON
7,3,ትልሚ,NCSG
7,4,ቋንቋ,NCSG
7,5,ሓደሽቲ,ADJPL
7,6,ዝኾነ,CREL
7,7,ኣብ,PREP
7,8,ዝተፈላለዩ,ADJPL
7,9,ፖለቲካዊ,ADJSG
7,10,፣,SP
7,11,ስነ,NCSG
7,12,ልሳናውን,ADJSGCON
7,13,ማሕበረሰባውን,ADJSGCON
7,14,ሽቶታት,NCPL
7,15,ንምዕዋት,PREPNGSG
7,16,መሰረት,NCSG
7,17,ጌሩ,VM
7,18,ዝቐርብ,CREL
7,19,ቋንቋ-ተኮር,NHYPSG
7,20,ውሳነ,NCSG
7,21,እንትኽውን,CREL
7,22,ትልሚ,NCSG
7,23,ቋንቋ,NCSG
7,24,ድማ,CON
7,25,ቋንቋ,NCSG
7,26,መሰረት,NCSG
7,27,ዝገበሩ,CREL
7,28,ፀገማት,NCPL
7,29,ንምፍታሕ,PREPNGSG
7,30,እንትበሃል,CREL
7,31,ዝሕንፀፅን,CRELCON
7,32,ብፖሊሲ,PREPNCSG
7,33,ኣቢሉ,VM
7,34,ድማ,CON
7,35,ዝፍፀም,CREL
7,36,ከይዲ,NCSG
7,37,ወይ,CON
7,38,ትልሚ,NCSG
7,39,እዩ,VM
7,40,።,EPM
8,1,መጠቓለሊ,NCSG
8,2,ትልሚ,NCSG
8,3,ቋንቋ,NCSG
8,4,ኣብ,PREP
8,5,ውሽጢ,PREP
8,6,ሓንቲ,NUCA
8,7,ዓዲ,NCSG
8,8,ማሕበረሰብ,NCSG
8,9,ዘለዉ,CREL
8,10,ቋንቋታት,NCPL
8,11,(,OPO
8,12,ላህጃታት,NCPL
8,13,),CPC
8,14,ደረጅኦም,NCSGBMO
8,15,፣,SP
8,16,ኣጠቓቕምኦም,NCSGBMO
8,17,፣,SP
8,18,ተግባሮም,NCSGBMO
8,19,ንምስፋሕ,PREPNGSG
8,20,ተባሂሉ,VM
8,21,ብሃገርለኽ,PREPNCSG
8,22,ኮነ,CON
8,23,ብውልቀ,PREPNCSG
8,24,ሰብ,NCSG
8,25,ደረጃ,NCSG
8,26,ናይቶም,PREPPRONDFDIPL
8,27,ተዛረብቲ,NCPL
8,28,ባህሪ,NCSG
8,29,ቋንቋ,NCSG
8,30,ብትልሚ,PREPNCSG
8,31,ንምልዋጥ,PREPNGSG
8,32,ዝግበር,CREL
8,33,ተግባር,NCSG
8,34,እዩ,VM
8,35,።,EPM
9,1,ነዚ,PREPPRONDNDISG
9,2,ተግባር,NCSG
9,3,ከም,PREP
9,4,መበገሲ,NCSG
9,5,ኮይኖም,VM
9,6,ከገልግሉ,VM
9,7,ዝኽእሉ,CREL
9,8,ድማ,CON
9,9,ዝተፈላለዩ,ADJPL
9,10,ርእዮተ,NCSG
9,11,ሓሳባት,NCPL
9,12,ልሳነ-ብሑሕነት,NHYPSG
9,13,፣,SP
9,14,ዓለምለኻዊነት,NCSG

9,15,፣,SP
9,16,አሃዳዊነ
ት,NCSG
9,17,ወይ,CO
N
9,18,ድማ,C
ON
9,19,ቨርናኩላ
ራይዜሽን,NB
ORSG
9,20,ክኾኑ,V
M
9,21,ይኽእሉ,
VAUX
9,22,።
,EPM
10,1,ወድዓዊ,
ADJSG
10,2,ኩነታት,
NCPL
10,3,ሃገራት,
NCPL
