WORD SEQUENCE PREDICTION FOR AMHARIC LANGUAGE USING DEEP LEARNING

A Thesis Presented

by

Yared Wolderufael Woldetsadik

to

The Faculty of Informatics

of

St. Mary's University

In Partial Fulfillment of the Requirements
For the Degree of Master of Science

in

Computer Science

February 2024
Addis Ababa, Ethiopia

ACCEPTANCE

Word Sequence Prediction for Amharic Language Using Deep Learning

By

Yared Wolderufael Woldetsadik

Accepted by the Faculty of Informatics, St. Mary's University, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Thesis Examination Committee:

Internal Examiner
Shimelis Tamiru (Ph.D)

External Examiner
Minale Ashagrie(Ph.D)

Dean, Faculty of Informatics
Alembante Mulu (Ph.D)

February 2024

DECLARATION

I, the undersigned, declare that this thesis work entitled **Word Sequence Prediction for Amharic Language using BiLSTM** is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been duly acknowledged.

Declared by

Yared Wolderufael Wolderufael
Full Name of Student

_____
Signature

Addis Ababa

Ethiopia

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

Alembante Mulu (Ph.D)
Full Name of Advisor

_____
Signature

Addis Ababa

Ethiopia

January 2024

Acknowledgments

First and foremost, I express my sincere gratitude to the almighty God for providing me with the opportunity to undertake this endeavor and for guiding me throughout this journey. I want to extend my deepest appreciation to my advisor, Dr. Alembante Mulu for his invaluable support, mentorship, and unwavering commitment to my academic pursuits. His guidance in refining my research and ensuring precise language usage has been instrumental in shaping the outcome of this study.

I would also like to express my heartfelt thanks to my wife, Selamawit Tadesse, whose unwavering support and constant inspiration have been a source of strength and motivation throughout this undertaking. Her encouragement and belief in my abilities have been pivotal in sustaining me during challenging times.

Lastly, I am grateful to my family and friends for their unwavering support and motivation. Their continuous encouragement and belief in my capabilities have been instrumental in my progress. I would like to acknowledge their significant contributions in providing a conducive environment for me to focus on my work and pursue excellence.

Table of Contents

List of Acronyms and abbreviations

AAC	Augmentative and alternative communication
BERT	Bidirectional Encoder Representations from Transformers
Bi-LSTM	Bidirectional long short-term memory
CNN	Convolutional neural network
DSRM	Design Science Research Methodology
DL	Deep Learning
ELMO	Embedding from Language Models
LSTM	Long short-term memory
NLP	Natural language processing
POS	Part of speech
RNN	Recurrent neural network
RvNN	Recursive neural networks
SGD	Stochastic gradient descent
SMS	Short Message Service

List of Figures

List of Tables

Abstract

Textual communication is globally prevalent, with individuals relying on email and social networking platforms for information exchange. Word prediction systems offer a time-saving solution by anticipating the next word during data entry. However, typing complete text can be time-consuming. Despite the development of language models for various languages, research on prediction models for Amharic is limited. Existing studies primarily utilize statistical language models for Amharic prediction, which struggle with data sparsity and fail to capture long-term dependencies.

To address these limitations, this study proposes a deep learning approach for Amharic next-word prediction. The dataset is preprocessed and collected with a vocabulary of 18,085 unique words. Bi-directional Long Short-Term Memory (Bi-LSTM) models are employed, along with popular pre-trained word embedding models (Word2vec, Fasttext, Glove, and Keras) for feature extraction.

Experiments encompass various hyperparameter values and optimization methods (Adam and Nadam), significantly influencing model training and performance. Model accuracy is compared to identify the most effective solution for Amharic word sequence prediction.

Evaluation is conducted using accuracy measurements to assess overall prediction system correctness. Among the tested models, the Fasttext model combined with Bi-LSTM architecture and Adam optimizer achieves the highest training accuracy (97.5%) and validation accuracy (95.6%), surpassing other embedding methods. This research contributes to Amharic language model development, demonstrating the capacity to capture long-term dependencies and accurately predict the next word in Amharic text. The findings highlight the potential of Bi-LSTM-based approaches in enhancing text prediction systems.

Keywords*: Word prediction, Amharic language, Bi-LSTM, Word embedding, Fasttext, Long-term dependencies*

CHAPTER ONE

1. INTRODUCTION

1.1.Background of the study

Natural language processing (NLP) is a fascinating topic at the heart of computer science and artificial intelligence. To provide computers the ability to examine and comprehend the complexity of human language, from spoken words to written text. NLP works with massive datasets, applying advanced algorithms to extract meaning, assess sentiment, and even synthesize natural language. This opens the door to ground-breaking applications in machine translation, text summarization, and the development of intelligent machines that understand our words [1].

Ethiopians speak Amharic, It is the mother tongue of the Amhara area as well as the official working language of the federal government. In the Amhara regional state, it is also utilized as a regional working language, and many people in the country use it as a second language. The Amharic Fidel script (Ge'ez abugida) is written from left to right [2]. Amharic, is vital in Ethiopian economic, commercial, and political contexts [3].

The growing usage of computers and mobile devices in Ethiopia has increased the number of persons communicating in Amharic [2]. However, producing text in Amharic is difficult because most physical keyboards are intended primarily for English. The Amharic script is made up of thirty-four base characters that are organized into seven columns [4], the initial column contains the basic characters, while the subsequent six columns display the derived vowel sounds from the fundamental characters. Amharic consists of a total of 265 characters, which includes 27 labialized characters typically representing two sounds. (e.g., ቐ/(ḳʰä) for ቀዋ/(Qʷä)) [3]. Due to the vast number of characters, a physical computer keyboard cannot accommodate a key for every character, leading to text that is challenging to decipher.

An effective method to tackle the challenge of composing Amharic text involves utilizing a word prediction system. This system, which falls under the umbrella of natural language processing, is designed to forecast the following word by analyzing preceding word sequences. Referred to as word prediction or language modeling [4], this tool assists users in entering words by predicting the next probable word and offering a selection of potential options [5].

Next word prediction can help people with physical limitations by reducing writing errors, speeding up communication, and conserving energy during text composition [6]. The potential benefits of word prediction systems for Amharic are multifaceted, extending beyond mere ease of typing. Research suggests such systems can foster typing skills, enhance user concentration, boost self-esteem, and promote autonomous writing [7]. However, the inherent grammatical complexity of Amharic poses a significant challenge to the development of robust word prediction models for this language.

N-gram models and other conventional word prediction techniques employ Markov chains to gauge the probability of the next word by considering the preceding sequence of words. Nevertheless, when implemented in Amharic, these approaches prove inadequate due to limited data availability, extensive complexity, and the incapacity to capture meaningful connections among words [8]. The intrinsic intricacies of the Amharic language, defined by a huge array of affixes and morphological variants, present substantial hurdles for traditional prediction approaches.

To tackle these challenges and make the Amharic language more accessible for daily activities, a word prediction system that reduces the time and effort required to write in Amharic script has been developed. The researcher is inspired to create a next-word prediction system that will save time and effort when writing in Amharic script.

1.2.Motivation

As computers and mobile devices become ever-present in Ethiopia, the ability to type effectively in Amharic, the dominant language, becomes crucial for daily communication. However, a significant barrier stands in the way: a lack of dedicated tools and resources for Amharic text input. While word prediction software exists for numerous languages, from English and Italian to Persian and Swedish [6, 9, 10, 11], Amharic remains largely neglected. This gap is particularly evident in the area of deep learning models, powerful tools for predicting word sequences with accuracy. Without Amharic-specific models, navigating the digital world remains clumsy and inefficient for millions of Ethiopians. This lack of research and development impedes not only individual productivity but also hinders wider participation in the digital sphere, potentially disadvantaging Amharic speakers in communication, education, and economic opportunities. Addressing this gap

through dedicated research and development efforts is essential to ensuring equitable access and inclusion in Ethiopia's digital future.

To address this issue, we are motivated to develop an Amharic Word Sequence Prediction system using a deep learning approach of the Bidirectional Long Short-Term Memory (BLSTM) model that can help speed up text entry and eliminate spelling errors. Our study focuses on word sequence prediction, which can be particularly beneficial for people with limited vocabulary or those who struggle with spelling. The morphological characteristics of Amharic, which include complex inflectional and derivational verb morphology, make it challenging to develop accurate and efficient word prediction systems. However, by using deep learning models like BLSTM, we can overcome these challenges and create a language-specific tool that can benefit a large number of people who communicate mainly in Amharic.

1.3.Problem Statement

Modern digital interactions rely significantly on text entry, which is frequently accomplished using keyboards intended primarily for English characters. This is particularly difficult for Amharic speakers, whose language employs a script unique from the common Latin alphabet. While there are alternate input techniques available, such as Amharic-specific software packages like Power Geez, GeezIME, and Abnet, they frequently need laborious multi-key combinations for individual characters, slowing down the writing process [12].

In response to this challenge, word prediction models offer a promising solution. By analyzing the sequence of already-typed words, these models can predict the most likely next word, potentially reducing both typing time and spelling errors. This technology holds the potential to significantly improve the efficiency and ease of Amharic text entry, benefiting communication, education, and overall digital participation for Ethiopia's Amharic-speaking population.

While word prediction software has proven to enhance typing speed and efficiency, especially for slower typists or those with limited keyboard space, its use in Amharic is mainly unexplored. This is owing, in part, to the language's complex morphological structure, which presents particular hurdles for word sequence prediction. While N-gram models were used in the initial investigations, their disadvantages, such as data sparsity and restricted management of long-term dependencies, necessitated the exploration of alternate methodologies.

To the best of our knowledge, there are only a few research attempts on word sequence prediction in the Amharic language, especially using deep learning models. Our study aims to fill this gap by developing an accurate and efficient Amharic word sequence prediction system that can improve text entry performance and promote digital inclusion in Ethiopia. With the rapid growth of electronic devices and computers in the country, language-specific tools like Amharic word sequence prediction can play a crucial role in improving digital communication and accessibility for all users, including those with limited vocabulary, poor spelling, or disabilities. By addressing the challenges posed by the complex morphology of Amharic, our study can contribute to the development of language-specific tools for under-resourced languages and promote the use of electronic devices and computers in Ethiopia

This research aims to address these limitations by developing an Amharic word sequence prediction model utilizing a Bidirectional Long Short-Term Memory (BLSTM) architecture. As a powerful deep learning technique specifically designed for sequential data, BLSTM offers the potential to overcome the aforementioned challenges and significantly improve Amharic text entry.

1.4. Research Question

The following research questions are addressed in this study:

1. What are the optimal hyperparameter values that can enhance performance in the Word Sequence Prediction Model?
2. Which pre-trained word embedding model produced the best results?
3. How effective are deep learning algorithms of Bi-LSTM for Amharic Word Sequence Prediction?
4. What are the most significant challenges for Amharic language models, according to research?

1.5. Objective of the Study

1.5.1. General Objective

The general objective of the proposed study is to develop a Word sequence Prediction Model for Amharic using Deep Learning.

1.5.2. Specific Objective

To achieve the above-mentioned general objective the following specific objectives are performed: -

- To review the existing literature on Amharic Word Sequence Prediction, providing an overview of current theories and methodologies while identifying research gaps for further investigation.
- To identify the challenge of the Amharic language in language model
- To collect and prepare representative dataset for Amharic word sequence prediction.
- To propose the development of a word sequence prediction model for the Amharic language by leveraging pre-trained word embedding models, optimizing hyperparameter values, and employing the Bi-LSTM deep learning algorithm.
- To evaluate performance of the word sequence prediction model

1.6.Methodology

This part explains the entire study procedure, which will take place sequentially with the goal of successfully and efficiently answering the presented research problem, complemented by suitable performance evaluations. As a result, each study process is detailed separately, as follows:

1.6.1. Research Design

The design science research approach methodology was applied in this research, It is a comprehensive scientific discipline that focuses on the creation and implementation of practical solutions in various fields including software engineering, computer science, information science, and information technology [13]. It encompasses a range of research techniques that employ design and development to gain insights into fundamental processes. The DSRM methodology consists of six distinct stages [13].

- ✓ Define the problem and motivate: It outlines the particular research problem and provides reasoning for the importance of a resolution. Justification of the solution's value serves two purposes: it encourages both the researcher and the research audience to seek the solution, and it enables the audience to recognize the researcher's grasp of the issue.
- ✓ Define the objectives for a solution: It is possible to deduce from the problem definition and understanding of what is achievable and practical. The objectives should be logically deduced from the problem specification.

- ✓ Design and creation of the artifact: a DSR artifact can encompass any designed object that incorporates a research contribution within its design. This process involves defining the desired functionality and structure of the artifact before proceeding to its creation.
- ✓ Demonstration: It is an exercise that showcases the application of the artifact to resolve one or multiple instances of the problem. This may encompass its utilization in experimentation, simulation, case study, proof, or any other suitable undertaking.
- ✓ Evaluation: it assesses the effectiveness of the artifact in facilitating a solution to the problem. This process includes analyzing whether the intended goals of the solution align with the practical outcomes observed during the use of the artifact within its specific context.
- ✓ Communication: all relevant stakeholders are informed about every aspect of the problem and the designed artifact. The communication methods used are tailored to the research goals and the audience, which may include practicing professionals.

1.6.2. Literature Review

Conducting a literature review is essential to conduct a comprehensive evaluation and analysis of prior research relevant to the current study. The focus is on the thesis, specifically the field of study, such as the language and datasets utilized; the methodology employed; the techniques for data collection and preparation; the tools utilized for prototype development and analysis; and the evaluation of the performance of these approaches and tools to ascertain their potential applicability to the newly proposed study.

1.6.3. Dataset Collection

This research aims to predict the next word in an Amharic sentence, considering the surrounding words and context. To achieve this, we gathered data from Amharic news articles, a common source of written language. A crucial challenge for any learning system is determining how much data it needs to perform well. Ideally, we would use a vast collection of text (corpus), but we also need to consider how to best choose which data to use for training. Therefore, we opted for a simple method where each sentence had an equal chance of being selected. This resulted in a dataset containing 3496 sentences from various news sources. From the collected sentence, we collect a unique word of 18,085 this dataset is divided into training, validation, and testing sets, allowing us to train our system and evaluate its performance in predicting the next word.

1.6.4. Implementation Tools

The study's experiments were conducted utilizing a prototype that was developed with the use of Python and open-source libraries.

Python: is a robust programming language renowned for its capability to manage various modules and conduct analysis on mathematical operations.

Keras is a Python API that interfaces with platforms such as TensorFlow, offering robust backing for a range of neural network components and enabling smooth incorporation with Python.

TensorFlow is an open-source numerical computation platform, is utilized to provide efficient computational abilities on both CPUs and GPUs, thus improving the overall performance of the model.

NumPy is an essential Python library that plays a crucial role in scientific computing. It offers comprehensive assistance for performing mathematical operations on arrays and matrices with multiple dimensions. These arrays and matrices are widely utilized in various data preprocessing and manipulation tasks.

Pandas is a versatile library for data manipulation and analysis. It offers data structures and functions to efficiently handle structured data, such as data frames, which are commonly used in preprocessing and organizing input data for machine learning models.

Scikit-learn is a comprehensive library for machine learning in Python. It provides a wide range of tools and algorithms for tasks such as data preprocessing, feature extraction, and model evaluation. In the context of word sequence prediction, sci-kit-learn can be used for data preprocessing and splitting datasets into training and testing sets.

Gensim is a Python library specifically designed for topic modeling and document similarity analysis. It also includes implementations of popular word embedding algorithms such as Word2Vec, which can be utilized for generating word embeddings in the BiLSTM model.

1.6.5. Model Evaluation

Accuracy: measures the percentage of words the model correctly predicts in a given sequence. Essentially, it tells you how often the model's guess for the next word matches the actual word in the text. To calculate accuracy, we compare the sequence of words predicted by the model with the actual sequence in the reference text. For each word, we mark a match if the predictions and reference words are identical. Finally, we divide the total number of matches by the total number of words and multiply by 100% to get the accuracy percentage. The formula for Accuracy is: [14]

$$Accuracy = \frac{\text{Total number of maches}}{\text{Total number of words}} * 100\%$$

Perplexity: measures the average difficulty of predicting the next word in a text, with lower perplexity indicating better performance. It is a key metric used to evaluate the performance of language models, like word prediction. Cross-entropy measures the model's uncertainty about the next word. Lower cross-entropy signifies higher confidence in the prediction, leading to a lower perplexity. The formula for perplexity (PP) is: [15]

$$\text{PP} = 2^{(average\ cross-entropy\ per\ word)}$$

Keystroke saving: a widely adopted metric for evaluating word prediction systems, quantifies the reduction in typing effort achieved by utilizing suggested words. It is defined as the difference in the number of keystrokes required to type a text without prediction (KT) and the effective number of keystrokes used with prediction (KE). [7, 8]

$$KSS = \frac{KT - KS}{KT} * 100\%$$

The higher the keystroke saving (KSS), the better the word prediction system performs. It's like a shortcut – the more keystrokes you save with suggestions, the smoother and faster your typing experience. Conversely, a low KSS indicates the system isn't offering much savings, suggesting its performance needs improvement.

1.7.Scope and Limitation

This research aims to develop a deep learning-based word sequence prediction for Amharic. The model utilizes frequent 7-gram sequences for prediction. Linguistic features like syntax, semantics, and pragmatics are not incorporated due to resource limitations and time constraints. However, word embedding techniques are employed to represent word features using distributed vectors. User interface design is out of the scope of this study due to time limitations.

1.8. Significance of the Research

This research holds significant value for the field of Amharic Natural Language Processing. By establishing a dedicated Amharic sentence dataset, identifying the most effective algorithm for word sequence prediction, and optimizing hyperparameters for Bi-LSTM models, this work lays the groundwork for future research and development in this domain. It empowers future researchers and developers with crucial resources and insights to create more accurate and efficient Amharic word prediction systems.

1.9. Organization of the Thesis

The remaining sections of this thesis are structured as follows. Chapter 2 conducts a concise literature review, examining key concepts in word prediction, available methods, and relevant research by other scholars. This includes their approaches and findings on word sequence prediction. Chapter 3 delves into the intricacies of the Amharic language, exploring its structure and grammatical norms. Chapter 4 offers a comprehensive explanation of the proposed word sequence prediction model, detailing its architecture, underlying strategy, and relevant considerations. Chapter 5 presents the experimental setup, conducted evaluations, and insightful discussions. Finally, Chapter 6 provides a conclusive summary of the research and outlines potential avenues for future work.

CHAPTER TWO

2. LITERATURE REVIEW AND RELATED WORK

2.1. Overview

This chapter lays the groundwork for understanding word sequence prediction and its various approaches. We explore fundamental concepts and methods, including statistical, knowledge-based, and heuristic techniques, in order to present a thorough analysis of the subject.. As this study focuses on building a word sequence prediction model for Amharic using deep learning, we delve deeper into key concepts like language models, deep learning principles, hyperparameter optimization, and relevant research on Amharic and foreign language prediction models. Each of these topics is covered in dedicated sections throughout the chapter

2.2. Word Sequence Prediction

In today's digital age, the sheer volume of text and documents generated electronically has propelled computers and technology to become crucial tools in our daily lives. We rely on numerous gadgets to enter textual information and transmit it across the digital world, whether we're writing emails, preparing reports, or engaging in online discussion. Over the last decade, there has been substantial progress in the ability to process Amharic documents using computers, mobile phones, and other electronic devices [2]. This has resulted in an exponential increase in electronically stored papers, information, and databases. Text input into computers can be accomplished using keyboards or other novel approaches.

The development of text prediction systems can be traced back to the post-World War II era when an increase in physical disabilities needed the development of assistive technologies. Word prediction, a pioneering form of assistive technology, has emerged as a critical tool for bridging the gap between those with physical limitations and the digital world. Researchers worked tirelessly to create systems that not only accommodated users' limitations but also enhanced their talents. Prediction systems have been firmly established as essential components of the assistive technology environment since the early 1980s [16].

