Tigrigna-English Bidirectional Machine Translation using Deep Learning

A Thesis Presented

by

Fitiwi Hailu

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

January, 2024

Declaration

I, the undersigned, declare that this thesis work 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.

Fitiwi Hailu

Full Name of Student

_____

Signature

Addis Ababa

Ethiopia

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

Alemebante Mulu Kumlign (PhD)

Full Name of Advisor

_____

Signature

Addis Ababa

Ethiopia

January 2024

Acknowledgment

First and foremost, I would like to thank my God and Ever-Virgin, St. Marry, Mother of our Lord, for your blessing and giving me the courage and wisdom to accomplish this thesis.

Next, I would like to thank Dr. Alemebante Mulu, my advisor, for his support from the very beginning of proposal writing to this end. He has been actively interested in my work and he has always been available when I need him. I am very grateful for his patience, motivation, and immense knowledge. It could not be easier to finish this thesis work without his valuable comments and corrections.

Next I would like to thank my parents, whose love and guidance are with me in whatever I pursue. They are the ultimate role models. Most importantly, I wish to thank my loving and supportive wife, Eleni Tsegay, and my three wonderful children, Naome, Herani and Neamen, who provide unending inspiration.

Finally I would like to thank Journalist Biniyam Tadesse and, teacher Gebru W/Michel, for their support during data translation. I would like to say “Thank you all” for your support in my work.

Table of Contents

Table of Acronyms

AI—Artificial Intelligence

ANN— Artificial Neural Network

DL—Deep Learning

CNN—Convolutional Neural Network

HCI—Human-Computer Interaction

LSTM—Long Short Term Memory

MT—Machine Translation NLP—

Natural language processing NMT—

Neural Machine Translation RMT—

Rule Based Machine Translation

SMT—Statistical Based Machine Translation

List of Figures

List of Table

Abstract

A language can be described by its rules or its symbols. Making computers understand sentences or words written in human languages is the goal of natural language processing (NLP). Machine translation (MT) is area of NLP where computers are used to translate one natural language into another. One of the languages that needs such translation systems is Tigrigna. Tigrinya is a Semitic language spoken in northern Ethiopia in the Tigray Region as well as in Eritrea. Previously some studies were conducted on machine translation of Tigrigna and English languages. However most of the studies were only one directional which is English to Tigrigna languages. Some studies that proposed bidirectional Tigrigna-English machine translation are also domain specific. In this study, the researcher developed bidirectional Tigrigna-English machine translation model using different machine translation approaches. In the study we collected 31000 Tigrigna-English parallel corpus from different sources and by translating English text to Tigrigna. We then preprocessed the dataset through cleaning, normalizing and tokenization stages. Using our dataset we have experimented different machine translation approaches. We have experimented approaches of encoder decoder model and attention based models using LSTM, Bi-LSTM and GRU deep learning algorithms. Based on the result of our experiments, our encoder decoder model using the Bi-LSTM algorithm has a better BLEU score. The encoder decoder model using the Bi-LSTM algorithm scored 24.8 for English-Tigrigna translation and 24.4 for Tigrigna-English translation. The model achieves a BLEU score of +0.8 from a baseline translation model on the area.

Keywords: deep learning, NLP, BLEU

CHAPTER ONE

1 INTRODUCTION

1.1. Background

A language can be described by its rules or its symbols. Information is transmitted or broadcast using a combination of symbols. Making computers understand sentences or words written in human languages is the goal of the field of artificial intelligence (AI) and linguistics known as natural language processing (NLP). Natural Language Understanding and Natural Language Generation, which advance the task of understanding and producing text, are the two components of natural language processing. In order to simplify users' life and satisfy their desire to communicate with computers in natural language, natural language processing was developed. As not all users are fluent in machine specific language, NLP caters to those users who do not have the time to learn new languages or perfect them [1]. The study of how people and computers communicate with one another using natural language is known as natural language processing (NLP). In this area, computational linguistics, artificial intelligence, and computer science all interact. NLP is a text analysis method that enables computers to comprehend spoken language. Numerous NLP tools enable tasks like automatic text summarization, sentiment analysis, topic extraction, named entity recognition, parts-of-speech tagging, connection extraction, stemming, and other useful tasks that support human-computer interaction. Examples of these tools include text mining, machine translation, and automated question answering [2].

One of natural language processing's earliest and most intriguing subfields is machine translation. The main goal is to break down language barriers by creating a machine translation system that can translate between different human languages. In the area of artificial intelligence known as machine translation, computers are used to translate one natural language into another. It is an interdisciplinary area of study that draws concepts from other disciplines, including languages, artificial intelligence, statistics, and mathematics. Communities from all around the world are connected in this digital age and share a wealth of resources. In this kind of digital world, language barriers make communication difficult. To get around this problem, researchers from several nations and significant corporations are developing machine translation systems. Before the 20th century, performing the necessary translation was a pipe dream [3].

The concepts and expressions of the human mind are represented by a language, which is a means of communication. As a result, the translation approach was used to transmit communications from one language to another. The field of machine translation has undergone revolutionary change thanks to advancements in information, communication, and technology (ICT). It is necessary to translate literary works from any foreign language into native languages, more precisely using a machine translation system. Typically, the machine translation system is given the text in the foreign language, and then the translation is completed. By enabling global access to diverse literary sources, such systems can overcome linguistic barriers [4]. Computational linguistics studies the use of software to translate text or speech from one language to another. One subfield of Computational linguistics is machine translation (MT). Machine translation only replaces words from one language with words from another, but this does not guarantee accurate translation. Using statistical and neural techniques is a more advanced approach that is also a growing topic to handle the problem of multiple phrase recognition [5].

Tigrinya is a Semitic language spoken in northern Ethiopia in the Tigray Region as well as in Eritrea. It is one of Eritrea's nine official languages. A 13th-century document of regional customary regulations is the earliest known written work in Tigrinya. It was discovered in Eritrea's Logo Sarda neighborhood in Akele Guzai [6]. Around 10 million people in Eritrea and Ethiopia speak the low-resource Semitic language of Tigrinya. Its intricate root and template morphology produces a wide range of word inflections through the use of prefix, infix, and suffix affixation. It is challenging to create a dense corpus that includes the majority of Tigrinya words due to the numerous inflections used in the language [7].

The majority of the languages spoken in Europe and western Asia, from Iceland to India, are linked to English because it is a member of the Indo-European language family. The ancestral language, known as Proto-Indo-European, was reportedly spoken by nomads who wandered the southeast European plains some 5,000 years ago. Since the beginning of time, English has absorbed terms from several languages. The phonology and morphology of the languages spoken around the world have been used to create the English that we have today. English comes third with 369.7 million native speakers, although it ranks first with 898.4 million second language speakers, according to the 2019 edition of Ethnologue published by SIL International. English has spread around the

world, either voluntarily or involuntarily. If we go through history's records, we see that colonists initially imposed this universal language [8].

The research on deep learning and artificial neural networks is a result of our desire to create a computer system that can mimic the functioning of the human brain. Understanding the operation of our cognitive system is necessary for the development of such a system. Ivakhnenko and Lapa employed thin but deep models with polynomial activation functions in 1965 to develop the first deep-learning-like algorithms. They then applied statistical techniques to examine their results. They used statistical techniques to pick the top features in each layer and passed them on to the subsequent layer. They used layer-by-layer least squares fitting instead of backpropagation to train their network end-to-end, where earlier layers were independently fitted from later levels [9].

In recent years, deep learning has changed a number of industries, from computer vision to game artificial intelligence (AI). Due to these advancements, the field of machine translation has shifted to the use of deep-learning neural-based techniques, which have essentially replaced earlier methods like rule-based systems or statistical phrase-based techniques. Now that deep learning approaches are available, such as neural machine translation (NMT) models, it is possible to access all of the information included in the source phrase and automatically determine which information is pertinent at each stage of the output text's synthesis. The elimination of earlier independence assumptions is the main cause of the enormous improvement in translation quality [10]. A radical improvement over earlier machine translation techniques is neural machine translation. In contrast to SMT, which uses discrete symbolic representations, NMT uses continuous representations. On the other hand, NMT eliminates the need for excessive feature engineering by modeling the entire translation process with a single, sizable neural network. As opposed to SMT's separately tuned components, NMT's training is end-to-end. In addition to being straightforward, NMT has attained cutting-edge performance across multiple languages. In actuality, NMT also ends up serving as the primary technology for many commercial MT systems [11]. Artificial neural networks are used in deep learning, a type of machine learning, to mimic the workings of the human brain. It has the capacity to process massive amounts of data and find crucial patterns that can support important decision-making. In many NLP tasks, including text categorization and language translation, it offers cutting-edge accuracy. When it comes to natural language processing tasks like speech recognition, audio recognition, bioinformatics, and machine translation, deep learning

architectures like deep belief networks, deep neural networks, and convolutional neural networks often outperform human experts [12].

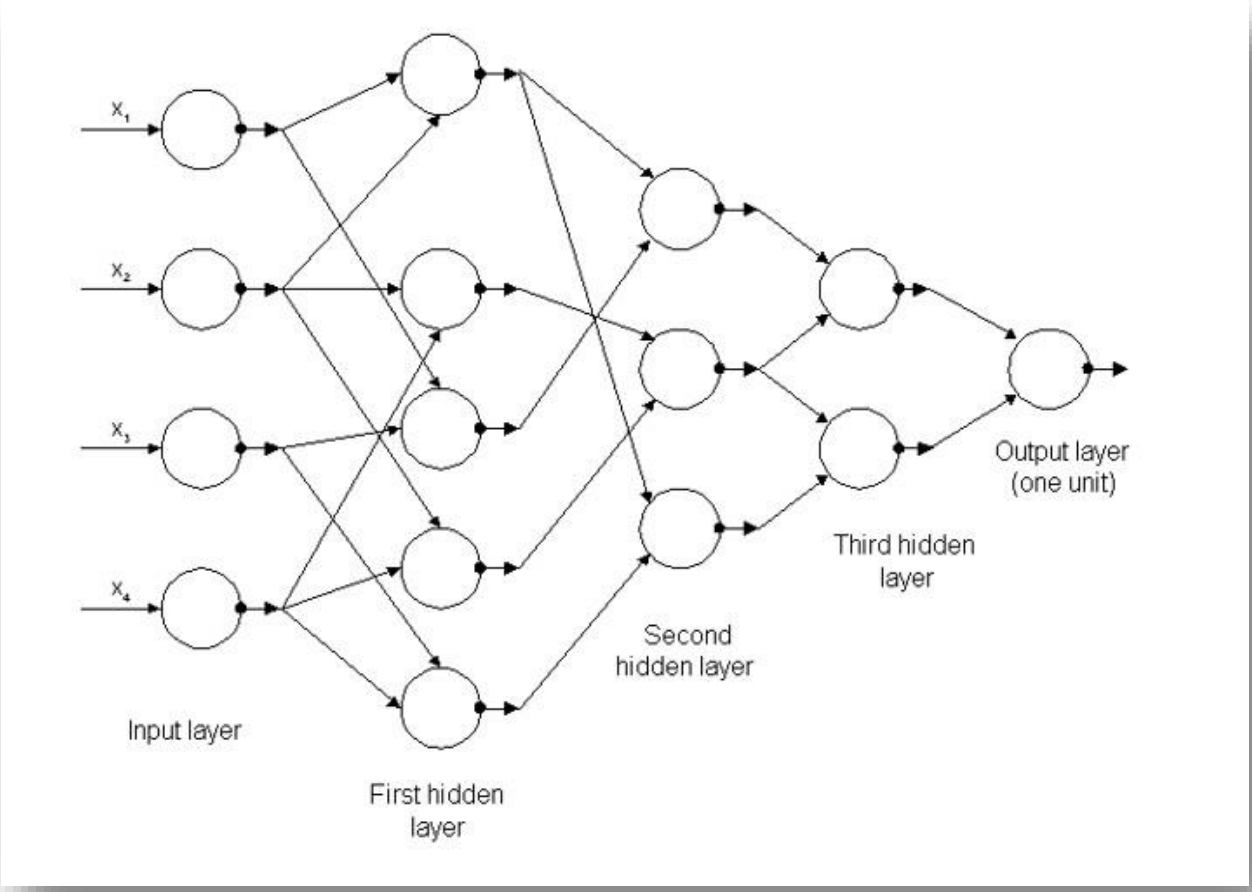

Figure 1 The architecture of the first known deep network which was trained by Alexey Grigorevich Ivakhnenko in 1965 [13].

1.2. Problem statement

Tigrinya is an Afro-Asiatic Semitic language that is spoken in the East African nations of Eritrea and Ethiopia. It is descended from the ancient Geez language. The Tigrigna script, in contrast to the Latin language, has more than 32 base letters with seven vowels each. A start letter has six different suffixes [14]. Previously some studies were conducted on machine translation of Tigrigna

and English languages. The study [4] proposed a statistical machine translator for English to Tigrigna translation. However the study were only one directional which is English to Tigrigna languages only. The study proposed [7] English to Tigrigna translation using neural machine translation. Again the study were only one directional which is English to Tigrigna languages only. In the study by [15] Tigrigna neural machine translation were proposed. The study focused on Tigrigna to English machine translation not the reverse using transfer learning. The study were conducted for domain specific case of humanitarian response. This makes the model developed in the study limited to humanitarian response domain only. In the study [16] a bidirectional English-Tigrigna machine translation were proposed. The study used a hybrid approach of statistical approach and post-processing technique. Even though the study reported good performance of their approach, the model is limited to four domains only. In the study [17] a bidirectional Tigrigna – English machine translation using statistical machine translation approach were conducted. However SMT technique may disregard the extended dependency that exists beyond the length of phrases resulting an errors in translation outcomes such as gender agreements that are wrong. Separate components, such as word aligners, translation rule extractors, and other feature extractors, are also affected [18]. In the study [19] English -Tigrigna factored statistical machine translation were conducted. The study used statistical machine translation and it was English to Tigrigna translation and not the reverse.

Tigrigna –English Machine Translation is required since a lot of documents are written in both languages. Due to the information communication between many regions employing different regional languages, translation systems are becoming more and more in demand today. As far as our knowledge is concerned there is no research done that fills the above limitations of previous studies. The goal of this research is to develop bidirectional Tigrigna-English machine translation model using Deep learning techniques.

1.3. Motivation

Machine translation systems remove language barriers by translating one human language to other languages. It is essential for improving communication between individuals who live in different places and for enabling people to use data and documents created in languages with abundant resources. The requirement for information sharing between languages with abundant and limited

resources creates a high demand for translation. The languages that have been translated the least are Ethiopian languages, like Tigrigna, which is thought to have fewer resources even though it is a language that is widely spoken in the country's north part and Eritrea. English, being the language that is used everywhere on the Internet, is the language that is translated the most, making it difficult for non-native speakers of the language to communicate and obtain resources. The community, private sector, and public sector in the Tigray region and Eritrea will all benefit greatly from improved machine translation of Tigrigna-English languages. Since Tigrigna is the official working language of the Tigray region and Eritrea, applying machine translation on the translation of various educational books or other materials can contribute to different government institutions like elementary education. Bidirectional English-Tigrigna Machine Translation can solve the aforementioned problems, which has motivated us to work on bidirectional English-Tigrigna Machine Translation.

1.4. Research questions

The following are research questions answered at the end of this study.

A. To what extent the proposed model is effective as compared to previous baseline models?
B. How to develop and evaluate a model that can translate sentences from Tigrigna-English and vice versa?
C. To what extent does the proposed bidirectional Tigrigna-English translation work?

1.5. The objective of the study

1.5.1. General objective

The general objective of the study is to develop bidirectional Tigrigna-English machine translation model using deep learning techniques.

1.5.2. Specific objectives

To realize the general objective of our study, we carried out the following specific objectives:

- ✓ To review literatures and related works done on local and foreign languages
- ✓ To collect parallel corpus for Tigrigna-English translation model
- ✓ To study the linguistic behaviors of both Tigrigna and English languages
- ✓ To design Tigrigna-English machine translation model
- ✓ To evaluate performance of different deep learning algorithms for Tigrigna-English machine translation model.

1.6. Scope and limitation of the study

The scope of the study is designing Tigrigna-English bi-directional machine translation to translate sentences written in Tigrigna text into English text and vice versa using deep learning. In this study tasks such as speech translation, morphological analysis and word sense disambiguation is not covered.

1.7. Significance

The researchers, translators, and society all gain from this research. It introduces researchers new insights and datasets for more research. Additionally, it is important for advancing local language NLP research and inspiring academics to carry out MT between local languages. The ability to translate Tigrigna written papers, religious texts, educational texts, Tigrigna documents, and many literal news stories into English and vice versa is useful for translators. It is utilized to reduce the time and expense involved with manual translation. In order to strengthen our local languages, it encourages the community to exchange knowledge and learn the language.

1.8. Thesis organization

This study is organized as follows. The second chapter presents a literature review of natural language processing, Tigrigna language, English language, machine translation and techniques of machine translation. Previous studies on Tigrigna-English machine translation and performance evaluation metrics is explored in chapter two. The third chapter discusses the methodology, which include the dataset preparation and the proposed model architecture. The fourth chapter discuss about the result, discussion and findings of the study. The last chapter discusses the conclusion of the study and recommendations based on the study's results and findings.

CHAPTER TWO

2 Literature Review

2.1. Introduction to Natural language

Language is more than just a collection of sounds or a vocabulary of meaningful words. People communicate primarily through language, and this language-based communication usually occurs in a social setting. Language is used by people to convey their friendliness, love, anger, and pain as well as to interact socially and emotionally. Ideas can be shared and tasks can be completed with the use of language (tell a narrative, provide information, complain, or seek for help). It takes more than just knowing the vocabulary of the language being used to communicate effectively. Understanding how people using the language utilize its words is necessary for effective communication. Children acquire spoken language naturally through social interactions. Oral

language development in children starts in the years before school, usually without explicit instruction. Childhood is a continuous period of this growth. Children require direct education, including instruction on language structure, in order to learn to read and write [25]. Language is more than just words; it's also the arrangement of those words, or grammar. Humans are unique in having the capacity to speak in intricate, nuanced ways; while bees, parrots, and chimps may imitate it, they are not able to do so with the same intricacy or spontaneity. It is uncertain whether Neanderthals could communicate in the same way as modern humans, despite well-known conjectures, and hypotheses suggesting that language originated from a single gene mutation some 30,000 years ago are also becoming more and more contentious [26]. Language, which can also refer to the structure, syntax, or arrangement of each of its components, is a means of communicating thoughts or feelings that are understood and utilized by a community. Written symbols, gestures, and vocalizations are all part of human language; nonetheless, it is difficult to say with certainty that language does not exist in other creatures. Animals are excellent communicators and information carriers. Gorillas primarily use twenty-five different vocalizations for group communication. The band uses sounds, such as grunts and barking, to identify each member's location when traveling. The main uses of distance calls by common chimpanzees are to attract attention, sound an alarm, and point out food sources or other members of the society. Similar to humans, bonobos mostly use vocalizations for communication. Although animal communication is not as sophisticated as human speech, animal communication is nonetheless effective enough to show the evolutionary benefit of group information exchange. Thus, the question of how animal communication evolved into human language has been debated for decades [27].

2.2. Natural language processing

The ability of a computer software to comprehend spoken and written human language is known as natural language processing, or NLP. It's a part of AI, or artificial intelligence. With its origins in the study of languages, NLP has been around for more than 50 years. It has numerous practical uses in a range of industries, such as business intelligence, search engines, and medical research [28]. In order to achieve human-like language processing for a variety of activities or applications, a theoretically motivated spectrum of computing techniques known as "natural language processing" is used to analyze and represent naturally occurring texts at one or more levels of linguistic analysis. Natural language processing (NLP) is regarded as a branch of artificial

intelligence (AI), as demonstrated by its human-like qualities. Furthermore, since NLP aims to perform as well as humans do, it is reasonable to classify it as an AI discipline, even though its entire genealogy depends on a number of other disciplines. Natural language processing has been the subject of research for many years, beginning in the late 1940s. The first computer-based application pertaining to natural language was machine translation (MT) [29].

Natural language processing is the use of computing techniques to the study of linguistic data, usually in the form of textual data like articles or papers. Using linguistic insights, natural language processing aims to create a representation of the text that gives the otherwise unstructured natural language some structure. This structure can be more semantic, capturing the text's content, or syntactic, capturing the grammatical relationships between the text's parts. Systems biologists utilize natural language processing to create applications that combine data from the literature with information from other biological data sources. A pipeline of components makes up the standard natural language processing system, which manipulates an input text in ever more complex ways. Each component's general goal is to provide the text structure so that it may be processed more easily later on. Early in the pipeline components deal with jobs that are similar to the text's surface strings, whereas later components seek to evaluate concepts and their relationships. Component tasks can be completed using a variety of techniques, from statistical and machine learning models to rule-based approaches like regular expressions and finite state automata [30].

The field of natural language processing (NLP) originated in the 1950s as a combination of linguistics and artificial intelligence. First, text information retrieval (IR) was distinguished from natural language processing (NLP) by employing highly scalable statistics-based techniques to rapidly index and search large amounts of text. But over time, there has been some convergence between NLP and IR. Since NLP now draws from many other fields, researchers and innovators in the field must significantly broaden their conceptual knowledge base [31]. In computational linguistics, grammar refers to the study of certain structures and rules found in language, such as determining the principles of sentence order and classifying words. Language Model and Part-of-Speech Tagging are two ways for expressing linear laws in these languages. Syntactic Structure or Dependency Relationship between words in the sentence can be used to represent nonlinear information in the sentence. Although the analysis and expression of sentence structure may not be the final goal of natural language processing problems, it is frequently a key step in solving the problem [32].

Natural language processing is an interdisciplinary field that aims to get computers to perform useful natural language tasks like enabling human-machine communication, improving human-human communication, or simply performing useful text and speech processing. It is also known as Human Language Technology, Language Technology, or Speech and Language Processing. The coding, recognition, interpretation, translation, and production of human language are among the research and development activities in this sector. The final products of these efforts include speech and language technologies including text classification, machine translation, speech recognition and synthesis, and text mining [33].

Natural language, as opposed to computer languages, is human language. The distinction between them is that uncertainty is present. There is no ambiguity in any well-designed computer language. All known natural languages, on the other hand, have the trait of ambiguity. Ambiguity happens when an input can be interpreted in multiple ways. Ambiguity exists at every level of human communication. The study of computer programs that take natural, or human, language as input is known as natural language processing. Natural language processing software can tackle a variety of tasks, from low-level tasks like attributing components of speech to words to high-level ones like answering questions. Natural language processing (NLP) is required to convert meaningful information contained in text into structured data that may be utilized by computer operations [34].

2.3. Background of Tigrigna language

Tigrinya is a Semitic language spoken in northern Ethiopia in the Tigray Region as well as in Eritrea. It is one of Eritrea's nine official languages. A 13th-century document of regional customary regulations is the earliest known written work in Tigrinya. It was discovered in Eritrea's Logo Sarda neighborhood in Akele Guzai [6].

One of the languages of the subfamily of South Semitic languages known as Ethio-Eritrean Semitics is Tigrinya. There are two subgroups of the Ethio-Eritrean Semitic languages: North Ethiopian and Eritrean Semitic languages, and South Ethiopian Semitic languages. Ge'ez, Tigré, and Tigrinya make up the former. While Tigré is exclusively spoken in Eritrea, Ge'ez and Tigrinya are shared by Ethiopia and Eritrea. South Ethiopian Semitics includes Amharic and more than 20 languages [35]. Tigrigna is a member of the Ethio-Semitic language family, which is part of the Afro-Asiatic super family, and is the third most spoken language in Ethiopia. Although immigrants speak Tigrigna throughout the world, it is mainly spoken in the Tigray area of Ethiopia and Eritrea. The Ge'ez script, commonly known as the Ethiopic script these days, is used by Tigrigna [36].

The Tigrigna language is syllabic; each combination of a vowel and a consonant is represented by a single symbol, the alphabet. Tigrigna displays the root and stem pattern morphological phenomenon as a result of its rich morphology. It is vital to comprehend the behavior of Tigrigna language stems and roots since the morphological diversity is the outcome of appending affixes to the root verbs or nouns to indicate number, gender, tense, possession, etc. While a stem is a verb whose final letter is "sads" (6th order) or a noun that indicates a single number, a root is a verb that indicates a third person singular masculine, such as በልዐ ፣ ሰተየ ፣ ሰርሐ, etc. and a noun that expresses a singular noun. Thus, depending on whether the word is a verb, a stem may or may not have meaning. In Tigrigna inflectional or derivational morphology of words is created by adding an affix at the beginning or middle (inside) or end of the root. After adding the affix the new words may be new in meaning and structure from their respective roots. The affixes can also be concatenated with each other. The affixes in Tigrigna can be classified in to four categories. The first is prefixes that come at the beginning of the root. Example prefixes such as ን፣ ዝ፣ እንተ፣ ም፣ ብም፣...፣. The second affix is suffixes that come at the end of the root. Example suffixes such as ና፣ ታት፣ ት፣ ነት፣ ን፣ ... ፣. The third is infixes that come inside the root. Example infixes such as ባ in ሰባበረ፣ላ in በላልዐ፣ ታ in ሰታተየ. The fourth is circumfixes that are attached before and after the base form at the same time [36].

Based on the work of [37] a finite verb and a subject are the two minimum components of Tigrigna sentences. Tigrigna words are divided into eight main categories according to the parts of speech. These are Nouns/ "ሹም", Verbs/"ግሳት", Adjectives/"ቅፅላት", Adverbs/"ተውሳኽ-ግሳት, Pronoun/ተውላጠ-ሹም/ክንዲ-ሹም, Preposition/መስተዋድድ, Conjunction/መስተፃምር and Interjection/ቃል አጋኖ". The syntactic structure is created by sequentially joining several words. Subject ("በዓል-ቤት) (beOel-Biet)" object ("ተስሓቢ) (tesHebi)" verb ("ግሲ) (gsi)" word order (SOV) is the structure of a Tigrigna sentence.

Nouns ("ሹም")

In Tigrigna, nouns are names for people, places, animals, emotions, attributes, actions, and concepts. There are further subcategories of nouns, including countable, abstract, concrete, proper, and common. The majority of nouns in the language terminate with "Sadese Fidel," which is the

sixth letter. However, this does not imply that other letters or Fidel never end. Tigrigna nouns can mean several different things in their plural forms [37].

There is no standard method for changing a singular form into its plural equivalents. There are two methods for creating a noun's plural forms, despite the fact that there is no standard method for doing so. The first is adding an ending (internal plurals), and the second is replacing patterns (broken plurals). Plural nouns created by substituting patterns are sometimes referred to as "broken plurals" or "internal plurals," while plural nouns created by appending suffixes are known as "external plurals." External plurals can be formed with the two ends -ãn (ኦነ) and -"ãt /ኦት". -"ãn" is primarily limited to nouns that refer to male humans [37]. Example of forming plural forms of a nouns is stated in below.

- ✓ Using pattern replacement (broken plurals): "መንበር mɛnbɛr" ..... "መናብር mɛnabɨr"
- ✓ Using addition of an ending (internal plurals): "ዓመት" ---------- "ዓመታት".

Adjectives ("ቅፅላት")

Tigrigna adjectives are based on property, size, shape, color and most adjectives in Tigrigna were found in front of a noun. Adjectives are one of the four major word classes, and their main function is to give clear explanation for a noun (i.e., talk about things behavior or characteristics, like shape, size, color, type, property) [38].

Example: "ዝተልአኩ አወዳት መልእኽቲ ይዛረቡ።"

Verbs ("ግሳት")

A verb is a term that describes an event, condition, or action. It usually serves as the primary component of the predicate in a phrase. In Tigrigna, verbs are typically used at the end of sentences. Verbs in Tigrigna can terminate in two different ways: one for the subject and one for the object. As a result, the verb's affix may agree with the subject and the object at the same time.

Adverbs ("ተውሳኽ-ግሳት")

In Tigrigna, a word that describes a verb's attribute is an adverb.

Pronoun ("ተውላጠ ስም/ክንዲ-ሹም**")**

Pronoun is utilized to replace a noun and the noun place in sentences. Pronoun has the same usage like that of noun. The personal, reflexive, relative, reciprocal, demonstrative, interrogative, indefinite, and possessive pronoun are different types of pronouns.

Conjunction (“መስተፃምር”)

Words that are used to connect clauses or sentences or o coordinate words in the same clause are conjunctions. In Tigrigna words that act as a conjunction are ን ፣ ወይ ፣ and ነገር ግን.

The following Table 1 shows all Tigrigna word classes and their examples.

Table 1 Word Class Category in Tigrigna Language [38]

Word Class	**Example**
Verb(ግስ)	Drive (ዘወረ), grow(ዓበየ), sing(ደረፈ)
Noun(ስም)	Sister (ሐፍተይ)፡Bus (ኣውቶቡስ)፣ house (ገዛ),
Adjective(ቅፅል)	Big (ዓቢ)፤ happy(ሕጉሰ) cleaver(ጎበዝ)
Adverb(ተውሳከግስ)	Happily (ተሐጊሰ), recently (ቀረባ), soon
Preposition(መሰተዋድድ)	of, over (ልዕሊ), with (ምስ), in (ትሕቲ)
Pronoun(ክንድ ስም)	He (ንሱ), she(ንሳ)
Conjunction (“መስተፃምር”)	And (ከምኡ ውን), or (ወይ ድማ)
Interjection (ቃል ኣጋኖ)	Wow (ዋው), gosh (ጎሽ)

2.4. Background of English language

Based on the study by [39] the English language's history is divided into three distinct periods. It was decided that "old English covers from the first Anglo-Saxon settlements in England, from about 450 to about 1100, Middle English from about 1100 to about 1500, and Modern English from 1500 to the present day," despite the fact that there are no clear boundaries between these periods. It was observed that English achieved success and a notable position during the early modern age. Nonetheless, there were still numerous unresolved issues with the language. For instance, the grammar had choice-based forms, the spelling was not established, and pronunciation variances existed. These were the causes of the late modern era's necessity to correct and harmonize the language.

In English, there are eight recognized parts of speech. Still, this is mostly based on ancient Latin grammars. Despite their many similarities, pronouns and nouns are traditionally categorized as distinct components of speech. Conversely, there is less similarity between different kinds of adverbs and pronouns [40]. Based on [41] the main word classes—nouns, verbs, adjectives, and adverbs—can be divided into groups according to the morphological (or "word-building") characteristics of each group. Generally, words in the same class accept the same set of suffixes (endings). Additionally, several word classes have specific suffixes that are typically employed to make their terms.

2.5. Machine translation

Automated translation, often known as machine translation (MT) or sometimes just "MT," is distinct from computer-aided translation (CAT), machine-aided human translation (MAHT), or interactive translation. It is the process of translating a text from one natural language—like English—to another—like Ibo—using computer software. After the United States established the Association for Machine Translation and Computational Linguistics in 1962 and the National Academy of Sciences established the Automatic Language Processing Advisory Committee (ALPAC) in 1964 to investigate machine translation, researchers kept entering the subject. However, actual development was significantly slower, and funding was drastically cut following the ALPAC report (1966), which concluded that the ten-year research project had fallen short of expectations. A. D. Booth and possibly others first floated the idea in 1946 of translating natural languages using digital computers [42].