10,4,ማሕበረ
ሰባት,NCPL
10,5,አብ,PR
EP
10,6,ግምት,
NCSG
10,7,ብምእታ
ው'ውን,CON
10,8,ትልሚ,
NCSG
10,9,ቋንቋ,N
CSG
10,10,ክብገሰ
ሎም,VM
10,11,ዝኽእ
ሉ,CREL
10,12,ሞዴላ
ት,NBORPL
10,13,ምህላ
ዎም,NGSGB
MO
10,14,ምርኣ
ይና,NGSGB
MO
10,15,ይዝከር
,VM
10,16,እዩ,V
AUX
10,17,።
,EPM
11,1,ካብዚ,P
REPPROND
NDISG
11,2,ብተወሳ
ኺ'ውን,CON
11,3,ዘፈራት,
NCPL
11,4,ትልሚ,
NCSG
11,5,ቋንቋ,N
CSG
11,6,አብ,PR
EP
11,7,ክልተ,N
UCA
11,8,ቀንዲ,A
DJSG
11,9,ኣካላት,
NCPL
11,10,ከምዝ
ምቀልን,CREL
CON
11,11,ንሳቶ
ም,PRONPR
TPPL
11,12,ድማ,C
ON
11,13,ደረጃ
ውን,ADJSGC
ON
11,14,ስነ,N
CSG
11,15,ልሳና
ውን,ADJSGC
ON
11,16,ትልም
ታት,NCPL
11,17,ቋንቋ,
NCSG
11,18,ተባሂ
ሎም,VM
11,19,ከምዝ
ፍለጡ,CREL
11,20,ተገሊ
ፁ,VM
11,21,እዩ,V
AUX
11,22,።
,EPM
12,1,እዞም,P
RONDNDIP
L
12,2,ትልምታ
ት,NCPL
12,3,ቋንቋ,N
CSG
12,4,ብመንፅ
ር,PREPNCS
G
12,5,ሓሳብነ
ት,NCSG
12,6,እንትረኣ
ዩ,CREL
12,7,ዝተፈላ
ለዩ,ADJPL
12,8,ክኾኑ,V
M
12,9,ይኽእሉ,
VAUX
12,10,አብ,P
REP
12,11,ተግባር
,NCSG
12,12,እንት
ውዕሉ,CREL
12,13,ግና,C
ON
12,14,እቲ,P
RONDFDIS
G
12,15,ሓደ,N
UCA
12,16,በቲ,P
REPPROND
FDISG
12,17,ሓደ,N
UCA
12,18,እንት
ሕገዝ,CREL
12,19,ይርአ,
VM
12,20,እዩ,V
AUX
12,21,።
,EPM
13,1,ዶሞርት
,NP
13,2,(,OP
13,3,2006,
NUCA
13,4,:,SP
13,5,374,N
UCA
13,6,),CP
13,7,አብ,PR
EP
13,8,መንጎ,P
REP
13,9,ደረጃው
ን,ADJSGCO
N
13,10,ስነ,N
CSG
13,11,ልሳና
ውን,ADJSGC
ON
13,12,ትልም
ታት,NCPL
13,13,ቋንቋ,
NCSG
13,14,ጥንኩ
ር,ADJSG
13,15,ርክብ,
NCSG
13,16,ምህላ
ዉ,NGSG
13,17,አብ,P
REP
13,18,ቋንቋ,
NCSG
13,19,እብራ
ይስጥ,NP
13,20,ብዝገበ
ሮ,CREL
13,21,መፅናዕ
ቲ,NCSG
13,22,አቢሉ,
VM
13,23,ንምርኣ
ይ,PREPNGS
G
13,24,ፈቲኑ,
VM
13,25,እዩ,V
AUX
13,26,።
,EPM
14,1,ቋንቋ,N
CSG
14,2,እብራይ
ስጥ,NP
14,3,መጀመር
ታ,NCSG
14,4,አብ,PR
EP
14,5,ምድሪ,
NCSG