Word prediction in natural language processing addresses the problem of forecasting the next word in a given text sequence [6]. This entails examining the prior words and utilizing statistical or machine-learning approaches to determine the most likely word to follow. The terms "text

prediction," "word prediction," and "word completion" are frequently used interchangeably, yet there are subtle differences between them. term prediction attempts to estimate the user's intended term based on the prior context, whereas word completion attempts to predict the word the user is now typing based on the partially inserted characters or words.

Word prediction models are intended to improve text entry efficiency across a variety of applications by reducing the number of keystrokes necessary. They are especially useful in language learning as they suggest appropriate words for non-native speakers, minimize spelling errors for users with limited language proficiency, and reduce the overall effort involved in text composition, especially for individuals with physical disabilities. These approaches efficiently speed up the writing process and enable users to communicate more effectively.

2.3. Word Sequences Prediction Approaches

There are three main ways to predict the next word in a sequence: statistically, with linguistic rules, and by adapting to user input and context [2]. Statistical methods rely on how often words appear together in a large dataset. The more often two words show up next to each other, the more likely the second word is to follow the first. Linguistic methods use the grammar and structure of the language to predict the next word. These methods rely on dictionaries and rules about how words can be combined. Adaptive methods learn and improve as they're used. They start with basic statistical or linguistic models, but then adjust their predictions based on real-time feedback from users and the specific context of what's being written [2].

2.3.1. Statistical Word Sequence Prediction

Statistical word prediction relies on the likelihood of words within a specific text to anticipate letters, words, and phrases. This method is grounded on the Markov assumption, which posits that the preceding n-1 words dictate the subsequent word. However, these models can produce incorrect or nonsensical predictions, and it is not possible to preserve all word forms in languages with complex morphology. Prediction using frequencies, which proposes the most frequent words based on a dictionary, and prediction using word probability tables, which contain conditional probabilities of words that follow each other, are two common ways. These models improve prediction accuracy but demand more data storage and computational complexity; they perform best for languages with no inflection [2].

2.3.2. Knowledge-based Word Sequence Prediction

This approach uses linguistic knowledge or rules to analyze the order of word types (syntax), meaning (semantics), and context (pragmatics) in an input sentence.

Syntactic prediction

To achieve grammatical accuracy, syntactic prediction in word suggestion takes into account part-of-speech tags and phrase structures. The algorithm can make reasonable predictions about the type of words that will follow by following grammar rules. Statistical syntax assigns probabilities to candidate words based on their syntactic categories and POS tags, while rule-based grammar parses the present phrase and determines word categories. These approaches improve the accuracy of suggested words by using syntactic information.

There are two methods for predicting syntax: probability tables and grammars [12] . The probability table technique combines syntactic information by evaluating the probability of each word and the likelihood of syntactic categories appearing consecutively. Based on the current position in the phrase, the algorithm suggests words with the most probable syntactic categories, resulting in more precise predictions compared to frequency-based techniques. The two-dimensional table containing category probabilities is smaller than the frequency-based approach, with fewer probabilities close to zero. The table and lexical frequencies can be adjusted to allow for system adaptation. On the other hand, the grammar-based approach analyzes sentences using either top-down or bottom-up techniques and utilizes natural language processing to identify categories with the highest likelihood of occurrence. Each language has its own set of syntactic rules, and the right-to-left structure assists in category decomposition. Morphological agreement constraints can be established between categories to ensure that suggested words possess the appropriate morphological characteristics. These systems have a higher computational complexity as they consider the entire beginning of a sentence. In these systems, word probabilities and syntactic rule weights can be modified for adaptation.

Semantic prediction

Semantic prediction is assessing words in real-time based on their semantic meaning, with each word connected with one or more semantic categories [2, 12]. The method and complexity of

semantic prediction are comparable to syntactic approaches based on grammar, although semantic prediction is more sophisticated and less often utilized.

The utilization of the lexical source and lexical chain is a common practice in semantic word prediction. The lexical source, exemplified by WordNet in English, assesses the probability of words being interconnected within a given context to ensure that the predicted words are contextually appropriate. On the other hand, the lexical chain gives priority to words that are semantically linked in the context by eliminating unrelated words from the list of predictions. While these predictions may be syntactically or semantically valid, they can still be flawed in terms of discourse due to the significant impact of pragmatics on the predictor's proficiency. Enhancing prediction accuracy is achieved by incorporating pragmatic knowledge during the training of the system.

Pragmatics prediction

Pragmatics prediction pertains to the capacity of a model or system to foresee the subsequent word in a sequence, surpassing mere grammatical accuracy and semantic coherence. It focuses on predicting the word that is most fitting and appropriate within the specific context of the discourse [2, 14].

2.3.3. Heuristic Word Prediction

Heuristic word prediction aims to personalize predictions by dynamically adapting to individual user preferences and behavior. This adaptation can be achieved through two primary approaches: short-term and long-term learning.

Short-Term Learning:

This method focuses on adapting predictions based on the present context of the user's input. Several strategies can be used in this case, including Recency promotion is the practice of preferring recently used words for suggestion. Topic guidance: customizing suggestions depending on the current text's inferred topic. Trigger-target pairs: linking often co-occurring words to provide context-aware suggestions; and finally, N-gram cache: using frequently encountered n-grams (word sequences) to predict the next word in a similar context [12]

.

Long-Term Learning

This method remembers the user's writing habits by looking at all the text they've written before, not just the current sentence. The more the user types, the better the system learns their preferences and adjusts its suggestions accordingly. This gradual adaptation is called heuristic learning [12]. It helps in Learning new words, Suggesting different forms of words based on the context and the user's writing style, and recommending compound words the user often uses together.

2.4.Language model

In the field of natural language processing, language models play a crucial role in facilitating machine comprehension of textual information. These models are primarily employed for probabilistic analysis, tasked with calculating the likelihood of a specific sequence of tokens occurring within a corpus – a curated collection of text documents [17]. Word prediction systems use two main approaches: statistical models and neural models. Statistical models rely on numbers - they analyze how often words appear together to guess the next likely word. Think of it like counting marbles in a bag - the most frequent colors are the most likely to be picked next. Neural models, inspired by the brain, use interconnected "neurons" to learn complex patterns. Imagine a maze these neurons navigate, getting better at predicting the next turn with each attempt.

2.4.1. Statistic language modeling

Statistical language models (SLMs) are strong natural language processing (NLP) tools that anticipate the next word in a sequence based on previous words. some of the statistical language model types are word frequency, word sequence frequency, Pos-tagging, and N-gram models [18].

2.4.1.1. Word Frequencies

Early systems simply looked at how often individual words appeared. This led to repetitive suggestions, ignoring the surrounding context [19].

2.4.1.2. Word Sequence Frequencies

Taking a step further, these models consider the previous word(s) to influence the next prediction. So, "the quick" is more likely to be followed by "brown" than "apple." This captures some grammar and meaning but still has limitations [18, 20].

2.4.1.3. Part-of-Speech Sequences

Instead of looking at individual words, this method tracks the sequence of grammatical tags like "noun" or "verb." This allows for a broader context but sacrifices some specific meaning [18, 21].

2.4.1.4. N-gram Language Models

These are the stars of statistical prediction. The likelihood of a word sequence is calculated by analyzing the preceding words. Think of it like connecting sentences of marbles - certain combinations appear more often together. N-grams are powerful tools in speech recognition and handwriting analysis but struggle with capturing larger contexts [22, 23].

2.4.2. Neural Language Models

Neural Language Models (NLMs) are a powerful class of machine learning models that utilize artificial neural networks to process and generate human language. NLMs best perform tasks like machine translation, text summarization, dialogue systems, and even creative text generation. Neural language models (NLMs) have indeed advanced N-gram language models in many aspects. N-grams only consider a fixed-size window of previous words, leading to issues with long-range dependencies and unseen n-grams. In addition, N-grams struggle to capture complex semantic and syntactic structures in language. NLMs have high accuracy and performance on various NLP tasks and improve their performance by learning from new data, making them adaptable to changing language patterns and trends. However, its benefits NLMs have challenges in training, and running large NLMs requires significant computational resources. [24, 25]

2.5.Deep Learning

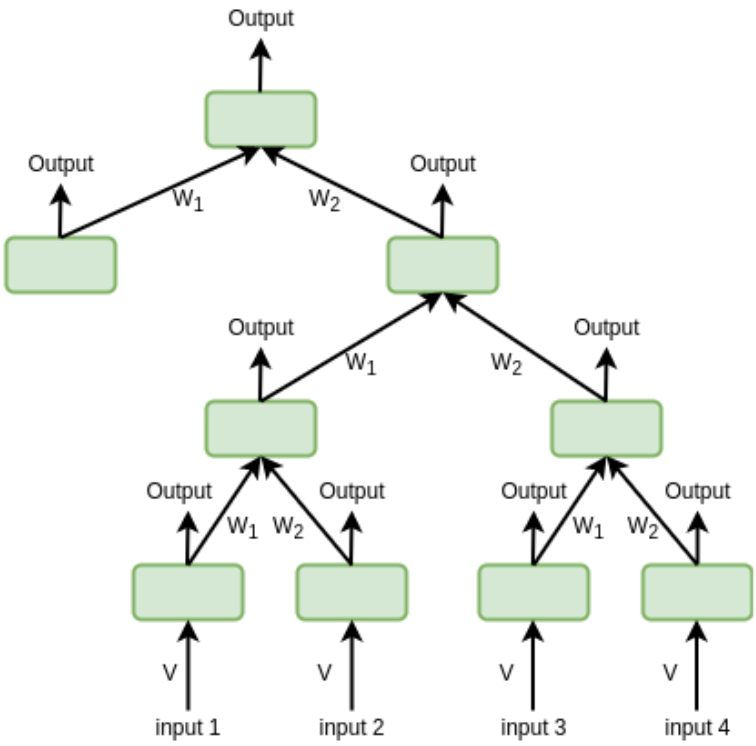
Deep learning is a revolutionary machine learning technique that excels where traditional methods falter: on complex, unstructured data. [26] Unlike traditional methods, which require extensive feature engineering, deep learning models can automatically learn features from data at multiple levels, progressively building a refined understanding. This ability, coupled with the explosion of available data and the rise of powerful computing hardware, has propelled deep learning into the forefront of machine learning, empowering it to tackle ever-more challenging tasks. [26, 27]

Deep learning's true power lies in its ability to learn from data in a way traditional methods simply cannot. By leveraging the power of multi-layered neural networks, deep learning models can extract intricate features from data at multiple levels, progressing from basic patterns to complex concepts [26, 27] This feat, once unimaginable, is now a reality thanks to the vast amount of data

available and the development of powerful hardware capable of handling the immense computational demands. As a result, deep learning has become a transformative force in machine learning, unlocking breakthroughs in areas ranging from computer vision to natural language processing, cybersecurity, bioinformatics, robotics and control, and medical information processing [27].

2.5.1. Recursive Neural Networks

Recursive Neural Networks are a powerful type of deep neural network designed to excel in two key areas: hierarchical forecasting and compositional vector classification. Unlike traditional neural networks, RvNNs thrive on data with inherent structure, such as graphs or trees. In RVNNS the network wouldn't simply analyze each word individually; it would understand the hierarchical relationships between words, phrases, and clauses. The understanding of the structure gives the power to forecast in a hierarchical fashion which means predicting future behavior based on the underlying structure of the data, not just individual elements. Moreover, structure understanding helps in classifying outputs using compositional vectors that represent the meaning of a complex object like a sentence as a combination of the meanings of its smaller parts like words, and phrases [27] .



Fifigure 2. 1 Recursive Neural Networks

2.5.2. Convolutional Neural Network

The **Convolutional Neural Network (CNN)** is a popular and powerful algorithm in the area of deep learning [27]. What sets it apart from older methods is its ability to automatically discover important features in data, without needing humans to hand-pick them. This makes it incredibly versatile and effective for various tasks like image recognition, audio analysis, and even face recognition. Just like traditional neural networks, CNNs draw inspiration from the way neurons work in the brain [27]. Imagine you're looking at a picture. A CNN would assign weights and values to different objects in the image, helping it distinguish one from another. This unique ability allows it to capture the relationships between pixels and understand the overall scene, making it less dependent on pre-processing compared to other algorithms. Figure 2 illustrates the basic structure of a CNN.

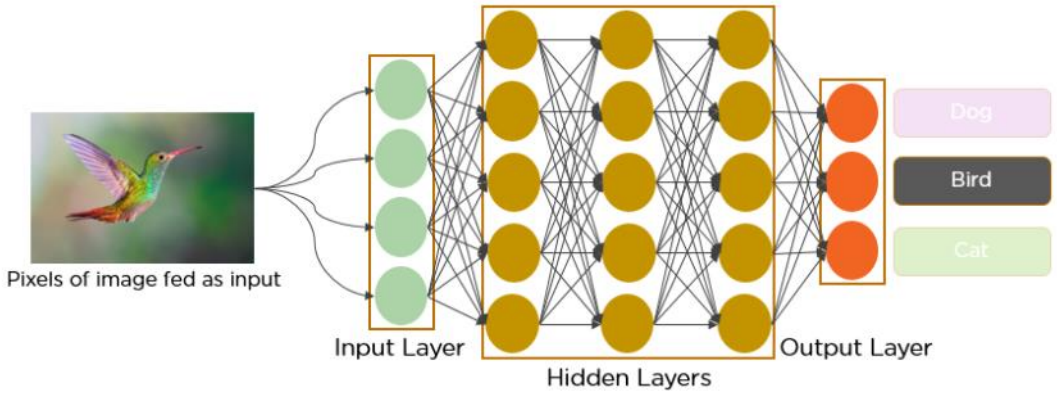


Figure 2. 2 Convolutional Neural Networks

2.5.3. Recurrent Neural Network

Recurrent Neural Network **(RNN)** is a form of artificial neural network that is designed to analyze sequential data such as time series or natural language. [28] Unlike feedforward neural networks, which linearly process data from input to output, RNNs have recurrent connections that allow them to maintain an internal state or memory. RNNs possess the unique advantage of recurrent connections. These connections form internal feedback loops, enabling the network to maintain a dynamic internal state, effectively serving as its memory. This inherent memory allows RNNs to capture temporal dependencies and patterns within sequential data, making them particularly adept at tasks involving temporal interactions [27].

The core operation of an RNN lies in its step-by-step processing of input sequences. Each step corresponds to a specific position or time frame within the sequence. At each step, the RNN receives an input vector and integrates it with the internal state inherited from the preceding step. This fusion generates an output while simultaneously updating the network's internal state. This iterative process repeats for each step in the sequence, permitting the RNN to progressively capture and propagate information across time [27, 28].

Assume x = (x1, x2,...., xT) represents a sequence of length T, and h_t represents RNN memory at time step t, and an RNN model updates its memory information using the following formula:

$$h_t = \sigma(W_x x_t + W_h h_t - 1 + b_t) \quad (1)$$

where σ is the activation function, often a non-linear function like sigmoid or tanh. It introduces non-linearity to allow the RNN to learn complex relationships. $W_x x_t$ is the weight matrix for the input at time step t. It determines how much each input feature contributes to the hidden state.$W_h h_1$ is the weight matrix for the previous hidden state ht-1 to control how much information from the past is carried forward. Bt is the bias vector at time step t. It helps adjust the overall output of the activation function.

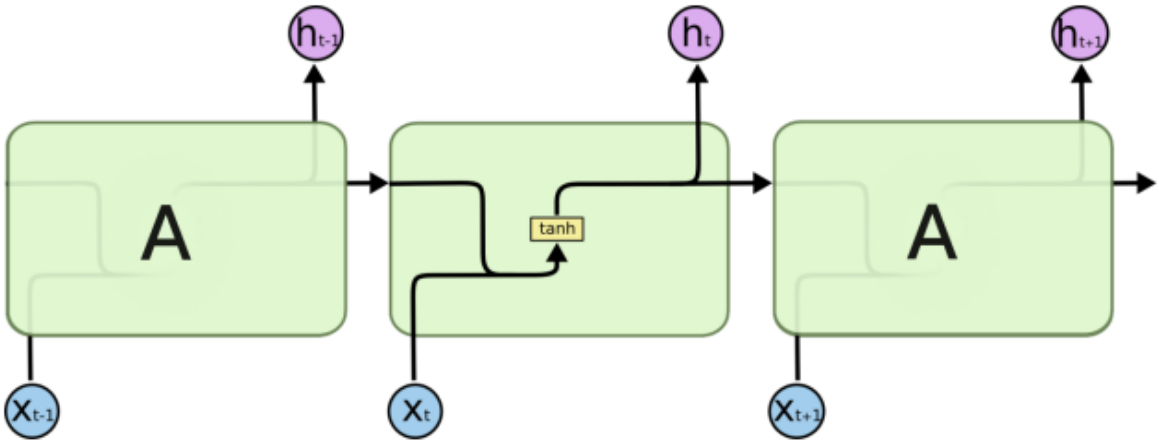


Figure 2. 3 RNN with the repeating module

2.5.4. Long Short-Term Memory

Long Short-Term Memory is a specialized RNN architecture that is adept at capturing long-term dependencies in sequential data, overcoming the issues of vanishing or exploding gradients that traditional RNNs face when processing lengthy sequences. LSTMs employ a clever internal structure that allows them to effectively learn over extended periods [29]. The core of an LSTM lies in four cell structure, which includes the forget gate, Input gate, cell state, and output gate. The forget gate will decide which information from the previous hidden state (ht-1) to retain by combining it with a new weight vector. The input gate chooses which parts of the current input (xt) to incorporate through linking with the new weight vector. The cell state stores the actual long-term memory and is updated using the activation function and based on the forget and input gates. The current hidden state (h_t) is determined by applying another activation function to the updated cell state and a weight vector.

In LSTM, gates operate like tiny decision mechanisms, regulating the flow of information through the LSTM cell. They ensure that only relevant data persists in the long-term memory while irrelevant or outdated information is discarded. LSTM can analyze not only single data points but also complete data sequences like as audio or video. LSTM, for example, is useful for tasks like networked handwriting recognition, speech recognition Machine translation, and Text generation. [30]

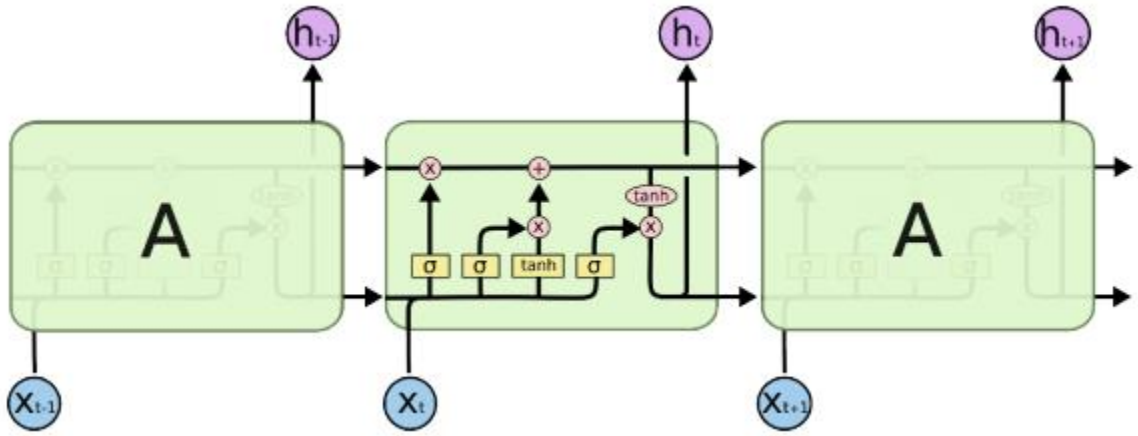


Figure 2. 4. An LSTM's repeating module consists of four interacting layers.

Figure 2. 5 shows the flow of information inside a LSTM network. Imagine each yellow box as a learning station

Figure 2.4 illustrates the information flow within an LSTM network, visualize each yellow box as a station for learning, processing information on its way through the network. The lines connect these stations, carrying information like arrows on a map. Pink circles represent simple operations like adding information together. One special feature of LSTMs is the cell state. Think of it as a long conveyor belt running through the entire network. Information can travel along this belt easily, almost untouched. This helps us remember important things for a long time, even as new information comes in.