The origins of machine translation (MT) can be found in the conversations and correspondence between British crystallographer Andrew D. Booth and Warren Weaver of the Rockefeller Foundation in 1947. Specifically, Weaver's 1949 memorandum to the Rockefeller Foundation, which contained two sentences, is where the actual development of MT can be found. The comparison between translation and decoding may seem oversimplified to an experienced reader (despite the complexity of coding, it is essentially a one-to-one replacement process with a single correct answer; translation is a far more subtle and complex process). However, Weaver later in the memorandum offered some additional, more advanced perspectives, and these helped to transform an apparently challenging task into one that could be tackled with the help of emerging computer technology (computers had been successfully used for cryptography during World War II) [43].

Because language is such an efficient means of communication in the modern world, there is a growing demand for language translations. Because more information is being exchanged between different regions using distinct regional languages, there has been a rise in demand for translation services in recent years. Information professionals have expressed worry about, for example, web documents' accessibility in different languages. Translating documents from one natural (human) language to another using computers is known as machine translation (MT), a branch of artificial intelligence. A variety of methods have been employed recently to create an MT system [42].

One of information science's most difficult "dreams" to come true, so far, is machine translation (MT). Researchers and funding agencies that were extremely disappointed with the performance of their MT systems after spending significant sums of money for five or even ten years have frequently declared that MT is "difficult" or "impossible." They have acknowledged the challenges of both translation and machine translation. In addition to linguistic expertise, translation demands the highest level of "general" human intellect. Even for human professional translators who are proficient in multiple languages but lack sufficient subject-matter expertise, translation remains a challenging task [44].

Multilingual translation has gained prominence as a result of the globalization trend and the increase in knowledge that has occurred in the second millennium. In order to acquire knowledge in academic domains, Machine Translation (MT) warrants attention from both an academic and practical standpoint. Since MT is a useful strategy used by qualified translators in a variety of professional fields, translating students should be aware of it. Machine translation is an ideal

method that translation trainers, translation learners, and professional translators should be familiar with for developing translating skills and finding a means to learn and teach through bi-lingual/multilingual translating functions in software. Indeed, many academics had placed a great weight on theories pertaining to computer help and machine translation [45].

Machine translation is one of the oldest and most exciting subfields in natural language processing. The main objective is to develop a machine translation system that can translate between human languages in order to overcome language barriers. A machine translation is an interdisciplinary area of research that combines ideas from several academic fields, including mathematics, artificial intelligence, languages, and statistics [3].

Approaches to MT can be divided into two categories based on methodology: rule-based methods and dataset-based methods. Rule-based methods dominated from the time the concept of MT was first proposed until the 1990s. Rule-based machine translation (RBMT) methods translate source language texts into target language texts using bilingual dictionaries and manually written rules. However, manually writing rules is time consuming. Furthermore, rules are difficult to keep and transfer from one domain to another, as well as from one language to another. As a result, rule-based systems are difficult to scale for open-domain translation and multilingual translation. MT systems were initially designed primarily for military applications. Georgetown University, in collaboration with the now-famous computer manufacturer International Business Machines Corporation (IBM), completed the first Russian-English MT experiment using the IBM-701 computer in 1954, demonstrating that the dream of MT had come true. Following the 1954 presentation, MT gained significant traction for over ten years. However, the boom came to an abrupt halt in 1966 with the release of the Automatic Language Processing Advisory Committee (ALPAC) report. It became extremely difficult to work on MT after the report, which was extremely skeptical of MT and resulted in a drastic cut in funding for MT research. The Association for Computational Linguistics (ACL), the dominant scientific society today, was originally named the Association for Machine Translation and Computational Linguistics in 1962, during the boom; however, it dropped the "MT" from its name in 1968, during the bust, following the ALPAC report. Meanwhile, MT researchers kept trying to improve translation quality. The first International Conference on Computational Linguistics, which focused on rule-based parsing and translation, was held in 1965 by NLP researchers. Beginning in the 1970s, RBMT methods became more refined. One of the first MT companies, SYSTRAN, launched a commercial

translation system in 1978, which was one of the most well-known examples of a commercially successful rule-based system at the time. SYSTRAN's services were used by Google until 2007 [46].

Machine translators can be purchased as commercial computer solutions (for example, SYSTRAN Enterprise Server and IBM WebSphere) or as free Web-based applications (eg, Google Translate and Microsoft Bing Translator). Most machine translators are text-based and provide instant translations between various languages; however, audio output is sometimes available. A variety of language keyboards are occasionally available. Google Translate, for example, has a keyboard icon that allows users to switch between different language scripts by toggling an on-screen keyboard. To use the virtual keyboard, select the language from which you want to translate (i.e., uncheck "Detect language" and select a language other than English). The virtual keyboard icon will appear in the text box's lower left-hand corner. Smartphone apps that connect to online machine translation programs are also on the rise [47].

2.6. Approaches of machine translation

Approaches of machine translation can be categorized using the fundamental techniques of MT systems. This classification is based on two primary paradigms: the dataset-based approach and the rule-based method. The rule-based approach requires a large amount of human expert input as human experts define a set of rules to characterize the translation process. In contrast, the dataset-based technique uses an analysis of translation instances from a parallel dataset created by human experts to automatically extract information. After merging the best features from the two main categories of machine translation systems, the Hybrid Machine Translation Approach was formed [42]. Figure shows different techniques of machine translation.

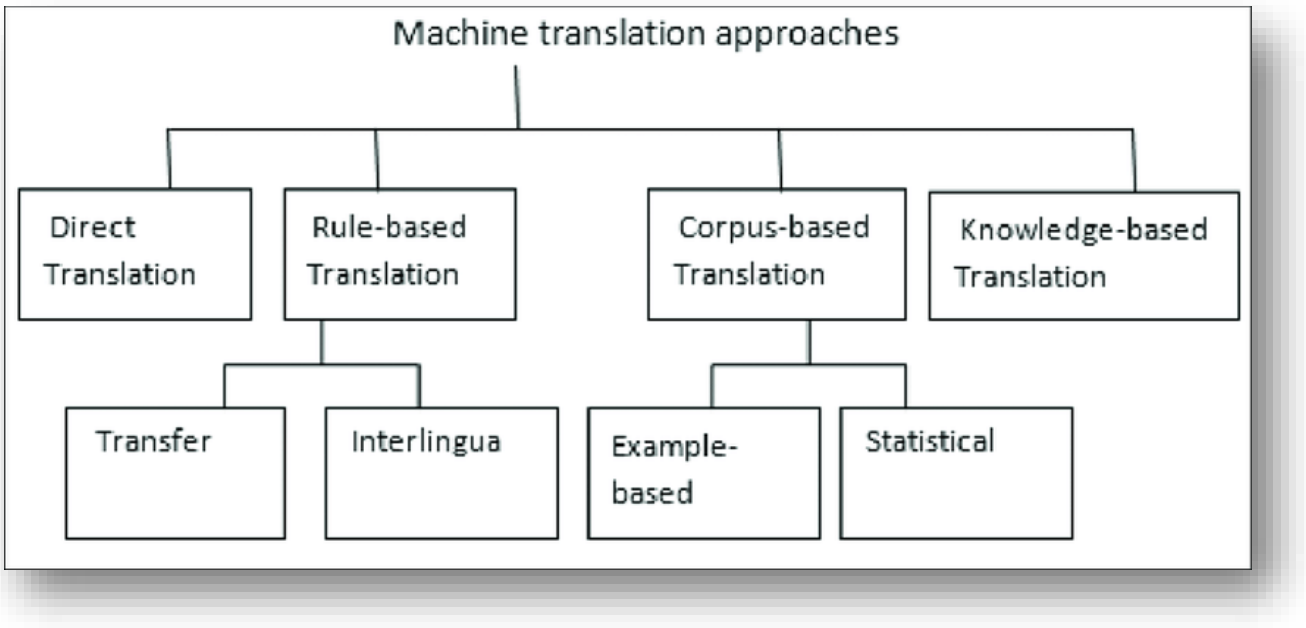

Figure 2 categories of machine translation

2.6.1. Rule-based machine translation (RBMT)

The rule-based machine translation (RBMT) approache is knowledge-based machine translation, which requires a lot of human efforts on the preparation of linguistic rule and resources of both source and target languages. The RBMT system translates a given input sentences to output sentences based on morphological, syntactic, and semantic analysis of both source and target languages. Rule-Based Machine Translation (RBMT) are the Classical Approach to MT, which is a general term for machine translation systems that are based on linguistic information about the source and target languages that is retrieved from (bilingual) dictionaries and grammars that cover the main semantic, morphological, and syntactic regularities of each language. Based on morphological, syntactic, and semantic analysis of the source and target languages involved in a given translation task, a RBMT system generates output sentences (in some target language) from input sentences (in some source language) [42]. There are three different methods in the rule-based machine translation technique. They are the Interlingua, Transfer, and Direct machine translation approaches. Despite being members of the RBMT, their approaches to achieving a representation of meaning or intent that is independent of language between the source and target languages varies [42].

2.6.1.1. Direct Machine Translation (DMT) Approach

At the base of the pyramid, the Direct Machine Translation Approach is the lowest level. The oldest and least used method is DMT. Direct translation is done at the word level. With this method, machine translation systems can translate between two languages: the target language (TL) and the source language (SL). The SL words are translated without the use of an additional/intermediary representation. The analysis of SL texts is limited to a single TL. Direct translation systems are primarily bilingual and unidirectional in nature. A minimal amount of syntactic and semantic analysis is required for the direct translation approach. SL analysis is focused on producing representations that are appropriate for a single TL. DMT is a word-for-word translation method with some minor grammatical changes [42]. Figure 1 shows the steps in direct machine translation approach.

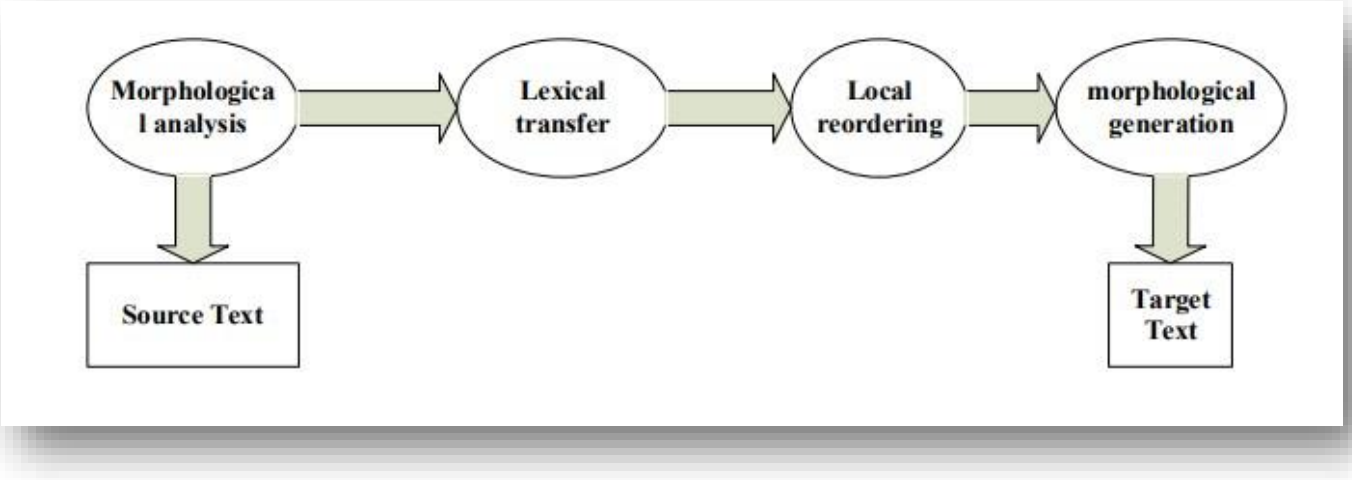

Figure 3 Steps of direct machine translation approaches

2.6.1.2. Transfer-based Machine Translation Approach

The transfer based machine translation generates generates a translation that mimics the original sentence's meaning from an intermediate representation. It is somewhat dependent on the language pair being translated, in contrast to interlingual machine translation. The Transfer-based Approach is a better rule-based translation method that was found as a result of the shortcomings of the Interlingua approach. Similar to interlingual machine translation, transfer-based machine translation creates a translation from an intermediate representation by simulating the meaning of the source sentence. Unlike interlingual MT, it depends to some extent on the language pair involved in the translation. Based on the structural distinctions between the source and destination

languages, a transfer system can be broken down into three stages: I analysis, ii) transfer, and iii) generation. The syntactic representation of an SL sentence is created in the first stage using the SL parser. In the next step, the output of the first stage is transformed into corresponding TL-oriented representations. The last stage of this translation methodology is to produce the final TL texts using a TL morphological analyzer. Translations using this technique can be of a respectably good caliber [42].

Figure 2 shows steps in Transfer-based approach of machine translation.

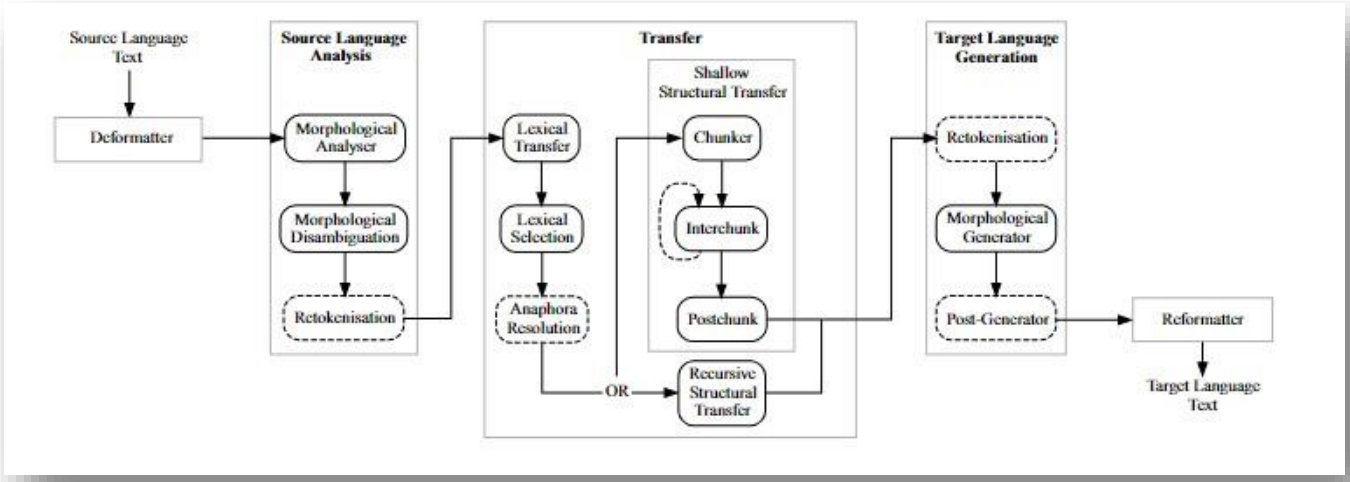

Figure 4 Transfer-based Approach of MT [48]

2.6.2. Dataset-based Machine Translation Approach

An alternate method of machine translation that tackles the rule-based machine translation issue of knowledge acquisition is dataset-based machine translation, commonly referred to as data-driven machine translation. As its name suggests, dataset-based machine translation (DBMT) leverages a multilingual parallel dataset to gather information for newly incoming translations. This approach uses parallel corpora, which are large-scale collections of raw data. This raw data includes text and translations. Translator training is based on these corpora. The dataset-based approach is further classified into two approaches. The statistical machine translation and example-based machine translation [42].

2.6.2.1. Example-based Machine Translation Approach

Translation by analogy is the core notion behind example-based machine translation (EBMT), which stands out for using a bilingual dataset with parallel texts as its primary knowledge. An EBMT system receives a series of sentences in the target language that correspond to each sentence's point-to-point translation in the source language. Sentences from the source language to the target language are translated using these examples. The four tasks that make up EBMT are example synthesis, example application, example base and administration, and example acquisition. The foundation of example-based machine translation is the idea of translation by analogy. The idea of translation by analogy is conveyed to example-based machine translation through the example translations used to train such a system [42].

Most example-based machine translation (MT) systems translate by using phrases or sentences as the example unit, which enables them to take case relations or idiomatic expressions into account. Example-based machine translation (MT) obtains instances that are similar to an input sentence by treating a bilingual dataset as a database. When certain examples conflict during retrieval, example-based MT selects the best example based on how similar the example's input and source components are. This implies that the accuracy of the translation of the given input sentence is not verified by example-based machine translation [49]. Compared to hard rule-based approaches, the example-based approach offers more flexible transfer, simple translation knowledge acquisition, and natural translations. The application of o1' linguistic rules, however, has several significant advantages. An example-based machine translation system can handle a wide range of input by using detailed linguistic analysis, since rules can be used to factor out all linguistic variations that do not affect the tile exampled) used transfer. Higher grammatical output quality can be achieved through rule-based language generation derived from detailed linguistic representations. Ultimately, the system can be expanded to much larger domains thanks to a modular system architecture that makes use of domain-independent linguistic regularities in distinct linguistic modules [50]. Figure 5 shows the architecture of example based machine translation.

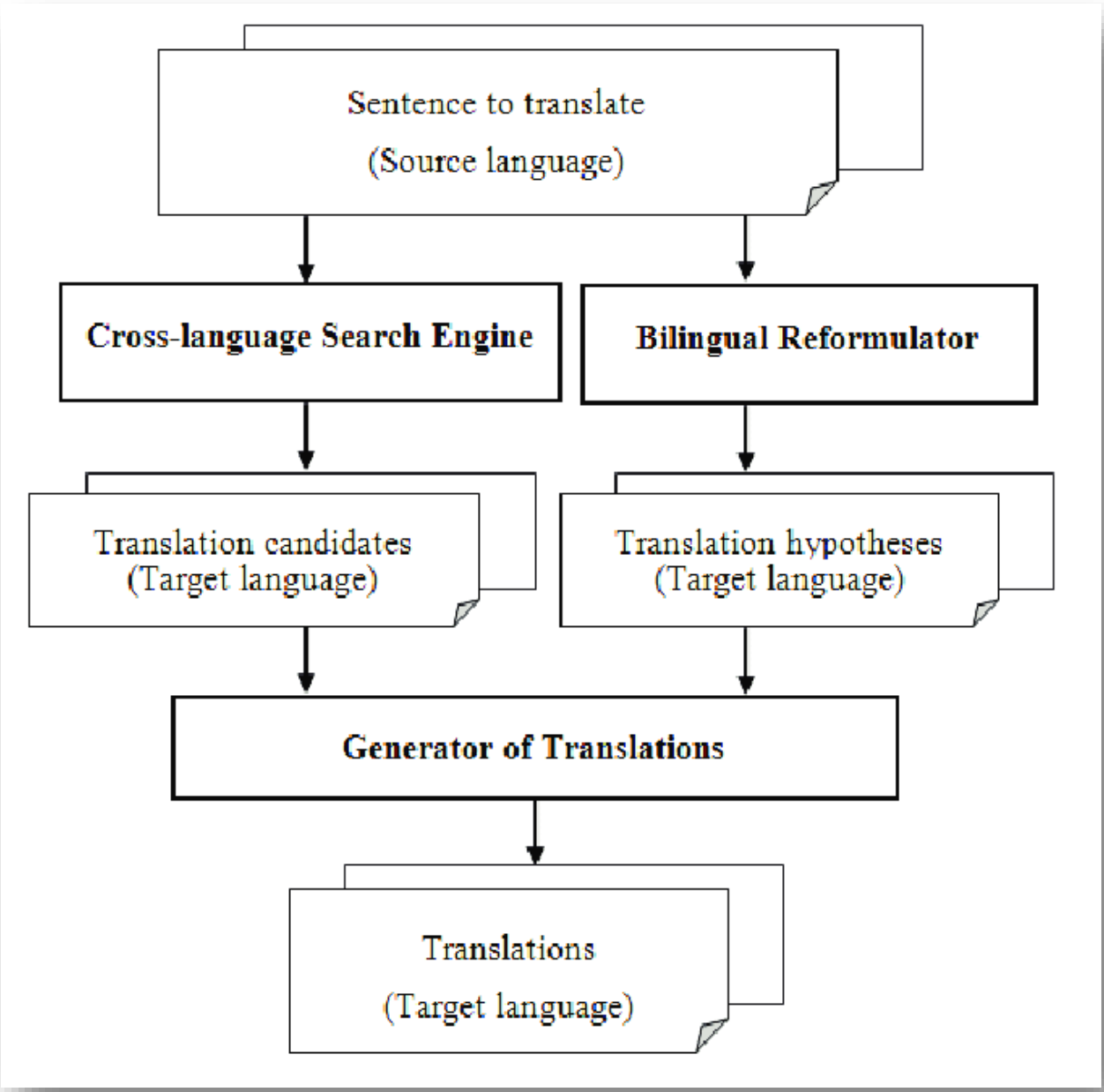

Figure 5 Example based MT architecture [51]

2.6.2.2. Statistical machine Translation

In 1949, Warren Weaver suggested applying statistical and cryptanalytic methods that were being developed from the then-emerging discipline of communication theory to translate text from one natural language to another. Though attempts in this way were soon abandoned for a number of theoretical and philosophical reasons, any such strategy was certain to fail at a time when the most sophisticated computers were comparable to modern digital watches. These days, anyone with a

decent workstation can study machine translation using statistical approaches and benefit from the research they do. A string of English words, e, can be translated into a string of French words in a variety of ways. The field of permissible French translations can be reduced by being aware of the larger context in which the word appears, but there will still be a wide range of acceptable translations; ultimately, preference will determine which one is preferred. We think that every French string, f, can be rendered as e via statistical translation. For every pair of strings (e,f), we assign a number Pr (f|e), which we understand as the probability that a translator given e will translate to f [52].

A Source Language Model and a Translation Model (S,T) produce a probability distribution over source-target sentence pairs. The sum of the conditional probability Pr (T|S) in the translation model and the probability Pr (S) in the language model yields the joint probability Pt (S, T) of the pair (S, T). These models' parameters are automatically estimated by means of a statistical technique that maximizes the fit between the models and the data, utilizing a vast database of source-target sentence pairings [53]. Figure 6 shows general architecture of Statistical machine translation.

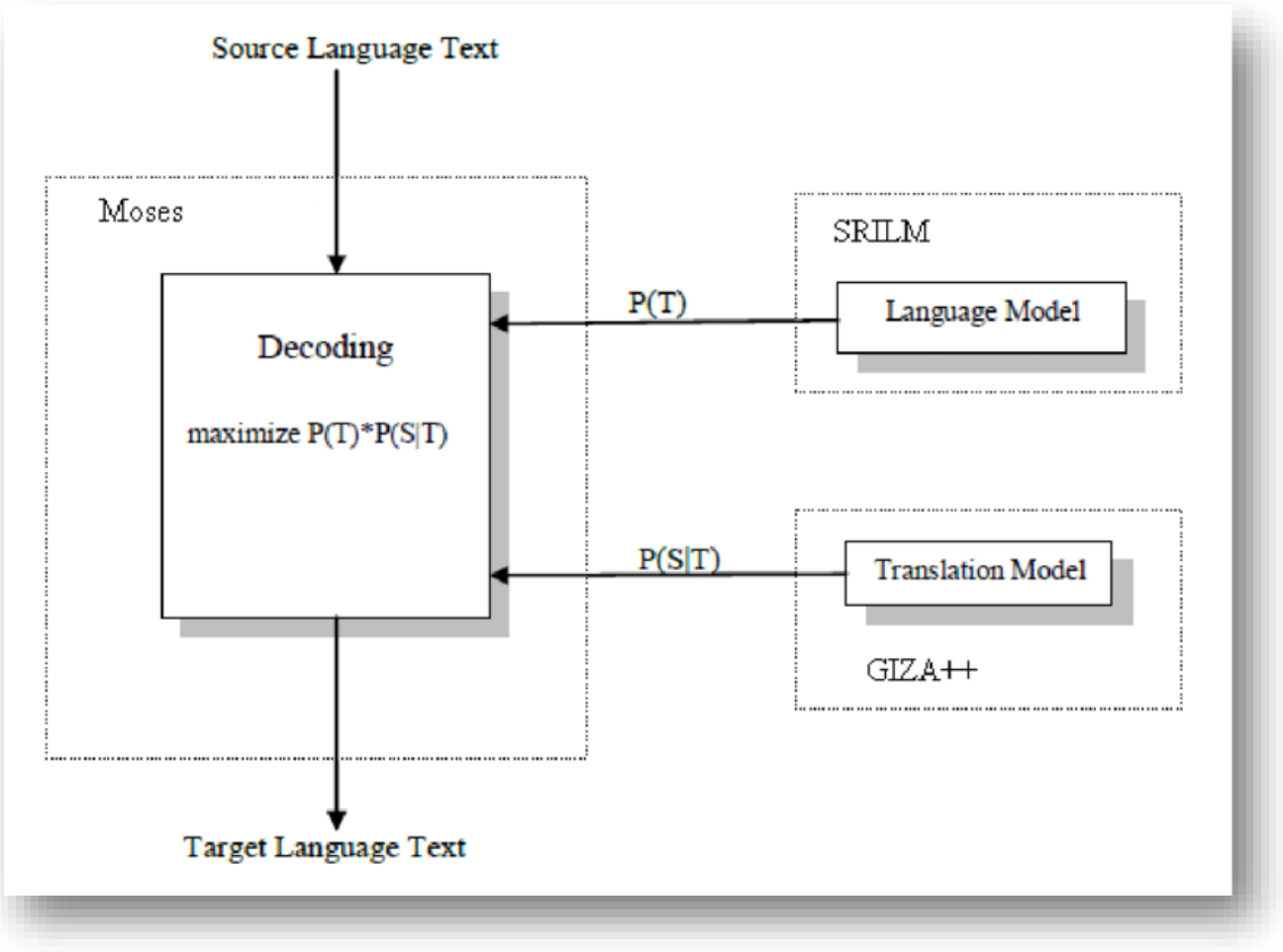

Figure 6 the general architecture of Statistical machine translation [54]

2.6.3. Machine translation using Neural Network

In order to facilitate the computation of the statistical probability assigned to each word in a sequence, neural networks were introduced as an advancement tool for SMT. The concept of simultaneously training and translating data from one natural language to another was modeled for pure neural network machine translation after further research by multiple researchers. Neural networks haven't attempted machine translation tasks in a very long time. But the performance was appalling in the early stages of the attempt. Neural network-based machine translation research has been neglected for a long time [55].

Deep neural networks were introduced for improved performance of existing machine translation systems in a number of ways, particularly in terms of translation quality. One of the main causes of this increase is deep neural networks' capacity to learn a logical representation of words. Deep

neural architecture-based machine translation is producing cutting-edge outcomes when translating European languages [56].

2.6.3.1. Neural Machine Translation (NMT)

With most language pairs, neural machine translation (NMT) is a well-researched method that produces the best results. The sequence-to-sequence model with attention, which uses single-layer recurrent neural networks in both the encoder and the decoder, is the foundation for the majority of systems. Recurrent neural network architectures with multiple layers allow different connections that result in different, orthogonal definitions of depth, which can affect the model performance depending on a given task. This is in contrast to feed-forward networks, where depth is simply defined as the number of non-input layers. The complexity of sequence-to-sequence models increases due to their multiple feed-forward or recurrent sub-networks, which can all be deep in different ways and result in a vast array of possible configurations [57]. NMT is made up of two neural networks: one encoder and one decoder.) The encoder converts the original sentence into a context vector c, which the decoder decodes to generate the target sentence. When the length of a sentence increases, encoding it into a fixed-length content vector v causes a problem. Incorporating the attention layer with the design can help to solve this problem and provide good performance. It is equal to finding a target sentence that optimizes the conditional probability, that is, arg max P(t|s), according to the probabilistic method. The encoder considers the source sentence S to be a series of vectors S = (x1, x2, x3,...) in vector v, also known as thought [3]. An end-to-end learning mode is used in neural machine translation, an automated translation technique. A neural network of machines is used to both encode and decode the source text. It's more like adhering to a set of rules that have been established. The quality of the translation has significantly increased thanks to neural machine translation (NMT), which can also handle conventional idioms and phrase-based content. An amazing tool that significantly affects translation precision and accuracy is the neural machine, a novel invention [58].

2.6.3.2. Different Neural Machine Translation Models

A relatively new approach to machine translation is called deep learning. Neural machine translation is a better option for more accurate translation and performance than traditional machine translation. Deep learning can be used to enhance and increase the efficiency of current systems. Various deep learning methods and libraries are needed to create a better machine translation system. For training purposes, the system that will translate the sentence from source

language to target language uses RNNs, LSTMs, and other neural networks. It is a wise decision to adjust suitable networks and deep learning techniques as this optimizes the system to maximize the translation system's accuracy relative to other systems [23]. The following are deep learning algorithms used in machine learning applications.

Recurrent neural network (RNN)

A specialized neural network with a feedback connection, the recurrent neural network (RNN) processes sequential or time series data by feeding back the output along with new input at each time step. The neural network can remember the previous data when processing the next output thanks to the feedback connection. Because this type of processing is characterized as recurring, the architecture is also referred to as a recurring neural network [59]. A simple neural network with a feedback connection is part of the simple RNN architecture, sometimes referred to as Simple RNN. Because of parameter sharing, which broadens the model's applicability to handle variable-length sequences, it can handle sequential data of variable length. RNNs share the same weights over a number of time steps, in contrast to feed forward neural networks, which have different weights for every input feature. The output of a current time step in an RNN is determined by the preceding time steps and is produced using the same update rule that produced the preceding outputs. The RNN unfolds into a deep computational graph where time steps are shared by the weights [59]. Recurrent Neural Networks (RNN) eliminate the need to specify the context's size N and can represent more diverse patterns. RNNs have input, hidden, and output layers, but they also have a recurrent matrix that connects the hidden layer to itself to enable time-delayed effects, or short-term memory. Recurrent Neural Networks have recently gained popularity in language modeling tasks, particularly neural machine translation (NMT). Recent NMT models are based on Encoder-Decoder, in which a deep LSTM-based encoder projects the source sentence to a fixed dimensional vector, and then another deep LSTM decodes the vector [60]. Figure 7 shows architecture of RNN.

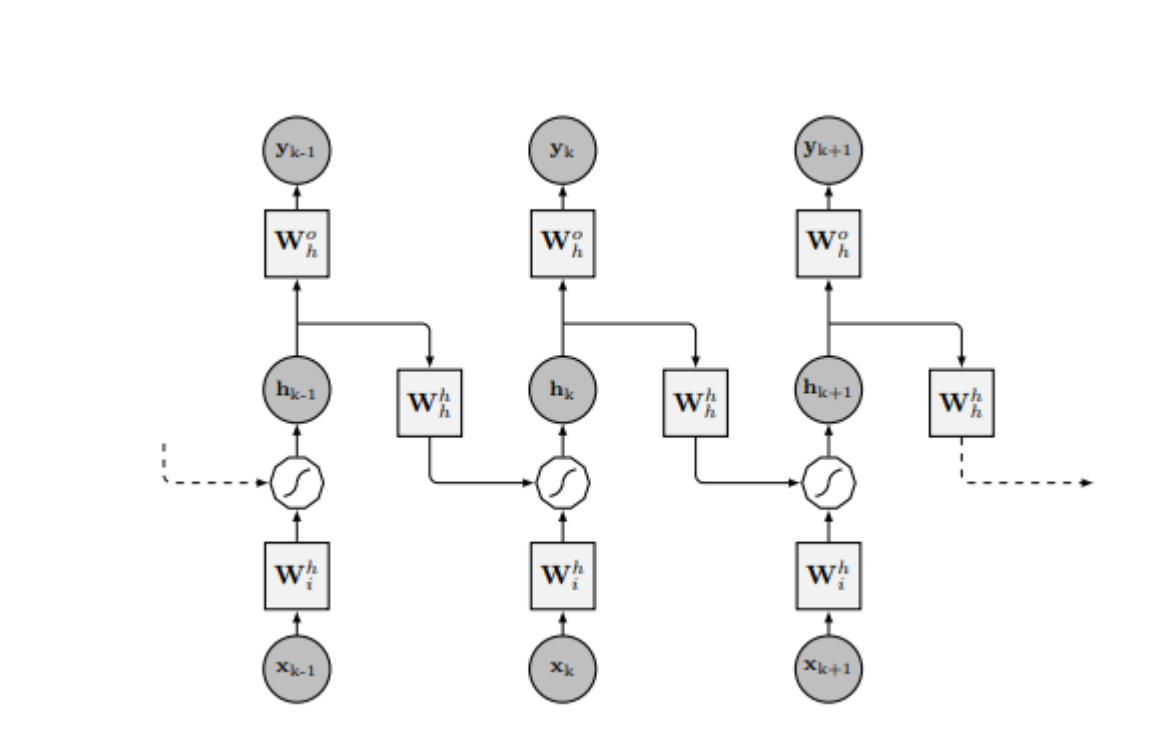

Figure 7 RNN architecture [61]

Long short-term memory (LSTM)

A particular type of recurrent neural network (RNN) architecture called Long Short-Term Memory (LSTM) was created to more accurately model temporal sequences and their long-term dependencies than standard RNNs [62]. Text is viewed as a sequence of words in RNN-based models, which are designed to capture word relationships and text structures. However, traditional RNN models are ineffective and frequently outperform feed-forward neural networks. The most prevalent RNN architecture is the Long Short-Term Memory (LSTM), which is designed to better capture long-term dependencies. By introducing a memory cell to remember values across arbitrary time periods and three gates (input gate, output gate, forget gate) to manage the flow of information into and out of the cell, LSTM addresses the gradient vanishing or exploding difficulties that vanilla RNNs suffer from. RNNs and LSTM models for TC have been improved by capturing additional information, such as natural language tree structures, long-span word relations in text, document subjects, and so on [63]. The external input gate, forget gate, and output gate are the three gates that make up an LSTM. The information that should be deleted from the

cell state is determined by the forget gate at time t and state s_i ($f_i^{(t)}$). Through the use of a sigmoid function σ, the gate sets the weight between 0 and 1, controlling the self-loop. Previous information is kept when the value is close to 1, and it is discarded when the value is close to 0. Following the forget gate, an update is made to the internal state $f_i^{(t)}$. With its own set of parameters, the calculation for an external input gate (g_i^t) is comparable to that of a forget gate using a sigmoid function to produce a value between 0 and 1. A sigmoid unit integrated into the LSTM output gate decides whether to output the value or to turn off the value b_i^t through the output gate q_i^t [59]. The structure of LSTM is shown in Figure 8.

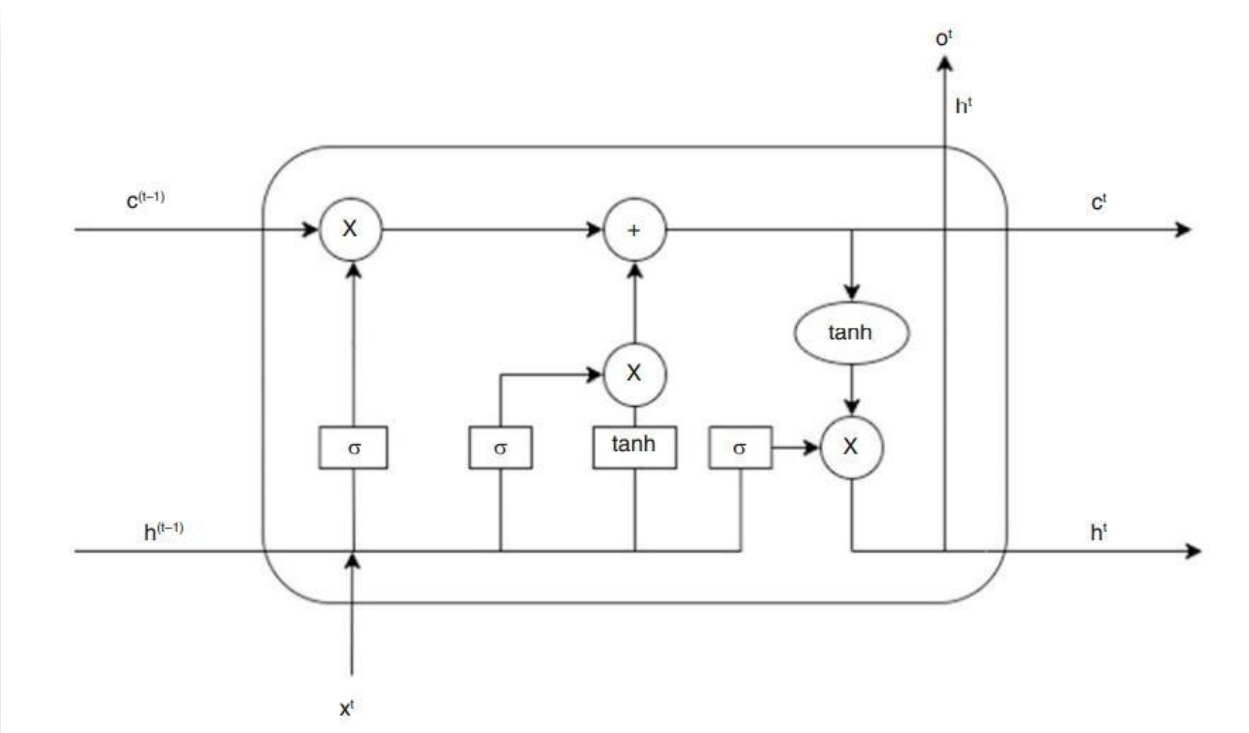

Figure 8 LSTM architecture [59]

Gated Recurrent Neural Network

Because back-propagation in LSTM involves a large number of parameters, computation times are lengthy. The gated recurrent unit (GRU), which shortens computation times, was suggested. Although the architecture of the GRU has been altered, its functionality is comparable to that of the LSTM. Similar to LSTM, GRU uses gating units to capture long-term dependencies in order to solve the vanishing and exploding gradient problem. The reset gate and the update gate are the

two gates that make up GRU. The amount of past data that must be forgotten is determined by the reset gate, and the amount that must be carried forward is determined by the update gate [59]. The structure of GRU is shown in Figure 9.

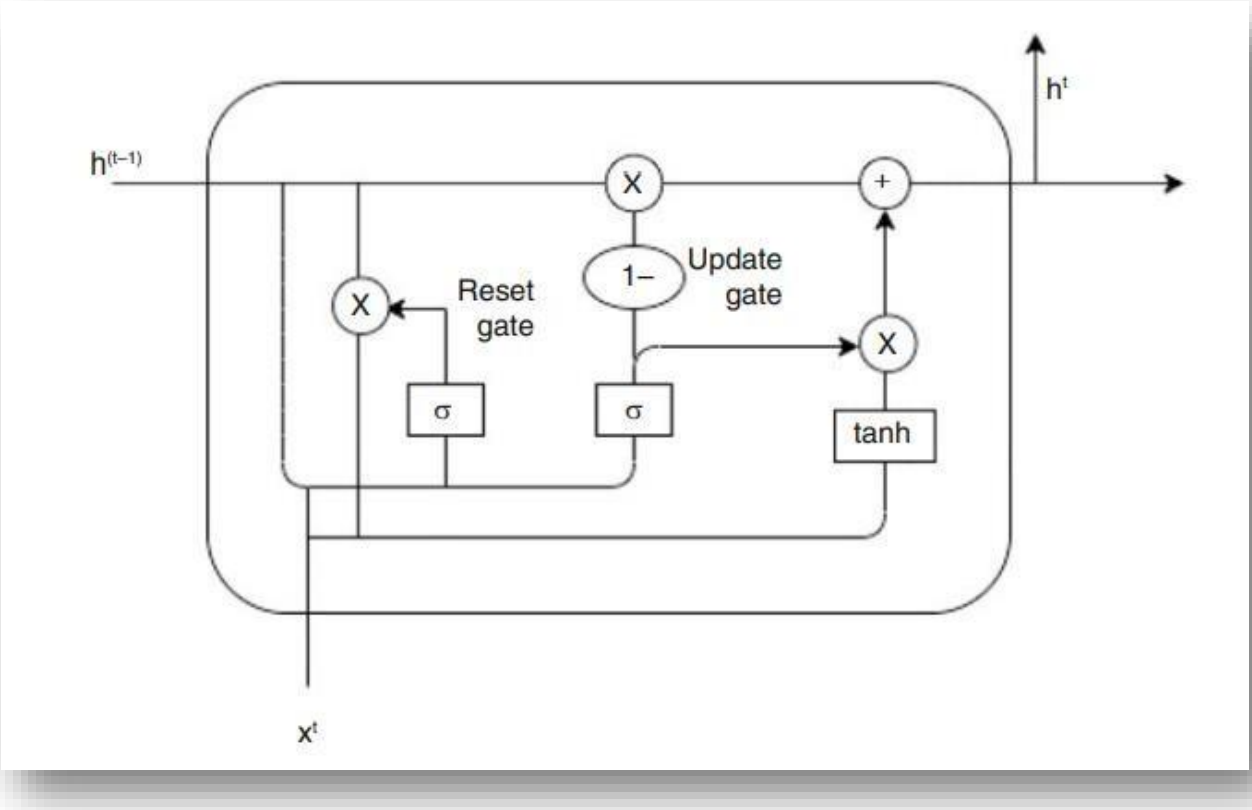

Figure 9 Gated recurrent neural network (GRU) [59]

Bidirectional long short-term memory (Bi-LSTM)

Two parallel LSTMs, one on the input sequence and the other on the output sequence, come together to form a bidirectional LSTM, whose hidden state records both past and future data. The bidirectional LSTM's hidden state is the concatenation of the forward and backward hidden states at each time step [64]. The Bi-LSTM neural network is made up of LSTM units that work in both directions to take into account past and future context. Long-term dependencies can be learned using Bi-LSTM without keeping redundant context information. As a result, it has shown to be quite good at solving sequential modeling problems and is commonly used for text categorization. The Bi-LSTM network includes two parallel layers that propagate in two directions using forward and reverse passes to capture interdependence in two contexts, unlike the LSTM network [65]. The structure of Bi-LSTM is shown in Figure 10.

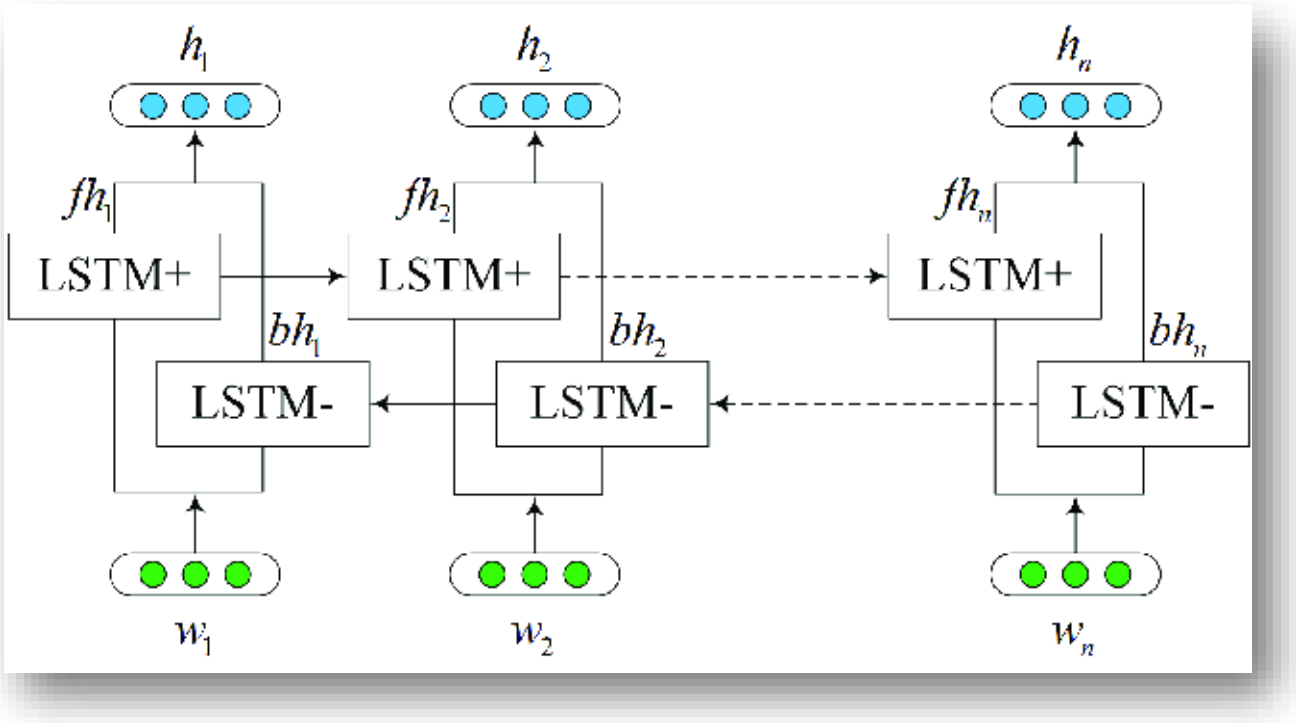

Figure 10 Structure of the Bi-LSTM network [66]

Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a popular deep learning method for handling challenging issues. It gets around the drawbacks of conventional machine learning techniques. The CNN model has received interest due to its performance on different NLP and image processing tasks. Layers with convolving filters are applied to local features in convolutional neural networks (CNN). CNN models, which were first developed for computer vision, have now been proven to be useful for NLP, with outstanding results in semantic parsing, search query retrieval, sentence modeling, and other standard NLP tasks [67]. Convolutional Neural Networks (CNNs) are convolution-based networks that use pooling strategies to get their results. The advantages of CNN over others include parameter sharing, sparse interactions, and similar representations. For using the bi-dimensional structure of input data, local connections and shared weights in the network are used instead of completely connected networks. The CNN is the most well-known and widely used algorithm in the field of deep learning. The fundamental advantage of CNN over its predecessors is that it automatically detects significant features without the need for human intervention [68]. Figure 11 shows architecture of a CNN algorithm.

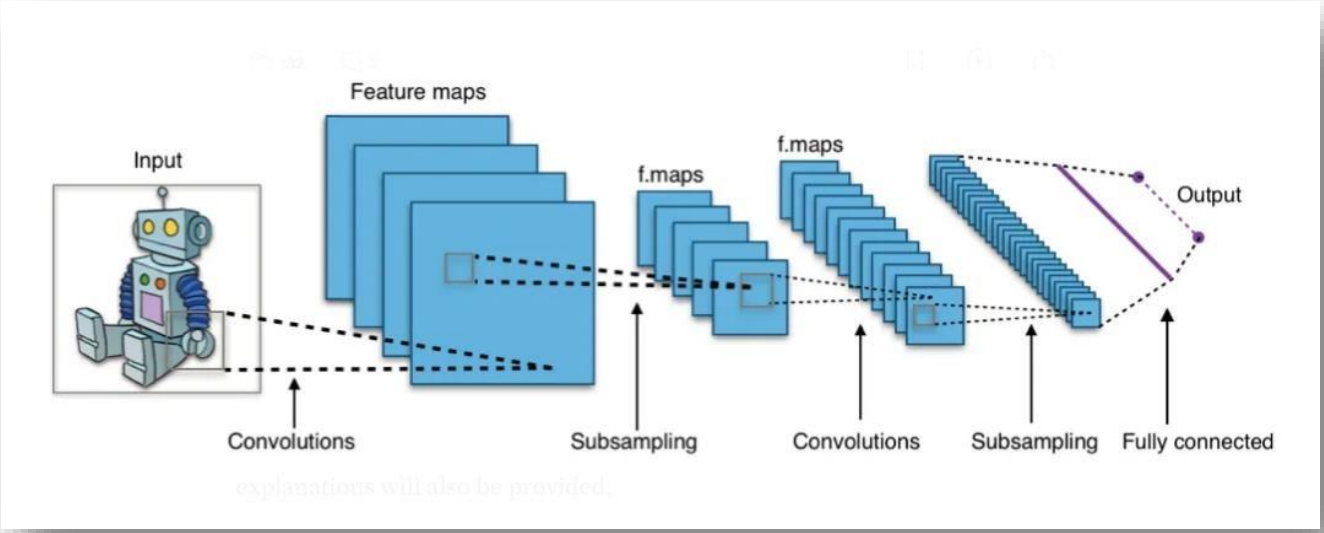

Figure 11 CNN architecture with two channels [69]

Encoder-Decoder Models

When it comes to sequence-to-sequence tasks like machine translation, current machine learning encoder-decoder architectures can achieve impressive performance. Text is converted to a numeric representation using a text encoder. Decoders, as opposed to encoders, unfold a vector that represents the state of the sequence and provide us with meaningful information, such as text, tags, or labels. A sequence of contextualized representations, h_1^n, is produced by an encoder given an input sequence, x_1^n. The context vector, c, functions as a function of h_1^n and provides the decoder with the essential information from the input. Additionally, there is a decoder that takes an input of c and produces an arbitrary length sequence of hidden states h_1^m from which an equivalent sequence of output states y_1^m can be calculated [70]. The architecture of encoder-decoder is commonly used in sequence-to-sequence modeling applications. Encoder-decoder neural networks remain the de facto neural network design for state-of-the-art models in machine translation, despite the transition from long short-term memory networks to Transformer networks as well as the introduction and development of attention mechanisms. Encoder-decoder is still the de facto neural network architecture for state of-the-art models. Sequence-to-sequence modeling is often approached with Neural Networks (NNs), prominently encoder-decoder NNs, nowadays. For the task of Machine Translation (MT), which is by definition also a sequence-to-sequence task, the default choice of NN topology is also an encoder decoder architecture [71]. From a variable-

length input sentence, the encoder extracts a fixed-length vector representation, from which the decoder produces an accurate, variable-length target translation. The task of translation can be understood from the perspective of machine learning as learning the conditional distribution p (f | e) of a source sentence e given a target sentence (translation) f. After a model has learned the conditional distribution, it can be used to sample a target sentence directly given a source sentence, either by actual sampling or by using a (approximate) search algorithm to find the maximum of the distribution [72].

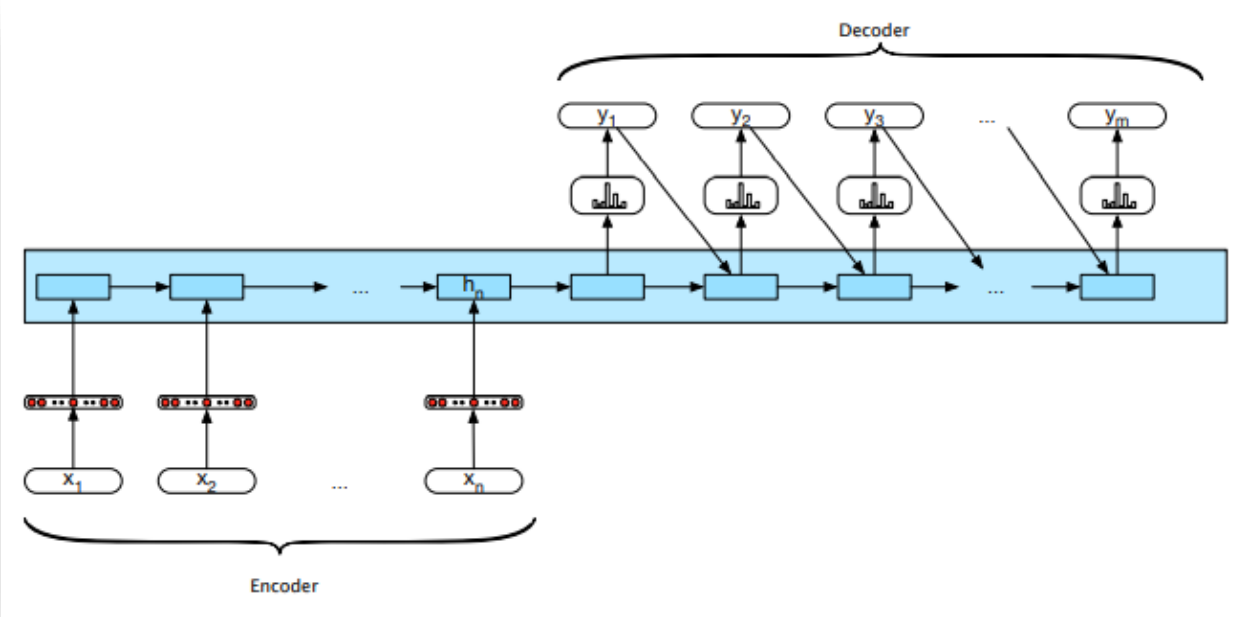

Figure 12 The encoder–decoder architecture [70]

Attention Mechanism

In order to enhance NMT performance, attention techniques were first developed to teach the alignment between source and target tokens. The classic word alignment in SMT, which learns the hard alignment between source and destination tokens, is different from the attention techniques. When creating a target token, attention mechanisms learn to take features from every source token. All of the concealed states of the source tokens are given weights. Larger weights are given to the concealed states that are more connected. Following that, attention mechanisms provide the decoder a context vector c_t that was retrieved from the encoder for target-side predictions [73].

The hidden state is represented with h and set is $\{h_1, h_2, \cdots, h_n\}$ in the encoder, where n is the number of source-side tokens. The context vector c_t is computed using

$$c_t = \alpha_t h$$

where α_t is the attention vector at time step t. α_t is a normalized distribution of a score computed by the hidden state set h and the decoder state s_t-1, as described by Equation 2:

$$\alpha_t = \text{softmax}(\text{score}(s_t-1, h))$$

The fact that early NMT models frequently provided inadequate translations for lengthy sentences is one issue that has yet to be fully resolved. The fixed-length source sentence encoding is the cause of this flaw. Sentences of various lengths transmit information in different ways. A fixed-length vector cannot therefore adequately capture a long sentence with a complex structure and meaning, even though it is fine for small sentences [74].

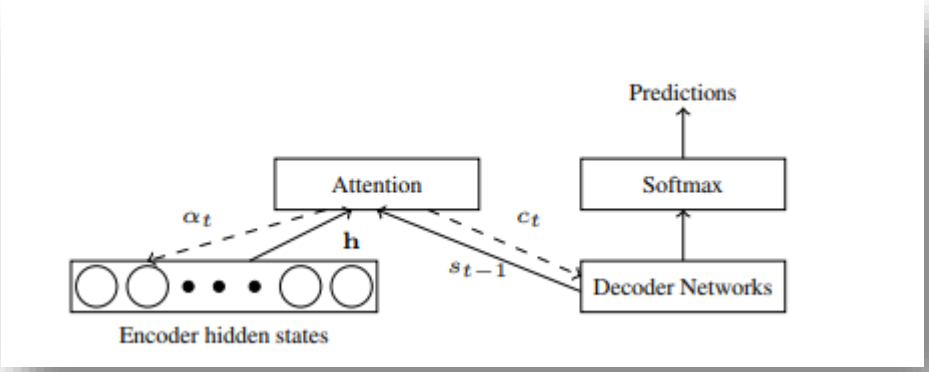

Figure 13 Vanilla attention mechanism [73]

Fully attention-based NMT has recently demonstrated promising performance. In particular, the attention mechanism has operated as a driving force rather than an assistant in text feature extraction. Transformer, which is a completely attention-based paradigm, is one of them. Transformer is a fully attention based NMT model, in contrast to earlier RNN- or CNN-based models. It can be a feature extractor that allows the complete sentence to be "read" and modelled once, meaning it is of self-attention with a feed-forward link. Multiple layers are frequently stacked, which improves the quality of the translation [18]. Figure 11 shows the architecture of the transformer.

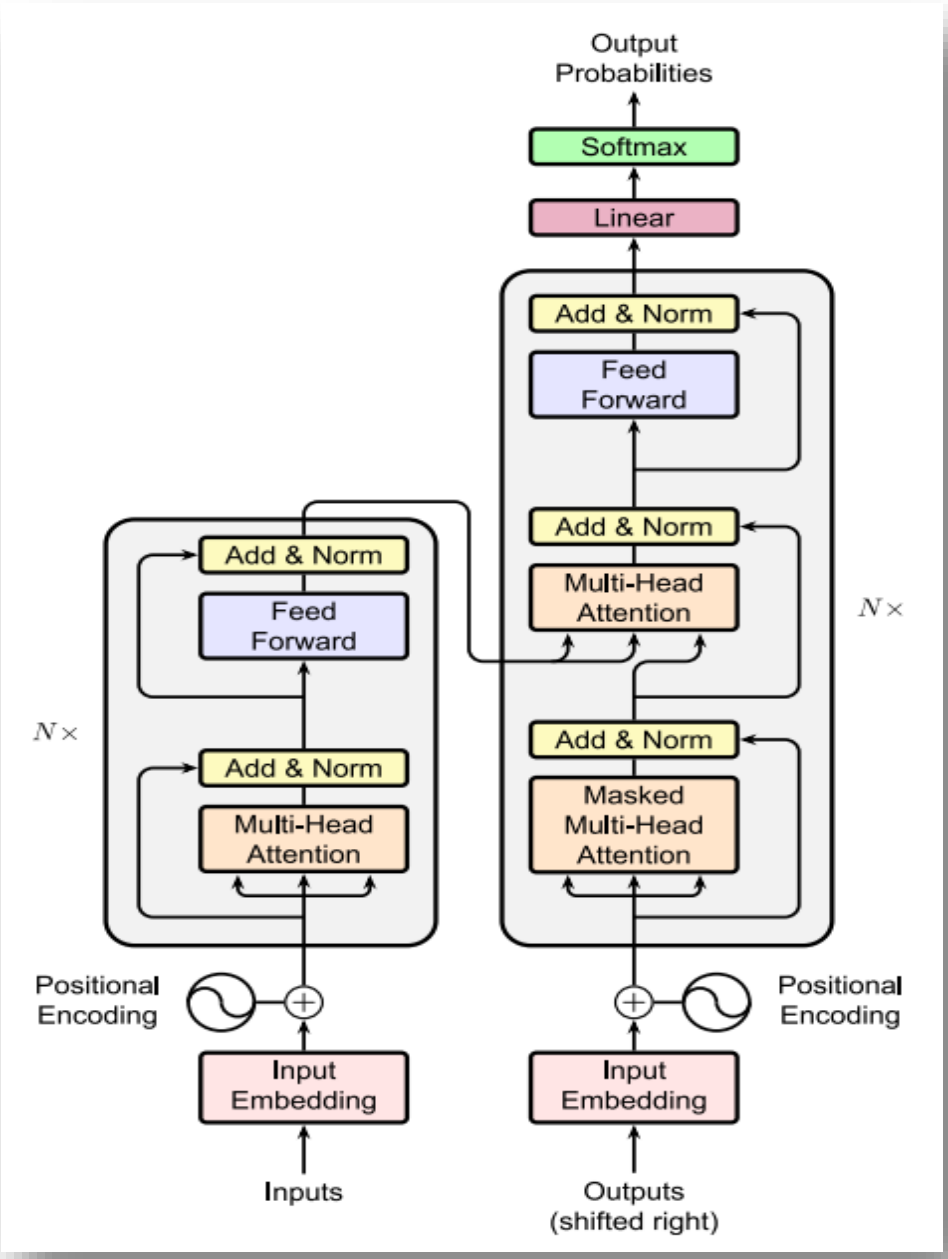

Figure 14 Transformer architecture [75]

2.7. Related work

Machine translation has been studied for many foreign languages and some Ethiopian languages. Some of the Ethiopian languages that have resources on the web such as Amharic, Oromo and Tigrigna has been explored for machine translation tasks. Previous researchers has employed different approaches of machine translation such as rule based and statistical based machine translation techniques for translation of Semitic languages.

Previously there are some studies conducted on machine translation of Tigrigna and English languages. The study [4] proposed a statistical machine translator for English to Tigrigna translation. However the study were only one directional which is English to Tigrigna languages only. On the other hand the study [7] proposed English to Tigrigna translation using neural machine translation. Again the study were only one directional which is English to Tigrigna languages only. In the study by [15] Tigrigna neural machine translation were proposed. The study focused on Tigrigna to English machine translation not the reverse using transfer learning. The study were conducted for domain specific case of humanitarian response. This makes the model developed in the study limited to humanitarian response domain only.

In the study [16] a bidirectional English-Tigrigna machine translation were proposed. The study used a hybrid approach of statistical approach and post-processing technique. Even though the study reported good performance of their approach, the model is limited to four domains only. Based on the literature review the author collected a total of 32,000 bilingual dictionaries and roughly 12,000 parallel sentences from four domains for the experiment, and two language models—one for Tigrigna and the other for English—were created. 1,200 sentences were selected at random for testing and 10,800 sentences for the training set from the parallelly collected corpus. The experiment used the Moses open source statistical machine translation system for training, tuning, and decoding. The Giza++ toolkit was utilized to align the parallel corpus, and SRILM was employed to construct the language model.

In the study [17] a bidirectional Tigrigna – English machine translation using statistical machine translation approach were conducted. However SMT technique may disregard the extended dependency that exists beyond the length of phrases resulting an errors in translation outcomes such as gender agreements that are wrong. Separate components, such as word aligners, translation rule extractors, and other feature extractors, are also affected [18]. In the study [19] English -

Tigrigna factored statistical machine translation were conducted. The study used statistical machine translation and it was English to Tigrigna translation and not the reverse.

The work of [76] proposed parallel corpora for bi-directional statistical machine translation for seven Ethiopian language pairs including Tigrigna. The study employed statistical based machine translation (SMT). However SMT has a limitation of handling long term sentences dependencies. The authors reported that their system performs less for Ethio-Semitic language family. The authors recommended ANN modelling as an attractive solution to the problems of machine translation that is the trend of the time.

The study by [77] proposed an Amharic-Tigrigna machine translation using the SMT approach. The author prepared the text corpus for Amharic-Tigrigna machine translation system from religious domain specifically from bible. The author recommended to enhance the performance of translator using other techniques. The author [77] proposed morpheme based bi-directional machine translation for Ge'ez to Tigrigna languages. The author experimented their model using a dataset collected ten bible books. The author used GIZA++ which is a statistical machine translation toolkit. The author reported as their experiment achieved a BLUE score of 9.23 % from Tigrigna to Ge'ez and 8.67% Ge‟ez to Tigrigna BLEU score respectively and recommend increasing the size and domain of the data set used for training the system for better results.