Forget Gate:- It receives two inputs: the previous hidden state (h_{t-1}), which **is** the accumulated information from previous elements in the sequence**,** and the weight vector (W_f**),** which assigns

importance to different parts of the hidden state. It then combines these inputs with an activation function of the sigmoid and the bias vector (b_f) to adjust the "forgetting factor (f_t) between 0 and 1. The forgetting factor is derived as follows:

$$f_t = \sigma(W_{fh}\,[h_{t-1}], W_{fx}\,[x_t], b_f) \quad (2)$$

Input Gate:-It receives three inputs, current input (x_t), previous hidden state (h_{t-1}), and weight Vectors(W_i and W_h). The current input(x_t) has new information from the current element in the sequence. The previous hidden state(h_{t-1}) has information about the past data. weight vector has assigned a weight to different parts of the input and hidden stats. The input gate harmonizes these inputs through a sigmoid activation function, yielding a "gate value" (i_t) bounded between 0 and 1. This value acts as a selective filter, determining the extent to which the current input permeates the cell state.

$$i_t = \sigma(W_{ih}\,[h_{t-1}], W_{ix}\,[x_t], b_i) \quad (3)$$

Then the gate constructs a candidate cell state (c_t) by applying a tanh function to the filtered input. This candidate holds potential additions to the long-term memory, awaiting final approval.

$$c_t = tanh(W_{ch}\,[h_{t-1}], W_{cx}\,[x_t], b_c) \quad (4)$$

The combination of these two layers provides an update for the LSTM memory in which the current value is forgotten using the forget gate layer through multiplication of the old value (i.e., c_{t-1}) followed by adding the new candidate value ($i_t * c_t$). The following equation represents its mathematical equation:

$$c_t = f_t * c_{t-1} + i_t * c_t \quad (5)$$

when f_t value must be a value between 0 and 1 where ft = 0 it indicates that completely get rid of the value; whereas, ft =1 implies completely reserving the value.

Output Gate:- has two inputs, which are updated cell state (c_t) and weight vector (w_o). The update cell state vector encapsulates the network's current long-term memory, containing knowledge distilled from past inputs and refined through the forget and input gates' meticulous filtering processes. Weight vector (w_o) guides the gate's focus towards specific elements within the cell state, highlighting those deemed most pertinent for the immediate task at hand.

$$o_t = \sigma(W_{oh}\,[h_{t-1}],\, W_{ox}\,[x_t],\, b_o) \quad (6)$$

The output gate methodically combines these inputs using a sigmoid activation function, providing a "gate value" (o_t) confined within the range of 0 to 1. This value acts as a selective filter, deciding how much information from the cell state permeates the hidden state.

$$h_t = o_t * tanh(c_t) \quad (7)$$

The cell state is transformed using the tanh activation function, resulting in a vector with a range of -1 to 1. This transformed state represents raw knowledge ready for possible transmission. The output gate then acts by multiplying the changed cell state by its gate value, thus filtering out irrelevant or excessive information. The generated vector becomes the current hidden state (h_t), ready to influence further network activities.

2.5.5. Bidirectional Long Short-Term Memory

Building upon the foundation of LSTMs, bidirectional LSTMs leverage two LSTMs for processing input data [31]. The first LSTM analyzes the original sequence in a forward direction, followed by a second LSTM analyzing the reversed sequence (backward direction) [31]. This dual-pass architecture strengthens the model's ability to capture long-term dependencies within the data, leading to improved accuracy compared to standard LSTM models.

Bidirectional Long Short-Term Memory (Bi-LSTM) networks offer several advantages over traditional LSTMs for sequence prediction tasks. [32] Unlike its predecessor, which only processes data in one direction, Bi-LSTM performs two iterations: one from beginning to end and another from end to beginning. [32] This bidirectional traversal allows the network to consider both historical context (past data) and future expectations (upcoming data) when making predictions, leading to enhanced accuracy and efficiency.

A key feature of Bi-LSTM is its feedback input mechanism. Each layer receives information not only from the previous layer but also from the corresponding layer in the reverse iteration. This allows for a more comprehensive and faster learning process as the network can leverage both past and future dependencies within the input sequence. In essence, Bi-LSTM's ability to consider both history and the near future simultaneously provides a richer context for prediction: By incorporating knowledge of the entire sequence, Bi-LSTM models can generate more precise and contextually relevant predictions. Moreover, the feedback loop mechanism facilitates efficient

information flow and enables the network to learn long-term dependencies within the data more quickly [32]. Therefore, Bi-LSTM's bidirectional processing and enhanced learning capabilities make it a powerful tool for various sequence prediction tasks, outperforming traditional LSTMs in accuracy and efficiency [32].

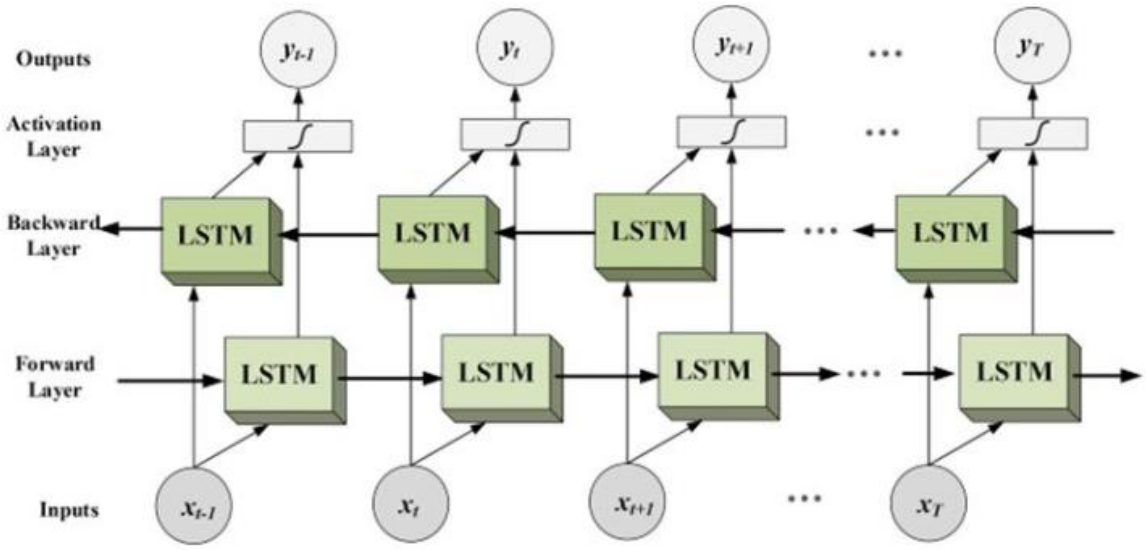


Figure 2. 6 Bidirectional LSTM Arcitecher

2.6.Hyperparameter

Deep learning algorithms are changing the way we evaluate and interact with language-based data by teaching machines to understand text and speech and execute automated tasks such as translation, summarization, classification, and extraction. It is, however, not an easy task. It requires careful parameter selection and optimization. In the NLP and machine learning areas, hyper-parameter optimization has gotten a lot of attention [33]. How well a learning algorithm performs, from simple ones like logistic regression to complex ones like neural networks, depends heavily on choosing the right settings for its "tuning knobs" called hyperparameters [33]. These settings can make a huge difference in how good the algorithm works, but finding the perfect ones can be tricky. It takes time because there are often many different knobs to adjust and not enough data to try out all the combinations. Some examples of these hyperparameters include the type of

word embedding used, how many hidden layers the model has, how many neurons are in each layer, how fast the model learns, how much information it randomly drops out during training, what kind of activation function it uses, and how it optimizes its performance.

2.6.1. Word Embedding

The quest for effective representations of text words has long been a cornerstone of natural language processing and other machine-learning fields. Initial attempts often employed discrete one-hot vectors, which, however, were plagued by the "curse of dimensionality" and failed to capture the inherent semantic relationships between words. Recent breakthroughs in machine learning have paved the way for low-dimensional and continuous vector representations known as word embeddings. These embeddings enable significant advancements in downstream tasks such as machine translation, natural language inference, and semantic analysis, opening doors to improved performance and broader application possibilities [34].

Word embedding is a powerful tool in the world of natural language processing (NLP). It helps "translate" words into numbers, but not just random numbers! These numbers capture the meaning and grammatical connections between words, like a secret code [35, 36]. This allows computers to understand how words relate to each other, even if they haven't seen those exact words before. Imagine two words like "ብርቅቤ" and "ብቸኛ." Both words meaning in English are “unique”. They have similar meanings, right? Word embedding captures this connection by giving them similar numbers. This way, the computer can learn about new words based on familiar ones, even if it's never encountered the new word directly. There are two main types of word embedding: Static: These are like fixed dictionaries; each word has a single, unchanging number code. Contextual: These are more flexible, taking into account the surrounding words to adjust the meaning of a particular word. For example, "ገና" in "Ethiopian Christmas" will have a different meaning than "ገና" in "something that doesn’t occur”

2.6.1.1. Static word embedding

Static word embedding models provide stand-alone re-presentations that are independent of the words or sentences around them (context [36]. It is produced by analyzing huge text corpora and recording co-occurrence patterns between words, and it is context-independent, which means

that the same word will have the same vector representation regardless of its context. Some of the static word embedding models are Word2vec, FastText, and Glov

I. Word2vec

Word2Vec is a method for representing individual words using a fixed-length vector, allowing for the encoding of semantic relationships between them. The meaning of words within a specific context is captured by this vector-based representation. Word2Vec employs two methods: Continuous Bag of Words (CBOW) and Skip-gram. The context word in Skip-gram is anticipated based on the core word provided, but the context word in Continuous Bag of Words (CBOW) is inferred from the available context information [37].

II. FastText

FastText is a word embedding technique that overcomes the issue of unusual and out-of-vocabulary terms that Word2Vec has difficulty with. FastText, unlike Word2Vec, uses sub-word information to construct word embeddings. It works at the character level, capturing the internal structure and morphological properties of words. FastText represents a word as the sum of its sub-word vector representations. FastText can effectively handle unusual and out-of-vocabulary terms thanks to this method, making it a powerful tool for language modeling and NLP applications [38].

III. Glove

Glove, a popular method for word embedding, can capture the meaning of words based on how often they appear together in large amounts of text (corpora). Imagine it like building a map of words, where those that frequently show up close to each other are considered similar in meaning. This helps the computer understand the relationships between words, even if it hasn't seen them before in the same context.

The key of Glove is that it analyzes word co-occurrence across various corpora [37], not just a single one. This broader perspective allows it to gather more comprehensive information about how words are used in different contexts, leading to more accurate and versatile word representations.

2.6.1.2. Contextual word embedding

Contextual word embeddings are a type of word embeddings that capture the meaning of words in their context. In contrast to traditional word embeddings, which assign a fixed vector representation to each word regardless of context, contextual embeddings provide different

representations for the same word based on its usage inside a specific sentence or document. This contextual information improves the ability of the embedding to capture nuances in word meaning and distinguish between distinct meanings of a word. Two prominent models for generating contextual word embedding

I. **Embeddings from Language Models (ELMO**) is a special way to turn words into numbers that computers can understand, called word embedding. Unlike older methods, ELMO considers the specific context of a word in a sentence. ELMO captures these differences by using two "layers" of information: which are the Character Layer and Bidirectional LSTM Layers. Character Layer breaks down the word into its letters and analyzes how they combine to create meaning. Where Bidirectional LSTM Layers consider both the word's beginning and ending, understanding how it relates to surrounding words in the sentence. By combining these layers, ELMO creates a unique word embedding that reflects its specific meaning in a given context. [37] [39]

II. **Bidirectional Encoder Representations from Transformers (BERT)** is another powerful tool for understanding language. It also uses word embedding but with a different approach called a "transformer." Imagine a team of translators, each focusing on a different part of a sentence and then sharing their insights. BERT works similarly, analyzing words about one another throughout a sentence. BERT has two ways of analyzing words one is the Masked Language Model and the other is Next Sentence Prediction. In masked language mode hides certain words in a sentence and tries to predict them based on the remaining context. This helps it understand how words connect and what meaning they bring. Next Sentence Prediction learns to predict whether two sentences belong together, further strengthening its understanding of relationships between words and ideas [37]. These combined approaches make BERT a powerful tool for various language tasks, from search engines to question answering.

2.6.1.3. Character Representation for Linguistic Labeling

Understanding the individual letters (characters) in a word, like prefixes and suffixes, can be extremely helpful for certain language tasks, particularly those involving identifying the grammatical role of a word (part-of-speech tagging). Traditionally, this relied on manually defining specific features to analyze [35]. However, research suggests that utilizing both

Convolutional Neural Networks (CNNs) and Bidirectional Long Short-Term Memory (Bi-LSTM) networks is a more effective and adaptable approach for representing characters and extracting their hidden meaning. This allows the model to automatically learn which features are important for linguistic tasks, potentially leading to greater accuracy and efficiency.

2.6.2. Dropout

The dropout technique is a regularization method often used in deep learning to reduce overfitting. During training, a proportion of neurons or units in a neural network are randomly deactivated. This means that the dropped-out units, as well as their connections, are temporarily removed from the network for a certain training iteration. Dropout has shown to be an excellent regularization technique and has been widely used in a variety of deep-learning applications. It aids in the prevention of overfitting, increases model generalization, and contributes to the overall performance and robustness of deep neural networks [26].

2.6.3. Opoptimizer

In deep learning, training a neural network involves adjusting its internal settings called parameters. Think of them like knobs you turn to get the network to perform well. Optimizers are algorithms that help turn these knobs in the right direction, minimizing the loss function. This function measures the gap between the network's predictions and the actual data it's learning from, like the difference between a guess and the correct answer. There are many optimizers, each with its strengths and weaknesses. Some popular ones include Adagrad, Adadelta, RMSProp, Adam, Nadam, and SGD [35].

Adam is an adaptive optimizer that automatically adjusts the learning rate for each parameter individually. It's easy to use, efficient and works well with different data scales. It also doesn't need much memory to work, [40] like a lightweight backpack for learning. SGD (Stochastic Gradient Descent): This classic optimizer uses the "slope" of the loss function to guide parameter updates. It's simple and efficient but can be slow and sensitive to how much you adjust the learning rate. Choosing the right optimizer depends on the specific task and your network's needs. Some optimizers are faster but less stable, while others are slower but more reliable.

2.6.4. Hidden layers and Number of nodes

Adding more hidden layers and nodes in a neural network sounds like a good way to boost accuracy, but it's not always a guaranteed win. Think of it like adding layers to a sieve. While more

layers might catch smaller details, too many can also capture unwanted noise, making the model "memorize" the training data without truly understanding it. This can lead to poor performance on unseen data. The ideal number of hidden layers and nodes depends on your specific data, task, and resources It's similar to finding the correct proportion of components in a recipe: explore and try various configurations to find the sweet spot for your model's performance [34].

2.6.5. Learning rate

In deep learning, it effectively defines how much a model adapts and alters itself during the training phase based on new inputs. A high learning rate might cause the model to learn quickly and make large alterations based on each new piece of input. Low learning rates result in slower learning and fewer modifications. It is safer and prevents instability, but it can take significantly longer to achieve the appropriate performance, especially in difficult situations. Adjusting the learning rate affects the weights of neural networks, regulating how much they update their connections depending on new data. Finding the correct rate is critical for balancing quick learning with stability and preventing the model from being stuck in local minima. Thus, determining an optimum learning rate is critical for efficient and effective learning [34].

2.6.6. Activation function

Within neural networks, activation functions serve as critical gatekeepers, dictating how the weighted sum of inputs translates into an output signal [40]. The typical network architecture comprises input, hidden, and output layers. The input layer ingests raw data, while hidden layers progressively process information and propagate it forward. Finally, the output layer delivers the network's predictions. Depending on the specific task, each layer may leverage a distinct activation function to optimize the complexity of connections and the precision of forecasts across diverse problem domains. Rectified Linear Units (ReLU), Logistic (Sigmoid), and Hyperbolic Tangent (Tanh) are prevalent choices for hidden layers, while Linear, Logistic (Sigmoid), and Softmax find widespread use in the output layer [35]. This rich tapestry of activation functions empowers neural networks to model

2.7.Hyperparameter optimization

Finding the best hyperparameters in training neural networks requires hyperparameter optimization. Manual search, grid search, random search, and Bayesian optimization are among the approaches that can be employed for this purpose. Manual search relies on intuition and

experience to locate optimal hyperparameter values, but it is dependent on professional competence. Grid search searches the hyperparameter space systematically, but its efficiency declines as the number of hyperparameters increases. Random search samples from the hyperparameter space effectively, making it more suitable for high-dimensional areas. Bayesian Optimization makes use of previous evaluations to direct the search for optimal hyperparameters. [41] KerasTuner is a scalable and user-friendly hyperparameter optimization framework that includes search techniques such as Bayesian Optimization, Hyperband Search, and Random Search. It solves the difficulties of hyperparameter search and assists in the discovery of optimal hyperparameter combinations [22].

2.8.Related Works

This chapter discusses word prediction research on Ethiopian languages such as Amharic, Tigrigna, and Afaan-Oromo, as well as foreign languages such as Italian, Swedish, Hindu, and Assamese. Many scholars have examined word prediction, and numerous studies, notably for non-inflected and less-inflected languages, have been proposed and implemented throughout the years. Some work on highly inflected languages has been completed. The researcher's work can be presented here by categorizing it as Word Prediction for Ethiopian Languages and Word Prediction for Foreign Languages.

2.8.1. Word Prediction for Ethiopian Languages

Nesredin Suleiman et al. [23] developed word prediction for Amharic online handwriting recognition. The aim was to develop a statistical word prediction model based on word frequency. The researchers created a corpus of 131,399 Amharic words and 17,137 names of people and places to develop their word prediction model. They created a corpus of Amharic words and names and extracted statistical information to calculate the value of n in the n-gram model and the average word length. Using a bi-gram model with n=2, they predicted words based on the first two characters. The prototype had a prediction accuracy of 81.39%. However, the model ignored context information and relied on a dictionary method for word prediction.

Tigist Tensou [2] developed Amharic word prediction using statistical methods and linguistic rules. It required building language models and generating predicted words. Morphological analysis, word sequence prediction, and morphological generation were critical components. The models were then constructed to capture the root/stem and morphological features of words, such

as aspect, voice, tense, and affixes. Furthermore, the models extracted preferred morpho-syntactic elements such as gender, number, and person from the user's content to ensure proper grammatical agreements between words. The models were trained on a wide range of news texts to capture morphological aspects as well as desired morpho-syntactic features. Keystroke savings of 20.5% (hybrid model), 17.4% (tri-gram model), and 13.1% (bi-gram model) were discovered throughout the study. While the hybrid model performed better, The study utilizes a series of words to take into account contextual information, yet it employs a statistical language model known as N-gram, which struggles to capture long-term dependencies effectively.

Alemebante Mulu et al. [3] used statistical approaches to develop an Amharic text prediction system for mobile phones. The Amharic corpus has 1,193,719 words, 242,383 Amharic lexicons, and a list of person and place names totaling 20,170. The word prediction system input on the first two letters is written and listed down, based on the frequency of the word and alphabetically if the frequency is the same. They have a 91.79% forecast accuracy. Due to the utilization of statistical methods, the research encounters challenges in capturing long-term dependencies.

D Endalie et al. [4] used the Bi-directional Long Short Term-Gated Recurrent Unit (BLSTGRU) network model to predict words in Amharic. Their network model was tested with 63000 Amharic bible sentences and produced an accuracy of 78.6%. The proposed model was compared against state-of-the-art models such as LSTM, GRU, and BLSTM by the researchers. This study employed data from a single source, which does not allow for the consideration of multiple forms of text. The epoch size is also 1000, which is too large for training huge amounts of training data and can result in overfitting. Furthermore, no out-of-vocabulary terms were handled, no text was standardized, and the effect of pre-trained word embedding was not explored.