The study [78] proposed Ge'ez Amharic machine translation using deep learning. The dataset for the study were collected from Bible and religious documents. The study reported that the performance of their system is much lower. This is because the study employed LSTM algorithm which has a drawback of handling long term dependencies that exist in a text. The dataset also makes the model limited to the domain where the dataset is collected. The study [79] proposed a morpheme based Ge'ez Amharic machine translation. The study used SMT which is based on morpheme and word level translation as a technique. However SMT approach may disregard the extended dependency that exists beyond the length of phrases and new words.

Studies also show that Neural Machine Translation (NMT) are better as compared to SMT systems. NMT is an end-to-end learning strategy for automated translation that has the potential to solve many of the flaws of traditional phrase-based translation systems. Unfortunately, both in training and translation inference, NMT systems are known to be computationally expensive – often excessively so in the case of very big data sets and complex models. NMT systems have also been

accused of being unreliable, particularly when input sentences contain unusual terms. NMT's application in actual deployments and services, where both accuracy and speed are critical, has been hampered by these limitations [80].

In the study [81] English Amharic machine translation using SMT were proposed. From the study it is shown that the task of developing MT for Amharic using a rule-based approach, which is considered one of the NLP scarce resource languages, is enormous. The same may not be true for languages with well-developed NLP resources. The rule-based MT strongly leverages integrated linguistic knowledge, rules, and resources of both the source and target languages, which makes it difficult for under-resourced languages. The source and target languages' linguistic knowledge includes tagging, parsing, morphological, syntactic, semantic, and lexical information. The linguistic rules include rules for analyzing, transferring (including syntactic, semantic, and lexical) the source and/or target languages, as well as rules for generating them.

An Amharic English machine translation were proposed in the work of [76]. From the work, we observed that for SMT the linguistic characteristics of the target languages have a significant impact on translation. The difficulties include everything from the writing system to word ordering and morphological sophistication. Different characters are employed in words that express the same meaning in the Ge'ez writing system, which is used by Amharic, Tigrigna, and Ge'ez languages. Peace, for example, can be written as: ሰላም or ሠላም. Such character differences have an impact on probability values, which have a direct impact on SMT performance.

CHAPTER THREE

3 RESEARCH METHODOLOGY

3.1. Introduction

In this chapter the methodology, detailed architectural design and implementation of the bidirectional Tigrigna-English machine translation model is described. We collected the Tigrigna-English parallel dataset from freely available online sources.

3.2. Research design

We used an experimental research design technique to analyze the result of our experiments. We used our collected data to undertake various experiments. With a scientific methodology, experimental research is carried out with dependent and independent variables. We chose an experimental research approach so that we could easily observe the impact of some variables on other variables and investigate potential causes and effects. In our case various model parameters are the variable we employed.

3.3. System design and architecture

In this study we used the encoder decoder and attention based machine translation approaches. In Figure 15 the proposed model architecture is shown and each step in the architecture is described in the following sections.

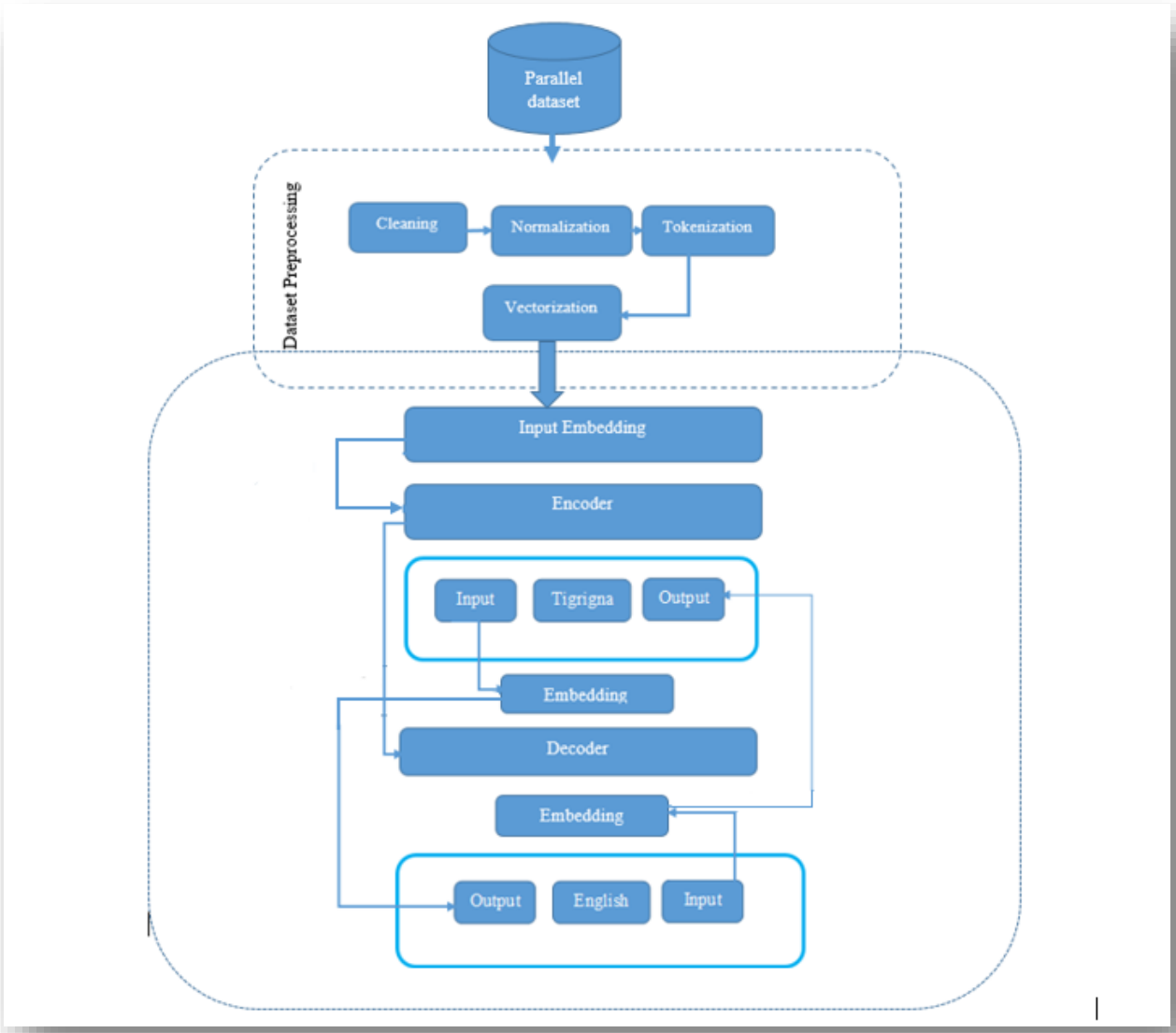

Figure 15 proposed model architecture

3.3.1. Dataset collection

We collected the Tigrigna English dataset from different sources like online repositories. Some of our dataset sources are Github, and the Translators without borders. The dataset contains two files one for Tigrigna and the other English text. In addition to the existed dataset, we have translated freely available English text corpus to Tigrigna.

3.3.2. Dataset preprocessing

In the preprocessing phase we make our dataset ready for our model's later processing tasks. During the text preprocessing stage of natural language processing, a text is transformed into a machine-readable format to facilitate the operation of learning algorithms. In our study we did preprocessing stages that includes cleaning text, normalization, and tokenization.

Dataset cleaning

Text cleaning involves removing unnecessary spaces, stop words, punctuation, special characters and numbers. Punctuation is not included in the corpus and is not particularly relevant to translation. In this stage, we removed these unnecessary spaces, punctuation and special characters. The algorithm we used to remove these characters is shown below in Appendix B and list of stop words in Appendix D.

Normalization

The process of mapping various word variants to a single string is known as normalization in NLP. Normalization helps in dataset preparation by converting words into a common format. For instance, in the Tigrigna language, all words "ሀ," "ሐ," and "ሠ" , "ሰ" can be represented as a single entity, allowing text representation techniques to provide comparable representations for words that are similar. The algorithm used to normalize the dataset in our study is shown in Appendix B.

Tokenization

The process of breaking down a given text into individual words, sentences, and subwords is known as tokenization. It is lexical analysis method used to separate sentences into their constituent tokens. When we performed tokenization, we found the vocabulary size, the maximum length of sequences, and the representation of words with unique numbers. In order to assign a distinct number to each word, we read the entire parallel dataset and a unique number is given to each word. With tokenization process the whole dataset is translated to integer form. The following Tigrigna English sentences shows how tokenization is done in our study. “ንሳ ተምሃሪ እዩ” tokenized as ‘ንሳ’, ‘ተምሃሪ’, ‘እዩ’. The Figure 18 below shows sample of the tokenization stage of dataset preprocessing for Tigrigna sentences.

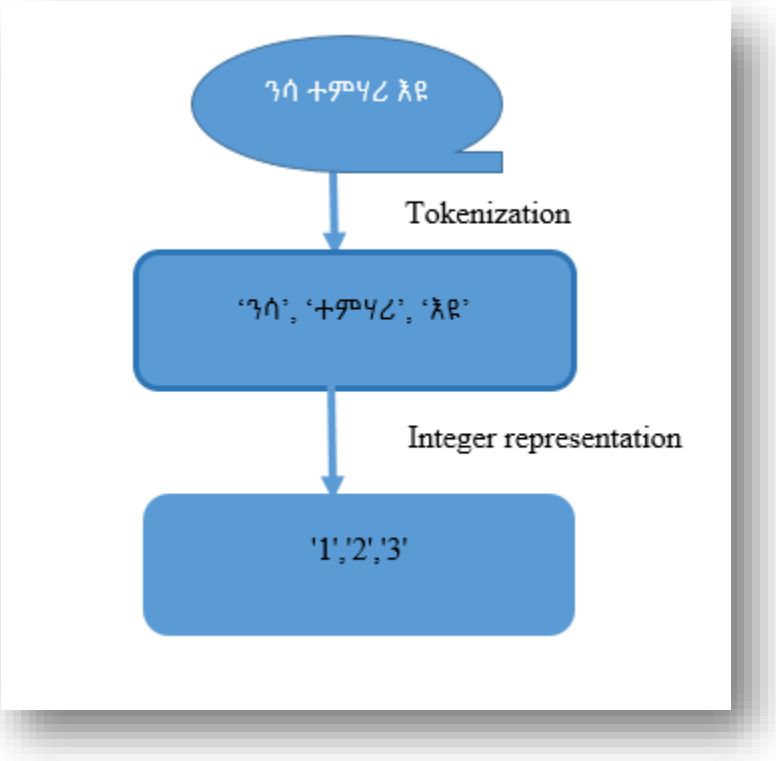

Figure 16 Tokenization of Tigrigna sentence

Similarly the corresponding English sentences is also tokenized the same way. “She is a student” is tokenized as 'she ', 'is', 'student'. However, in order to know the boundary of end of the sentences we used <sos> to indicate the beginning of a new sentence and <eos> to indicate the end of the sentences like ‘<sos>’, 'she' ,'is' ,'student' <eos>’. The Figure 19 below shows sample of the tokenization stage of dataset preprocessing for Amharic sentences.

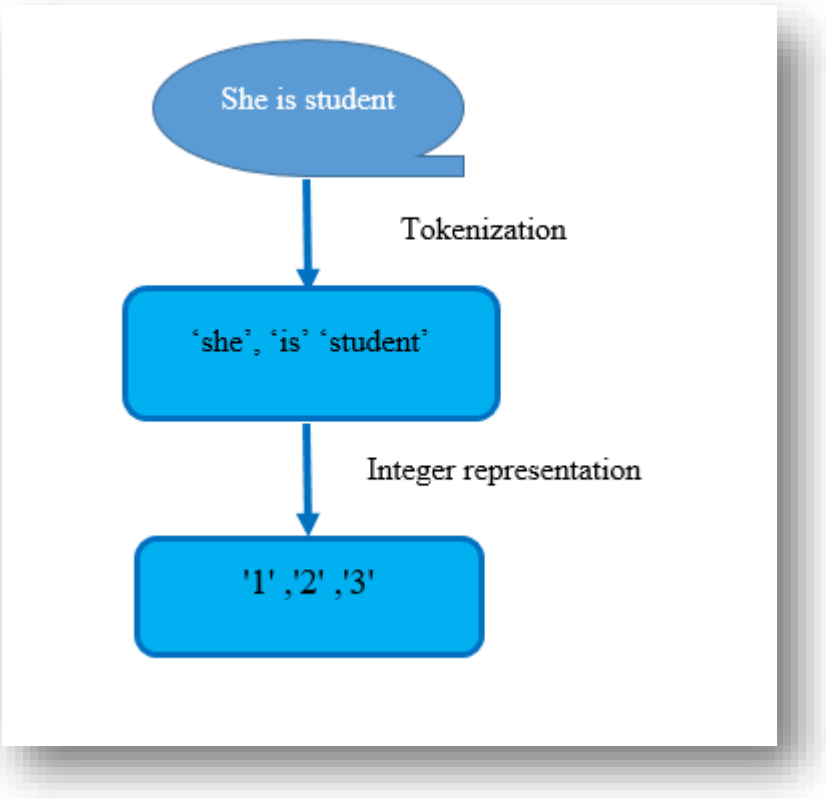

Figure 17 Tokenization of English sentences

The algorithm we used to tokenize the dataset is shown below in Appendix B.

Vectorization

In the tokenization stage the dataset is changed in to integer representation. However the integer representation cannot be directly input for the translation model. This is because the neural network does not directly operate on the vocabulary represented with integer. In order to operate the vocabulary by the neural network it should be changed in to vector representation, which is called one hot vector representation. In this stage the integer represented data is changed in to two-dimensional vector which is, one-hot vector representation. The one hot vector representation uses unique vector representation for each word of the sentences.

3.3.3. Model training

3.3.3.1. Input embedding

Embedding is the process of creating an initial representation for every word in the input language that corresponds to its corresponding numerical value for processing. The encoder layer's bottom is where the embedding step takes place. Each word passes through both of the encoder's layers

after being embedded in the input sequence. Subsequently, the encoder creates the key, query, and value vectors for every word in Transformers using the embedding. Next, a list of vectors based on [82] is sent to the encoder with size the length of the longest sentence in the training dataset.

3.3.3.2. Encoder decoder model

Encoder

Encoding is the process of transforming a given data into the required format. The encoder creates an internal representation called a context vector from the input, which the decoder utilizes to generate the output sequence in a typical Seq2Seq encoder-decoder model. The lengths of the input and output sequences can vary since there isn't a definite one-to-one correspondence between them. Here, a stack of RNN, LSTM, or GRU units makes up the encoder and decoder. It operates in two phases. First, the LSTM in the encoder processes the entire input sentence and encodes it into a context vector. The context vector then becomes the final LSTM or RNN's hidden state. The input sentence should be accurately summarized in this way. The final state is interpreted as the decoder's initial hidden state, and all other encoder intermediate stages are ignored. The main problem with this approach is an event: if the encoder generates an incorrect context vector, the translation will be inaccurate. The encoder produces a poor summary when it attempts to understand longer sentences. The RNN or LSTM long-range dependency. The encoder architecture used in this case is shown in Figure 21.

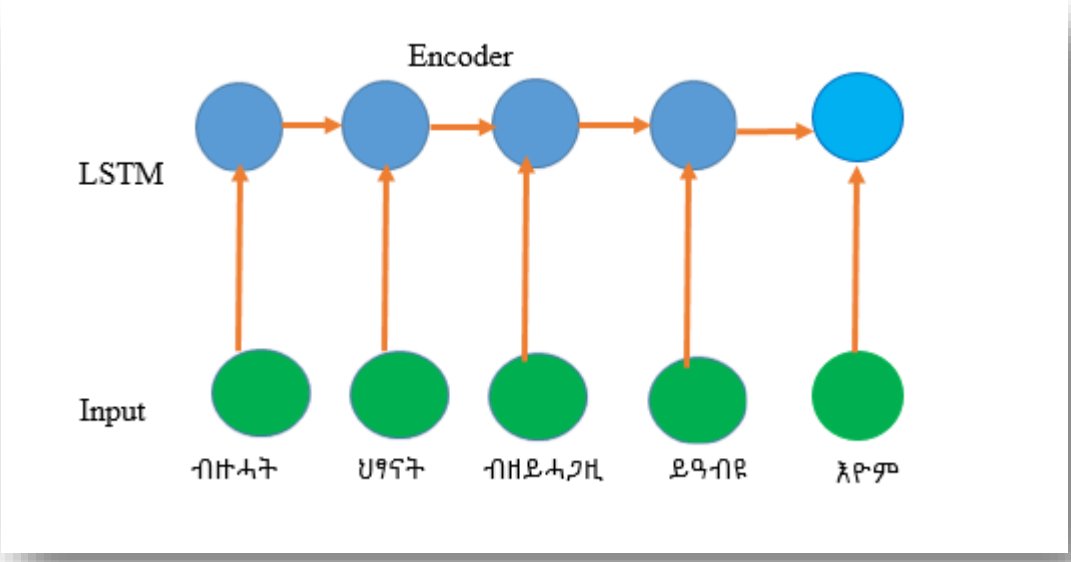

Figure 18 the encoder architecture

Decoder

Like the encoder, LSTM (or occasionally GRU or Bi-LSTM) models make up the decoder of the standard Seq2Seq architecture. Additionally, the final hidden states of the encoder are initialized into this decoder's first state. The decoder begins creating the output sequence using these initial states in such a way that the input it receives at each time step is the output from the time step before. By doing this, the encoded meaning of the input sentence is transferred to the decoder and converted into a sentence in the target language. The translated sentence produced by the Decoder, in contrast to the Encoder, will have a variable length. As a result, until it produces a whole sentence, the Decoder will output a prediction word at each time step. To begin, input a <sos> tag as the input for the Decoder's first time step. The Decoder will update its hidden state using the input at time-step t=1, just like the Encoder does. Nevertheless, the Decoder will employ an extra weight matrix to generate a probability over each word in the output vocabulary rather than simply moving on to the next time step. The output vocabulary word with the highest probability will be the first word in the predicted output sentence, and this process will continue until the prediction of the <eos>. Figure 22 below shows the decoder architecture described above.

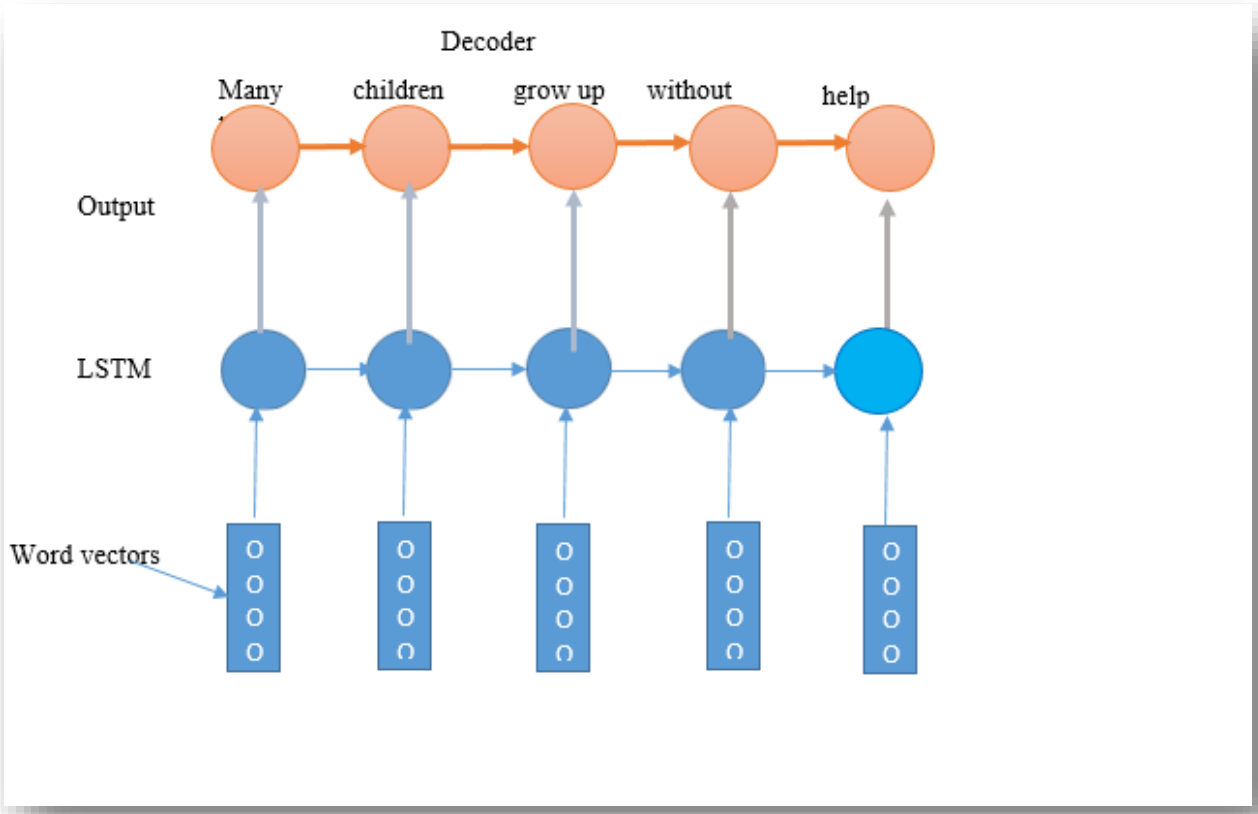

Figure 19 the decoder architecture

3.3.3.3. Attention based model

In our research we implement the Bahdanau Attention [83] layer. The main distinction between this method and the fundamental encoder-decoder is that it avoids attempting to compress the entire input sentence into a single fixed-length vector. Instead, it encodes the input sentence into a series of vectors, and then, while decoding the translation, it arbitrarily selects a subset of these vectors. The attention layer helps the decoder to access the data that the encoder extracted. The attention layer creates a vector out of the whole context sequence and adds it to the output of the decoder. A single vector from the full sequence is calculated by taking the average over the sequence. Similar to a context layer, an attention layer computes a weighted average over the context sequence. Then the context and "query" vectors are used to generate the weights. Figure 26 shows encoder decoder with attention mechanism for this case.

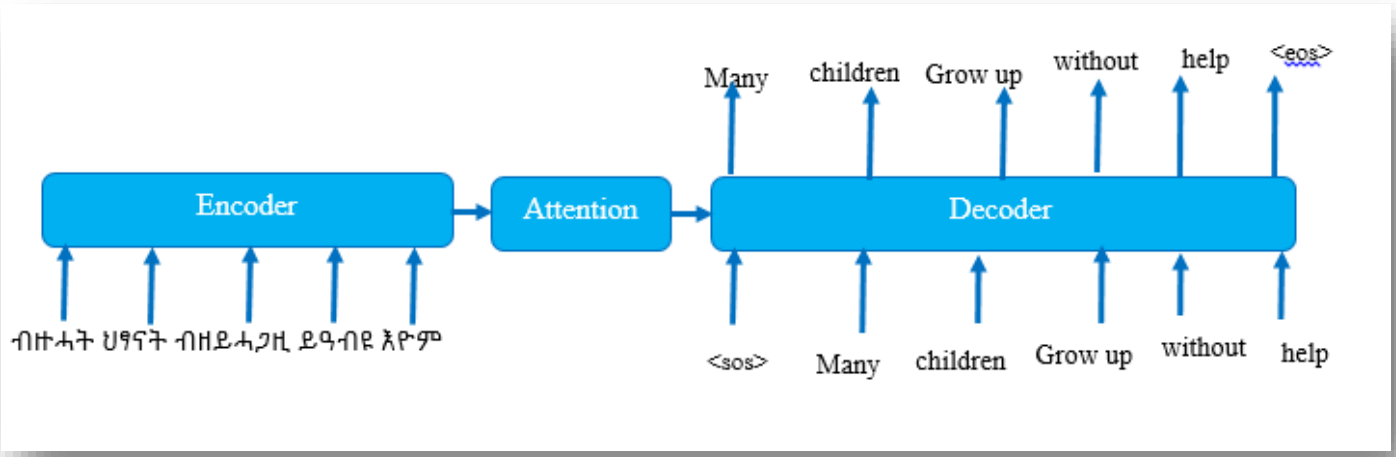

Figure 20 attention mechanism

3.3.4. Model evaluation

In this study, we used the Bilingual Evaluation Understudy (BLEU) score to assess the performance of the proposed model. BLEU score can automatically evaluate the machine translation model. The text's quality is determined by comparing the results of machine translation to those of human translation. If the machine translated text is closely related to the human translated text, the result is then considered to be higher quality, and the model is effective.

3.3.5. Development toolsPython

Python is a high-level, interpreted, dynamically semantic programming language. Because of its high-level built-in data structures, dynamic typing, and dynamic binding, it's perfect for scripting or as a language to connect existing components while doing Rapid Application Development. Python's short, simple-to-learn syntax encourages readability, which reduces the cost of software maintenance. Python supports modules and packages, which makes code reuse and program modularity easier. We wrote our software in the Python programming language.

Jupyter notebook

The Jupyter Notebook App is a web-based server-client application for editing and running notebook documents. The Jupyter Notebook App can be run locally on a computer without internet access (as explained in this paper) or remotely on a server and accessible via the internet. We used jupyter notebook to write our python code.

Keras

Keras is a Python-based high-level neural network library that can be used with either TensorFlow or Theano. It is critical to be able to move quickly from idea to result when conducting research. We used Keras to train our deep learning model.

Anaconda

Aimed for streamlining package management and deployment in scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, and so on), Anaconda is a Python and R programming language distribution. The bundle includes data-science packages for Windows, Linux, and macOS. This platform was utilized by the researcher to create the suggested model.

Matplotlib

Matplotlib is a plotting library available as a component of NumPy, a big data numerical handling resource, for the Python programming language. Matplotlib embeds plots in Python applications using an object-oriented API. We used this library to visualize the performance of the model during different experiments. We visualize the accuracy and loss of the model using different algorithms.

TensorFlow

TensorFlow is a complete open source machine learning platform. It has a comprehensive, adaptable ecosystem of tools, libraries, and community resources that enable researchers to push the boundaries of ML and developers to easily build and deploy ML-powered applications. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation)-based simulations, competing with frameworks such as PyTorch and Apache MXNet. Best of all, TensorFlow can predict production at scale using the same models that were used for training.

CHAPTER FOUR

4. EXPERIMENTATION AND RESULTS

4.1. Introduction

This chapter discusses the experimental results by showing experimental setups and performance of testing results of the systems using BLEU score metrics. The dataset, experimental setup, evaluation processes and results are described in detail.

4.2. Dataset Collection and Preparation

We collected the Tigrigna English dataset from different sources like Github, and the Translators without borders. In addition to this we have translated freely available English text to Tigrigna. The translator for our new dataset were English teachers whose mother tang is Tigrigna. We have translated 2k English sentences to Tigrigna. We have generally collected and prepared 31k English- Tigrigna parallel sentences.

Figure 25 shows the sample of prepared Tigrigna-English machine translation dataset.

Out[46]:

	english	tigregna
0	Is that your new friend?	ሓዲሽ ዓርከኻ /መሓዛካ ድዩ?
1	Jacob wasn't interested in baseball.	ያዕቆብ ኣብ ሻኩ ይግደስ ኣይነበረን።
2	Adam told me that Alice had a new boyfriend.	ኣሊስ ሓዲሽ ዓርኪ ከምዝሓዘት ኣዳም ነጊሩኒ።
3	The radio didn't inform about the accident.	እታ ሬድዮ ብዛዕባ እቲ ሓደጋ ኣይሓበረትን.
4	Adam is worried we'll get lost.	ኣዳም ክይንጠፍእ ተሻቒሉ ኣሎ።
...	...	...
495	Why in the world would you want to live in Aus...	ስለምንታይ ኣብ ኣውስትራልያ ኽትነብር ደሊኻ?
496	Carmen said she didn't want to talk about work.	ካርመን ብዛዕባ ስራሕ ክትዛረብ ከም ዘይትደሊ ተዛሪባ።
497	I wonder whether Harry and Laur were able to d...	ሃሪን ላውርን ነዚ ኽንብርዎ ዝኽእሉ እንተነይሮም ርግጸኛ ኣይኮንኩን።
498	Harry and Mary said that's not what they neede...	ሃሪን ሜሪን እቲ ኽንብርዎ ደልዮም ዝነበሩ፡ እዚ ኸም ዘይኮነ ተዛሪቦም።
499	I can't face the truth.	ነቲ ሓቂ ክቕበሎ ኣይከኣልኩን።

Figure 21 sample of the dataset

4.3. Experimental setups

The experimentation for this research is done on windows operating system. The machine we used has 8GB of RAM and a GPU NVIDIA GTX 770, which helps us to process experiments with a

short training time. To build our system, we used the Python programming language along with the Keras, TensorFlow, NumPy, and libraries. The experiment is done using 31k Tigrigna-English parallel sentences. During experimentation, the researcher used 80 to 20 percentages of dataset proportion as a train test splitting ratio. The researcher used 15 percent of the training dataset as a validation set. In addition to the train test split ratio, the researcher used different experiment setups with respect to different dropout values, number of neurons in layers, batch sizes and number of epochs.

4.4. Parameter Selection

To achieve the desired result, we conducted various experiments on various parameter combinations using our training data. We started by selecting embedding dimension, which we selected the embedding dimension 64, 128 and 256. In order to select the best parameters using the training data, we have done the experiments using different combinations of embedding dimension, optimizers, dropout rates and activation functions. Finally, we got the following parameter combination with best results specified in the Table 1. These parameter combination results almost the same performances, only slightly different.

Table 2 Parameters

Parameter	**Setup 1**	**Setup 2**	**Setup 3**
Batch size	30	60	128
Number of Epoch	50	100	120
Dropout	0.2	0.2	0.5
Learning rate	0.002	0.002	0.005
Optimizer	Adam	Adam	Adam
Embedding dimension	64	128	256
Number of neuron	64	64	64
Activation function	Softmax	Softmax	Softmax

From the above experiments, the best result were scored with parameter combinations of 128 embedding dimension, 100 epochs, 60 a batch size and 0.002 learning rate. The time taken for the experiment with this parameter combinations were around 4 hours.

4.5. Performance evaluation

4.5.1. Training and validation accuracy of English to Tigrigna translation model

In this section, the researcher discussed the performance of the proposed model using the training and validation accuracy during the training phase. All the performance detail discussed in this section is for English-Tigrigna translation.

In Figure 29, the training and validation accuracy of the encoder decoder model using the LSTM algorithm is shown. As shown in the Figure 29, the model performs 64.21% and 63.1 % training accuracy and validation accuracy respectively.

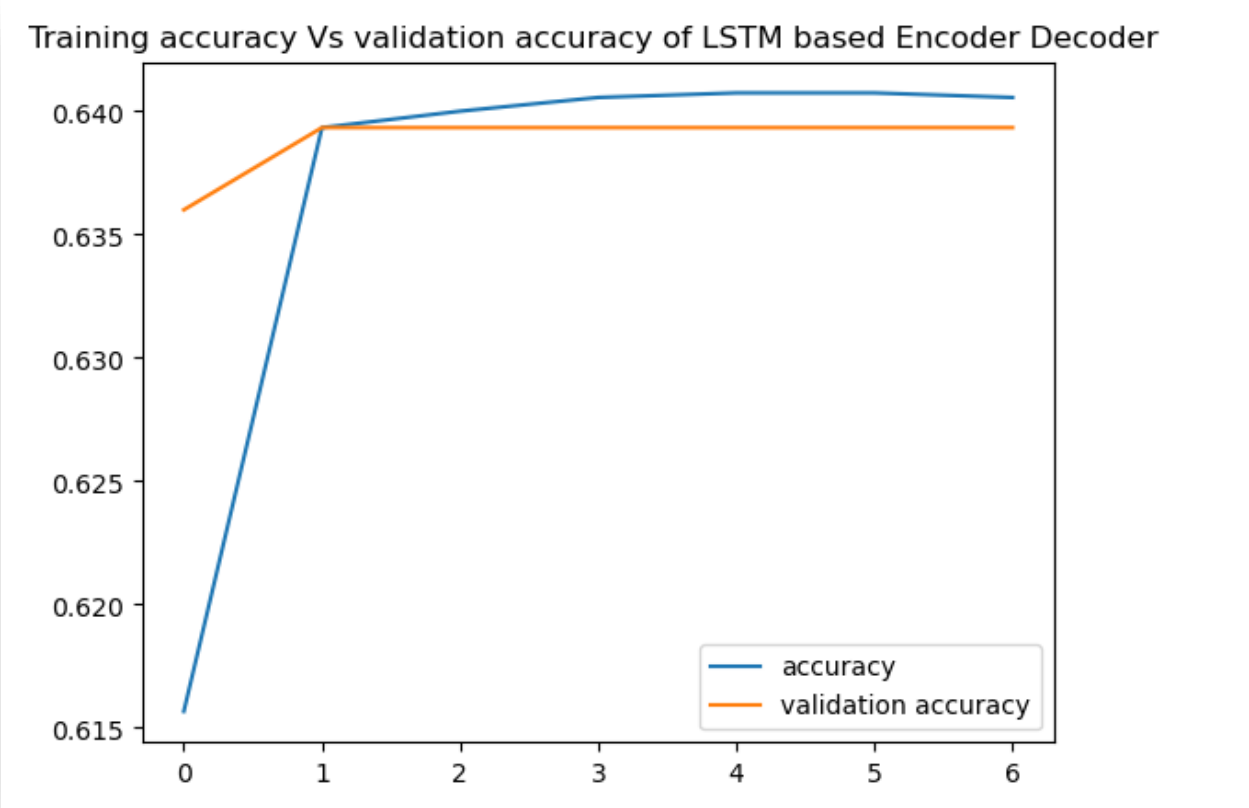

Figure 22 training Vs validation accuracy of the encoder decoder model using the LSTM

In Figure 29, the training and validation accuracy of the encoder decoder model using the GRU algorithm is shown. As shown in the Figure 29, the model performs 64.5% and 64 % training accuracy and validation accuracy respectively.

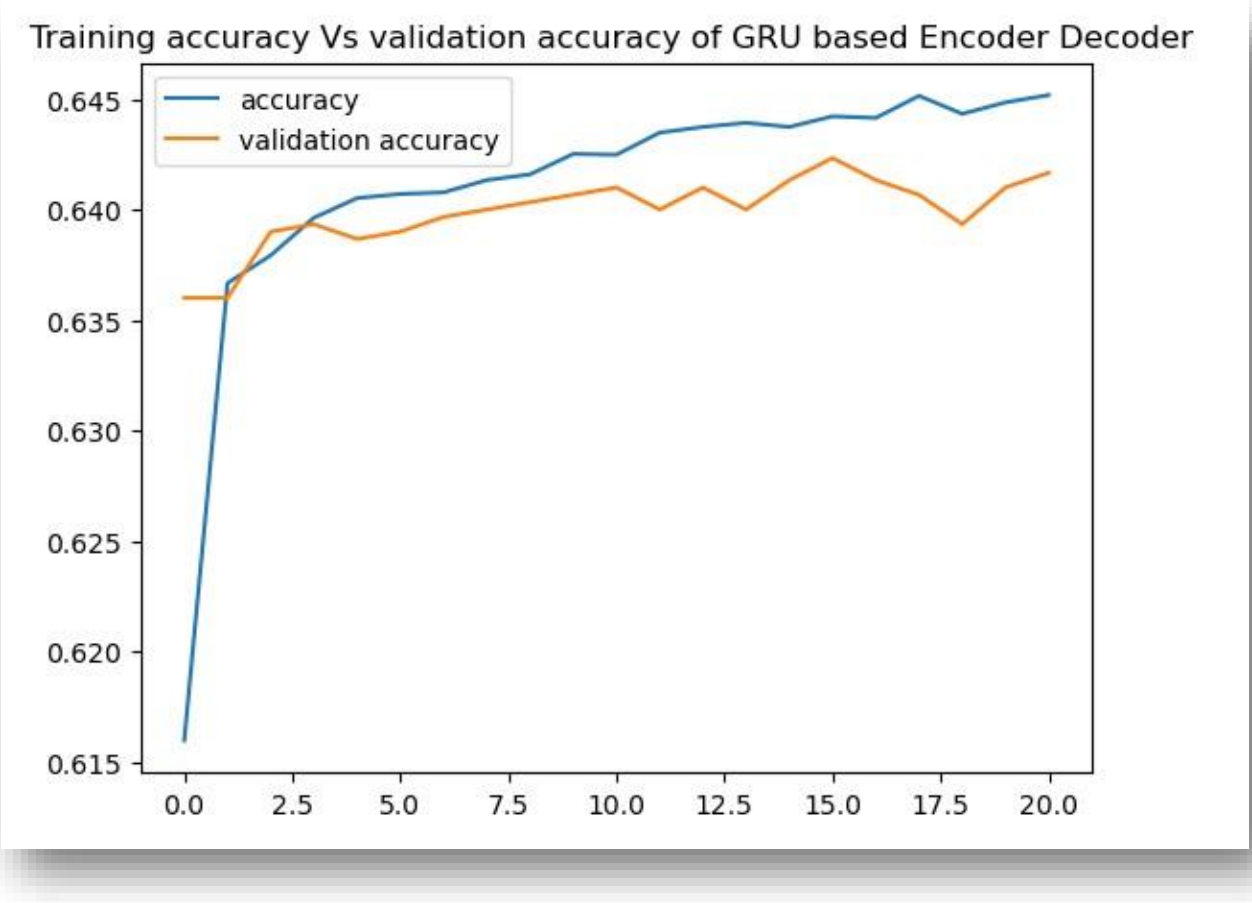

Figure 23 training Vs validation accuracy of the encoder decoder model using the GRU

In Figure 29, the training and validation accuracy of the Attention based LSTM model. As shown in the Figure 29, the model performs 53.21% and 50.9 % training accuracy and validation accuracy respectively.

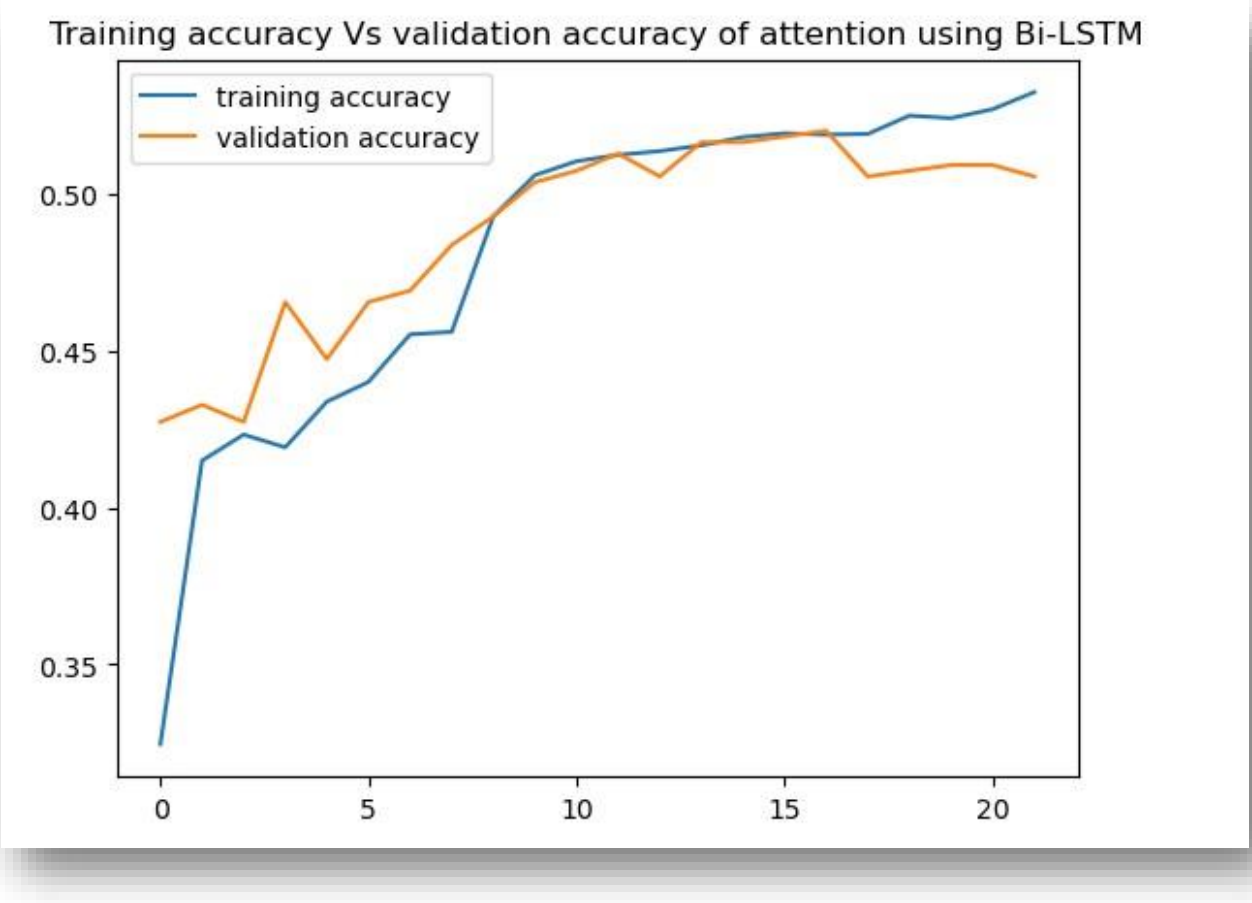

Figure 24 Training Vs validation accuracy of attention using Bi-LSTM

4.5.2. Training and validation accuracy of Tigrigna to English translation model

In this section, the researcher described the training and validation accuracy history of the proposed translation model for Tigrigna to English translation.

In Figure 25, the training and validation accuracy of the encoder decoder model using the LSTM algorithm is shown. As shown in the Figure 25, the model performs 68.21% and 67.2 % training accuracy and validation accuracy respectively.

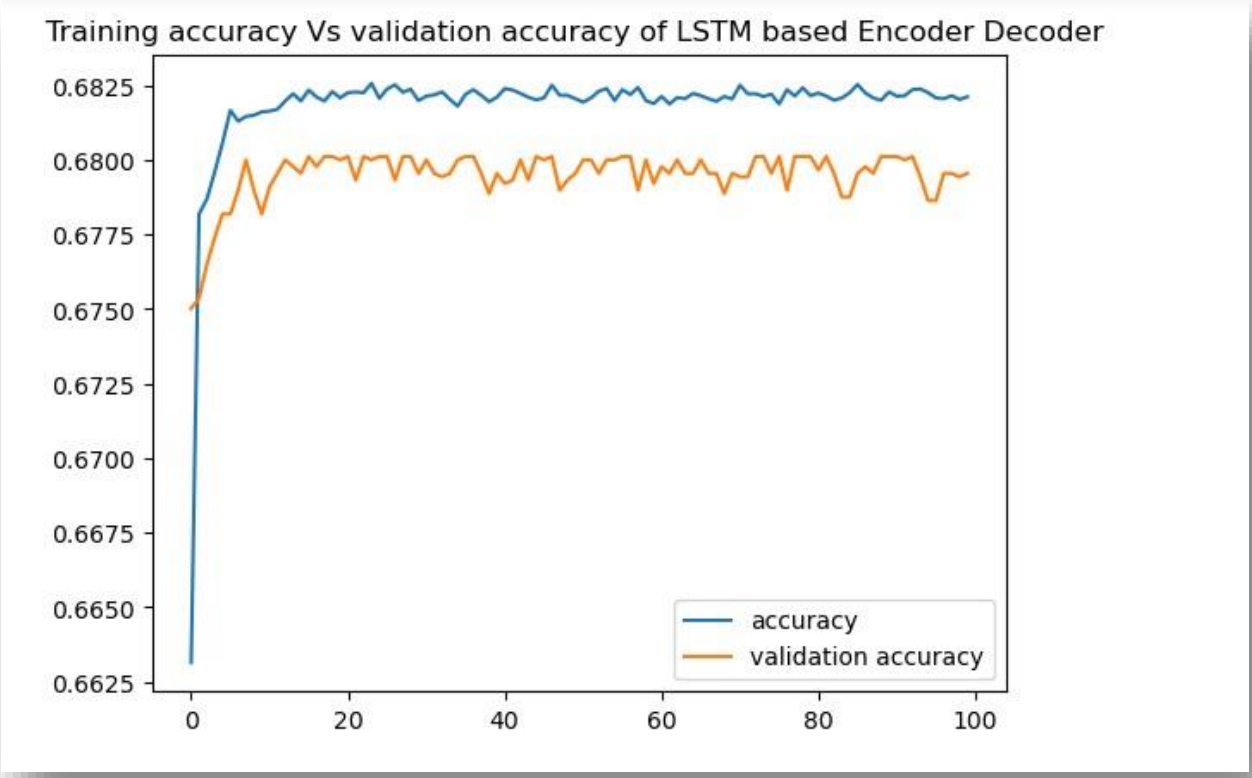

Figure 25 Training and validation accuracy of encoder decoder model with LSTM algorithm

In Figure 26, the training and validation accuracy of the encoder decoder model using the GRU algorithm is shown. As shown in the Figure 26, the model performs 82.28% and 81.9 % training accuracy and validation accuracy respectively.

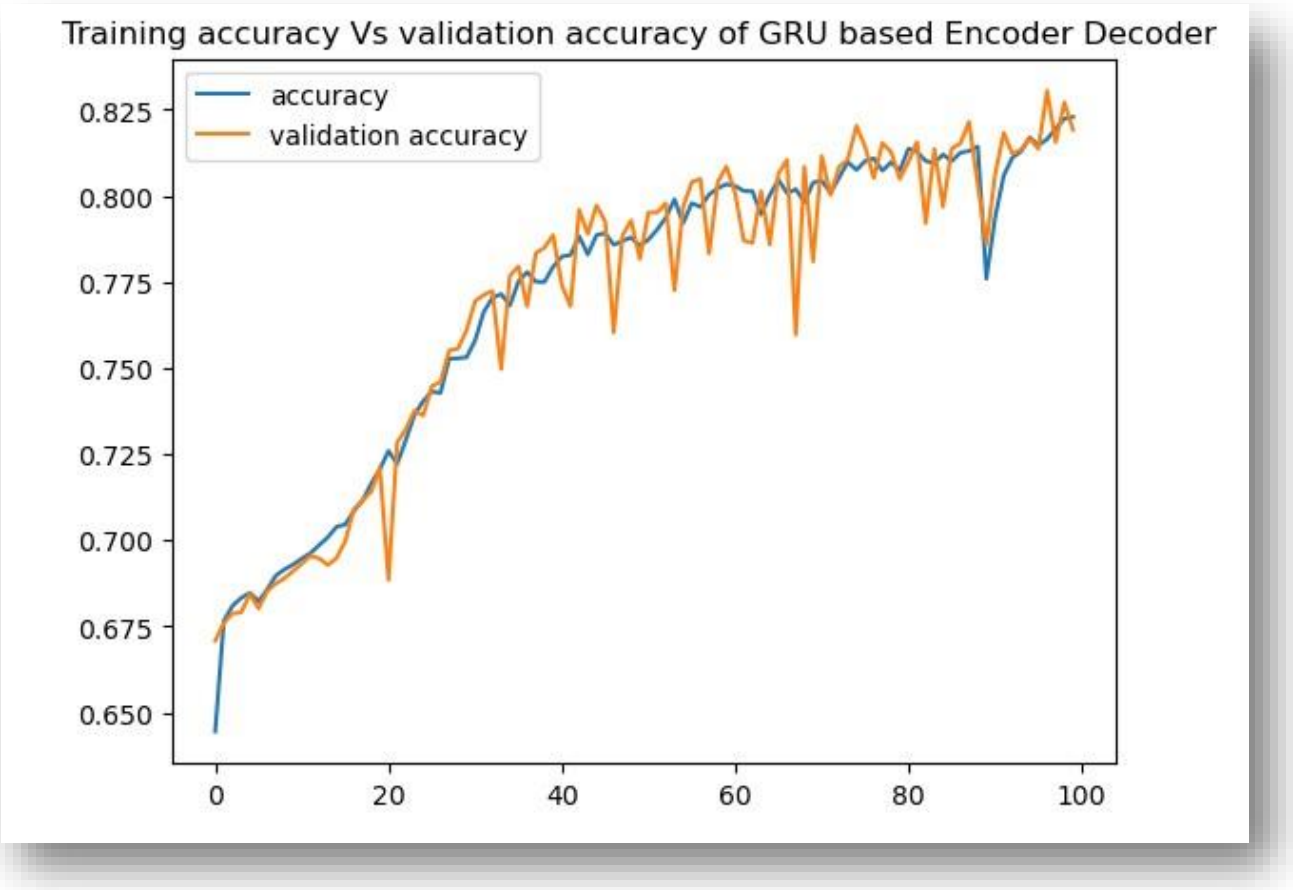

Figure 26 Training and validation accuracy of encoder decoder model with GRU algorithm

In Figure 27, the training and validation accuracy of the encoder decoder model using the Bi-LSTM algorithm is shown. As shown in the Figure 27, the model performs 93.5% and 85.5 % training accuracy and validation accuracy respectively.

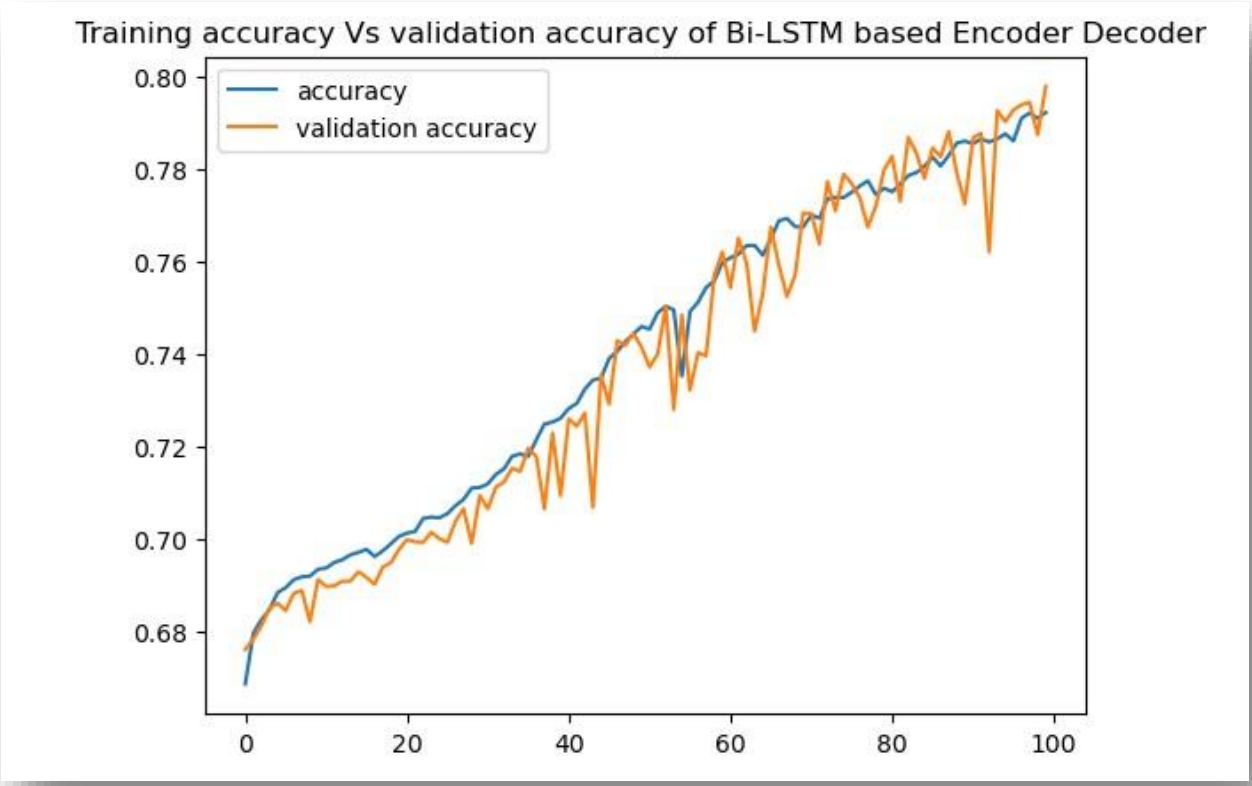

Figure 27 Training and validation accuracy of encoder decoder model with Bi-LSTM algorithm

In Figure 28 the training and validation accuracy of attention based model using the Bi-LSTM algorithm is shown. As shown in the Figure 28, the model performs 65.5% and 62.5 % training accuracy and validation accuracy respectively.

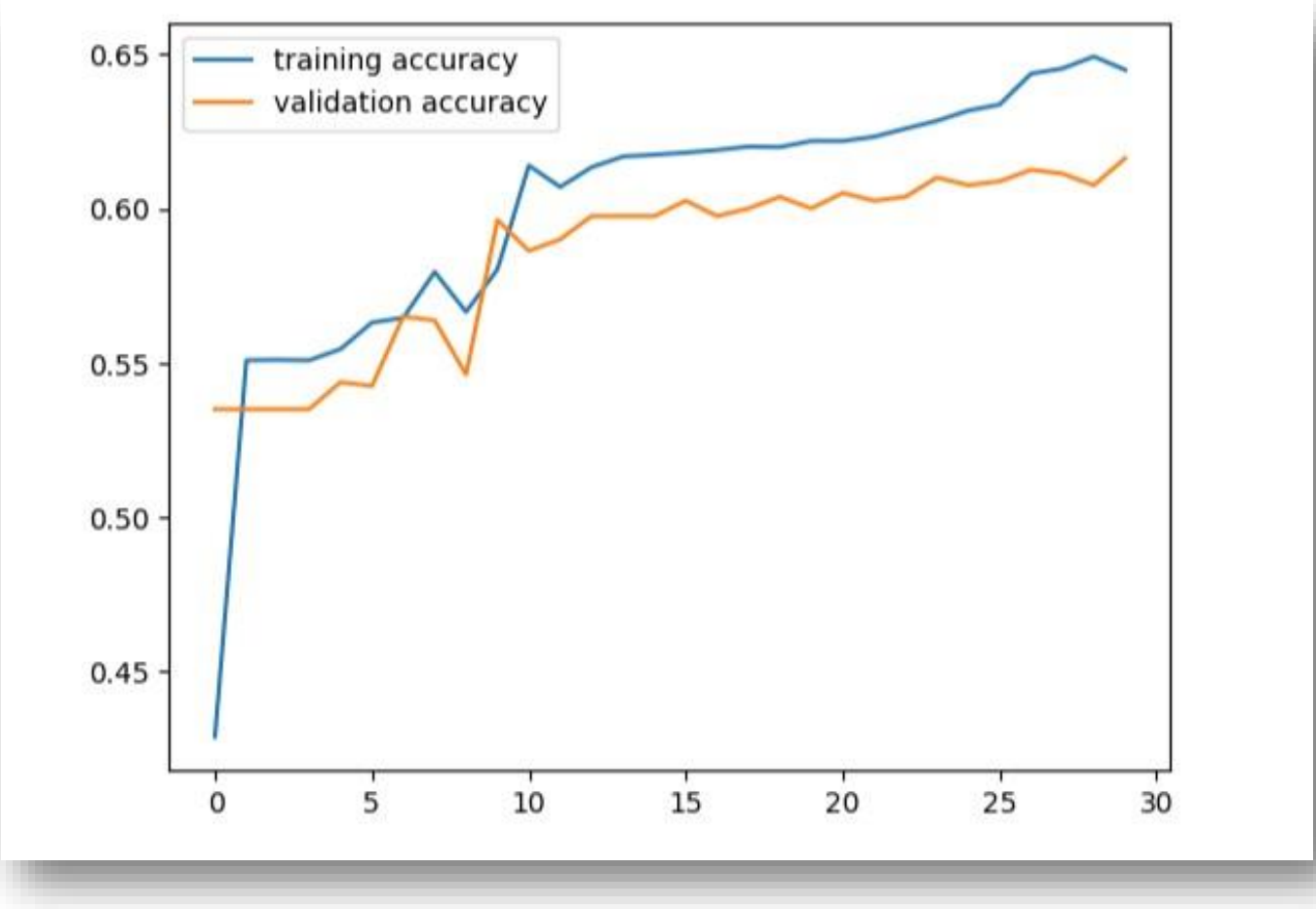

Figure 28 training and validation accuracy of attention based model using the Bi-LSTM algorithm

4.5.3. Training and validation loss of English to Tigrigna translation model

In this section, the researcher described the training and validation loss of the proposed translation model for English to Tigrigna text.

In figure 29, the training and validation loss of encoder decoder model using LSTM algorithm is shown. From Figure 29, we observe that the model scores a training loss of 2.7 and a validation loss of 2.8.

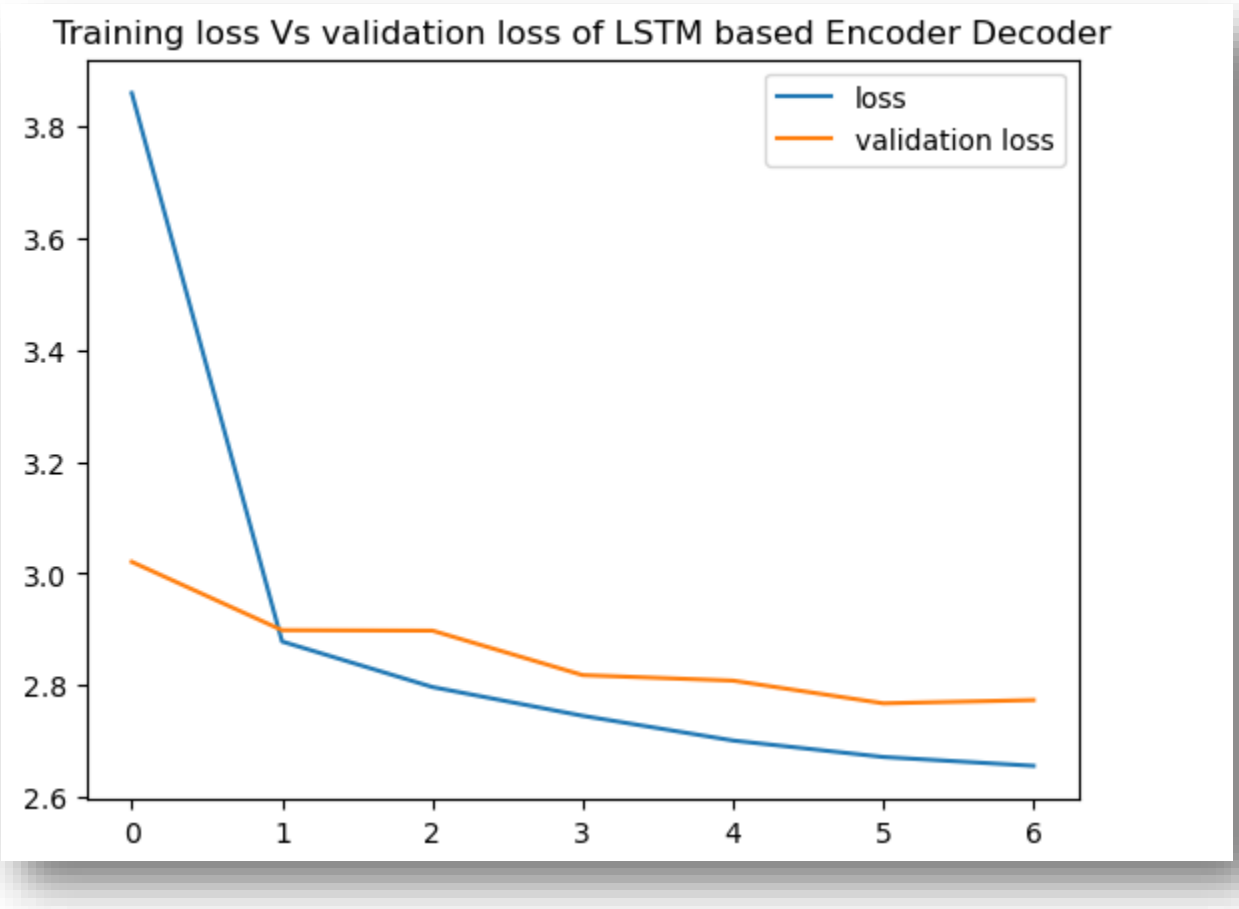

Figure 29 training and validation accuracy of attention based model using the STM algorithm

In figure 30, the training and validation loss of encoder decoder model using GRU algorithm is shown. From Figure 30, we observe that the model scores a training loss of 2.2 and a validation loss of 2.4.