Y.T.Tesema [21] explored the application of Bi-LSTM, a deep learning approach, to predict the next word in Amharic text. The corpus comprised 37,565 word sequences, encompassing 13,314 unique words. Employing a design science research methodology, the study aimed to identify optimal hyperparameters and embedding techniques for the model. Experiments revealed that the combination of 102 LSTM units, tanh activation function, 0.5 dropout rate, a learning rate of 0.00066283, batch size of 32, 10 training epochs, and the Adam optimizer yielded the best performance. Notably, FastText word embedding outperformed both word2vec and Keras embedding, achieving 91.02% accuracy on the training set and 90% on the testing set. These

findings suggest the potential of Bi-LSTM with FastText embedding for Amharic next-word prediction. However, Increasing the data volume and training epoch size will further increase the performance of the model.

W.T.Mamow [42] developed a Next Word Prediction System for the Geez language by combining a statistical method of N-gram with back off and a morphological analyzer. Normalization of letters with the same sound is not possible in geez because the letters have distinct meanings than in Amharic. This method saved 35.7% of keystrokes.

S.K. Berhe [12] developed word sequence prediction for the Tigrigna language using n-gram statistical models based on two Markov language models. The model was evaluated using an accuracy evaluation metric, and it achieved an average performance of 85% for correctly predicted words using a sequence of two tags and 81.5% using a sequence of three tags. The results showed that word prediction using a sequence of two tags outperformed the sequence of three tags

A. B. Delbeto [43] developed Word sequence prediction for the Afaan Oromo language using the statistics method. The study used the n-gram approach of bi and tri-word statistics and syntactic. The model is built for a root or stems with tags like nouns, verb, adjectives, pronoun, adverbs, and conjunctions The corpus has 23,400 sentences and a 49,143 unique words. Keystroke Saving (KSS) is used to measure system performance once the model has been trained. According to the results of the evaluation, the primary word-based statistical system achieved 20.5% KSS, while the second system, which employed syntactic categories with word statistics, achieved 22.5% KSS. As a result, statistical and linguistic rules have great promise for Afaan Oromo word sequence prediction.

2.8.2. Word Prediction for Foreign Languages

Aliprandi et al. [44] concentrated on creating FastType, a word prediction system for the Italian language. To improve keystroke efficiency, the system employs statistical and lexical approaches, as well as robust language resources. It includes a user interface, a prediction engine, and linguistic resources. The key component is the predictive engine, which manages communication with the user interface and keeps track of prediction status and entered words. The system makes use of morpho-syntactic agreement, lexicon coverage, and access to language models and a large lexicon. POS n-grams and Tagged word (TW) n-grams are used to enrich the prediction engine's

morphological information. For the Italian language, the method combines POS trigrams and simple word bigrams models. To construct POS and tagged word n-grams, training data from diverse sources, including newspapers, magazines, documents, letters, and emails, is employed. A different test set is used to evaluate the system's performance based on keystroke saving (KS), keystroke until completion (KUC), and word type saving (WTS) factors. The results show a significant gain in typing efficiency, with a 51% savings compared to non-inflected languages. Furthermore, a 29% word type saving and 2.5 keystrokes until completion are noted.

S. Radhika et al [32] propose a new way of predicting the next word in a Hindi sentence, to reduce the user's keystrokes. For this job, the paper investigates two deep learning strategies, Long Short Term Memory (LSTM) and Bi-LSTM. The LSTM accuracy was 59.46% and the Bi-LSTM accuracy was 81.07%.

P.B. Partha et al. [45] developed in the Assamese language an LSTM model that was provided to the user to predict the next word(s) given a set of current words. They deal with the problem of words with various synonyms by storing the Tran-scripted Assamese language, according to the International Phonetic Association (IPA). Their model walks through their dataset of transcribed Assamese words and predicts the next word using LSTM with an accuracy of 88.20%. The results reveal that LSTM is quite effective in next-word prediction, especially for under-resourced languages.

S. Rajakumar et al. [46]developed next-word prediction using machine learning for the English language. They suggest a novel technique for anticipating the following word in an English phrase. They used deep learning algorithms LSTM and Bi-LSTM to analyze the problem of next-word prediction. Based on the evaluation they have got an accuracy of 59.46% using LSTM and 81.07% with the Bi-LSTM. The research is helpful on keystrokes assisting users in saving typing time reducing user spelling errors and helping non-native speakers learn the language by suggesting new and correct next-word predictions.

Afika Rianti et al. [25]designed a next-word prediction using the deep learning method of LSTM. They used a dataset that contains 180 Indonesian destinations from nine provinces. The model could be used to predict the next word by giving the input of the destination. They trained the model with 200 epochs and based on their evaluation they have an accuracy of 75 % and a loss of 55%.

Omor Faruk Rakib et al. [47] propose a novel next-word prediction system for the Bangla language utilizing the Gated Recurrent Unit (GRU) deep learning method. Their system predicts subsequent words and suggests corresponding full sentences. This approach leverages an n-gram dataset to create a language model capable of predicting the next word in a provided sequence. The dataset, meticulously curated from diverse Bangla sources, ensures a comprehensive representation of the language. The authors evaluate their proposed model against Long Short-Term Memory (LSTM) and a baseline model utilizing Naïve Bayes with Latent Semantic Analysis (LSA). Their system demonstrates impressive performance, achieving an average accuracy of 99.70% for a 5-gram model, 99.24% for a 4-gram model, and 94.84% for a Tri-gram model. Notably, the performance diminishes for lower n-gram models, reaching 78.15% and 32.17% for Bi-gram and Uni-gram models, respectively. These results highlight the effectiveness of the proposed GRU-based approach in capturing complex word dependencies and generating accurate next-word predictions in Bangla.

2.9.Summary

Word prediction best applicable in non-inflected languages, but it is challenging in inflected languages like Amharic. Inflected languages constantly twist and turn words, demanding the model to store a vast library of all possible variations. Analyzing word structure and grammar perfectly becomes crucial, like deciphering a complex puzzle, and even then, offering too many suggestions can drown users in choices, while too few leave them lost and frustrated. It's a delicate dance between memory, accuracy, and user experience, making word prediction in these languages a challenging task [2, 48].

The related works in Amharic word sequence prediction using the BLSTM network model focused on developing models to assist in typing and text prediction for the Amharic language. This is important because Amharic is a Semitic language spoken in Ethiopia and has a complex writing system and grammar, making it challenging for non-native speakers to type accurately. The studies used various deep-learning techniques to develop models for tasks such as text prediction, classification, sentiment analysis, speech recognition, and clustering. They also used different datasets and evaluated their models using various metrics such as accuracy, F1-score, and more. The results showed that the BLSTM models combined with word embedding produced promising results with high accuracy in Amharic word sequence prediction compared to other state-of-the-art models. These studies provide insights into the

development of Amharic language models and show the potential for using deep learning techniques to improve typing accuracy and efficiency in Amharic.

CHAPTER THREE

3. AMHARIC LANGUAGE

3.1. Amharic Language and its Writing System

Amharic is an official language of the country's, it serves as the voice of government and daily life in the north-central region. Its influence extends beyond borders, growing in Ethiopian cities and among Ethiopian communities distributed across the Middle East and North America [49].

Amharic, a Semitic language, has profound linguistic relations with Hebrew, Arabic, and Syrian. Its written form, created with the ancient Ge'ez script (also employed in Ethiopian Orthodox Church ceremonies), exposes 33 distinct characters and an extra seven consonant-vowel combinations [2, 49].

The Amharic writing system employs a unique repertoire of 238 characters. This intricate system is built upon 33 fundamental characters, known as "fidäl," each capable of assuming seven distinct forms. The basic form represents the consonant itself, while the remaining six non-basic forms depict various consonant-vowel combinations, forming syllables. Notably, these non-basic forms are generated from their corresponding basic forms through a set of generally predictable modifications. For instance, second-order characters are often formed by adding strokes to the right side of the core character [50].

3.2. Amharic Part of Speech

POS tagging is an essential task in natural language processing that aids in the analysis of sentence syntactic structure and facilitates subsequent language processing tasks. Words in Amharic, like words in any other language, can be categorized into various parts of speech based on their grammatical functions and syntactic roles within a sentence. Here are some common Amharic parts of speech: noun, pronoun, adjective, verb, adverb, conjunction, and preposition.

3.2.1. Nouns

Nouns are words used to describe persons, places, things, or abstract concepts. In Amharic, nouns can have up to two prefixes and four suffixes for each stem. Nouns can be identified by their suffix since they can add suffixed by bound morphemes such as ኤ/E, ኡ/u, ኦች/och, ዎች/woc, and thus allows nouns to be easily identified [2]. Table 3.1 shows examples of gender, number, and case marker suffixes for Amharic nouns.

Table 3. 1 Amharic Noun suffixes in gender, number, and case Markers

	Gender marker		Number marker		Case marker	
Word	Masculine	Feminine	Singular	plural	nominative	Accusative
“ጎልማሳ”/ “golemasa”	“ጎልማሳ”/ “golemasa”	“ጎልማሳ-ኢት”/ “golemasa –it”	” ጎልማሳ”/ “golemasa”	“ጎልማሳ-ኦች”/ “golemasa -och”	“ጎልማሳ”/ “golemasa”	” ጎልማሳ-ን”/ “golemasa -n”
“ዶሮ”/ “doro”	“ዶሮ”/ “doro”	ዶሮ-ኢት”/ “doro -it”	“ዶሮ”/ “doro”	“ዶሮ-ኦች”/ “doro -och”	“ዶሮ”/ “doro”	” ዶሮ-ን”/ “doro -n”

3.2.2. Pronoun

Pronouns are a small group of words with a variety of linguistic purposes. Pronouns are classified into various categories, including demonstrative, possessive, interrogative, and personal pronouns. Serving as substitutes for the speaker, the listener, and any other individuals or objects mentioned in the conversation, these pronouns play a crucial role in effective communication. They are further divided into groups according to person, gender, and number; distinct affixes serve as indicators of these groups [2]. Table 3.2 is a list of sample pronouns arranged methodically based on the person, number, and gender features to give a thorough understanding of pronouns.

Table 3. 2 Amharic pronouns

	Personal Pronouns			
Person	Gender		Singular	Plural
1st		I, We	እኔ/Ene	እኛ/Ega
2nd	Masculine	You	አንተ/Ante	እናንተ/Enante
	Feminine	You	አንቺ/Anchi	
3rd	Masculine	He, They	እሱ/Esu	እነሱ‖/‖‘Nesu‖
	Feminine	She	እሷ/Eswa	
	Polite	You	እርስዎ /Erswo , አንቱ/Ante	

3.2.2.1. Demonstrative pronouns

Pronouns that convey an object's location or proximity to the speaker or spectator fall into the category of demonstrative pronouns. These pronouns can be used to describe objects that are either nearby or far away from the person identifying them or the viewer. As a result, these pronouns are categorized according to the gender of the object being referred to as well as its distance from the specified object [2]. For example, in Amharic, the demonstrative pronoun "ይህ" (yəh) designates a masculine-gender item that is near the speaker or observer. Conversely, the pronoun "ይቺ" (yci)is used to designate a nearby object that is feminine.

3.2.2.2. Reflexive pronouns

Reflexive pronouns are employed alongside personal pronouns. Reflexive pronouns include words like ⁅እኔ ራሴ⁆/⁅‘EnE rasE,/myself, —አንተ ራስህ⁆/⁅‘ante rash⁆/yourself, እሱ ራሱ/⁆/⁅Esu rasu‘/himself, እኛ ራሳችን/⁆/⁅Ena rasachne/ourselves, and እናንተ ራሳችሁ‘/’’enante rasachu /themselves. They refer to a person or object. When something is odd or unusual, we can use a reflexive pronoun to emphasize it. To emphasize a reflexive pronoun, we can use it in conjunction with the nouns. We can use it for singular as well as plural forms

3.2.2.3. Interrogative pronouns

An interrogative pronoun is used to ask questions. There are just five interrogative pronouns in the English language. Each one is used to pose either a direct or indirect query. Some words, like as "who" and "whom," solely relate to persons. Others can be used to describe things or individuals. In similar Amharic language have an interrogative pounce. For example :- Who/ማን⁆/ ⁅man⁆/, What/ምን⁆/ ⁅min⁆/, where/የት⁆/⁅yet⁆/, when/መቼ⁆/⁅mecE⁆/, and How /እንዴት⁆/⁅‘ndEt⁆

3.2.2.4. Possessive pronouns

Possessive pronouns are used to indicate possession of something and are generated by adding the prefix /ye to personal pronouns. [2]. Possessive pronouns can be singular or plural. For example, My//የእኔ⁆/⁆/ye'nE, Your/የአንተ⁆/⁅ye‘ante⁆ and His/የአንተ⁆/⁅ye‘ante⁆ are for singular possessive pronouns, and Our//ye'Na, and Their//ye'nante are for plural possessive pronouns.

3.2.3. Adjectives

Adjectives are linguistic elements that describe or modify nouns or pronouns, and they usually come before the word they modify. They provide additional information about the noun or pronoun with which they are linked. Various characteristics distinguish items from one another, such as shape, behavior, color, and so on, and these distinctions are communicated through the use of adjectives. Adjectives, like nouns, can change in gender, number, and case [2].Examples: ,

"ቀይ በግ"/ "keye beg" /Red Sheep ሰነፍ ተማሪ፡/ senfe temari / lazy student In the first sentence, the term "ቀይ"/ keye/Red is an adjective that modifies the noun "በግ"/beg/Sheep, providing further details regarding the hue of the Sheep. In the next sentence, the term ሰነፍ / senfe / lazy is an adjective that qualifies the noun "ተማሪ"፡/temari/student, providing further details regarding the student, who is lazy.

3.2.4. Verb

A verb is a word used to represent the occurrence of an action or to denote the presence of a specific state or situation. Verbs in Amharic are highly intricate, comprising of a stem along with a maximum of four prefixes and four suffixes. The verb forms are modified to convey various aspects such as individual, gender, quantity, and time, with the standard verb form being the third person masculine singular. Distinct suffixes that change depending on the person and number are what characterize passive voice verbs [2].

3.2.5. Adverb

Adverbs are language constructions that function to modify verbs, just like adjectives qualify nouns. They are essential in giving further details on the action being carried out. Adverbs can be categorized according to several factors, including place, time, situations, and more. [2] The phrase "በቀሰታ" (bekesta) modifies the main verb "እየተራመደች" (eyeteramdech), meaning "walking," in the Amharic line "ልጅትዎ በቀሰታ እየተራመደች ነበረ" (ljetwa bekesta eyeteramdech neber), which translates to "The girl was walking slowly." The use of the word "bekesta" in this context adds details about the girl's gait, emphasizing that it was slow.

3.2.6. Preposition

Prepositions are a limited group of words that are frequently used in front of nouns to indicate how they relate to the other words in the phrase. These terms are essential for denoting chronological,

geographical, or logical relationships. Some of the example prepositions in the Amharic language are "ለ" (le), "እንደ" ('nde), and "ከ" (ke).

3.2.7. Conjunction

A conjunction is a word that serves as a connector, linking various elements such as words, phrases, clauses, and sentences. Although they are relatively few, Conjunctions possess the flexibility to be utilized with verbs, nouns, and adjectives. Example:‖እና ‖/‖ ‘na‖, —ስለሆነም‖/‖slehonem‖, —ነገር ግን‖/‖negergn‖, etc.

3.3. Amharic Morphology

Morphology encompasses the analysis of words and their formation, as well as the complex relationships that words have with other lexemes within a language. [1] A morpheme is the smallest unit of a word that contains meaning. [2] Morphology studies the various types of morphemes, such as roots, prefixes, and suffixes, and how they interact to form words. words can be altered through processes such as inflection and derivation. [2] Morphemes are classified as either free or bound in the Amharic language. Free morphemes can express meanings that are autonomous and self-contained, but bound morphemes require attachment to other free morphemes to transmit semantic importance.

Examples:

Free	Bound	Free + Bound
በግ‖/‖beg‖	ኤ‖/‖-_E‖	በጌ‖/‖begE‖
አጎት‖/‖agote‖	ህ‖/‖-h‖	አጎትህ‖/‖agoteh‖

The language Amharic demonstrates a high level of morphological complexity, with a root-pattern structure. A cluster of consonants makes up a root in this structure, and vowels are added between the consonants to produce stems. Given Amharic's extensive morphology, it is possible for a single root word to give rise to several word variants, each with unique linguistic characteristics.

Take the word "አጠበች" (atebech), for example. It can be broken down into two morphemes: "አጠበ" (atebe), which denotes the word's root or stem, and "-ች" (-c), which denotes a further significant part of the word.

3.3.1. Inflectional Morphology

Inflectional morphology is the process of assigning grammar features to nouns, verbs, and adjectives. These characteristics include person, gender, number, case, definiteness, and time. Noun inflection utilizes suffixes to indicate gender, number, and case. Person, gender, number, case, definiteness, and time are examples of these characteristics. Suffixes are used in noun inflection to indicate gender, number, and case. The default form of the verb is third person, masculine, and singular, although it has the ability to be modified for person, gender, number, and tense.Typically, the perfect tense is employed to express the past tense. Prefixes denote masculine and feminine subjects, but suffixes denote first, second, and third-person future forms. Adjectives, like nouns, undergo inflection for gender, number, and case [2].

3.3.2. Derivational Morphology

Amharic demonstrates a variety of affixation processes in the area of derivational morphology, including the application of prefixes, infixes, and suffixes to create new nouns from base nouns, adjectives, verbs, stems, and roots. Conversely, verbs, nouns, verbal roots, and stems can all be given a suffix to create an adjective. Specifically, infixation functions as a process that separates adjectives from verbal roots, setting it apart from other word classes. Notably, Amharic verbs have a quite restricted propensity to originate from other parts of speech (POS).

3.4.Amharic Grammar

The structure of sentences, clauses, phrases, and words in natural language is governed by a set of structural rules known as grammar. These rules give instructions on how to organize words in sentences so that they make sense and are coherent. Word order and morphological agreements are two major components of Amharic grammar in this context. Every word in an Amharic phrase must follow the proper grammatical agreements and word order to accurately convey meaning to readers.

Formal Amharic texts adhere to a subject-object-verb (SOV) word order, in contrast to English, which employs a subject-verb-object (SVO) arrangement. Although there are cases of OSV

sequences like /ljun Alemayehu mekerew/The boy is advised by Alemayehu in some Amharic texts, where the object is suffixed by object marker /n, this word order is not commonly seen in formal Amharic texts.

Table 3. 3 A Structure of word order in an Amharic sentence.

	ፖሊሱ ልጅን መታው(SOV) (Polisu legun metawu)	The police officer kicked the boy(SVO)
Subject	ፖሊሱ(polisu)	The police officer
Object	ልጅን (legun)	kicked
Verb	መታው(metawu)	the boy

Research on Natural Language Processing (NLP) needs to look at and evaluate different word sequences that appear in linguistic constructs. The word order patterns about adjectives and nouns, adverbs and verbs, and the main verb's placement about the sentence's finish are especially notable. An essential grammatical rule of the Amharic language states that adjectives must always come before the nouns they describe, regardless of any words that may come in between. In the same vein, adverbs always come before the verbs they modify, following a recognized syntactic pattern

3.4.1. Subject and Verb Agreement

The term "subject" in linguistic analysis refers to a component of a sentence or expression, usually a noun, noun phrase, pronoun, or its synonym, that functions as the object about which the rest of the sentence makes claims and with which the verb concurs. Often, the verb's action is represented by the subject. In Amharic, subjects are usually found at the start of sentences. It is necessary for the subject and the accompanying verb to agree on a number of grammatical features.

Take, for example, the expression "ፖሀሱ ልጅን መታው" (Polisu legun metawu), which means "The policeman kicked the boy." Here, the subject "ፖሊሱ" (Polisu) expresses information about person, gender, and number in the following ways: it is singular, masculine, and third-person, respectively. The verb "መታው" (metawu), which means "kicked," reflects these morphological characteristics. Should any of these grammatical attributes be incorrectly given to the verb, the phrase will not

follow the correct grammatical structure, leaving the readers in the dark. For example, the above sentence would disagree with the subject and be grammatically incorrect as well as possibly difficult to understand if it were written incorrectly as "ፖሀሱ ልጅን መታውቸው" (Polisu legun metatachew), where the verb misrepresents feminine gender agreement.