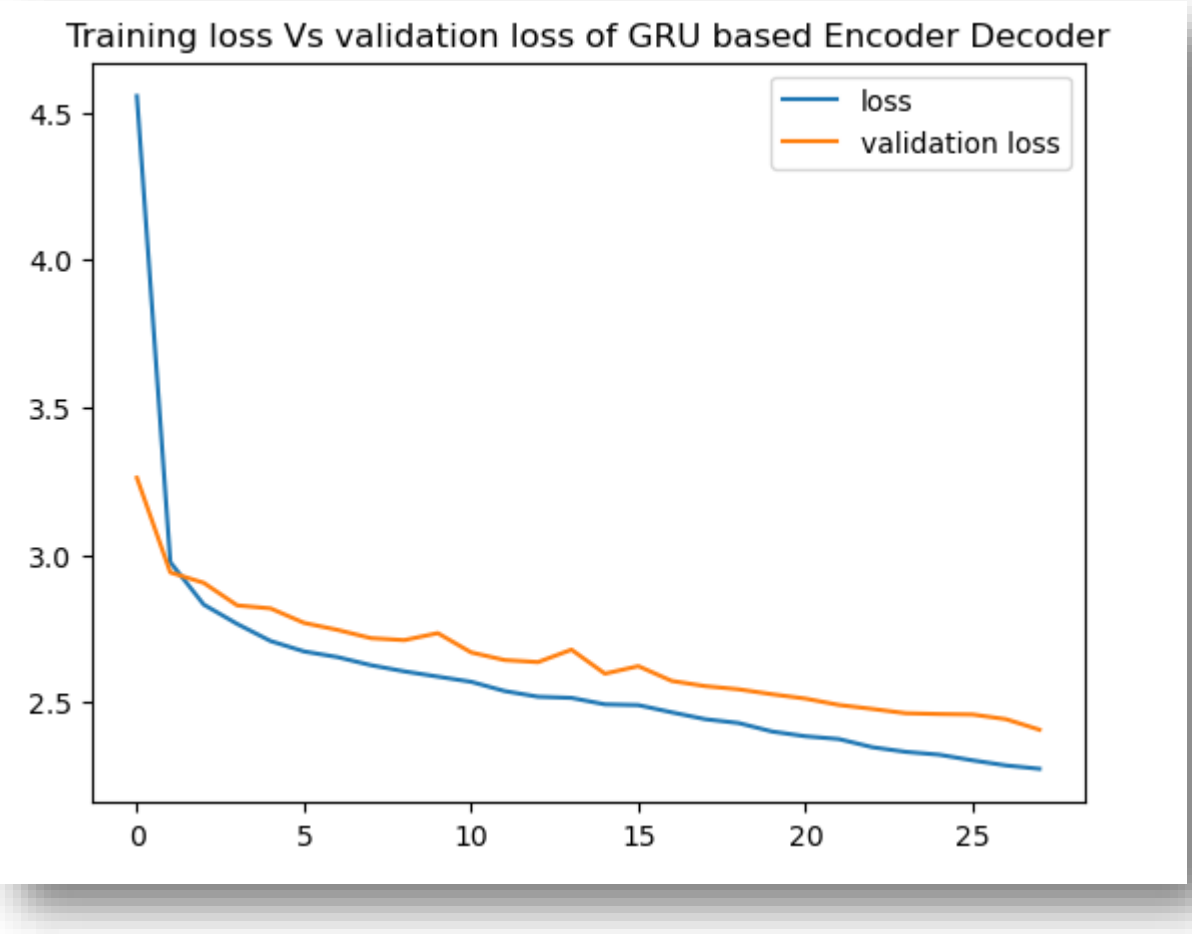

Figure 30 Training and validation loss of encoder decoder model with GRU algorithm

In figure 31, the training and validation loss of encoder decoder model using Bi-LSTM algorithm is shown. From Figure 31, we observe that the model scores a training loss of 0.34 and a validation loss of 0.38.

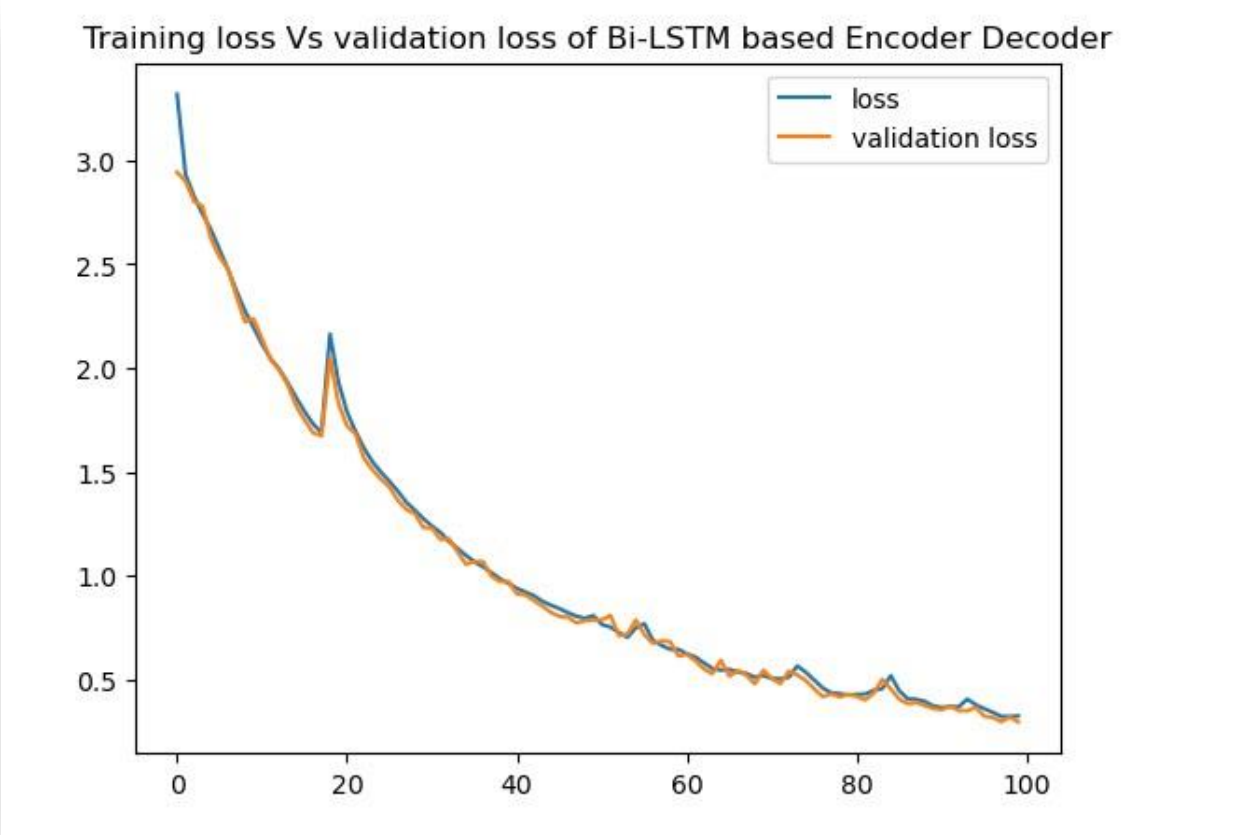

Figure 31 Training and validation loss of encoder decoder model with Bi-LSTM algorithm

4.5.4. Training and validation loss of Tigrigna to English translation model

In Figure 32, the training and validation loss of the encoder decoder model using the LSTM algorithm for Tigrigna-English translation is shown. As shown in the Figure 32, the model scored 2.25 and 3.1 training loss and validation loss respectively.

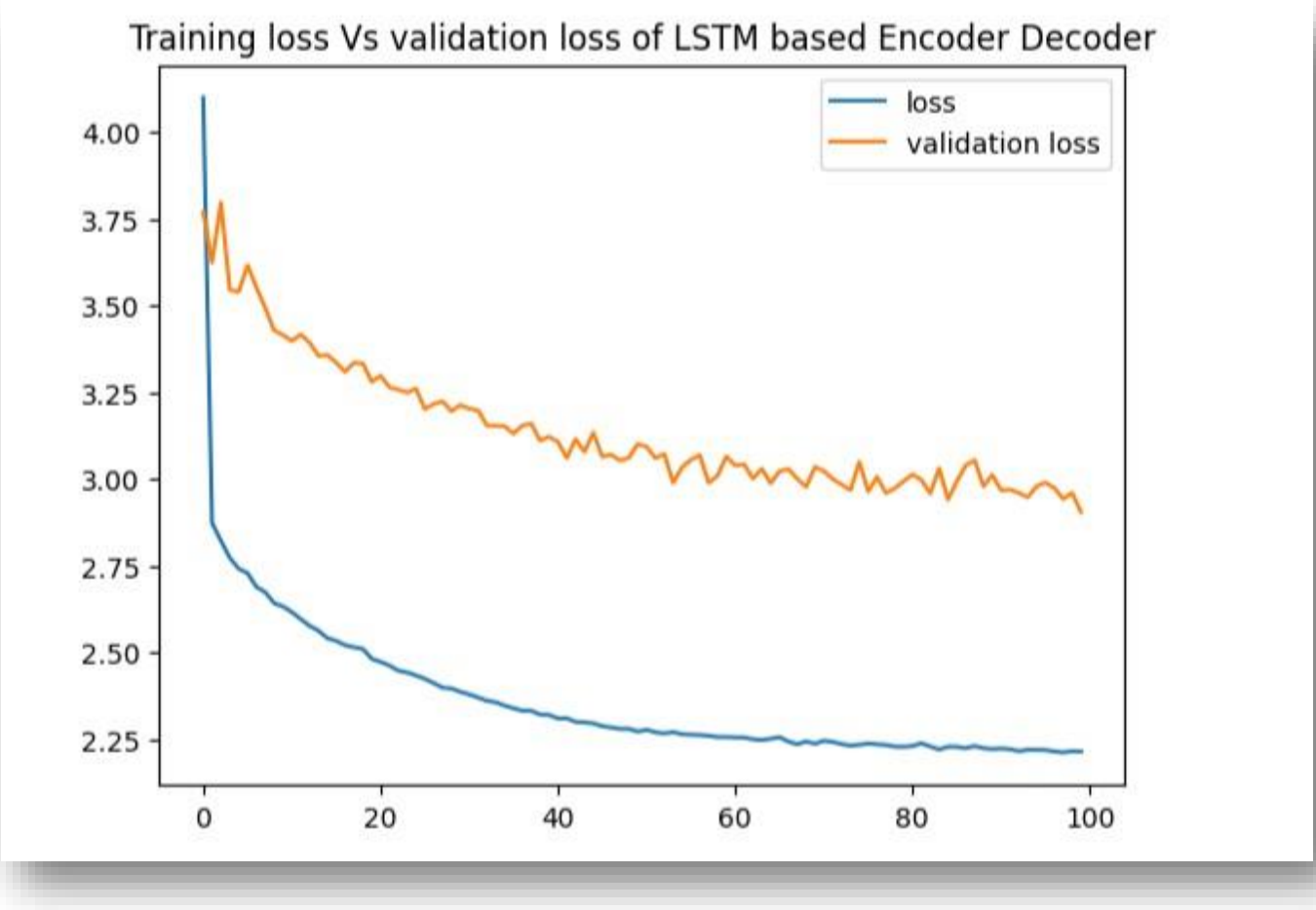

Figure 32 Training and validation loss of encoder decoder model with LSTM algorithm

In Figure 33, the training and validation loss of the encoder decoder model using the GRU algorithm is shown. As shown in the Figure 33, the model scored 2.1 and 2.4 training loss and validation loss respectively.

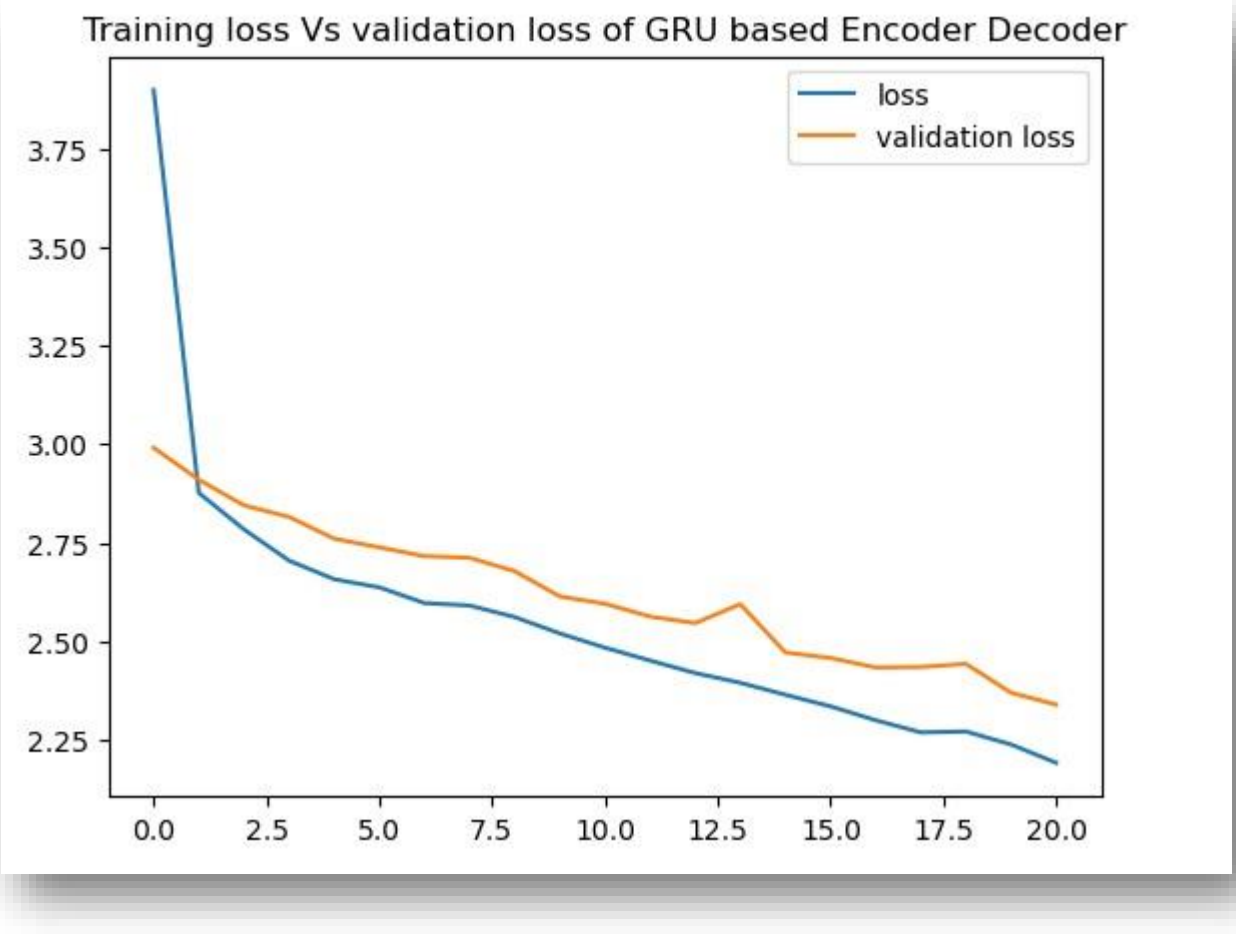

Figure 33 Training and validation loss of encoder decoder model with GRU algorithm

In Figure 34, the training and validation loss of the encoder decoder model using the Bi-LSTM algorithm is shown. As shown in the Figure 34, the model scored 0.6 and 0.71 training loss and validation loss respectively.

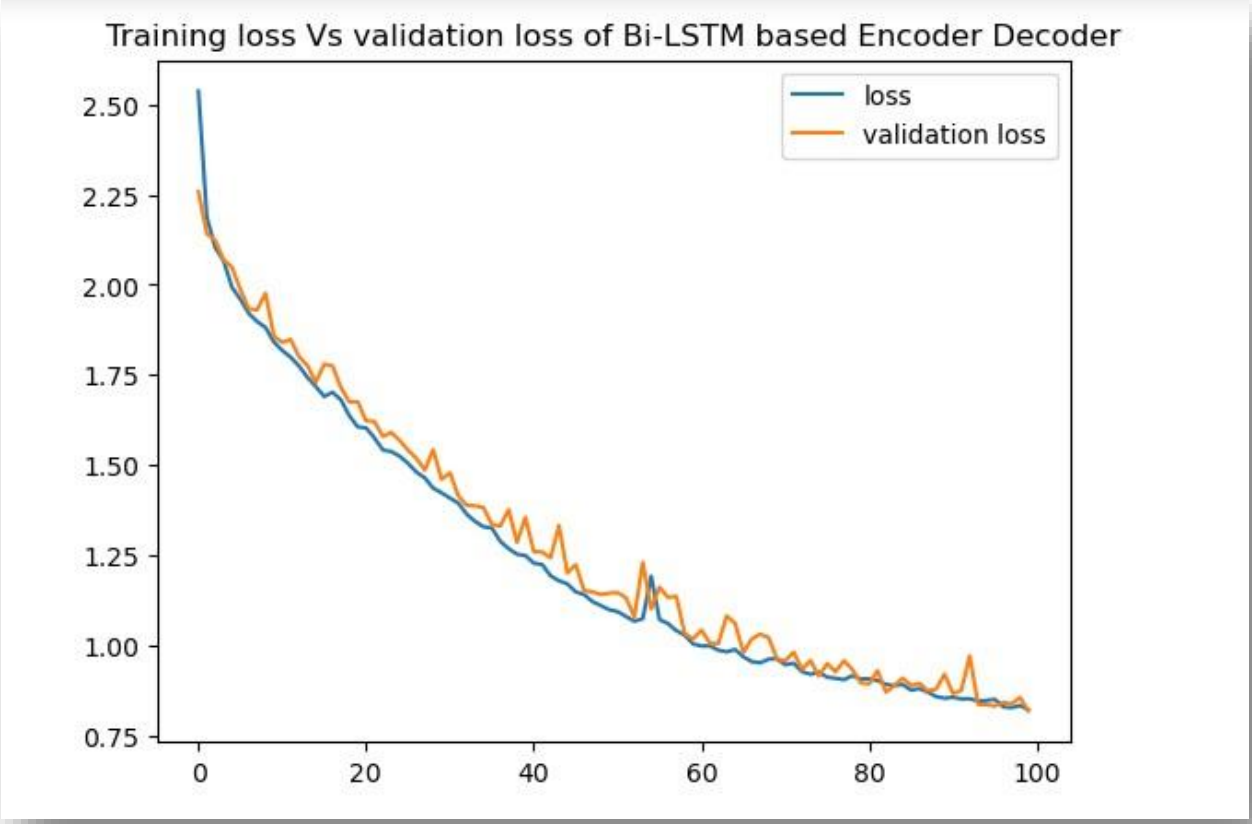

Figure 34 Training and validation loss of encoder decoder model with Bi-LSTM algorithm

In Figure 25, the training and validation loss of attention based model using the Bi-LSTM algorithm is shown. As shown in the Figure 35, the model scored 1.8 and 3.0 training loss and validation loss respectively.

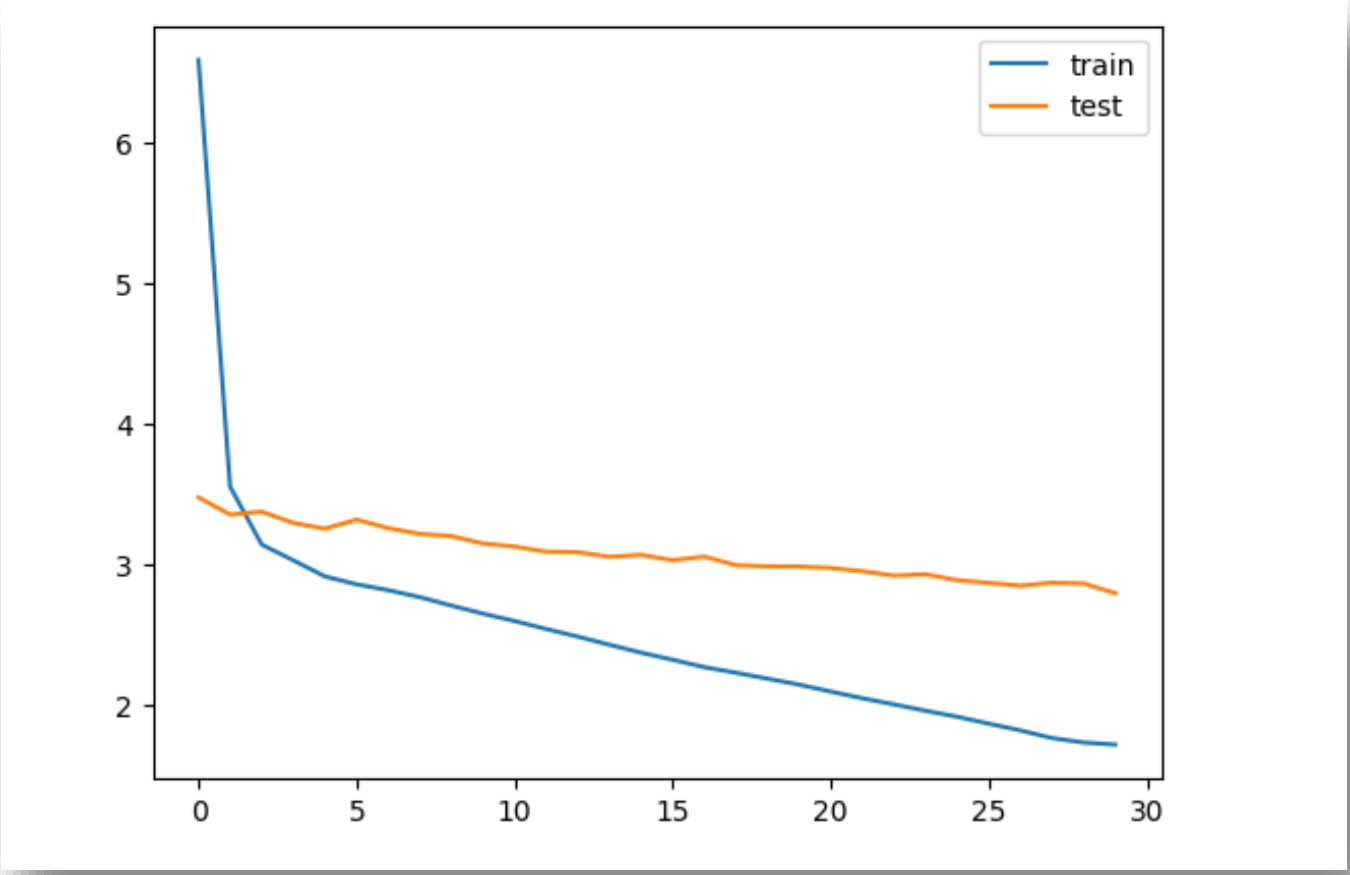

Figure 35 training and validation loss of attention based model using the Bi-LSTM algorithm

4.1. Experimental results and discussion

In this study, we have followed experimental research methodology that enables to conduct many investigations by taking different combination of model parameters and dataset distributions. After conducting different experiments with different values of parameter combination, the researcher selected parameter values with best performance discussed in section 4.4. Using the experiments conducted above, the researcher developed bidirectional Tigrigna-English machine translation. We have conducted different experiments using different machine translation techniques. To evaluate our model we used the BLEU score. We used the encoder decoder model and the attention based models in our experiment. We have experimented encoder decoder model using the LSTM, GRU and Bi-LSTM algorithms. Based on our experiment we got best BLEU score result using Bi-LSTM based encoder decoder model. Observing that the encoder decoder model using Bi-LSTM performs better, we also experimented the attention based model using the Bi-LSTM algorithm.

According to the experiments done, encoder decoder using the Bi-LSTM algorithm performs better than the encoder decoder model using the GRU and LSTM. This is because the model using the Bi-LSTM method can learn long-term bidirectional text associations. Based on our experiments our best scored model outperforms the BLEU score of the baseline model, obtained by previous researcher [15] who investigated the Tigrigna to English translation for specific domain of humanitarian response with BLEU scores of +0.8. Performance of different models using different approaches is shown in Table 2 and Table 3.

Table 3 Performance of encoder decoder model for English-Tigrigna translation

Encoder decoder model using	English to Tigrigna translation	Tigrigna to English translation
	Bleu score	Bleu score
LSTM Algorithm	18	19
Bi-LSTM Algorithm	**24.8**	**24.4**
GRU Algorithm	22	20

Table 4 performance of attention based model

Attention based models using	English- Tigrigna translation	Tigrigna- English translation
	Bleu score	Bleu score
Bi-LSTM Algorithm	21	22

The bar graphs in Figure 36 and 37 shows the comparison of different mation transaltion approaches in this study for English-Tigrigna and Vice versa transaltion.

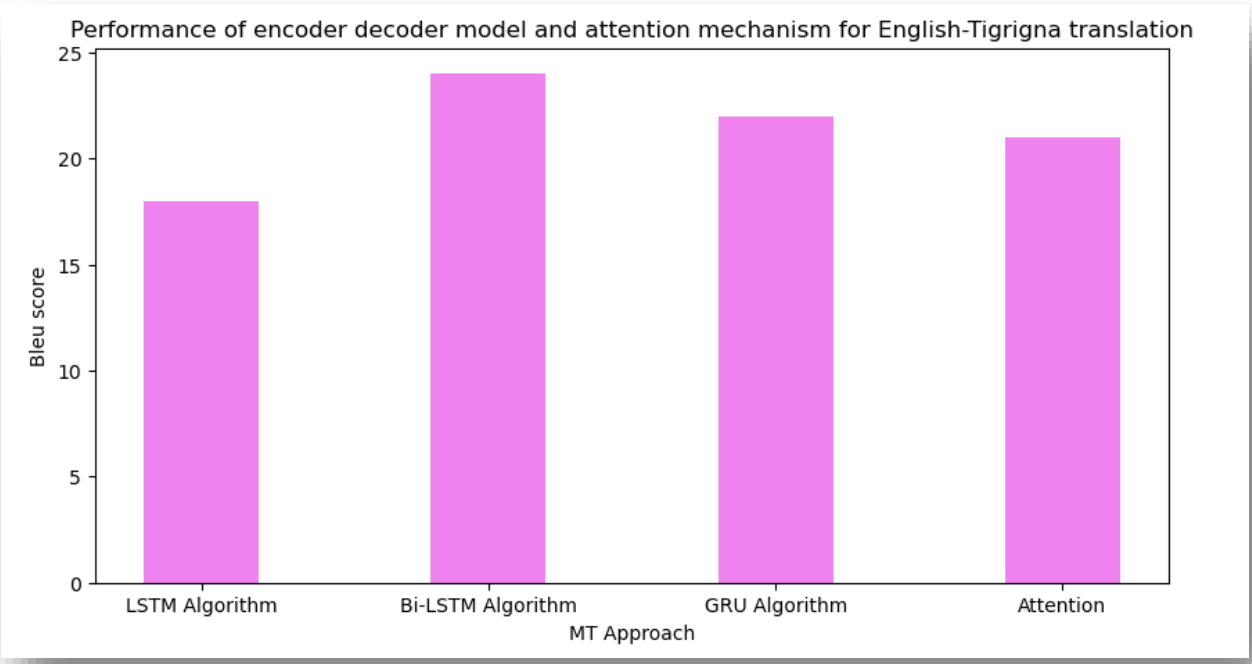

Figure 36 performance of different models for English-Tigrigna MT

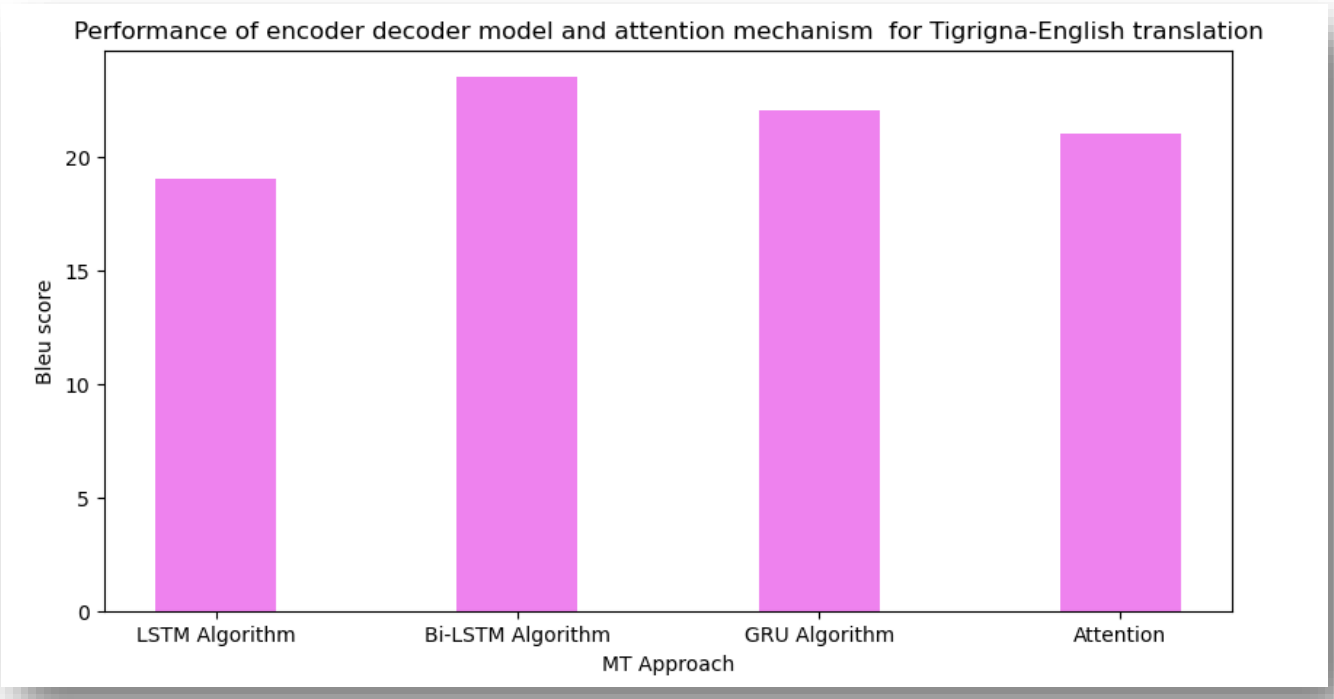

Figure 37 performance of different models for Tigrigna-English MT

4.2. Comparison of the model with previous works

In this stage we have compared the proposed model with previous studies. In table 5, the performance of different models is presented. As sown in the table our model performs a BLUE score of +0.8 from the baseline study by [15].

Table 5 comparison of proposed model with previous works

Model	Data size	Translation Approach	BLUE score	
			Tigrigna to English	English to Tigrigna
Proposed model	31000	Encoder Decoder with Bi-LSTM	**24.4**	**24.8**
Alp Oktem, Mirko Plitt & Grace Tang [15]	12000	Transfer Learning	23.6	---
M. Azath, Tsegay Kiros [4]	17,338	Statistical Machine Translation	---	23.27
Isayas Berhe Adhanom [7]	Not mentioned	Neural machine translation	---	21.4

4.3. Prediction

The trained translation model predicts the Tigrigna translation of English text and vice versa during the prediction phase. The text needs to be preprocessed, then fed to the loaded. The Tigrigna translation or English and vice versa of the text is displayed by the model. Samples of the model's prediction results are shown in below snapshots. The Figures are samples snapshots of the prediction phase of different translation approaches followed in our experiment. Figure 37 shows the prediction output of Bi-LSTM based encoder decoder model for English-Tigrigna translation.