3.4.2. Object and Verb Agreement

An object, encompassing nouns, pronouns, or noun phrases, represents the entity upon which an action is exerted or impacted by the verb. In the context of using a noun as an object within a sentence, it is possible for the object to be suffixed. It is crucial for the object in a sentence to exhibit concordance with the accompanying verb in terms of gender, number, person, and case[2].

For instance, consider the sentence "ፖሊሱ ልጅን መታው" (Polisu lejun metawu), meaning "The police officer kicked the boy". Here, the subject "ፖሊሱ" (Polisu) as the police officer indicates person, gender, and number information, specifically third person, masculine, and singular, respectively. These morphological properties are reflected in the verb "መታው" (metawu) denoting "kicked". Any misalignment of these grammatical features in relation to the verb leads to the sentence deviating from proper grammatical structure, thus introducing ambiguity for the readers.

For example, if the above sentence is mistakenly written as "ፖሊሱ ልጅን መታቸው" (Polisu lejun metachewu), where the verb incorrectly reflects feminine gender agreement, it results in a disagreement with the subject. Similarly, inconsistency in person and number can pose challenges in Amharic sentences. For instance, a sentence like "ፖሊሱ ልጅን መታአት" (Polisu lejun metaete), where the verb erroneously reflects a plural agreement, exhibits a discrepancy in number as the singular subject of the sentence is inaccurately matched with a plural verb.

3.5. Amharic punctuation

Punctuation is used in various languages, including Amharic, to help in reading and comprehension, to show sentence structure, and to improve clarity. Even if the precise punctuation symbols used in Amharic may not be the same as those in English, they yet have comparable purposes. Here are a few Amharic punctuation symbols that are often used: Table 3.5 shows Amharic punctuations with their corresponding purpose

Table 3. 4 Table Amharic punctuations adopted from *[21]*

No	Symbol	Punctuation mark	Purpose
1	፧	Question mark	Marks the end of a question. Similar to a question mark in English.
2	፨	Exclamation mark	The exclamation mark is used to express strong emotions, surprise, or emphasis. It is placed at the end of a sentence or phrase to convey an exclamatory tone
3	፣	Comma	The comma in Amharic serves to separate words, phrases, or clauses within a sentence. It helps to create pauses and clarify the structure of a sentence.
4	፤	Semi-colon	The semi-colon is used to separate closely related independent clauses or to separate items in a list when commas are already present.
5	...	Three dots	Similar to an ellipsis in English. Indicates an omission, unfinished thought, or trailing off.
6	( )	Parenthesis	To enclose elaboration
7	« »	Quotation mark	Used at the beginning and the end of quoted words, phrases, etc.
8	፠	Four dots	Mark the end of a sentence
9	፥	Colon	The colon is used to introduce a list, explanation, or example. It indicates that what follows is a continuation or elaboration of the preceding statement.

CHAPTER FOUR

4. SYSTEM DESIGN AND MODELING

4.1. Overview

This research focuses on constructing a Word Sequence Prediction Model for the Amharic language, utilizing the robust capabilities deep learning method of a Bi-directional LSTM (BILSTM) architecture. This section dissects the proposed model's architecture, design, data preparation strategies, and the evaluation methods employed to gauge its effectiveness. The model comprises dedicated components tailored for the prediction task, each trained and tested on distinct datasets. Its operational flow encompasses three crucial stages: training, validation, and testing. To assess the model's performance rigorously, we employ various evaluation metrics, including accuracy and perplexity, providing a comprehensive understanding of its strengths and weaknesses.

4.2. System Architecture

The system design for a word sequence prediction model for Amharic languages based on the deep learning method of BILSTM consists of five major components: data set collection and preprocessing, pre-trained model selection, fine-tuning the model, model evaluation, and the proposed BILSTM model architecture.

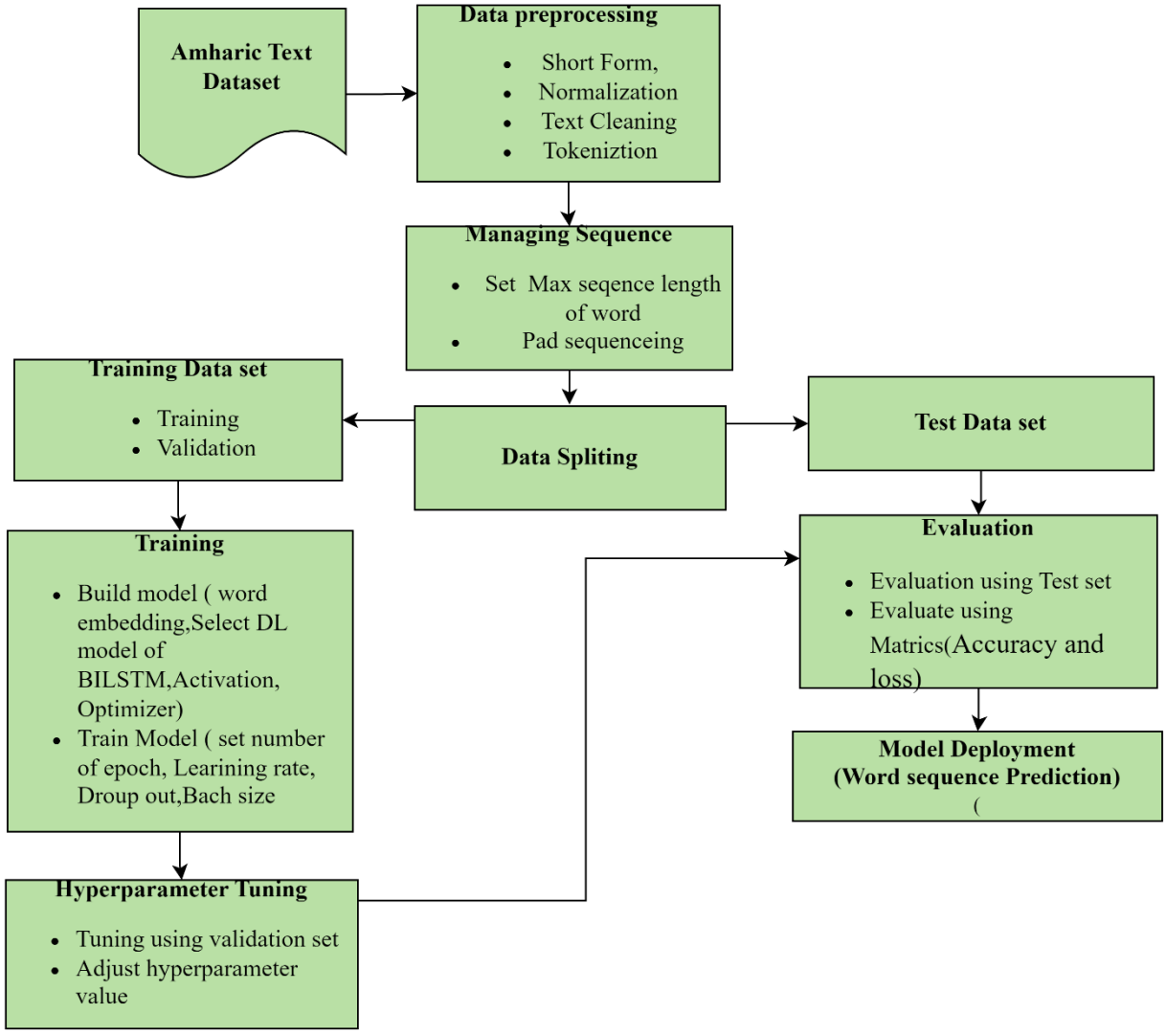


Figure 4. 1 Proposed Architecture

4.3.Data collection

The corpus of Amharic texts is compiled from various mass media sources. Fana, Walta, Ethiopia Press, and ALALAN Amharic news are data sources. These sources were utilized to improve the dataset's quality. It is also feasible to obtain a significant amount of mixed data because they can broadcast many social, political, and economic topics while using different linguistic expressions for each of them. As a result, gathering data from those media boosted the data's quality and diversity.

4.4.Data Preprocessing

Text preprocessing is essential for Natural Language Processing (NLP) systems because it lays the foundations for their overall success. Text preprocessing enhances the efficient and accurate functioning of Deep Learning algorithms by translating raw text into a more structured and uniform format. The model's performance is directly influenced by the quality of the preprocessed text. Inconsistencies and irregularities in the data can cause unexpected behavior and inferior performance if the input text is not correctly preprocessed. As a result, thorough text preprocessing is required to fully realize the potential of NLP models.

4.4.1. Data cleaning

The data cleaning procedure carefully refines raw text data by removing unnecessary items that can impede the performance of natural language processing (NLP) models. This includes deleting non-Amharic text, Arabic numbers (0-9), Geez numerals, English and Amharic punctuation, special symbols, non-Amharic characters, and unneeded whitespace. By removing these features, the data cleaning procedure guarantees that the text data remains focused on the core Amharic language, allowing NLP models to operate more effectively and reliably.

4.4.2. Short-form Expansion

Amharic short forms, often known as abbreviations, are extensively used in everyday communication to convey lengthier words or phrases in a compact manner. These abbreviations can be seen in different types of written Amharic, including social media, news articles, and casual conversations. Expanding Amharic short forms entails replacing the reduced form with its full-length equivalent. This procedure is critical for effective comprehension and interpretation of Amharic literature.

4.4.3. Text Normalization

Normalization is the process of transforming a collection of orthographically diverse words into a more uniform and standardized representation. It encompasses a series of interrelated tasks aimed at establishing a consistent foundation for text processing by eliminating punctuation, converting numerals to their corresponding word forms, and ensuring that all words are treated equally. This process plays a crucial role in enabling consistent and accurate text analysis by addressing homophones, instances where characters with distinct symbols share the same pronunciation. In Amharic, there are different characters that have the same sound but are written in different forms

For example, the characters {ሠ፣ ሰ}, {ጸ፣ ፀ}, {ሀ፣ ኃ፣ ሐ}, and {አ፣ ዓ፣ ዐ}, along with their seven subcomponents, are among the characters that can be interchangeably used to form orthographically variant words. Once these homophones are identified, they are normalized to their most prevalent representation throughout the corpus. [51]

4.4.4. Tokenization

Tokenization involves segmenting strings of text into smaller units called tokens. This process can be applied to various linguistic levels, from breaking down large text blocks into sentences to dividing sentences into individual words. In our case, we have employed tokenization to split sentences into their constituent words. We have utilized the whitespace between words as the delimiter for word separation. Additionally, we have treated the Amharic full-stop character (።), which serves as a sentence-ending marker, as a distinct token.

4.4.5. Determining the Sequence Length

The amount of words in each sentence differs after tokenizing sentences into words. However, neural networks require identical sequence length, and this sequence length is set to 7 because large sequences may not be noticed in the dataset and hence cannot be used frequently. As a result, lengthier Sentences with word lengths of more than seven cannot be utilized directly for training 39 without being converted into subsequences. So, by slicing a window size of 7 from left to right, each sentence was turned into a seven-sentence subsequence.

4.4.6. Extracting frequent word sequences

To facilitate word sequence prediction, the model employs a dedicated dictionary. This dictionary associates keys – sequences of seven words or less – with their corresponding frequency within the training corpus. However, due to resource limitations, only sequences appearing more than once within the dataset were incorporated into the dictionary while constructing the Amharic word sequence prediction model.

4.4.7. Sequence padding

Sequence padding is a typical approach for dealing with variable-length sequences of words or tokens. Because neural networks function on fixed-length input data, sequence padding ensures that all input sequences are the same length, allowing for fast processing and training. For this

research, sentences that have word lengths shorter than seven and longer than two are extended with padding prior to reaching the maximum word sequence length of seven.

4.4.8. Input and output

Following the completion of sequence padding, each word sequence is transformed into input and output pairs, where the initial six words form the input, and the seventh word represents the predicted or target term.

4.4.9. Train-test split

The data was divided into three categories: training, testing, and validation. The data was divided as follows: 70% for training, 20% for validation, and 10% for testing. A training set was employed to accomplish this. The validation set is used before training to test the model's performance on previously unknown data and to optimize the model's hyperparameters. After training, testing data is used to evaluate the model's performance on previously unseen data at the time of training.

4.5. Word embedding

To enable machines to comprehend words, it is essential to represent them as vectors, which serve as input for various tasks like text classification and natural language generation. Bidirectional Long Short-Term Memory (Bi-LSTM) networks are commonly employed for language modeling. Within the context of this research, the focus lies on word embedding techniques. Word embedding involves generating continuous vector representations for individual words, facilitating the measurement of similarity between words based on their proximity in vector space. Noteworthy word embedding algorithms encompass Word2Vec, FastText, Keras, GloVe, BERT, and ELMO. In this study, the researcher developed a word sequence prediction model utilizing FastText, Word2Vec, GloVe, and Keras embedding models. The performance of these models was subsequently compared in the context of word prediction, providing insights into their effectiveness.

FastText shines in capturing rare and out-of-vocabulary words, a vital advantage for morphologically rich languages like Amharic. It achieves this by breaking down words into smaller units called character n-grams, then summing the vector representations of these sub-words. This clever approach allows unseen words, even those never encountered during training, to inherit meaning from familiar sub-words they share with known vocabulary. Words with

overlapping character sequences naturally exhibit semantic similarities, reflected by their close proximity within the vector space. [38]

4.6.BILSTM Language Model

Deep learning algorithms are increasingly being used for solving dynamic classification problems, where data features and class labels can change over time. This necessitates algorithms that can handle such evolving dynamics, such as recurrent neural networks (RNNs), and Long Short-Term Memory (LSTM) networks. RNNs are particularly powerful for tasks requiring long-term dependence on past data, such as time series prediction. However, they are susceptible to vanishing gradients, making it difficult to learn long-range dependencies and update earlier weights. LSTMs were specifically designed to overcome this limitation by incorporating a gated memory cell that controls information flow and allows them to learn long-range relationships effectively. [52]

This study investigates the use of Bi-LSTM networks to build an Amharic Word Sequence Prediction Model. Unlike traditional LSTMs, Bi-LSTMs may process input in both forward and backward directions, allowing them to include context from previous and subsequent parts. This improved contextual understanding results in a significant improvement in the precision of predicting the next word.

The core of the Bi-LSTM architecture lies in its gated components: the input gate, forget gate, cell state, and output gate. The input gate selectively adds relevant new information to the cell state, while the forget gate removes irrelevant or outdated information. The cell state acts as a repository of long-term memory within the sequence, holding valuable information that may be useful in the future. Subsequently, the output gate intelligently selects the most pertinent information from the cell state and transfers it to the subsequent layer of the network [32].

This interplay between the gates and the cell state enables Bi-LSTMs to efficiently analyze sequential data, capture long-range dependencies, and ultimately achieve superior performance in next-word prediction tasks for the Amharic language. The architecture representation of the BiLSTM model employed in the present study is illustrated in Figure 4.2.

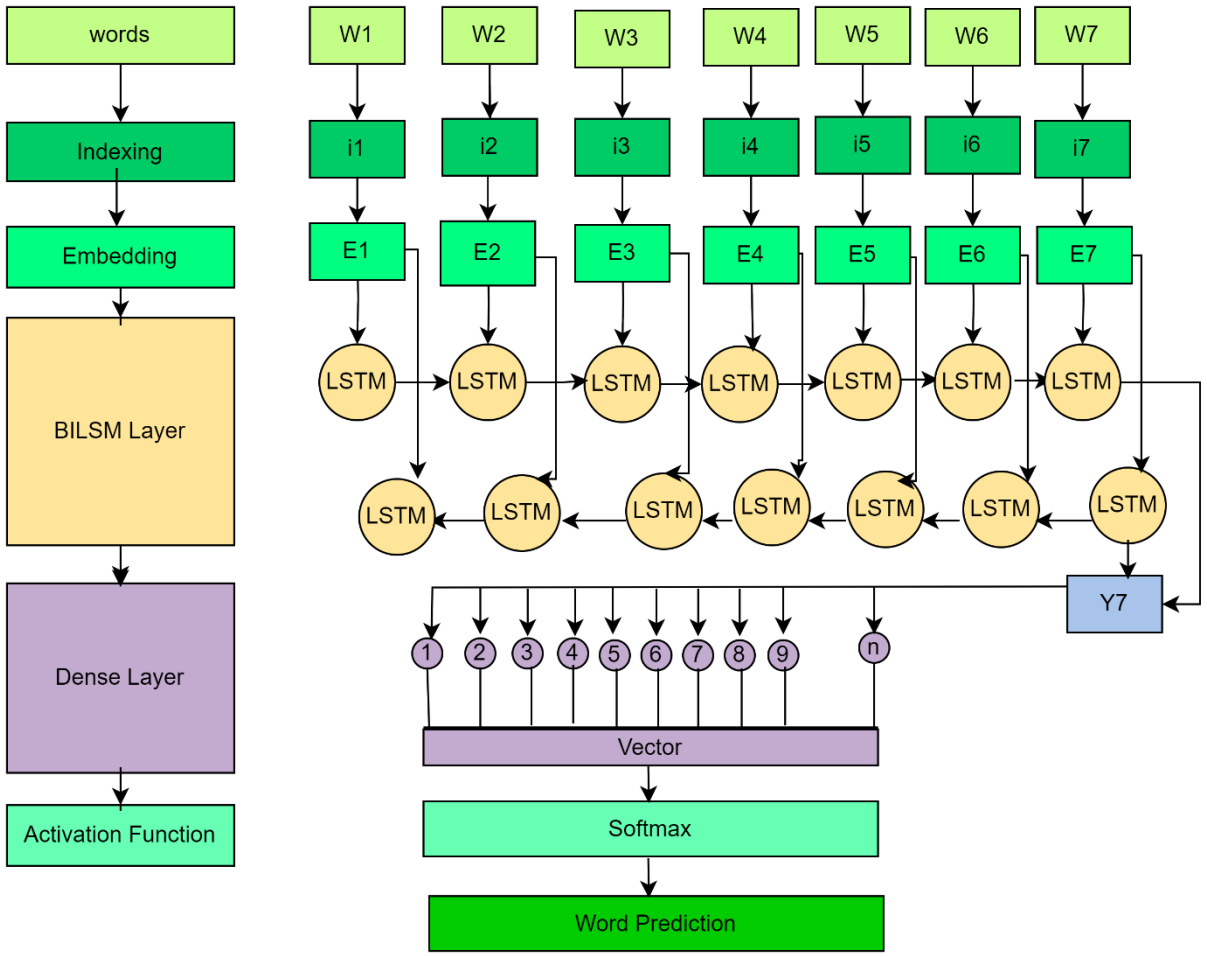


Figure 4. 2. BILSTM Flow diagram

4.7. Hyperparameter

The four important hyperparameters shared by Word2vec, FastText, GloVe, and Kerase embedding are vector dimension, epoch, and learning rate. The vector dimension controls the level of detail captured in word representations, with bigger sizes necessitating more training data and resources. Epochs dictate how many times the model iterates over the data, whereas the learning rate governs how quickly it converges. Notably, FastText provides the sub-word length hyperparameter, which defines the size of character n-grams for subword representation, a feature missing from Word2vec and GloVe's whole-word approaches [53, 54, 55].