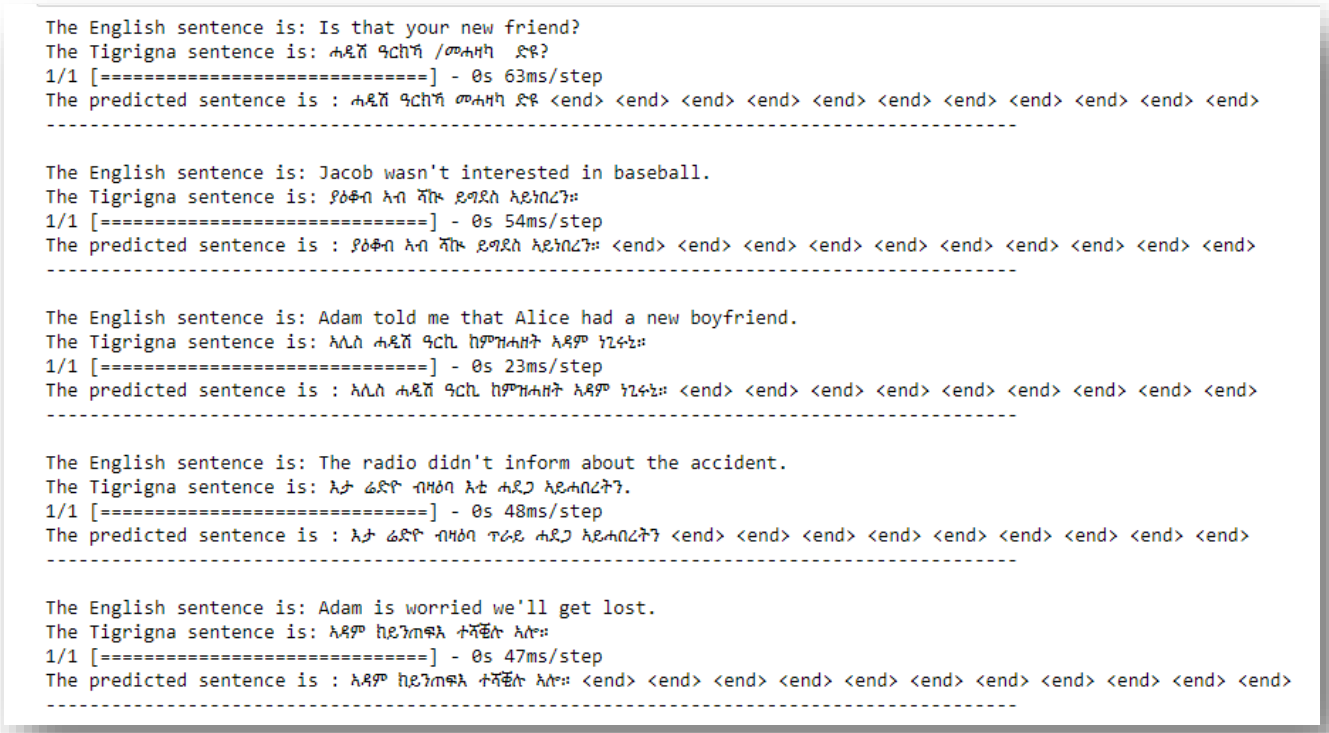

Figure 38 prediction output of Bi-LSTM based encoder decoder model for English-Tigrigna translation

Figure 39 shows the prediction output of Bi-LSTM based encoder decoder model Tigrigna-English translation

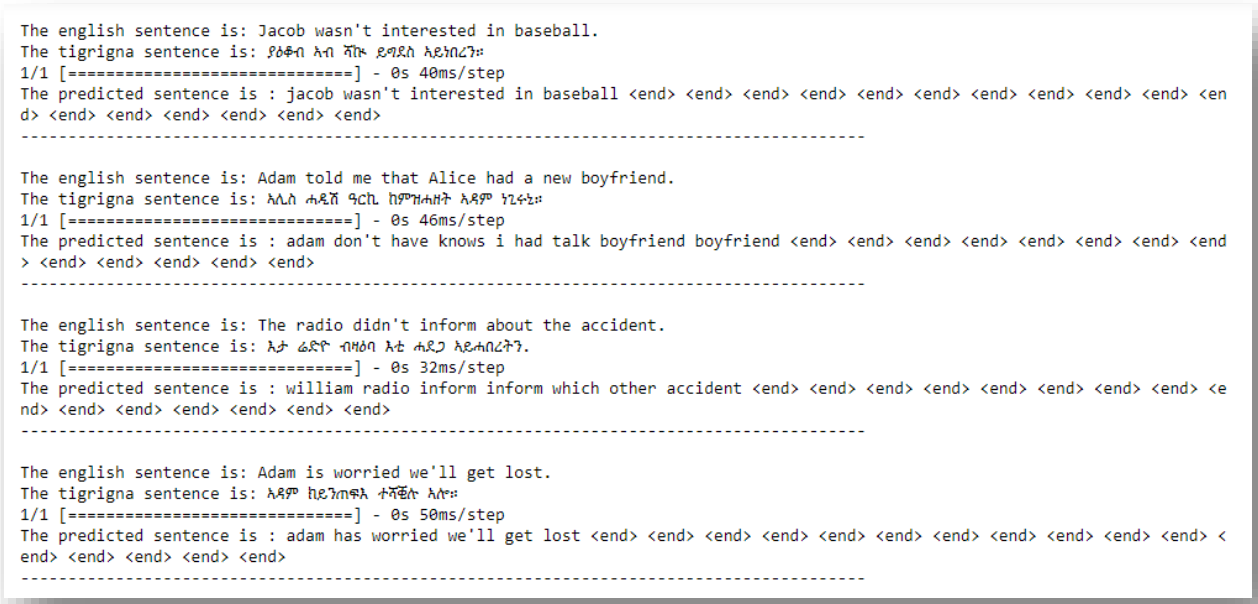

Figure 39 prediction output of Bi-LSTM based encoder decoder model for Tigrigna-English translation

Figure 40 shows the prediction output of Bi-LSTM based attention model.

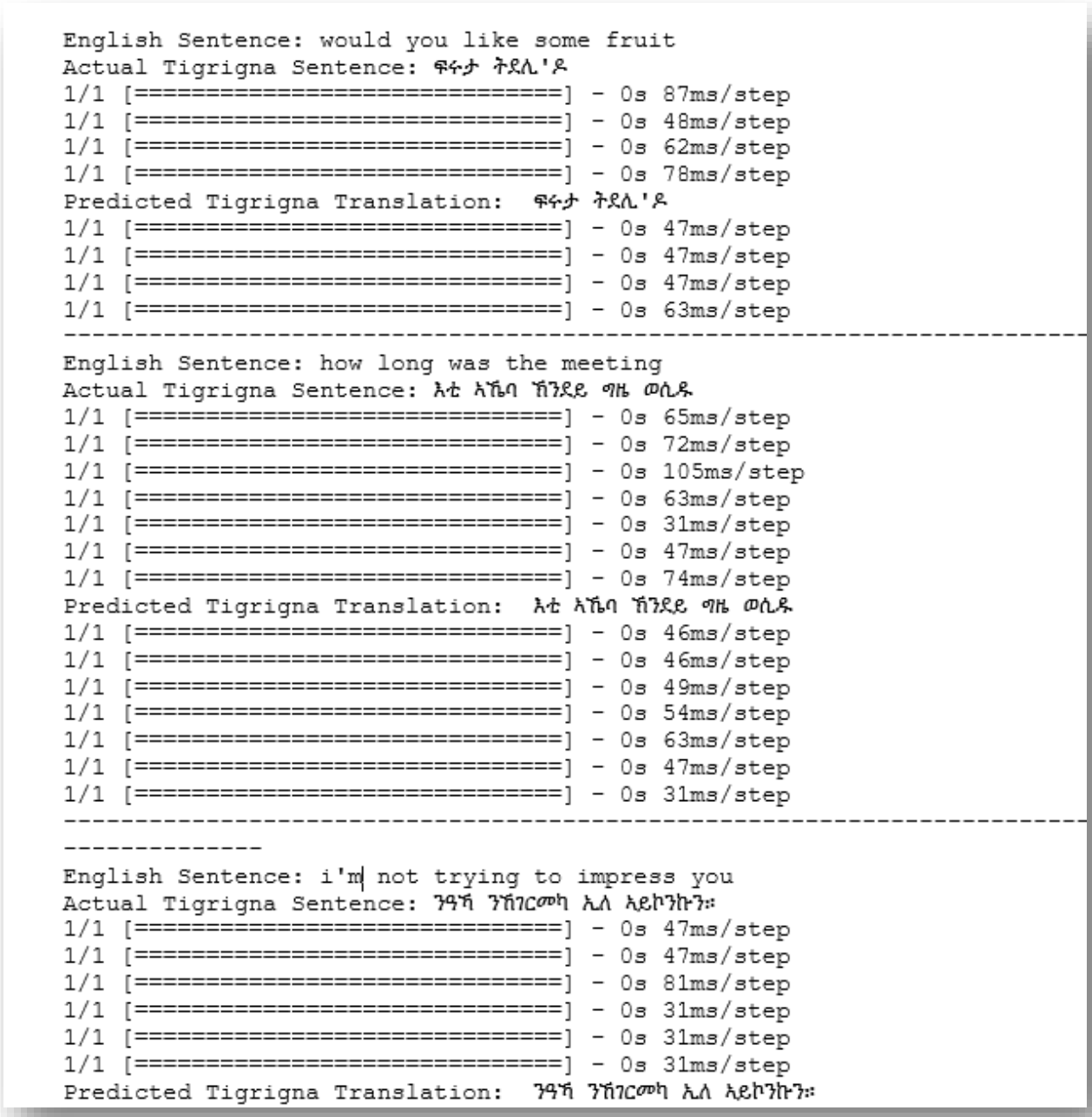

Figure 40 prediction of Bi-LSTM based attention model

CHAPTER FIVE

5. CONCLUSION AND RECOMMENDATION

5.1. Introduction

This chapter discusses the research findings and recommendations for future researchers interested in working on Tigrigna-English machine translation. In this study the researcher has developed the translation model using different deep learning algorithms. The model is able to predict the Tigrigna translation of English text and vice versa. In this study encoder decoder and attention based models are tested for their performance on Tigrigna-English machine translation.

5.2. Conclusion

Machine translation (MT) studies how to utilize computer software to translate text or speech from one language to another without using humans. Because language is such an efficient means of communication in the modern world, there is a growing demand for language translations. Because more information is being exchanged between different regions using distinct regional languages, there has been a rise in demand for translation services in recent years.

In this study, the researcher developed bidirectional Tigrigna-English machine translation model using different machine translation approaches. In the study we have collected Tigrigna-English parallel corpus from different sources and by translating English text to Tigrigna. Our model is trained on 31000 Tigrigna-English parallel sentences. Using our dataset we have experimented different translation approaches. We have experimented approaches of encoder decoder model and attention based models using LSTM, Bi-LSTM and GRU deep learning algorithms.

Based on the result of our experiments, our encoder decoder model using the Bi-LSTM algorithm has a better BLEU score. The encoder decoder model using the Bi-LSTM algorithm scored 24.8 for English-Tigrigna translation and 24.4 for Tigrigna- English translation. The encoder decoder model using the LSTM algorithm scored 18 for English-Tigrigna translation and 19 for Tigrigna-English translation. The encoder decoder model using the GRU algorithm scored 22 for English-Tigrigna translation and 20 for Tigrigna- English translation. From the experiment the encoder decoder model using the Bi-LSTM algorithm took long training time of 4 hours. The attention based model is also experimented using our dataset. Attention based model is experimented using the Bi-LSTM algorithms. The attention model using the Bi-LSTM algorithm scored a BLEU score of 21 for English- Tigrigna and 22 for Tigrigna- English translation.

5.3. Contribution

After the end of this study, the researcher has contributed the following things for other researchers and anyone who is interested on Tigrigna-English machine translation.

- We have prepared 2k parallel Tigrigna-English sentences
- The researcher has showed that encoder decoder model using Bi-LSTM algorithm outperforms other approaches in our experiments.
- The researcher has developed best performed model as compared to baseline models.
- The researcher has built bidirectional Tigrigna-English translation model.

5.4. Recommendation

After all the researcher recommend the following issues to be addressed for future.

- This translation model is trained with limited dataset, extending the study with a large dataset can be one task.
- Exploring word embedding techniques such as word2vec, fastText and BERT for text representation can be one task.
- Morphological analysis of the languages might increase the performance of the translation model and is recommended for future researchers
- Exploring other approaches to improve the performance of the model can be one task
- Since the goal of this study is to implement machine translation for text-to-text translation, we recommend that future work to conduct speech-to-speech machine translation between these language pairs.

References

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

[2] J. Allen, "Natural Language Understanding," pp. 1–24, 1995.

[3] A. Wahid, "Machine Translation System Using Deep Learning for English to Urdu," vol. 2022, 2022.

[4] M. Azath and T. Kiros, "Statistical machine translator for english to tigrigna translation," *Int. J. Sci. Technol. Res.*, vol. 9, no. 1, pp. 2095–2099, 2020.

[5] M. V. S. Rishita, M. A. Raju, and T. A. Harris, "Machine translation using natural language processing," *MATEC Web Conf.*, vol. 277, p. 02004, 2019, doi: 10.1051/matecconf/201927702004.

[6] Negash A., "The Origin and Development of Tigrinya Language Publications (1886 - 1991)," *Univ. Libr. 131. http//scholarcommons.scu.edu/library/131*, vol. 1, pp. 1–106, 2016.

[7] I. B. Adhanom, "A First Look into Neural Machine Translation for Tigrinya A First Look into Neural Machine Translation for Tigrinya," no. March, 2021, doi: 10.13140/RG.2.2.10698.90562.

[8] S. Dr. and D. V. Singh, "A Study on the History of English Language," *Iarjset*, vol. 8, no. 6, pp. 207–211, 2020, doi: 10.17148/iarjset.2021.8636.

[9] H. Wang and B. Raj, "On the Origin of Deep Learning," pp. 1–72, 2017, [Online]. Available: http://arxiv.org/abs/1702.07800

[10] M. Popel, M. Tomkova, Ł. Kaiser, and J. Uszkoreit, "Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals," pp. 1–15, 2020, doi: 10.1038/s41467-020-18073-9.

[11] Z. Tan *et al.*, "Neural machine translation: A review of methods, resources, and tools," *AI Open*, vol. 1, no. March, pp. 5–21, 2020, doi: 10.1016/j.aiopen.2020.11.001.

[12] D. Goularas and S. Kamis, “Evaluation of Deep Learning Techniques in Sentiment Analysis from Twitter Data,” *2019 Int. Conf. Deep Learn. Mach. Learn. Emerg. Appl.*, pp. 12–17, 2019, doi: 10.1109/Deep-ML.2019.00011.

[13] T. Dettmers, “Deep Learning in a Nutshell: History and Training.” [Online]. Available: https://developer.nvidia.com/blog/deep-learning-nutshell-history-training/

[14] Y. Shi, Y. Tian, Y. Wang, W. Zeng, and T. Huang, “Learning long-term dependencies for action recognition with a biologically-inspired deep network,” pp. 716–725.

[15] A. Oktem, “T IGRINYA N EURAL M ACHINE T RANSLATION WITH T RANSFER L EARNING FOR H UMANITARIAN R E -,” pp. 2–5, 2020.

[16] Z. Berihu, G. Mesfin, and M. Atsibaha, “Enhancing Bi-directional English-Tigrigna Machine Translation Using Hybrid Approach Overview of the Tigrinya language The Tigrinya writing system Research methodology Architecture of the proposed system”.

[17] Mulubrhan Hailegebreal, “A Bidirectional Tigrigna – English Statistical Machine Translation,” *Masters Thesis summited to Addis Ababa Univ.*, no. October, 2017.

[18] L. Zhao, W. Gao, and J. Fang, “applied sciences High-Performance English – Chinese Machine Translation Based on GPU-Enabled Deep Neural Networks with Domain Corpus,” 2021.

[19] T. TSEGAYE, “English -Tigrigna Factored Statistical,” no. June, 2014.

[20] P. S. Foundation, “What is Python? Executive Summary”, [Online]. Available: https://www.python.org/doc/essays/blurb/

[21] S. Yegulalp, “What is TensorFlow? The machine learning library explained.” [Online]. Available: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html

[22] “About Keras.” [Online]. Available: https://keras.io/about/

[23] S. P. Singh *et al.*, “Machine Translation using Deep Learning: An Overview,” pp. 162–167, 2017.

[24] S. Behera and S. Behera, “Machine translation using deep learning: A survey,” *Int. J.*

Psychosoc. Rehabil., vol. 23, no. 5, pp. 545–550, 2019, doi: 10.37200/IJPR/V23I5/PR190653.

[25] F. Fabbro, "Introduction to language and cerebellum," *J. Neurolinguistics*, vol. 13, no. 2–3, pp. 83–94, 2000, doi: 10.1016/S0911-6044(00)00005-1.

[26] J. Mcwhorter, "The Story of Human Language".

[27] W. Paper, K. Hyun, and K. Individual, "Origin of Human Language," no. February, 2016.

[28] B. Lutkevich, "natural language processing (NLP)." [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP

[29] E. . Liddy, "Encyclopedia of Library and Information Science, 2 Ed. Marcel Decker, Inc.," *Encycl. Libr. Inf. Sci. 2nd Ed.*, 2001, [Online]. Available: http://surface.syr.edu/cgi/viewcontent.cgi?article=1043&context=istpub%5Cnhttp://surface.syr.edu/istpub/63/

[30] K. M. Verspoor and K. B. Cohen, "Natural Language Processing," *Encycl. Syst. Biol.*, no. January 2013, 2013, doi: 10.1007/978-1-4419-9863-7.

[31] P. M. Nadkarni, L. Ohno-machado, and W. W. Chapman, "Natural language processing : an introduction," 2011, doi: 10.1136/amiajnl-2011-000464.

[32] Y. Zhou, "Natural Language Processing with Improved Deep Learning," vol. 2022, 2022.

[33] M. Y. Tachbelie, "No Title," no. August, 2010.

[34] K. B. Cohen, *Biomedical Natural Language Processing and Text Mining*, Error. Elsevier Inc., 2014. doi: 10.1016/B978-0-12-401678-1.00006-3.

[35] N. A. Kifle, "Tigrinya Applicatives in Lexical-Functional Grammar," 2011.

[36] H. Beyene, "DESIGN AND DEVELOPMENT OF TIGRIGNA SEARCH Engine," 2013.

[37] H. A. Nega, "Morpheme Based Bi-Directional Machine Translation," 2023.

[38] M. A. SIUM, "AUTOMATIC PART-OF-SPEECH TAGGER FOR TIGRIGNA LAN_GUAGE USING HYBRID APPROACH," *Corresp. Análisis*, no. 15018, pp. 1–23,

2016.

[39] B. M. H. Strang, G. Bourcier, and J. Lyons, “An Introduction to the History of the English Language,” *Mod. Lang. Rev.*, vol. 79, no. 2, p. 410, 1984, doi: 10.2307/3730036.

[40] “Morphology , morphemes , allomorphs , word classes”.

[41] “The morphology of the major word classes.” [Online]. Available: https://www.awelu.lu.se/language/selective-mini-grammar/the-morphology-of-the-major-word-classes/

[42] M. D. Okpor, “Machine Translation Approaches,” vol. 11, no. 5, pp. 159–165, 2014.

[43] D. Arnold and L. Sadler, “Machine Translation: an Introductory Guide,” no. November, 2017.

[44] J. Tsujii, “Machine Translation and Machine-Aided Translation: What’s Going On’,” *Transl. Comput.*, vol. 12, no. November 1990, pp. 3–24, 1991.

[45] P. S. C. Chien and G. Hui-chin Lin, “Machine Translation for Academic Purposes,” *Proc. Int. Conf. TESOL Transl.*, no. December, pp. 133–148, 2009.

[46] H. Wang, H. Wu, Z. He, L. Huang, and K. Ward, “Progress in Machine Translation,” *Engineering*, 2021, doi: 10.1016/j.eng.2021.03.023.

[47] G. Randhawa, M. Ferreyra, R. Ahmed, O. Ezzat, and K. Pottie, “Using machine translation in clinical practice.,” *Can. Fam. Physician*, vol. 59, no. 4, pp. 382–383, Apr. 2013.

[48] T. Khanna *et al.*, “Recent advances in Apertium, a free/open-source rule-based machine translation platform for low-resource languages,” *Mach. Transl.*, vol. 35, pp. 1–28, 2021, doi: 10.1007/s10590-021-09260-6.

[49] K. Imamura, H. Okuma, T. Watanabe, and E. Sumita, “Example-based machine translation based on syntactic transfer with statistical models,” *COLING 2004 - Proc. 20th Int. Conf. Comput. Linguist.*, no. Figure 2, 2004, doi: 10.3115/1220355.1220370.

[50] A. Franz, K. Horiguchi, L. Duan, D. Ecker, E. Koontz, and K. Uchida, “An integrated architecture for example-based machine translation,” p. 1031, 2000, doi:

10.3115/992730.992799.

[51] N. Semmar, O. Zennaki, and M. Laib, “Improving the performance of an example-based machine translation system using a domain-specific bilingual lexicon,” *29th Pacific Asia Conf. Lang. Inf. Comput. PACLIC 2015*, no. September, pp. 106–115, 2015.

[52] P. E. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer, “The Mathematics of Statistical Machine Translation : Parameter Estimation,” vol. 10598, 1993.

[53] P. F. Brown *et al.*, “Statistical approach to machine translation,” vol. 16, no. 2, pp. 79–85, 1990.

[54] R. Sankaravelayuthan and G. Vasuki, “English To Tamil Machine Translation System Using Parallel Corpus,” *India’s High. Educ. Auth. UGC Approv. List Journals Ser. Number*, vol. 19, no. 5, 2019, [Online]. Available: www.languageinindia.com

[55] Z. Yang, Y. Gao, W. Wang, and H. Ney, “Predicting and Using Target Length in Neural Machine Translation,” *Proc. 1st Conf. Asia-Pacific Chapter Assoc. Comput. Linguist. 10th Int. Jt. Conf. Nat. Lang. Process.*, pp. 389–395, 2020, [Online]. Available: https://www.aclweb.org/anthology/2020.aacl-main.41

[56] B. Premjith, M. A. Kumar, and K. P. Soman, “Neural machine translation system for English to Indian language translation using MTIL parallel corpus,” *J. Intell. Syst.*, vol. 28, no. 3, pp. 387–398, 2019, doi: 10.1515/jisys-2019-2510.

[57] A. V. M. Barone, J. Helcl, R. Sennrich, B. Haddow, and A. Birch, “Deep architectures for neural machine translation,” *WMT 2017 - 2nd Conf. Mach. Transl. Proc.*, no. January, pp. 99–107, 2017, doi: 10.18653/v1/w17-4710.

[58] J. Yang, S. Ma, D. Zhang, Z. Li, and M. Zhou, “Improving neural machine translation with soft template prediction,” *Proc. Annu. Meet. Assoc. Comput. Linguist.*, pp. 5979–5989, 2020, doi: 10.18653/v1/2020.acl-main.531.

[59] S. Editor Wolfgang Walz, *Machine Learning for Brain Disorders*, vol. 197. 2023. [Online]. Available: https://link.springer.com/10.1007/978-1-0716-3195-9

[60] M. Parmar and V. S. Devi, “Neural Machine Translation with Recurrent Highway

Networks," pp. 1–10.

[61] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, *An overview and comparative analysis of Recurrent Neural Networks for Short Term Load Forecasting*, no. November. 2017. doi: 10.1007/978-3-319-70338-1.

[62] D. Lee *et al.*, "Long short-term memory recurrent neural network-based acoustic model using connectionist temporal classification on a large-scale training corpus," *China Commun.*, vol. 14, no. 9, pp. 23–31, 2017, doi: 10.1109/CC.2017.8068761.

[63] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep Learning Based Text Classification: A Comprehensive Review," *arXiv*, vol. 1, no. 1, pp. 1–43, 2020.

[64] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-Term memory networks," *ACL-IJCNLP 2015 - 53rd Annu. Meet. Assoc. Comput. Linguist. 7th Int. Jt. Conf. Nat. Lang. Process. Asian Fed. Nat. Lang. Process. Proc. Conf.*, vol. 1, pp. 1556–1566, 2015, doi: 10.3115/v1/p15-1150.

[65] B. Jang, M. Kim, G. Harerimana, S. Kang, and J. W. Kim, "applied sciences Bi-LSTM Model to Increase Accuracy in Text Classification : Combining Word2vec CNN and Attention Mechanism," 2020.

[66] J. Xie, B. Chen, X. Gu, F. Liang, and X. Xu, "Self-Attention-Based BiLSTM Model for Short Text Fine-Grained Sentiment Classification," *IEEE Access*, vol. 7, pp. 180558–180570, 2019, doi: 10.1109/ACCESS.2019.2957510.

[67] M. Z. Amin and N. Nadeem, "Convolutional Neural Network : Text Classification Model for Open Domain Question Answering System".

[68] L. Alzubaidi *et al.*, *Review of deep learning : concepts , CNN architectures , challenges , applications , future directions*. Springer International Publishing, 2021. doi: 10.1186/s40537-021-00444-8.

[69] T. Binhuraib, "NLP with CNNs." [Online]. Available: https://towardsdatascience.com/nlp-with-cnns-a6aa743bdc1e

[70] D. Jurafsky and J. Martin, “Encoder-Decoder Models, Attention, and Contextual Embeddings,” *Speech Lang. Process.*, p. Chapter 10, 2020.

[71] Y. Gao, C. Herold, and Z. Yang, “Is Encoder-Decoder Redundant for Neural Machine Translation?,” 2019.

[72] B. Van Merri, “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches,” 2014.

[73] G. Tang, “An Analysis of Attention Mechanisms : The Case of Word Sense Disambiguation in Neural Machine Translation,” vol. 1, pp. 26–35, 2018.

[74] F. Stahlberg, “Neural machine translation: A review,” *J. Artif. Intell. Res.*, vol. 69, pp. 343–418, 2020, doi: 10.1613/JAIR.1.12007.

[75] Z. Tan, S. Wang, Z. Yang, G. Chen, and X. Huang, “Neural machine translation : A review of methods , resources , and tools,” *AI Open*, vol. 1, no. November 2020, pp. 5–21, 2021, doi: 10.1016/j.aiopen.2020.11.001.

[76] S. T. Abate and S. Atinafu, “Parallel Corpora for bi-Directional Statistical Machine Translation for Seven Ethiopian Language Pairs,” pp. 83–90, 2018.

[77] M. M. Woldeyohannis and M. Meshesha, “Experimenting statistical machine translation for ethiopic semitic languages: The case of Amharic-Tigrigna,” Springer International Publishing, 2018. doi: 10.1007/978-3-319-95153-9_13.

[78] G. W. GEBEYEHU, “GE EZ-AMHARIC MACHINE TRANSLATION USING DEEP LEARNING,” 2021.

[79] T. Kassa, “Morpheme-Based Bi-directional Ge’ez -Amharic Machine Translation,” 2018.

[80] Y. Wu *et al.*, “Google ’ s Neural Machine Translation System : Bridging the Gap between Human and Machine Translation,” pp. 1–23.

[81] M. G. Teshome, L. Besacier, G. Taye, and D. Teferi, “Phoneme-based English-Amharic Statistical Machine Translation,” in *AFRICON 2015*, 2015, pp. 1–5. doi: 10.1109/AFRCON.2015.7331921.

[82] A. Vaswani, “Attention Is All You Need,” *Conf. Neural Inf. Process. Syst. (NIPS 2017),*

Long Beach, CA, USA, no. Nips, 2017.

[83] D. Bahdanau, K. H. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

Appendix A. Tigrigna language orthographic Table

	ä	**u**	**i**	**a**	**e**	**(ə)**	**o**	**wä**	**wi**	**wa**	**we**	**wə**
h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ					
l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ					
ḥ	ሐ	ፎ	ሒ	ሓ	ሔ	ሕ	ሖ					

m	ᠮᠸ	ᠮᠦ	ᠮᠢ	ᠮᠠ	ᠮᠡ	ᠮᠣ	ᠮᠤ					
r	ᠷᠸ	ᠷᠦ	ᠷᠢ	ᠷᠠ	ᠷᠡ	ᠷᠣ	ᠷᠤ					
s	ᠰᠸ	ᠰᠦ	ᠰᠢ	ᠰᠠ	ᠰᠡ	ᠰᠣ	ᠰᠤ					
š	ᠱᠸ	ᠱᠦ	ᠱᠢ	ᠱᠠ	ᠱᠡ	ᠱᠣ	ᠱᠤ					
q	ᠺᠸ	ᠺᠦ	ᠺᠢ	ᠺᠠ	ᠺᠡ	ᠺᠣ	ᠺᠤ	ᠺᠸᠠ	ᠺᠸᠣ	ᠺᠠᠠ	ᠺᠡᠡ	ᠺᠣᠣ
ɋ	ᠼᠸ	ᠼᠦ	ᠼᠢ	ᠼᠠ	ᠼᠡ	ᠼᠣ	ᠼᠤ	ᠼᠸᠠ	ᠼᠸᠣ	ᠼᠠᠠ	ᠼᠡᠡ	ᠼᠣᠣ
b	ᠪᠸ	ᠪᠦ	ᠪᠢ	ᠪᠠ	ᠪᠡ	ᠪᠣ	ᠪᠤ					
v	ᠸᠸ	ᠸᠦ	ᠸᠢ	ᠸᠠ	ᠸᠡ	ᠸᠣ	ᠸᠤ					
t	ᠲᠸ	ᠲᠦ	ᠲᠢ	ᠲᠠ	ᠲᠡ	ᠲᠣ	ᠲᠤ					
č	ᠴᠸ	ᠴᠦ	ᠴᠢ	ᠴᠠ	ᠴᠡ	ᠴᠣ	ᠴᠤ					
n	ᠨᠸ	ᠨᠦ	ᠨᠢ	ᠨᠠ	ᠨᠡ	ᠨᠣ	ᠨᠤ					
ñ	ᠩᠸ	ᠩᠦ	ᠩᠢ	ᠩᠠ	ᠩᠡ	ᠩᠣ	ᠩᠤ					
ʼ	ᠠᠸ	ᠠᠦ	ᠠᠢ	ᠠᠠ	ᠠᠡ	ᠠᠣ	ᠠᠤ					

k	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኰ	ኲ	ኳ	ኴ	ኵ
k̲	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ	ዀ	Th	ኳ	ኴ	ኵ
w	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ					
ʿ	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ					
z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ					
ž	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ					
y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ					
d	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ					
ǧ	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ					
g	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጐ	ጓ	ጔ	ጕ	
ṭ	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ					
č̣	ጩ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ					
p̣	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ					

ṣ	𖹠	𖹡	𖹢	𖹣	𖹤	𖹥	𖹦					
ṣ	𖹧	𖹨	𖹩	𖹪	𖹫	𖹬	𖹭					
f	𖹮	𖹯	𖹰	𖹱	𖹲	𖹳	𖹴					
p	𖹵	𖹶	𖹷	𖹸	𖹹	𖹺	𖹻					
	ä	**u**	**i**	**a**	**e**	**(ə)**	**o**	**wä**	**wi**	**wa**	**we**	**wə**

Appendix B: Algorithms

```
ALGORITHM FOR TEXT CLEANING:
    DEFINE REGULAR EXPRESSION (R) #R Contains punctuation marks
    READ PARALLEL DATASET #Input
    WHILE READING LINE DO
     FOR C IN LINE #Read each character from each line
     IF C IS IN R #if C is punctuation or special character
     THEN
     REPLACE C WITH SPACE
     END IF
    END WHILE
    CLOSE FILE
```

Figure 41 Algorithm to remove special characters

```
ALGORITHM FOR NORMALIZATION OF DATASET:
    DEFINE NORMALIZATION LIST (N) WITH VALUE (V) # E.g. ሃኀሐኻ=>ሀ, ኍሑኹ=>ሁ...ሠ=>ሰ...
    READ THE TIGRIGNA CORPORA # This is input
    WHILE READING LINE DO
    FOR C IN LINE #Read each character from each line
     IF C IS IN N # if C is in a normalization list
     THEN
     REPLACE C WITH NORMALIZED VALUE (V) # E.g. if C is ሃ, ኀ or ሐ then replace C with U V
     END IF
    END WHILE
    CLOSE FILE
```