4.8.Activation function

Activation functions play a crucial role in neural networks, determining how information flows between layers. Their choice depends on several factors, including the task type, layer type, and

network architecture. For instance, hidden layers in Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) often utilize ReLU functions for their efficiency. Recurrent Neural Networks (RNNs) commonly employ Tanh or Sigmoid activations in their hidden layers for their smooth gradients. The output layer's activation function further depends on the specific problem. Sigmoid is suitable for binary classification tasks. while Softmax excels in multi-class scenarios. Linear activations are preferred for regression tasks. In this study, given the multi-label nature of next-word prediction, the output layer utilizes a Softmax activation function. [56]

A sigmoid function is an activation function that is used in neural networks to compress input values into a range between 0 and 1. It is defined by the formula

$$S(x) = \frac{1}{(1+e)^{-x}}$$

The Softmax function is a powerful tool in deep learning, particularly for tasks like multi-class classification. It takes a vector of real numbers and transforms it into a vector of probabilities, ensuring they sum to 1. The equation for the softmax function is expressed as:-

$$softmax(z) = \frac{\exp(z_{-i})}{sum(\exp(z_{-j})} \text{ for all i in n}$$

where: z is the input vector of logits, z_i is the i-th element in the vector, n is the length of the vector (number of categories), exp is the exponential function and sum is the sum function applied to all elements in the vector.

4.9. Optimizer

Optimizers play a crucial role in deep learning, guiding the model towards the optimal solution during training. They iteratively update the model's parameters (weights and biases) based on the loss function, aiming to minimize the error between the model's predictions and the actual values.

Choosing the right optimizer can significantly impact the training speed, convergence behavior, and overall performance of the proposed model. [57]

4.10. Hyperparameter tuning

Before training the model, the researcher carefully designed its hyperparameters for optimization. This involved specifying a range of possible values for each hyperparameter, including optimizer, dropout rate, learning rate, epoch, batch size, number of LSTM units, and LSTM activation function. To find the best combination of hyperparameters, the researcher employed a random search algorithm from Keras Tuner, supplemented with manual trial-and-error adjustments. This combined approach leveraged the efficiency of automated searching while allowing for targeted adjustments based on specific observations.

4.11. Model Training

Once the model architecture was finalized with optimal hyperparameters, its training regimen was configured. This involved specifying the optimizer algorithm, learning rate, loss function, and desired performance metrics. Subsequently, the model was trained through iterative learning cycles, fitting the training data to the designed architecture within defined epochs and batch sizes.

4.12. Model Evaluation

Following the model's training on the selected training corpus, a thorough evaluation of its generalizability on previously encountered data is required. This involves the use of a distinct testing set that must be carefully maintained throughout the development process. The testing set, which serves as a critical benchmark, reveals the model's performance in managing fresh textual encounters. In this study, accuracy is chosen as the major performance metric. The formula for calculating accuracy is written as:

$$Accuracy = \frac{\text{Total number of maches}}{\text{Total number of words}} * 100\%$$

CHAPTER FIVE

5. EXPERIMENT RESULT AND DISCUSSION

5.1. Overview

In this chapter, we examine the experimental implementation details and provide an overview of the outcomes achieved. The major goal of this research was to develop a model for predicting word sequences for the Amharic language using BILSTM deep learning methods. The experimental findings part includes, such as model training, and model testing.

5.2. Tools and Experimentation Environment

Python was the programming language utilized for both the experiment and the design solution. Naturally, an extremely powerful language for experimenting with NLP and other fields. Because it contains many built-in libraries and makes them easy to utilize for the user, it is simple to write.

We are used to a laptop and a server. The Dell server was utilized for the real model training, testing, and validation, with the laptop computer being used for the remaining tasks. These are the specifications of the computer:

- **Server(**Dell server**)**
  - ✓ Windows Server 2019 Standard Edition OS
  - ✓ 128GB RAM extendable to 256GB
  - ✓ 16TB SSD Hard Disk extendable to 32TB
  - ✓ Processor Intel® Xeon® 4216 Silver Scalable processors up to 28 cores per @ 2.70GHz, 2694 MHz,
- **Laptop(**HP Pavilion Notebook)
  - ✓ Windows 10 64-bit operating system
  - ✓ Processor Intel(R) Core (TM) i7- 6500U CPU @ 2.50GHZ 2.59 GHZ
  - ✓ Ram 8 GB RAM.
  - ✓ 1TB Hard disk

5.3. Dataset

A considerable amount of text data is required to build a corpus for Amharic word sequence prediction. Unfortunately, there is no widely available corpus for Amharic. To address this issue, we gathered information from a variety of sources, including mass media outlets (FANA,

WALTA, Ethiopian Press, and Alalan), Politics, social issues, sports, and entertainment were among the themes covered in the survey. We obtained a corpus of 3,496 sentences and after filtering the collected data, we obtained 18,086 unique words. To aid processing by the Python tool, all files were converted to text format.

Table 5. 1 Corpus summary

Type of Data	**Amount**
Sentence	3496
Unique Words	18,086
7-grams	76,984
Training set(70%)	57,738
Validation Set(20%)	11,548
Testing set(10%)	7,698

5.4.Experimental results

Our research explores the performance of the proposed Bi-LSTM model through a series of meticulously designed experiments. These experiments systematically varied crucial hyperparameters, including LSTM unit count, optimizer selection, activation functions, dropout rates, and learning rates, during model training and testing. Furthermore, we investigated the impact of employing Bi-LSTM architectures and utilizing different pre-trained word embedding methods like Word2Vec, FastText, GloVe, and Keras embeddings. The efficacy of each experiment was rigorously assessed using the aforementioned evaluation metrics. A comprehensive table, Table 5.2 details the specific model parameters employed in each experiment, ensuring transparency and facilitating the potential replication of our work.

Table 5. 2 Hyperparameter Parameters

Hyperparameter	**Value**
Batch size	32
LSTM unit	100
Learning rate	0.0001
Dropout rate	0.5
Epoch	100
Optimizer	Adam, Nadam
Activation function	Relu

5.4.1. Training with Glove Embedding

In our initial implementation, the Glove embedding technique was employed, incorporating a predetermined set of hyperparameters that encompassed 100-dimensional vectors and a training duration of 100 epochs. Additionally, we investigated the model's performance by considering two different optimizers, namely Adam and Nadam, along with the utilization of the Rectified Linear Unit (ReLU) activation function. To assess the model's efficacy, training curves were generated and analyzed.

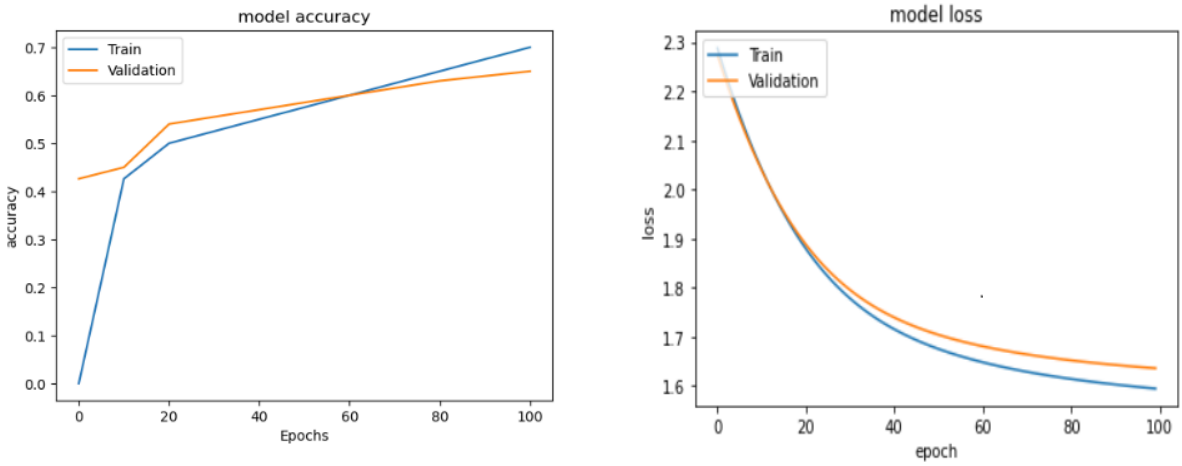


Figure 5. 1 Glove model accuracy and loss

Figure 5.1 illustrates the training and validation performance curves for the model. As seen, the validation accuracy reaches a peak of only 63%, while the training accuracy reaches 63%. This disparity between training and validation results suggests that the model is overfitting the training data. Overfitting occurs when the model memorizes the training patterns too closely, failing to generalize effectively to unseen data. This is further evidenced by the validation loss, which remains relatively high at 1.63, indicating the model struggles to learn generalizable patterns from the training data.

5.4.2. Training with Fasttext Embeddings

Following similar hyperparameters (100-dimensional vectors and 100 epochs), we trained a subsequent model with FastText embedding. Again, we compared the effectiveness of the two

optimizers (Adam, and Nadam) in combination with the activation function Relu. Training curves are provided to assess the model's learning behavior.

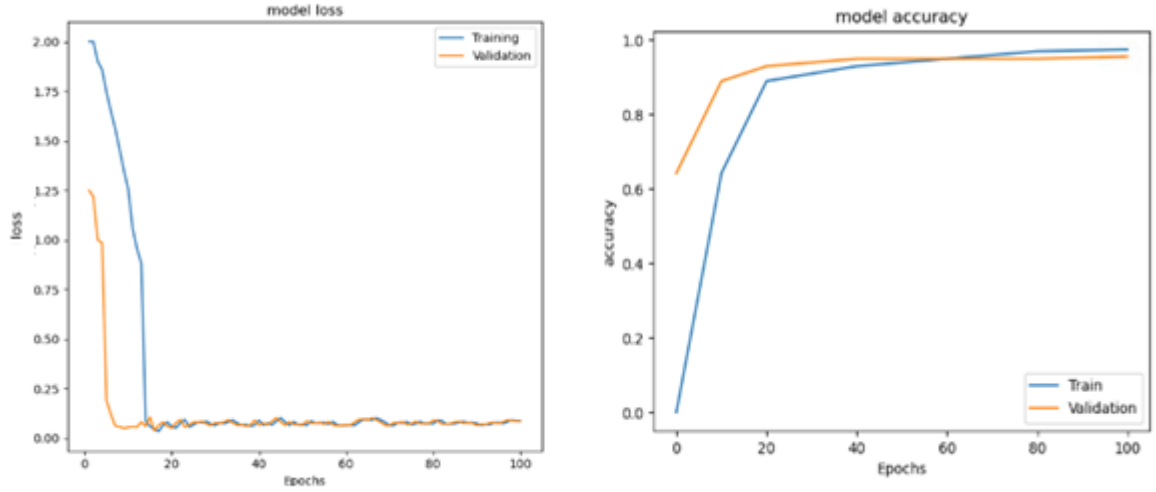


Figure 5. 2 Fasttext model loss and accuracy

Figure 5.2 shows the best results for the model using the fastest word embedding method. Both the training and validation sets reached exceptional accuracy scores: 97.5% for training and 95.6% for validation. This demonstrates that the model effectively learned from the training data and generalized well to unseen data. Additionally, the Adam optimizer demonstrated superior performance compared to Nadam, further contributing to the model's enhanced accuracy score. The validation loss recorded a low value of 0.03, further confirming the model's strong performance. These results signify that the fastest word embedding method outperformed other the three methods in terms of accuracy, achieving the best accuracy on both training and validation sets. The model seems to be working well based on our dataset. However, it's important to test it on a much larger dataset. To be truly confident in its performance.

5.4.3. Training with Word2vec embedding

We then trained a model utilizing Word2vec embedding with the same standard configuration (100-dimensional vectors and 100 epochs). As in previous models, we experimented with the performance of two optimizers (Adam, and Nadam) paired with the activation functions of Relu, The corresponding training curves are presented for further analysis.

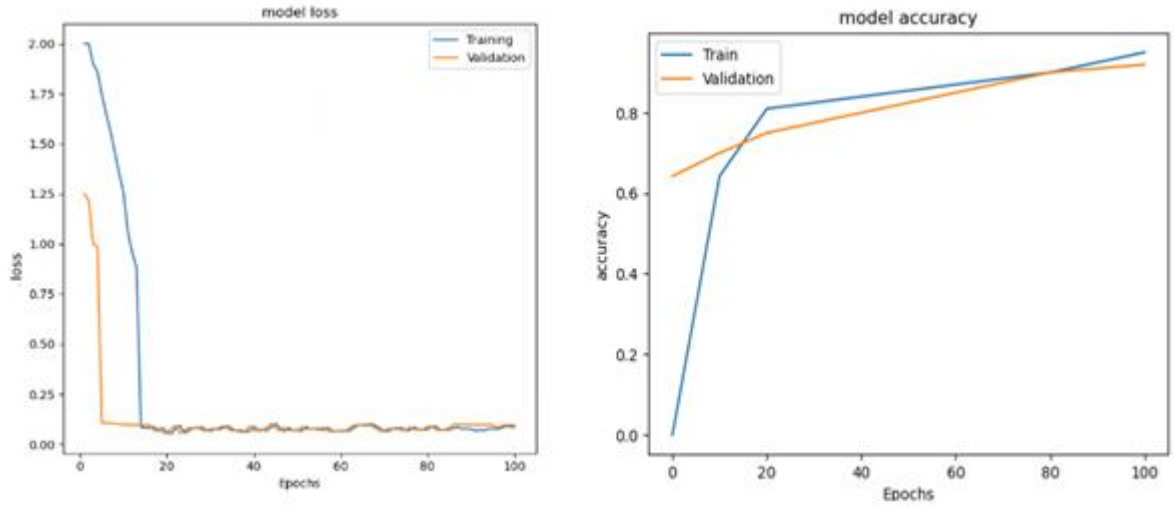


Figure 5. 3 Word2vec model loss and accuracy

Figure 5.3 shows impressive results for the model using the Word2vec word embedding method. Both the training and validation sets reached exceptional accuracy scores: 94.52% for training and 91.68% for validation. This demonstrates that the model effectively learned from the training data and generalized well to unseen data. Additionally, the Adam optimizer demonstrated superior performance compared to Nadam, further contributing to the model's enhanced accuracy score. the validation loss recorded a low value of 0.09, further confirming the model's strong performance. These results signify that the Word2vec word embedding method Also has better score compared with Glove and keras embedding methods in terms of accuracy, achieving good accuracy on both training and validation sets.

5.4.4. Training with Keras Embedding

Finally, we explored the use of Keras embedding in a model trained with similar hyperparameters (100-dimensional vectors and 100 epochs). We evaluated the efficacy of two optimizers (Adam and Nadam) alongside the activation functions of Relu, and Sigmoid). The training curves are depicted for performance evaluation.

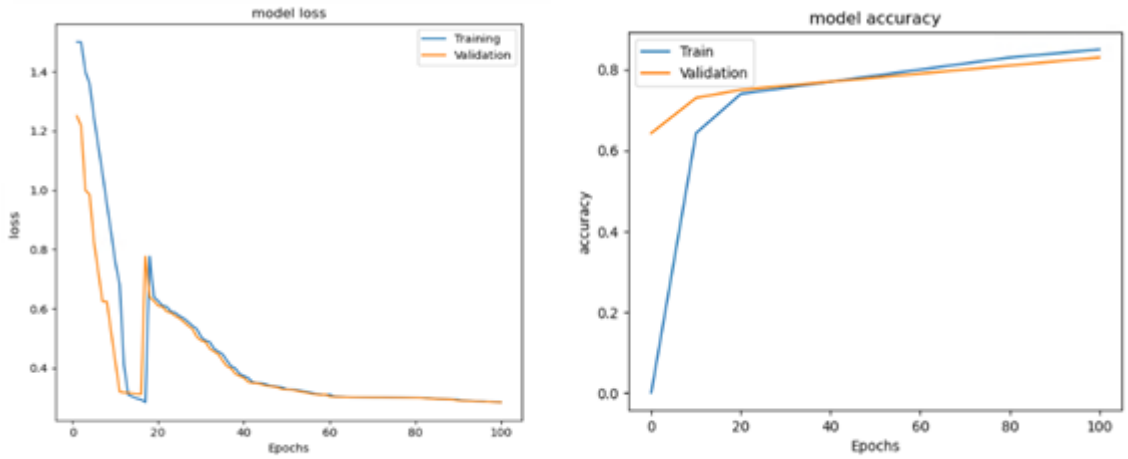


Figure5.4Keras embedding loss and accuracy

Figure 5.4 reveals that the Keras embedding method outperforms Glove in terms of accuracy score. The Keras model achieved an accuracy of 83% on the training set and 81% on the validation set, while Glove yielded 68% and 63%, respectively. This suggests that the Keras model learned more effectively from the training data and generalized better to unseen data. Additionally, the Adam optimizer demonstrated superior performance compared to Nadam, further contributing to the model's enhanced accuracy score. However, it's important to acknowledge that the model's performance still falls short of the best results achieved with other embedding methods, FastText and Word2vec. Finally, the validation loss of 0.31 indicates ineffective learning, although further optimization might be necessary for optimal performance.

Table 5.12. Summary of Word Embedding Models Comparison Results

Embedding Model	Epoch	Optimize	Accuracy		Loss	
			Training	Validation	Training	Validation
Glove	100	Adam	0.68	0.63	1.63	1.63
		Nadam	0.58	0.53	1.61	1.64
Fasttext	100	Adam	0.97	0.95	0.03	0.03
		Nadam	0.91	0.89	0.08	0.09
Word2vec	100	Adam	0.94	0.91	0.08	0.09
		Nadam	0.86	0.87	0.16	0.18
Karase Embedding	100	Adam	0.83	0.81	0.28	0.31
		Nadam	0.79	0.78	0.34	0.33

5.6 Discussion

In the preceding section, our findings, as depicted in Table 5.12, demonstrate that various pre-trained word embedding techniques can have an impact on the performance of deep learning models. Our experimentation revealed that the Fasttext with BiLSTM deep learning model exhibited superior accuracy in word prediction compared to the three other pre-trained models. Moreover, we noted that the utilization of Fasttext pre-trained word embedding techniques influenced the accuracy of the model in predicting word sequences in the Amharic language. Following Fasttext, word2vec also yielded favorable outcomes in comparison to Glove and Keras embedding.

In the context of this study, we employed Fasttext trained with BiLSTM deep learning, employing set hyperparameter values. The trained model achieved an accuracy score of 97.5%. Our experimentation revealed that the Fasttext word embedding method demonstrated enhanced performance in fitting the stance word prediction model using BiLSTM, with a training accuracy of 97.5% and a validation accuracy of 95.6%. Based on our investigations, it can be asserted that for Amharic word sequence prediction, the Fasttext embedding model produces the most favorable outcomes when compared to alternative pre-trained models

CHAPTER SIX

6. CONCLUSION AND RECOMMENDATION

6.1 Conclusion

This research aimed to create a word sequence prediction model for the Amharic language using deep learning techniques. The fundamental goal of word prediction systems is to provide choices for the most likely next word based on contextual information, hence saving time and effort when writing. To achieve this goal, preprocessed data from a corpus including 18,085 unique words and 76,984 word sequences was gathered and split into training, testing, and validation sets.

The suggested model was evaluated by training four pre-trained word embedding methods: Word2vec, Fasttext, Glove, and Keras, using Bidirectional Long-Short Memory (Bi-LSTM) algorithms, a popular deep learning methodology. The training process for each embedding method was improved by using two optimization approaches, Adam and Nadam, in combination with a set of hyperparameters.

The study was intended to develop an effective model for predicting Amharic word sequences using the Bi-LSTM deep learning approach, with a focus on properly forecasting the next word. Among all the examined word embedding and optimization techniques, the combination of the Fasttext method and the Adam optimizer methodology with a batch size of 32 produced the highest training accuracy of 97.5% and validation accuracy of 95.6%. These findings suggest that bi-LSTM-based models, when combined with appropriate hyperparameters and embedding approaches, have considerable potential for improving text prediction systems in Amharic.

The findings of this study are expected to have significant implications for the development of word sequence prediction systems for the under-resourced Amharic language, as well as for future work in the field of Amharic language modeling.

6.2. Future work

To improve the effectiveness of the Amharic word sequence prediction model, numerous options for future study can be pursued. This includes:

- ✓ Incorporating additional NLP modules: Integrating modules for tasks like morphological analysis, part-of-speech tagging, and named entity recognition could provide a richer context for the model and improve its predictions.
- ✓ Exploring alternative sequence learning models: Investigating other recurrent neural networks, such as gated recurrent units (GRUs), or even transformer-based architectures could potentially lead to better performance.
- ✓ Utilizing larger and more diverse datasets: Training the model on a larger and more diverse corpus of Amharic text could improve its generalizability and adaptability to different writing styles and domains.
- ✓ Examining domain-specific adaptations: Investigating ways to tailor the model for specific domains, such as news articles, legal documents, or social media, could further enhance its accuracy within those contexts.
- ✓ Explore the impact of alternative word embedding methods, and further optimize hyperparameter settings beyond those investigated in this study.
- ✓ Future studies could investigate how well deep neural network compression methods such as weight tying and matrix factorization can decrease the computational demands and memory usage of the Amharic language task.

References

[1] I. . Z. "Natural Language Processing of Semitic Languages," *Springer, Heidelberg, Berlin,* 2014.

[2] T. T. Tessema, "Word Sequence Prediction for Amharic Language," *Theses Addis Ababa University,* 2014.

[3] A. M. M. and V. G. , "Amharic Text Predict System for Mobile Phone," *International Journal of Computer Science Trends and Technology ,* 2015.

[4] D. E. G. H. and W. T. , "Bi-directional long short term memory-gated recurrent unit model for Amharic next word prediction," *PLOS ONE,* 2022.

[5] A. P. "Augmentative and Alternative Communication systems for the moto disabled," *National and Kapodistrian University of Athens,* 2014.

[6] M. G. and E. D. , "A POS-Based Word Prediction System for the Persian Language," *Iran National Science Foundation,* 2008.

[7] D. A. P. M. M. M. P. H. W. H. S. and H. M. , "The Effects of Word Completion and Word Prediction on Typing Rates Using On-Screen Keyboards," *The National Center for Biotechnology Information,* 2015.

[8] Y. M. a. W. C. W. and T. Q. , "Design of Word Input Prediction System Based on LSTM,," *International Journal of Scientific Research and Innovative Technology,* 2021.

[9] J. M. M. B. and H. T. , "FASTY A multi-lingual approach to text prediction‖," *n Computers Helping People with Special Needs,* pp. pp. 243- 250, 2002..

[10] S. A. and S. A. , "Context based word prediction for texting language," *RIAO '07: Large Scale Semantic Access to Content (Text, Image, Video, and Sound),* p. Pages 360–368, 2007.

[11] C. A. N. C. N. D. P. M. and M. R. , "Advances in NLP applied to Word Prediction," *Department of Computer Science – University of Pisa, Italy,* 2008.

[12] S. K. BERHE , "Word Sequence Prediction Model for Tigrigna Language," *Theses Addis Ababa University ,* 2020.

[13] J. v. B. A. H. and A. M. , "Introduction to Design Science Research," *researchgate,* 2020.

[14] J. and J. , "A Survey of Text Generation Techniques," *Journal of Artificial Intelligence,* 2020.

[15] S. C. D. B. and R. R. , "Evaluation Metrics For Language Models," *Researchgate,* 2001.

[16] M. G. and S. M. , "An overview on the existing language models for prediction systems as writing assistant tools," *Man and Cybernetics,IEEE International Conference,* pp. pp. 5083 -5087, 2009.

[17] M. Q. and V. M. , "A Survey on Language Models," *Lakehead University, Canada,* 2020.

[18] E. . G. and E. P. , "A Swedish Grammar for Word Prediction," *Language Engineering Programme,Uppsala universty,* 2003.

[19] R. H. . r. B. and R. L. , "Word Frequency Distributions and Lexical Semantics," *Computers and the Humanities, Kluwer Academic Publishers,* 1997.

[20] H. S. S. S. J. B. E. C. and S. V. , "Recent Advances in Recurrent Neural Networks," *Computer Science Neural and Evolutionary Computing arXiv,* 2018.

[21] . Y. . T. Tessema, "NEXT WORD PREDICTION FOR AMHARIC LANGUAGE USING BI-LSTM," *M.Sc. Thesis HAWASSA UNIVERSITY,* 2023.

[22] A. F. R. and E. V. M. , "Automation of the process of selecting hyperparameters for artificial neural networks for processing retrospective text information," *OP Conference Series: Earth and Environmental Science,* 2020.

[23] N. S. and S. A. , "Word Prediction for Amharic Online Handwriting Recognition," *Master's Thesis, Addis Ababa University,* 2008.

[24] T. M. I. S. K. C. G. C. and J. D. , "Distributed representations of words and phrases from massive text corpora," *Conference on Neural Information,* pp. pp. 212-223, 2013.

[25] A. R. S. W. and A. D. A. F. B. H. , "Next word prediction using LSTM," *Information Technology and IT Uitilization,* vol. v0lum 1, 2022.

[26] A. M. P. A. and S. S. , "Deep Learning Techniques: An Overview," *Avinashilingam Institute for Home Science and Higher Education for Women,* 2021.

[27] L. A. J. Z. A. J. H. A. A. Y. D. O. A. J. S. M. A. F. M. A. M. A. and L. F. , "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *J Big Data,* 2021.

[28] Y. B. P. S. and P. F. , "Learning Long-Term Dependencies with Gradient Descent is Diffcult," *in IEEE Transactions on Neural Networks,* 1994.

[29] S. S.-N. N. T. and A. S. N. , "A Comparative Analysis of Forecasting Financial Time Series Using ARIMA, LSTM, and BiLSTM," *arXiv:1911.09512v1 [cs.LG] ,* 2019.

[30] A. F. G. and F. K. , "Predicting next Word using RNN and LSTM cells: Stastical Language Modeling," *2019 Fifth International Conference on Image Information Processing ,* no. Aejaz Farooq Ganai;

Farida Khursheed “Predicting next Word using RNN and LSTM cells: Stastical Language Modeling” 2(ICIIP), 2019., 2019.

[31] A. R. . K. N. T. X. . E. J. . I. S. and . Y. T. , "Learning transferable language models from massive datasets," *In International Conference on Machine Learning,* no. Radford, A., Narayanan, K., Xu, T., Janosik, E., Sutskever, I., & Tay, Y. (2018, May). Learning transferable language , pp. pp. 4783-4801, 2018.

[32] R. S. N. G. N. A. P. K. and C. . P. , "Next Word Prediction in Hindi Using Deep Learning Techniques,," *Fifth international Conference on Data Science and Engineering,* 2019.

[33] L. W. M. F. B. . Z. B. X. and S. M. , "Efficient Hyper-parameter Optimization for NLP Applications," *IBM Watson, T. J. Watson Research Center, NY,* 2015.

[34] Q. J. and . S. . Z. , "A Brief Survey of Word Embedding and Its Recent Development," *IEEE 5th Advanced Information Technology,* 2021.

[35] N. R. and I. G. , "Optimal Hyperparameters for Deep LSTM-Networks for Sequence labeling task," *Ubiquitous Knowledge Processing Lab (UKP-DIPF) German Institute for Educational Research,* 2017.

[36] P. G. and M. J. , "Better Static Word Embeddings Using Contextual Embedding Models," *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing,* 2021.

[37] C. W. P. N. and D. L. , "A Comparative Study on Word Embeddings in Deep Learning for Text Classification," *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval,,* pp. PP 37-46, 2020.

[38] A. E. G. T. and T. . A. , "Learning Word and Sub-word Vectors for Amharic (Less Resourced Language)," *International Journal of Advanced Engineering Research and Science (IJAERS) ,* Vols. Vol-7, no. Issue-8, Aug- 2020].

[39] M. E. P. M. N. M. I. M. G. C. C. K. L. and L. . Z. , "Deep contextualized word representations," *in arXiv preprint arXiv:1802.05365.,* 2018.

[40] D. P. K. and J. B. , "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION," *in International Conference on Learning Representations (ICLR),* 2015.

[41] S. S. A. B. and A. S. , "Comparison of Hyper-Parameter Optimization Methods for Deep Neural Networks," *j.inst.Korean.electr.electron.eng,* 2020.

[42] W. M. "Designing Geez Next word prediction Model using statistical," *Theses Bahir Dar University ,* 2020.

[43] A. B. Delbeto, "Word Sequence Prediction for Afaan Oromo,," *Theses Addis Ababa University,* 2018.

[44] C. A. N. C. and P. M. , "An Inflected-Sensitive Letter and Word Prediction System," *International Journal of Computing & Information Sciences,* vol. Vol. 5, 2007.

[45] P. P. B. and A. B. , "A RNN based Approach for next word prediction in Assamese Phonetic Transcription," *Dept ofCSE, DUIET, Dibrugarh University, Dibrugarh-786004, Assam, India,* 2018.

[46] D. S. R. ,. V. R. N. Dr. D. Usha, K. R. B. S. and S. P. R. , "EXO NEXT WORD PRERDICTION using Machine Learning," *jornal of Survay fisher science,* 2023.

[47] O. F. S. . A. M. A. A. K. and K. M. H. , "Bangla Word Prediction and Sentence Completion Using GRU: An Extended Version of RNN on N-gram Language Model," *2019 International Conference on Sustainable Technologie,* 2019.

[48] N. K. Abebe, "WORD SEQUENCE PREDICTION FOR AMHARIC," *Ms.Theses, Addis Ababa University,* 2011.

[49] B. M. Hailu , "SEMANTIC ROLE LABELING FOR AMHARIC TEXT USING DEEP LEARNING," *Theses Addis Ababa University,,* 2021.

[50] A. A. Argaw and L. A. , "An Amharic stemmer: Reducing words to their citation forms.," *Proceedings of the 2007 workshop on computational approaches to Semitic languages: Common issues and resources, Association for Computational Linguistics.,* 2007.

[51] G. M. Misganew, "SEMANTIC-AWARE AMHARIC TEXT CLASSIFICATION USING DEEP LEARNING APPROACH," *Thesis Addis Ababa University,* 2020.

[52] R. C. S. and E. R. M. , "Understanding LSTM a tutorial into Long Short-Term Memory Recurrent Neural Networks," *arXiv.org,* 2019.

[53] U. K. A. H. M. U. A. W. S. and M. O. B. , "Co-occurrences using Fasttext embeddings for word similarity tasks in Urdu," *arXiv:2102.10957v1,* 2021.

[54] N. B. F. K. and A. H. C. , "Combining FastText and Glove Word Embedding for Offensive and Hate speech Text Detection," *26th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems,* 2022.

[55] B. M. Hailu, Y. A. and Y. B. Sinshaw, "Semantic Role Labeling for Amharic Text Using Multiple Embeddings and Deep Neural Network," *Department of Computer Science, Addis Ababa University,* 2023..

[56] S. R. S. K. and B. B. , "Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark," *Computer Vision and Biometrics Laboratory, Indian Institute of Information Technology, Allahabad, Indi,* 2022.

[57] A. L. and F. S. F. , "A Survey of Optimization Techniques for Deep," *International Journal for Research in Engi neering Application & Management,* 2019.

[58] T. Y. J. M. and Y. A. , "Morphologically Annotated Amharic Text Corpora," *The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval,* pp. pp.2349-2355, 2021.

[59] R. S. A. P. J. W. J. C. C. D. . A. N. and C. . P. , "Recursive deep models for semantic compositionality over a sentiment treebank.," *the 2013 conference on empirical methods in NLP,* pp. PP 1631-1642, 2021.

[60] L. A. J. Z. A. . J. H. . A. A. D. Y. D. O. A. S. J. S. M. A. F. M. A. A. and L. F. , "Review of deep learning: concepts, CNN architectures, challenges, applications," *Journal of Big Data,* 2021.

[61] C. Abhimanyu , P. Abhinav and S. Chandresh , "Natural Language Processing," *International Journal of Technology Enhancements and Emerging Engineering Reserch,* vol. Vol 1, no. Issue 4 131 Issn, pp. 2347-4289, 2013.

[62] D. Z. and D. . W. , "Relation Classification via Recurrent Neural Network," *Tsinghua National Lab for Information Science and Technology,* 2015.

Appendices

Amharic Character

	Ge 'ez ä	*Ka'eb* u	*Salis* ï	*Rab'e* a	*Hamis* é	*Sadis* i	*Sab 'e* o
h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
ḥ	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
m	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
s	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
r	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
s	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
q	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
b	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
t	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
h	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
n	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
a	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
k	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
w	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
ẹ	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
d	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
g	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
ṭ	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
p	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
ts	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
ts	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
f	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ

Short form Expansion

```
def convert_short_form(short_form):
    # Define your Amharic short form to full form mappings here
    short_to_full = {
        # Add your mappings here
        'ወ/ሮ': 'ወይዘሮ',
        'ወ.ሮ' : 'ወይዘሮ',
        'ዶ/ር' : 'ዶክተር',
        'ዶ.ር' : 'ዶክተር',
        'ዓ/ም' : 'አመተ ምህረት',
        'አ.ም' : 'አመተ ምህረት',
        'አ/ም' : 'አመተ ምህረት',
        'ጠ/ሚ' : ' ጠቅላይ ሚኒስትር',
        'በጠ/ሚ' : 'በጠቅላይ ሚኒስትር',
        'ጠ.ሚ' : ' ጠቅላይ ሚኒስትር',
        'ጠ/ሚ/ር' :'ጠቅላይ ሚኒስትር',
        'ጠ/ሚንስትር': 'ጠቅላይ ሚኒስትር',
        'ጠ/ሚኒስትር': 'ጠቅላይ ሚኒስትር',
        'ት/ቤት' : 'ትምህርት ቤት',
        'መ/ቤቶችና' :'መስሪያ ቤቶችና',
        'መ/ቤት' : 'መስሪያ ቤት',
        'ም/ቤት' : 'ምክር ቤት',
        'በም/ቤቱ': 'በምክር ቤቱ',
        'ም/ቤቱ' : 'ምክር ቤቱ',
        'ፕ/ት'  : 'ፕሬዝዳንት',
        'ቅ/ገብርኤል' : 'ቅዱስ ገብርኤል',
        'ም/ሊቀመንበር': 'ምክትል ሊቀመንበር',
        'ም/ሊ /መንበሩ' :'ምክትልሊቀመንበር',
```

Normalize Geez and Arabic Number MIs-match

```
def normalize_char_level_mismatch(self, input_token):
    rep1 = re.sub('[ሃኅኃሐሓኻ]', 'ሀ', input_token)
    rep2 = re.sub('[ሑኁዅ]', 'ሁ', rep1)
    rep3 = re.sub('[ኂሒኺ]', 'ሂ', rep2)
    rep4 = re.sub('[ኌሔኼ]', 'ሄ', rep3)
    rep5 = re.sub('[ሕኅ]', 'ህ', rep4)
    rep6 = re.sub('[ኆሖኾ]', 'ሆ', rep5)
    rep7 = re.sub('[ሠ]', 'ሰ', rep6)
    rep8 = re.sub('[ሡ]', 'ሱ', rep7)
    rep9 = re.sub('[ሢ]', 'ሲ', rep8)
    rep10 = re.sub('[ሣ]', 'ሳ', rep9)
    rep11 = re.sub('[ሤ]', 'ሴ', rep10)
    rep12 = re.sub('[ሥ]', 'ስ', rep11)
    rep13 = re.sub('[ሦ]', 'ሶ', rep12)
    rep14 = re.sub('[ዓኣዐ]', 'አ', rep13)
    rep15 = re.sub('[ዑ]', 'ኡ', rep14)
    rep16 = re.sub('[ዒ]', 'ኢ', rep15)
    rep17 = re.sub('[ዔ]', 'ኤ', rep16)
    rep18 = re.sub('[ዕ]', 'እ', rep17)
    rep19 = re.sub('[ዖ]', 'ኦ', rep18)
    rep20 = re.sub('[ጸ]', 'ፀ', rep19)
    rep21 = re.sub('[ጹ]', 'ፁ', rep20)
    rep22 = re.sub('[ጺ]', 'ፂ', rep21)
    rep23 = re.sub('[ጻ]', 'ፃ', rep22)
    rep24 = re.sub('[ጼ]', 'ፄ', rep23)
    rep25 = re.sub('[ጽ]', 'ፅ', rep24)
    rep26 = re.sub('[ጸ]', 'ፆ', rep25)
    rep27 = re.sub('(ሱ[ዋአ])', 'ኗ', rep26)
    rep28 = re.sub('(ሙ[ዋአ])', 'ሟ', rep27)
    rep29 = re.sub('(ቱ[ዋአ])', 'ቷ', rep28)
    rep30 = re.sub('(ሩ[ዋአ])', 'ሯ', rep29)
    rep31 = re.sub('(ሱ[ዋአ])', 'ሳ', rep30)
```

Text Cleaning

```
def read_and_remove_punctuation(file_path):
    with open("normalized_text.txt", 'r', encoding='utf-8') as file:
        text = file.read()

    # Remove punctuation using regular expression
    cleaned_text = re.sub(r'[\!\@\#\$\%\^\«\»\&\*\(\)\…\[\]\{\}\;\"\”\›\’\‘\"\':\,\.\<\/\<\>\?\\\\|\`\´\~\-\=\+\:\¤\£\;\:-\¤\

    return cleaned_text

def remove_ascii_and_numbers(cleaned_text):
    # Remove all ASCII characters and Arabic/Amharic numbers
    rm_num_and_ascii = re.sub('[A-Za-z0-9]', '', cleaned_text)
    return re.sub('[\'\u1369-\u137C\']+','', rm_num_and_ascii)

# Example usage
file_path = 'your_text_file.txt'  # Replace with the path to your text file
cleaned_text = read_and_remove_punctuation(file_path)
cleaned_text = remove_ascii_and_numbers(cleaned_text)

# Display the first 10 lines of the cleaned text
lines_to_display = cleaned_text.split('\n')[:10]
for line in lines_to_display:
    print(line)

# Save the normalized text to a new file
output_file_path = 'normalized_text2.txt'  # Replace with your desired output file path
with open(output_file_path, 'w', encoding='utf-8') as output_file:
    output_file.write(cleaned_text)
```


Word Tokenizer

```
# Your existing code
with open("normalized_text2.txt", 'r', encoding='utf-8') as myfile:
    mytext = myfile.read()

mytokenizer = Tokenizer()
mytokenizer.fit_on_texts([mytext])
total_words = len(mytokenizer.word_index) + 1

# Display results using PrettyTable
table = PrettyTable()
table.field_names = ["Word", "Index"]

# Extract words and their corresponding indices from the tokenizer
word_index = mytokenizer.word_index
for word, index in word_index.items():
    table.add_row([word, index])

# Print the table
print(table)
```

```
|          ሌሽ          | 18068 |
|         በብድሩ         | 18069 |
|       መዘጋታቸውን       | 18070 |
|         ጠቁመዉ         | 18071 |
|        ልንሰርዝ         | 18072 |
|        ኢማኑዌል        | 18073 |
```

Frequent word sequence extraction

```
max_sequence_len = 82

# Read the text file
with open("normalized_text2.txt", 'r', encoding='utf-8') as myfile:
    mytext = myfile.read()

# Tokenize the text
mytokenizer = Tokenizer()
mytokenizer.fit_on_texts([mytext])

# Get word counts
word_counts = mytokenizer.word_counts

# Sort words by frequency in descending order
sorted_words = sorted(word_counts.items(), key=lambda x: x[1], reverse=True)

# Print the top 10 frequent words and their counts
top_frequent_words = sorted_words[:10]
for word, count in top_frequent_words:
    print(f'{word}: {count} times')

# Get total number of unique words
total_words = len(mytokenizer.word_index) + 1
print(f'Total unique words: {total_words}')
```

```
ሳይ: 778 times
ክው: 679 times
እር: 445 times
አሞ: 424 times
ጋር: 352 times
ወጸ: 308 times
መሆኑን: 290 times
ውስጥ: 272 times
ሲሆን: 256 times
ከ: 250 times
Total unique words: 18086
```

Set Sequence length

```
my_input_sequences = []
for line in mytext.split('\n'):
    #print(line)
    token_list = mytokenizer.texts_to_sequences([line])[0]
    #print(token_list)
    for i in range(6, len(token_list)):
        my_n_gram_sequence_indices = token_list[i-6:i+1]
        my_n_gram_sequence_words = [mytokenizer.index_word[idx] for idx in my_n_gram_sequence_indices]
        print(my_n_gram_sequence_words)
        my_input_sequences.append(my_n_gram_sequence_indices)
```

Output of Seven-gram sequence of Amharic words

```
['ተግባሩን', 'በከፊል', 'ጀምሮ', 'እንደነበር', 'ጠቁመው', 'ሆኖም', 'ዘንድሮ']
['በከፊል', 'ጀምሮ', 'እንደነበር', 'ጠቁመው', 'ሆኖም', 'ዘንድሮ', 'በአገዳ']
['ጀምሮ', 'እንደነበር', 'ጠቁመው', 'ሆኖም', 'ዘንድሮ', 'በአገዳ', 'አቅርቦት']
['እንደነበር', 'ጠቁመው', 'ሆኖም', 'ዘንድሮ', 'በአገዳ', 'አቅርቦት', 'እጥረት']
['ጠቁመው', 'ሆኖም', 'ዘንድሮ', 'በአገዳ', 'አቅርቦት', 'እጥረት', 'ሳቢያ']
['ሆኖም', 'ዘንድሮ', 'በአገዳ', 'አቅርቦት', 'እጥረት', 'ሳቢያ', 'ለሁለት']
['ዘንድሮ', 'በአገዳ', 'አቅርቦት', 'እጥረት', 'ሳቢያ', 'ለሁለት', 'ወራት']
['በአገዳ', 'አቅርቦት', 'እጥረት', 'ሳቢያ', 'ለሁለት', 'ወራት', 'ብቻ']
```

Output of seven gram sequence

```
[309, 3697, 744, 1989, 3698, 2913, 3699]
[3697, 744, 1989, 3698, 2913, 3699, 2914]
[744, 1989, 3698, 2913, 3699, 2914, 2915]
[1989, 3698, 2913, 3699, 2914, 2915, 133]
[3698, 2913, 3699, 2914, 2915, 133, 1990]
[2913, 3699, 2914, 2915, 133, 1990, 26]
[3699, 2914, 2915, 133, 1990, 26, 621]
[2914, 2915, 133, 1990, 26, 621, 2916]
[2915, 133, 1990, 26, 621, 2916, 1495]
[133, 1990, 26, 621, 2916, 1495, 98]
[1990, 26, 621, 2916, 1495, 98, 101]
```

Sequence padding

```
max_sequence_len = max([len(seq) for seq in my_input_sequences])
input_sequences = np.array(pad_sequences(my_input_sequences, maxlen=max_sequence_len, padding='pre'))
```

Splitting the data in to training, validation and test

```
# Split the data into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_valid, X_test, y_valid, y_test = train_test_split(X_temp, y_temp, test_size=0.33, random_state=42)
```

Load the pre-trained FastText model

```
from gensim.models import FastText
fasttext_model_path = 'cc.am.300.bin'
fasttext_model = FastText.load_fasttext_format(fasttext_model_path)
```

Model Building

```
model = Sequential()
model.add(Embedding(total_words, 100, input_length=max_sequence_len-1))
model.add(Bidirectional(LSTM(150)))
model.add(Dense(100, activation='relu'))
model.add(Dense(total_words, activation='softmax'))
print(model.summary())
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 6, 100)            1808600

 bidirectional_2 (Bidirecti  (None, 300)               301200
 onal)

 dense_3 (Dense)             (None, 100)               30100

 dense_4 (Dense)             (None, 18086)             1826686

=================================================================
Total params: 3966586 (15.13 MB)
Trainable params: 3966586 (15.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Set the weights of the Embedding layer to the Fasttext embedding matrix

```
embedding_matrix = embedding_matrix[:, :100]  # Keep only the first 100 dimensions
model.layers[0].set_weights([embedding_matrix])
model.layers[0].trainable = False
```

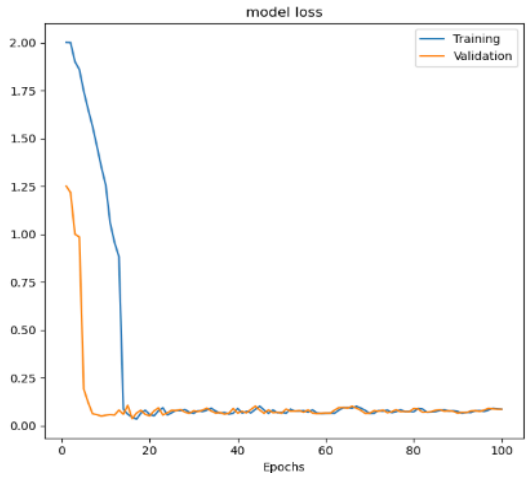
Training

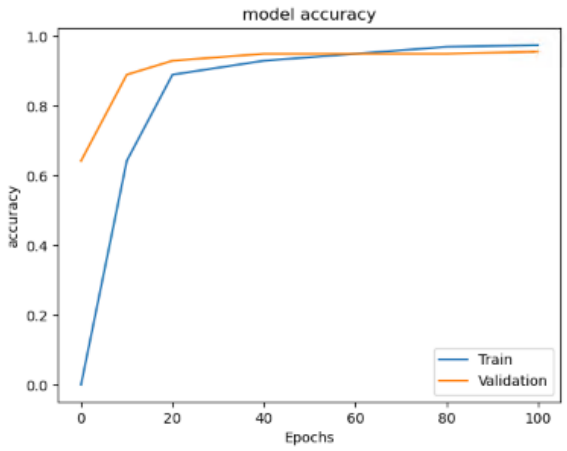
```
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
# Model training
history = model.fit(
    x=X_train,
    y=y_train,
    validation_data=(X_valid, y_valid),
    epochs=100,
    verbose=2
)
```

The Last 10 epoch Training with Fasttest word embedding

```
Epoch 91/100
547/547 - 483s - loss: 0.0451 - accuracy: 0.9766 - val_loss: 0.1914 - val_accuracy: 0.9589 - 483s/epoch - 883ms/step
Epoch 92/100
547/547 - 484s - loss: 0.0360 - accuracy: 0.9761 - val_loss: 0.1217 - val_accuracy: 0.9632 - 484s/epoch - 885ms/step
Epoch 93/100
547/547 - 480s - loss: 0.0361 - accuracy: 0.9758 - val_loss: 0.1065 - val_accuracy: 0.9583 - 480s/epoch - 878ms/step
Epoch 94/100
547/547 - 486s - loss: 0.0361 - accuracy: 0.9756 - val_loss: 0.0394 - val_accuracy: 0.9588 - 486s/epoch - 889ms/step
Epoch 95/100
547/547 - 479s - loss: 0.0361 - accuracy: 0.9754 - val_loss: 0.0846 - val_accuracy: 0.9579 - 479s/epoch - 876ms/step
Epoch 96/100
547/547 - 478s - loss: 0.0374 - accuracy: 0.9418 - val_loss: 0.0380 - val_accuracy: 0.9575 - 478s/epoch - 875ms/step
Epoch 97/100
547/547 - 475s - loss: 0.0361 - accuracy: 0.9752 - val_loss: 0.0348 - val_accuracy: 0.9571 - 475s/epoch - 868ms/step
Epoch 98/100
547/547 - 476s - loss: 0.0346 - accuracy: 0.9754 - val_loss: 0.0378 - val_accuracy: 0.9564 - 476s/epoch - 870ms/step
Epoch 99/100
547/547 - 476s - loss: 0.0345 - accuracy: 0.9752 - val_loss: 0.0376 - val_accuracy: 0.9572 - 476s/epoch - 871ms/step
Epoch 100/100
547/547 - 478s - loss: 0.0343 - accuracy: 0.9750 - val_loss: 0.0373 - val_accuracy: 0.9568 - 478s/epoch - 874ms/step
```

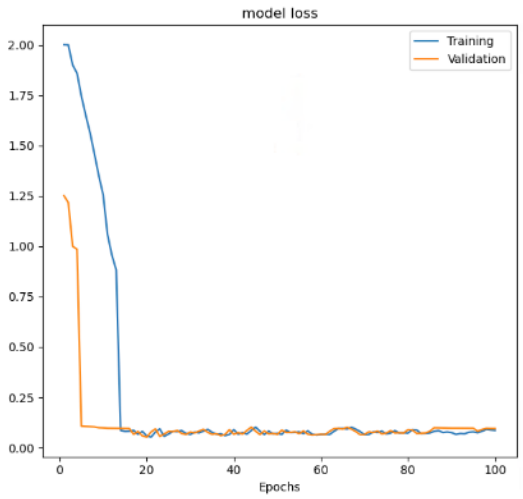


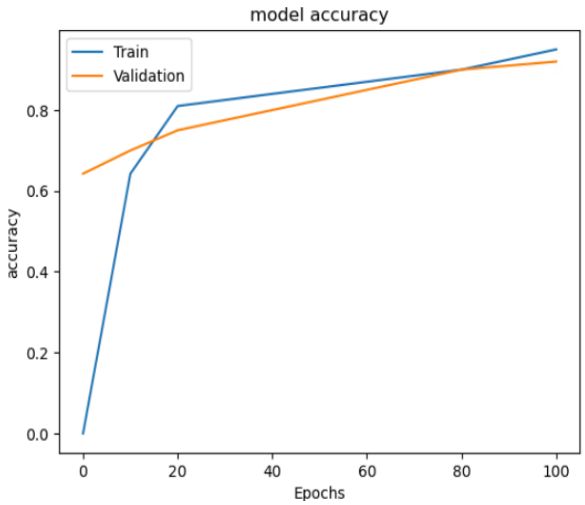


Fasttext loss and accuracy Graph

The Last 10 epoch Training With Word2vec word embedding

```
Epoch 91/100
547/547 - 483s - loss: 0.0859 - accuracy: 0.9784 - val_loss: 0.9493 - val_accuracy: 0.9612 - 483s/epoch - 883ms/step
Epoch 92/100
547/547 - 484s - loss: 0.0795 - accuracy: 0.9489 - val_loss: 0.9483 - val_accuracy: 0.9332 - 484s/epoch - 885ms/step
Epoch 93/100
547/547 - 480s - loss: 0.0860 - accuracy: 0.9493 - val_loss: 0.9475 - val_accuracy: 0.9274 - 480s/epoch - 878ms/step
Epoch 94/100
547/547 - 486s - loss: 0.0868 - accuracy: 0.9475 - val_loss: 0.9453 - val_accuracy: 0.9478 - 486s/epoch - 889ms/step
Epoch 95/100
547/547 - 479s - loss: 0.0859 - accuracy: 0.9453 - val_loss: 0.9428 - val_accuracy: 0.9468 - 479s/epoch - 876ms/step
Epoch 96/100
547/547 - 478s - loss: 0.0864 - accuracy: 0.9478 - val_loss: 0.9437 - val_accuracy: 0.9368 - 478s/epoch - 875ms/step
Epoch 97/100
547/547 - 475s - loss: 0.0854 - accuracy: 0.9456 - val_loss: 0.9456 - val_accuracy: 0.9344 - 475s/epoch - 868ms/step
Epoch 98/100
547/547 - 476s - loss: 0.0861 - accuracy: 0.9438 - val_loss: 0.9448 - val_accuracy: 0.8620 - 476s/epoch - 870ms/step
Epoch 99/100
547/547 - 476s - loss: 0.0858 - accuracy: 0.9441 - val_loss: 0.9438 - val_accuracy: 0.9280 - 476s/epoch - 871ms/step
Epoch 100/100
547/547 - 478s - loss: 0.0861 - accuracy: 0.9450 - val_loss: 0.0952 - val_accuracy: 0.9168 - 478s/epoch - 874ms/step
```

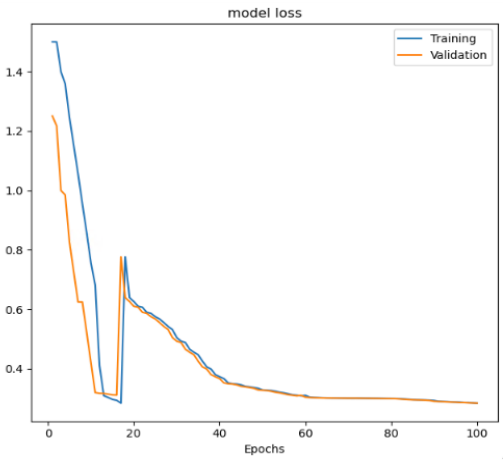


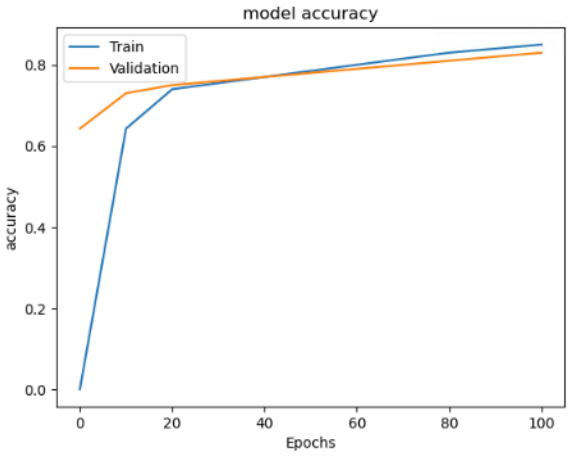


Word2vec loss and accuracy Graph

The Last 10 epoch Training with Keras word embedding

```
Epoch 91/100
547/547 - 483s - loss: 0.3146 - accuracy: 0.8191 - val_loss: 0.3246 - val_accuracy: 0.8589 - 483s/epoch - 883ms/step
Epoch 92/100
547/547 - 484s - loss: 0.3105 - accuracy: 0.8198 - val_loss: 0.3225 - val_accuracy: 0.8420 - 484s/epoch - 885ms/step
Epoch 93/100
547/547 - 480s - loss: 0.3094 - accuracy: 0.8238 - val_loss: 0.3204 - val_accuracy: 0.8310 - 480s/epoch - 878ms/step
Epoch 94/100
547/547 - 486s - loss: 0.3068 - accuracy: 0.8261 - val_loss: 0.3198 - val_accuracy: 0.8240 - 486s/epoch - 889ms/step
Epoch 95/100
547/547 - 479s - loss: 0.3034 - accuracy: 0.8285 - val_loss: 0.3174 - val_accuracy: 0.8140 - 479s/epoch - 876ms/step
Epoch 96/100
547/547 - 478s - loss: 0.2969 - accuracy: 0.8894 - val_loss: 0.3169 - val_accuracy: 0.8142 - 478s/epoch - 875ms/step
Epoch 97/100
547/547 - 475s - loss: 0.2936 - accuracy: 0.8299 - val_loss: 0.3146 - val_accuracy: 0.8143 - 475s/epoch - 868ms/step
Epoch 98/100
547/547 - 476s - loss: 0.2905 - accuracy: 0.8321 - val_loss: 0.3131 - val_accuracy: 0.8144 - 476s/epoch - 870ms/step
Epoch 99/100
547/547 - 476s - loss: 0.2862 - accuracy: 0.8342 - val_loss: 0.3121 - val_accuracy: 0.8145 - 476s/epoch - 871ms/step
Epoch 100/100
547/547 - 478s - loss: 0.2842 - accuracy: 0.8344 - val_loss: 0.3115 - val_accuracy: 0.8144 - 478s/epoch - 874ms/step
```

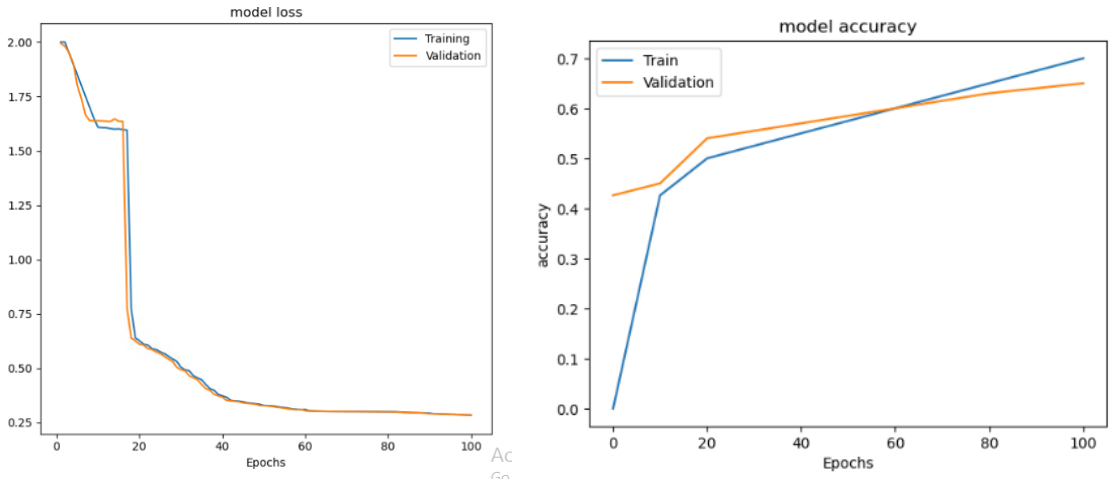




Keras loss and accuracy Graph

The Last 10 epoch Training with Glove word embedding

```
Epoch 91/100
547/547 - 483s - loss: 1.6042 - accuracy: 0.6282 - val_loss: 1.6392 - val_accuracy: 0.8589 - 483s/epoch - 883ms/step
Epoch 92/100
547/547 - 484s - loss: 1.6042 - accuracy: 0.6382 - val_loss: 1.6390 - val_accuracy: 0.7098 - 484s/epoch - 885ms/step
Epoch 93/100
547/547 - 480s - loss: 1.6039 - accuracy: 0.6399 - val_loss: 1.6388 - val_accuracy: 0.6598 - 480s/epoch - 878ms/step
Epoch 94/100
547/547 - 486s - loss: 1.6018 - accuracy: 0.6421 - val_loss: 1.6380 - val_accuracy: 0.6101 - 486s/epoch - 889ms/step
Epoch 95/100
547/547 - 479s - loss: 1.6010 - accuracy: 0.6426 - val_loss: 1.6378 - val_accuracy: 0.6203 - 479s/epoch - 876ms/step
Epoch 96/100
547/547 - 478s - loss: 1.6001 - accuracy: 0.6428 - val_loss: 1.6363 - val_accuracy: 0.6099 - 478s/epoch - 875ms/step
Epoch 97/100
547/547 - 475s - loss: 1.5990 - accuracy: 0.6429 - val_loss: 1.6350 - val_accuracy: 0.6101 - 475s/epoch - 868ms/step
Epoch 98/100
547/547 - 476s - loss: 1.5980 - accuracy: 0.6432 - val_loss: 1.6471 - val_accuracy: 0.6202 - 476s/epoch - 870ms/step
Epoch 99/100
547/547 - 476s - loss: 1.6071 - accuracy: 0.6432 - val_loss: 1.6363 - val_accuracy: 0.6102 - 476s/epoch - 871ms/step
Epoch 100/100
547/547 - 478s - loss: 1.5953 - accuracy: 0.6431 - val_loss: 1.6356 - val_accuracy: 0.6392 - 478s/epoch - 874ms/step
```



Glove loss and accuracy Graph

Word Prediction

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.sequence import pad_sequences

def make_prediction(model, text, n_words):
    for i in range(n_words):
        text_tokenize = tokenizer.texts_to_sequences([text])
        text_padded = pad_sequences(text_tokenize, maxlen=76932)  # Pad to the correct length
        prediction_index = np.argmax(model.predict(text_padded))
        prediction = str(vocab_array[prediction_index - 1])
        print(prediction)
        text += " " + prediction
    return text

# Example usage
result = make_prediction(loaded_model, "ሰላም", 5)
print(result)
```

```
1/1 [==============================] - 7s 7s/step
ሁኑን
1/1 [==============================] - 7s 7s/step
ብለናል
1/1 [==============================] - 7s 7s/step
ይበልጥ
1/1 [==============================] - 7s 7s/step
መውጣት
1/1 [==============================] - 7s 7s/step
ችለው
ሰላም ሁኑን ብለናል ይበልጥ መውጣት ችለው
```