Figure 42 Algorithm to normalize the dataset

```
ALGORITHM FOR DATASET TOKENIZATION :
    DEFINE TOKENIZATION (STRING_LIST: LIST[STR], INT K) -> VOCAB: LIST[STR]:
    VOCAB = <LIST OF UNIQUE CHARACTERS IN STRING_LIST>
    READ PARALLEL CORPORA #The Input
    WHILE READING LINE DO
    FOR I IN RANGE(0, K+1):
    C_LEFT,C_RIGHT=MOST FREQUENT PAIR OF ADJACENT TOKENS IN STRING_LIST
    C_NEW = C_LEFT + C_RIGHT
    VOCAB = VOCAB + C_NEW
    REPLACE EACH OCCURENCE OF C_LEFT, C_RIGHT WITH C_NEW #Update the corpus
    RETURN(VOCAB)
    END WHILE
    CLOSE FILE
```

Figure 43 Algorithm used to tokenize the dataset

Appendix C: Training History

Training History of the model using Encoder Decoder with Bi-LSTM

```
Epoch 1/100
120/120 [==============================] - 43s 126ms/step - loss: 3.3074 -
accuracy: 0.6311 - val_loss: 2.9162 - val_accuracy: 0.6407
Epoch 2/100
120/120 [==============================] - 13s 111ms/step - loss: 2.8554 -
accuracy: 0.6396 - val_loss: 2.8156 - val_accuracy: 0.6412
Epoch 3/100
```

```
120/120 [==============================] - 13s 107ms/step - loss: 2.7345 -
accuracy: 0.6403 - val_loss: 2.6832 - val_accuracy: 0.6398
Epoch 4/100
120/120 [==============================] - 13s 108ms/step - loss: 2.6067 -
accuracy: 0.6412 - val_loss: 2.5747 - val_accuracy: 0.6390
Epoch 5/100
120/120 [==============================] - 12s 102ms/step - loss: 2.4762 -
accuracy: 0.6417 - val_loss: 2.4294 - val_accuracy: 0.6408
Epoch 6/100
120/120 [==============================] - 13s 107ms/step - loss: 2.3485 -
accuracy: 0.6421 - val_loss: 2.2976 - val_accuracy: 0.6428
Epoch 7/100
120/120 [==============================] - 13s 108ms/step - loss: 2.2382 -
accuracy: 0.6431 - val_loss: 2.2201 - val_accuracy: 0.6432
Epoch 8/100
120/120 [==============================] - 13s 109ms/step - loss: 2.1189 -
accuracy: 0.6450 - val_loss: 2.0608 - val_accuracy: 0.6447
Epoch 9/100
120/120 [==============================] - 14s 115ms/step - loss: 2.0020 -
accuracy: 0.6462 - val_loss: 1.9622 - val_accuracy: 0.6463
Epoch 10/100
120/120 [==============================] - 13s 112ms/step - loss: 1.9190 -
accuracy: 0.6489 - val_loss: 1.8858 - val_accuracy: 0.6500
Epoch 11/100
120/120 [==============================] - 13s 109ms/step - loss: 1.8298 -
accuracy: 0.6520 - val_loss: 1.7740 - val_accuracy: 0.6557
Epoch 12/100
120/120 [==============================] - 14s 117ms/step - loss: 1.7576 -
accuracy: 0.6559 - val_loss: 1.7282 - val_accuracy: 0.6568
Epoch 13/100
120/120 [==============================] - 16s 136ms/step - loss: 1.6877 -
accuracy: 0.6596 - val_loss: 1.6458 - val_accuracy: 0.6648
Epoch 14/100
120/120 [==============================] - 13s 110ms/step - loss: 1.6053 -
accuracy: 0.6656 - val_loss: 1.5778 - val_accuracy: 0.6698
Epoch 15/100
120/120 [==============================] - 13s 108ms/step - loss: 1.5327 -
accuracy: 0.6740 - val_loss: 1.5488 - val_accuracy: 0.6695
Epoch 16/100
120/120 [==============================] - 14s 116ms/step - loss: 1.4528 -
accuracy: 0.6816 - val_loss: 1.4148 - val_accuracy: 0.6862
Epoch 17/100
120/120 [==============================] - 14s 114ms/step - loss: 1.3956 -
accuracy: 0.6892 - val_loss: 1.3672 - val_accuracy: 0.6938
Epoch 18/100
120/120 [==============================] - 15s 121ms/step - loss: 1.3244 -
accuracy: 0.6996 - val_loss: 1.2962 - val_accuracy: 0.7072
Epoch 19/100
120/120 [==============================] - 13s 109ms/step - loss: 1.2751 -
accuracy: 0.7061 - val_loss: 1.2301 - val_accuracy: 0.7182
Epoch 20/100
120/120 [==============================] - 13s 109ms/step - loss: 1.2209 -
accuracy: 0.7146 - val_loss: 1.2008 - val_accuracy: 0.7250
Epoch 21/100
```

```
120/120 [==============================] - 15s 121ms/step - loss: 1.2004 -
accuracy: 0.7176 - val_loss: 1.1861 - val_accuracy: 0.7253
Epoch 22/100
120/120 [==============================] - 14s 119ms/step - loss: 1.1593 -
accuracy: 0.7256 - val_loss: 1.1228 - val_accuracy: 0.7362
Epoch 23/100
120/120 [==============================] - 13s 110ms/step - loss: 1.0757 -
accuracy: 0.7408 - val_loss: 1.0533 - val_accuracy: 0.7580
Epoch 24/100
120/120 [==============================] - 13s 107ms/step - loss: 1.0284 -
accuracy: 0.7487 - val_loss: 1.0198 - val_accuracy: 0.7570
Epoch 25/100
120/120 [==============================] - 13s 108ms/step - loss: 0.9957 -
accuracy: 0.7559 - val_loss: 0.9638 - val_accuracy: 0.7733
Epoch 26/100
120/120 [==============================] - 13s 106ms/step - loss: 0.9668 -
accuracy: 0.7627 - val_loss: 0.9549 - val_accuracy: 0.7677
Epoch 27/100
120/120 [==============================] - 12s 102ms/step - loss: 0.9388 -
accuracy: 0.7683 - val_loss: 0.9325 - val_accuracy: 0.7710
Epoch 28/100
120/120 [==============================] - 12s 101ms/step - loss: 0.9191 -
accuracy: 0.7722 - val_loss: 0.8541 - val_accuracy: 0.7920
Epoch 29/100
120/120 [==============================] - 13s 106ms/step - loss: 0.8776 -
accuracy: 0.7794 - val_loss: 0.8421 - val_accuracy: 0.7895
Epoch 30/100
120/120 [==============================] - 13s 106ms/step - loss: 0.8599 -
accuracy: 0.7837 - val_loss: 0.8436 - val_accuracy: 0.7920
Epoch 31/100
120/120 [==============================] - 13s 108ms/step - loss: 0.8504 -
accuracy: 0.7864 - val_loss: 0.7967 - val_accuracy: 0.8005
Epoch 32/100
120/120 [==============================] - 13s 110ms/step - loss: 0.8151 -
accuracy: 0.7918 - val_loss: 0.7892 - val_accuracy: 0.7998
Epoch 33/100
120/120 [==============================] - 13s 106ms/step - loss: 0.8000 -
accuracy: 0.7957 - val_loss: 0.7567 - val_accuracy: 0.8107
Epoch 34/100
120/120 [==============================] - 13s 109ms/step - loss: 0.7694 -
accuracy: 0.8019 - val_loss: 0.7085 - val_accuracy: 0.8198
Epoch 35/100
120/120 [==============================] - 14s 120ms/step - loss: 0.7424 -
accuracy: 0.8088 - val_loss: 0.7529 - val_accuracy: 0.8125
Epoch 36/100
120/120 [==============================] - 14s 114ms/step - loss: 0.7401 -
accuracy: 0.8089 - val_loss: 0.7071 - val_accuracy: 0.8232
Epoch 37/100
120/120 [==============================] - 14s 114ms/step - loss: 0.7288 -
accuracy: 0.8098 - val_loss: 0.7044 - val_accuracy: 0.8170
Epoch 38/100
120/120 [==============================] - 13s 110ms/step - loss: 0.6930 -
accuracy: 0.8200 - val_loss: 0.6702 - val_accuracy: 0.8280
Epoch 39/100
```

```
120/120 [==============================] - 14s 114ms/step - loss: 0.6614 -
accuracy: 0.8263 - val_loss: 0.6574 - val_accuracy: 0.8323
Epoch 40/100
120/120 [==============================] - 14s 120ms/step - loss: 0.6507 -
accuracy: 0.8303 - val_loss: 0.6462 - val_accuracy: 0.8305
Epoch 41/100
120/120 [==============================] - 12s 103ms/step - loss: 0.6457 -
accuracy: 0.8287 - val_loss: 0.5707 - val_accuracy: 0.8497
Epoch 42/100
120/120 [==============================] - 13s 110ms/step - loss: 0.6607 -
accuracy: 0.8246 - val_loss: 0.5696 - val_accuracy: 0.8520
Epoch 43/100
120/120 [==============================] - 15s 126ms/step - loss: 0.6147 -
accuracy: 0.8362 - val_loss: 0.5674 - val_accuracy: 0.8530
Epoch 44/100
120/120 [==============================] - 12s 101ms/step - loss: 0.6303 -
accuracy: 0.8322 - val_loss: 0.5956 - val_accuracy: 0.8455
Epoch 45/100
120/120 [==============================] - 12s 101ms/step - loss: 0.5977 -
accuracy: 0.8426 - val_loss: 0.5303 - val_accuracy: 0.8597
Epoch 46/100
120/120 [==============================] - 14s 120ms/step - loss: 0.5707 -
accuracy: 0.8472 - val_loss: 0.5296 - val_accuracy: 0.8633
Epoch 47/100
120/120 [==============================] - 13s 109ms/step - loss: 0.5675 -
accuracy: 0.8471 - val_loss: 0.5557 - val_accuracy: 0.8532
Epoch 48/100
120/120 [==============================] - 13s 111ms/step - loss: 0.6239 -
accuracy: 0.8336 - val_loss: 0.6955 - val_accuracy: 0.8197
Epoch 49/100
120/120 [==============================] - 13s 107ms/step - loss: 0.6841 -
accuracy: 0.8181 - val_loss: 0.6063 - val_accuracy: 0.8392
Epoch 50/100
120/120 [==============================] - 13s 110ms/step - loss: 0.6423 -
accuracy: 0.8292 - val_loss: 0.7147 - val_accuracy: 0.8185
Epoch 51/100
120/120 [==============================] - 13s 110ms/step - loss: 0.6767 -
accuracy: 0.8210 - val_loss: 0.6880 - val_accuracy: 0.8252
```

Training history of the model using Encoder Decoder with LSTM

```
Epoch 1/100
75/75 [==============================] - 17s 94ms/step - loss: 2.8734 - accur
acy: 0.6574 - val_loss: 2.2046 - val_accuracy: 0.6755
Epoch 2/100
75/75 [==============================] - 6s 82ms/step - loss: 2.1581 - accura
cy: 0.6765 - val_loss: 2.1248 - val_accuracy: 0.6756
Epoch 3/100
75/75 [==============================] - 8s 109ms/step - loss: 2.1318 - accur
acy: 0.6770 - val_loss: 2.1108 - val_accuracy: 0.6756
Epoch 4/100
```

```
75/75 [==============================] - 7s 100ms/step - loss: 2.0994 - accur
acy: 0.6769 - val_loss: 2.1108 - val_accuracy: 0.6764
Epoch 5/100
75/75 [==============================] - 4s 52ms/step - loss: 2.0796 - accura
cy: 0.6771 - val_loss: 2.0681 - val_accuracy: 0.6771
Epoch 6/100
75/75 [==============================] - 4s 52ms/step - loss: 2.0558 - accura
cy: 0.6780 - val_loss: 2.0690 - val_accuracy: 0.6756
Epoch 7/100
75/75 [==============================] - 4s 51ms/step - loss: 2.0367 - accura
cy: 0.6795 - val_loss: 2.0411 - val_accuracy: 0.6771
Epoch 8/100
75/75 [==============================] - 4s 52ms/step - loss: 2.0293 - accura
cy: 0.6795 - val_loss: 2.0193 - val_accuracy: 0.6796
Epoch 9/100
75/75 [==============================] - 4s 57ms/step - loss: 2.0126 - accura
cy: 0.6808 - val_loss: 2.0157 - val_accuracy: 0.6809
Epoch 10/100
75/75 [==============================] - 4s 55ms/step - loss: 2.0055 - accura
cy: 0.6806 - val_loss: 2.0013 - val_accuracy: 0.6809
Epoch 11/100
75/75 [==============================] - 4s 51ms/step - loss: 1.9936 - accura
cy: 0.6812 - val_loss: 1.9959 - val_accuracy: 0.6804
Epoch 12/100
75/75 [==============================] - 4s 53ms/step - loss: 1.9868 - accura
cy: 0.6806 - val_loss: 1.9866 - val_accuracy: 0.6820
Epoch 13/100
75/75 [==============================] - 4s 53ms/step - loss: 1.9766 - accura
cy: 0.6811 - val_loss: 1.9744 - val_accuracy: 0.6807
Epoch 14/100
75/75 [==============================] - 4s 56ms/step - loss: 1.9675 - accura
cy: 0.6811 - val_loss: 1.9711 - val_accuracy: 0.6818
Epoch 15/100
75/75 [==============================] - 4s 55ms/step - loss: 1.9612 - accura
cy: 0.6808 - val_loss: 1.9621 - val_accuracy: 0.6818
Epoch 16/100
75/75 [==============================] - 4s 54ms/step - loss: 1.9344 - accura
cy: 0.6817 - val_loss: 1.9086 - val_accuracy: 0.6831
Epoch 17/100
75/75 [==============================] - 4s 56ms/step - loss: 1.8920 - accura
cy: 0.6826 - val_loss: 1.8650 - val_accuracy: 0.6831
Epoch 18/100
75/75 [==============================] - 4s 54ms/step - loss: 1.8448 - accura
cy: 0.6838 - val_loss: 1.8120 - val_accuracy: 0.6842
```

```
Epoch 19/100
75/75 [==============================] - 4s 56ms/step - loss: 1.8080 - accura
cy: 0.6853 - val_loss: 1.7911 - val_accuracy: 0.6873
Epoch 20/100
75/75 [==============================] - 4s 56ms/step - loss: 1.7682 - accura
cy: 0.6874 - val_loss: 1.7515 - val_accuracy: 0.6876
Epoch 21/100
75/75 [==============================] - 4s 54ms/step - loss: 1.7408 - accura
cy: 0.6894 - val_loss: 1.7332 - val_accuracy: 0.6896
Epoch 22/100
75/75 [==============================] - 4s 57ms/step - loss: 1.7232 - accura
cy: 0.6899 - val_loss: 1.7035 - val_accuracy: 0.6893
Epoch 23/100
75/75 [==============================] - 5s 60ms/step - loss: 1.6994 - accura
cy: 0.6900 - val_loss: 1.6810 - val_accuracy: 0.6904
Epoch 24/100
75/75 [==============================] - 4s 53ms/step - loss: 1.6836 - accura
cy: 0.6904 - val_loss: 1.6743 - val_accuracy: 0.6898
Epoch 25/100
75/75 [==============================] - 4s 60ms/step - loss: 1.6636 - accura
cy: 0.6917 - val_loss: 1.6544 - val_accuracy: 0.6915
Epoch 26/100
75/75 [==============================] - 4s 57ms/step - loss: 1.6422 - accura
cy: 0.6915 - val_loss: 1.6260 - val_accuracy: 0.6920
Epoch 27/100
75/75 [==============================] - 4s 53ms/step - loss: 1.6196 - accura
cy: 0.6924 - val_loss: 1.6084 - val_accuracy: 0.6944
Epoch 28/100
75/75 [==============================] - 4s 51ms/step - loss: 1.6100 - accura
cy: 0.6921 - val_loss: 1.5992 - val_accuracy: 0.6918
Epoch 29/100
75/75 [==============================] - 4s 57ms/step - loss: 1.5981 - accura
cy: 0.6921 - val_loss: 1.5661 - val_accuracy: 0.6929
Epoch 30/100
75/75 [==============================] - 4s 56ms/step - loss: 1.5654 - accura
cy: 0.6932 - val_loss: 1.5569 - val_accuracy: 0.6935
Epoch 31/100
75/75 [==============================] - 5s 61ms/step - loss: 1.5508 - accura
cy: 0.6936 - val_loss: 1.5526 - val_accuracy: 0.6925
Epoch 32/100
75/75 [==============================] - 5s 65ms/step - loss: 1.5411 - accura
cy: 0.6936 - val_loss: 1.5314 - val_accuracy: 0.6938
Epoch 33/100
```

```
75/75 [==============================] - 6s 78ms/step - loss: 1.5327 - accura
cy: 0.6936 - val_loss: 1.5205 - val_accuracy: 0.6947
Epoch 34/100
75/75 [==============================] - 5s 71ms/step - loss: 1.5148 - accura
cy: 0.6939 - val_loss: 1.5048 - val_accuracy: 0.6935
Epoch 35/100
75/75 [==============================] - 5s 61ms/step - loss: 1.5061 - accura
cy: 0.6951 - val_loss: 1.4950 - val_accuracy: 0.6947
Epoch 36/100
75/75 [==============================] - 4s 50ms/step - loss: 1.4893 - accura
cy: 0.6954 - val_loss: 1.4699 - val_accuracy: 0.6955
Epoch 37/100
75/75 [==============================] - 4s 48ms/step - loss: 1.4736 - accura
cy: 0.6964 - val_loss: 1.4571 - val_accuracy: 0.6956
Epoch 38/100
75/75 [==============================] - 4s 52ms/step - loss: 1.4630 - accura
cy: 0.6966 - val_loss: 1.4530 - val_accuracy: 0.6962
Epoch 39/100
75/75 [==============================] - 4s 50ms/step - loss: 1.4559 - accura
cy: 0.6975 - val_loss: 1.4483 - val_accuracy: 0.6976
Epoch 40/100
75/75 [==============================] - 5s 61ms/step - loss: 1.4373 - accura
cy: 0.6986 - val_loss: 1.4111 - val_accuracy: 0.6973
Epoch 41/100
75/75 [==============================] - 7s 93ms/step - loss: 1.4235 - accura
cy: 0.6979 - val_loss: 1.4070 - val_accuracy: 0.6964
Epoch 42/100
75/75 [==============================] - 7s 92ms/step - loss: 1.4102 - accura
cy: 0.6988 - val_loss: 1.3964 - val_accuracy: 0.6982
Epoch 43/100
75/75 [==============================] - 7s 88ms/step - loss: 1.3957 - accura
cy: 0.6993 - val_loss: 1.3805 - val_accuracy: 0.6980
Epoch 44/100
75/75 [==============================] - 4s 57ms/step - loss: 1.3837 - accura
cy: 0.7000 - val_loss: 1.3692 - val_accuracy: 0.7005
Epoch 45/100
75/75 [==============================] - 4s 53ms/step - loss: 1.3740 - accura
cy: 0.7003 - val_loss: 1.3754 - val_accuracy: 0.7005
Epoch 46/100
75/75 [==============================] - 4s 51ms/step - loss: 1.3698 - accura
cy: 0.6998 - val_loss: 1.3461 - val_accuracy: 0.6991
Epoch 47/100
75/75 [==============================] - 4s 59ms/step - loss: 1.3646 - accura
cy: 0.7005 - val_loss: 1.3541 - val_accuracy: 0.7011
```

```
Epoch 48/100
75/75 [==============================] - 4s 56ms/step - loss: 1.3495 - accura
cy: 0.7014 - val_loss: 1.3300 - val_accuracy: 0.7033
Epoch 49/100
75/75 [==============================] - 5s 68ms/step - loss: 1.3355 - accura
cy: 0.7033 - val_loss: 1.3303 - val_accuracy: 0.7002
Epoch 50/100
75/75 [==============================] - 5s 63ms/step - loss: 1.3227 - accura
cy: 0.7044 - val_loss: 1.3080 - val_accuracy: 0.7064
Epoch 51/100
75/75 [==============================] - 4s 59ms/step - loss: 1.3133 - accura
cy: 0.7050 - val_loss: 1.3239 - val_accuracy: 0.7038
Epoch 52/100
75/75 [==============================] - 5s 63ms/step - loss: 1.3140 - accura
cy: 0.7050 - val_loss: 1.2954 - val_accuracy: 0.7073
Epoch 53/100
75/75 [==============================] - 5s 65ms/step - loss: 1.2989 - accura
cy: 0.7061 - val_loss: 1.2914 - val_accuracy: 0.7060
Epoch 54/100
75/75 [==============================] - 5s 71ms/step - loss: 1.2872 - accura
cy: 0.7088 - val_loss: 1.2638 - val_accuracy: 0.7109
Epoch 55/100
75/75 [==============================] - 5s 73ms/step - loss: 1.2753 - accura
cy: 0.7092 - val_loss: 1.2605 - val_accuracy: 0.7125
Epoch 56/100
75/75 [==============================] - 5s 64ms/step - loss: 1.2705 - accura
cy: 0.7107 - val_loss: 1.2533 - val_accuracy: 0.7122
Epoch 57/100
75/75 [==============================] - 4s 58ms/step - loss: 1.2584 - accura
cy: 0.7118 - val_loss: 1.2445 - val_accuracy: 0.7142
Epoch 58/100
75/75 [==============================] - 4s 51ms/step - loss: 1.2549 - accura
cy: 0.7114 - val_loss: 1.2364 - val_accuracy: 0.7155
Epoch 59/100
75/75 [==============================] - 4s 53ms/step - loss: 1.2378 - accura
cy: 0.7163 - val_loss: 1.2355 - val_accuracy: 0.7167
Epoch 60/100
75/75 [==============================] - 4s 52ms/step - loss: 1.2446 - accura
cy: 0.7146 - val_loss: 1.2469 - val_accuracy: 0.7187
Epoch 61/100
75/75 [==============================] - 4s 53ms/step - loss: 1.2355 - accura
cy: 0.7178 - val_loss: 1.2330 - val_accuracy: 0.7193
Epoch 62/100
```

```
75/75 [==============================] - 4s 55ms/step - loss: 1.2258 - accura
cy: 0.7174 - val_loss: 1.2163 - val_accuracy: 0.7184
Epoch 63/100
75/75 [==============================] - 4s 55ms/step - loss: 1.2244 - accura
cy: 0.7184 - val_loss: 1.2073 - val_accuracy: 0.7218
Epoch 64/100
75/75 [==============================] - 4s 54ms/step - loss: 1.2059 - accura
cy: 0.7198 - val_loss: 1.1829 - val_accuracy: 0.7267
Epoch 65/100
75/75 [==============================] - 4s 54ms/step - loss: 1.1933 - accura
cy: 0.7225 - val_loss: 1.1679 - val_accuracy: 0.7245
Epoch 66/100
75/75 [==============================] - 4s 55ms/step - loss: 1.1744 - accura
cy: 0.7249 - val_loss: 1.1534 - val_accuracy: 0.7315
Epoch 67/100
75/75 [==============================] - 4s 53ms/step - loss: 1.1704 - accura
cy: 0.7253 - val_loss: 1.1483 - val_accuracy: 0.7302
Epoch 68/100
75/75 [==============================] - 4s 55ms/step - loss: 1.1669 - accura
cy: 0.7262 - val_loss: 1.1331 - val_accuracy: 0.7375
Epoch 69/100
75/75 [==============================] - 4s 56ms/step - loss: 1.1526 - accura
cy: 0.7292 - val_loss: 1.1566 - val_accuracy: 0.7300
Epoch 70/100
75/75 [==============================] - 4s 55ms/step - loss: 1.1468 - accura
cy: 0.7294 - val_loss: 1.1220 - val_accuracy: 0.7373
Epoch 71/100
75/75 [==============================] - 4s 60ms/step - loss: 1.1437 - accura
cy: 0.7311 - val_loss: 1.1388 - val_accuracy: 0.7304
Epoch 72/100
75/75 [==============================] - 4s 56ms/step - loss: 1.1512 - accura
cy: 0.7290 - val_loss: 1.1105 - val_accuracy: 0.7409
Epoch 73/100
75/75 [==============================] - 4s 55ms/step - loss: 1.1320 - accura
cy: 0.7310 - val_loss: 1.1289 - val_accuracy: 0.7364
Epoch 74/100
75/75 [==============================] - 4s 57ms/step - loss: 1.1269 - accura
cy: 0.7322 - val_loss: 1.1200 - val_accuracy: 0.7364
Epoch 75/100
75/75 [==============================] - 4s 58ms/step - loss: 1.1219 - accura
cy: 0.7323 - val_loss: 1.1081 - val_accuracy: 0.7340
Epoch 76/100
75/75 [==============================] - 4s 56ms/step - loss: 1.1152 - accura
cy: 0.7331 - val_loss: 1.0946 - val_accuracy: 0.7402
```

```
Epoch 77/100
75/75 [==============================] - 5s 61ms/step - loss: 1.0991 - accura
cy: 0.7370 - val_loss: 1.1040 - val_accuracy: 0.7416
Epoch 78/100
75/75 [==============================] - 8s 110ms/step - loss: 1.1004 - accur
acy: 0.7380 - val_loss: 1.0815 - val_accuracy: 0.7433
Epoch 79/100
75/75 [==============================] - 8s 110ms/step - loss: 1.0890 - accur
acy: 0.7382 - val_loss: 1.0466 - val_accuracy: 0.7476
Epoch 80/100
75/75 [==============================] - 7s 90ms/step - loss: 1.0678 - accura
cy: 0.7429 - val_loss: 1.0591 - val_accuracy: 0.7471
Epoch 81/100
75/75 [==============================] - 6s 85ms/step - loss: 1.0735 - accura
cy: 0.7423 - val_loss: 1.0495 - val_accuracy: 0.7480
Epoch 82/100
75/75 [==============================] - 4s 59ms/step - loss: 1.0686 - accura
cy: 0.7420 - val_loss: 1.0602 - val_accuracy: 0.7433
Epoch 83/100
75/75 [==============================] - 4s 59ms/step - loss: 1.0574 - accura
cy: 0.7445 - val_loss: 1.0408 - val_accuracy: 0.7491
Epoch 84/100
75/75 [==============================] - 4s 56ms/step - loss: 1.0522 - accura
cy: 0.7448 - val_loss: 1.0159 - val_accuracy: 0.7515
Epoch 85/100
75/75 [==============================] - 4s 59ms/step - loss: 1.0373 - accura
cy: 0.7480 - val_loss: 1.0202 - val_accuracy: 0.7495
Epoch 86/100
75/75 [==============================] - 4s 56ms/step - loss: 1.0360 - accura
cy: 0.7477 - val_loss: 1.0245 - val_accuracy: 0.7547
Epoch 87/100
75/75 [==============================] - 5s 62ms/step - loss: 1.0300 - accura
cy: 0.7498 - val_loss: 1.0239 - val_accuracy: 0.7551
Epoch 88/100
75/75 [==============================] - 4s 56ms/step - loss: 1.0237 - accura
cy: 0.7498 - val_loss: 1.0064 - val_accuracy: 0.7576
Epoch 89/100
75/75 [==============================] - 4s 52ms/step - loss: 1.0130 - accura
cy: 0.7525 - val_loss: 0.9891 - val_accuracy: 0.7609
Epoch 90/100
75/75 [==============================] - 4s 60ms/step - loss: 1.0088 - accura
cy: 0.7537 - val_loss: 0.9946 - val_accuracy: 0.7602
Epoch 91/100
```

```
75/75 [==============================] - 4s 55ms/step - loss: 1.0009 - accura
cy: 0.7551 - val_loss: 0.9728 - val_accuracy: 0.7584
Epoch 92/100
75/75 [==============================] - 4s 55ms/step - loss: 0.9883 - accura
cy: 0.7570 - val_loss: 0.9685 - val_accuracy: 0.7649
Epoch 93/100
75/75 [==============================] - 4s 58ms/step - loss: 0.9802 - accura
cy: 0.7600 - val_loss: 0.9554 - val_accuracy: 0.7685
Epoch 94/100
75/75 [==============================] - 4s 56ms/step - loss: 0.9749 - accura
cy: 0.7610 - val_loss: 0.9405 - val_accuracy: 0.7649
Epoch 95/100
75/75 [==============================] - 4s 58ms/step - loss: 0.9645 - accura
cy: 0.7616 - val_loss: 0.9412 - val_accuracy: 0.7704
Epoch 96/100
75/75 [==============================] - 4s 54ms/step - loss: 0.9651 - accura
cy: 0.7613 - val_loss: 0.9420 - val_accuracy: 0.7665
Epoch 97/100
75/75 [==============================] - 4s 55ms/step - loss: 0.9569 - accura
cy: 0.7625 - val_loss: 0.9426 - val_accuracy: 0.7704
Epoch 98/100
75/75 [==============================] - 4s 50ms/step - loss: 0.9522 - accura
cy: 0.7643 - val_loss: 0.9439 - val_accuracy: 0.7680
Epoch 99/100
75/75 [==============================] - 4s 50ms/step - loss: 0.9471 - accura
cy: 0.7646 - val_loss: 0.9390 - val_accuracy: 0.7675
Epoch 100/100
75/75 [==============================] - 4s 51ms/step - loss: 0.9494 - accura
cy: 0.7638 - val_loss: 0.9276 - val_accuracy: 0.7713
```

Training history of the model using Attention mechanism

```
Epoch 1/60
5/5 [==============================] - ETA: 0s - loss: 6.9325 - accuracy: 0.3
246INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 50s 8s/step - loss: 6.9325 - accuracy:
0.3246 - val_loss: 4.4323 - val_accuracy: 0.4273
Epoch 2/60
5/5 [==============================] - ETA: 0s - loss: 4.3500 - accuracy: 0.4
149INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 35s 8s/step - loss: 4.3500 - accuracy:
0.4149 - val_loss: 4.1993 - val_accuracy: 0.4327
```

```
Epoch 3/60
5/5 [==============================] - ETA: 0s - loss: 3.9490 - accuracy: 0.4
232INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 36s 9s/step - loss: 3.9490 - accuracy:
0.4232 - val_loss: 4.2830 - val_accuracy: 0.4273
Epoch 4/60
5/5 [==============================] - ETA: 0s - loss: 3.7824 - accuracy: 0.4
192INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 36s 8s/step - loss: 3.7824 - accuracy:
0.4192 - val_loss: 4.2629 - val_accuracy: 0.4655
Epoch 5/60
5/5 [==============================] - ETA: 0s - loss: 3.6830 - accuracy: 0.4
337INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 32s 7s/step - loss: 3.6830 - accuracy:
0.4337 - val_loss: 4.3303 - val_accuracy: 0.4473
Epoch 6/60
5/5 [==============================] - ETA: 0s - loss: 3.6018 - accuracy: 0.4
400INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 38s 9s/step - loss: 3.6018 - accuracy:
0.4400 - val_loss: 4.2815 - val_accuracy: 0.4655
Epoch 7/60
5/5 [==============================] - ETA: 0s - loss: 3.5092 - accuracy: 0.4
552INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 35s 8s/step - loss: 3.5092 - accuracy:
0.4552 - val_loss: 4.3541 - val_accuracy: 0.4691
Epoch 8/60
5/5 [==============================] - ETA: 0s - loss: 3.4295 - accuracy: 0.4
560INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 32s 7s/step - loss: 3.4295 - accuracy:
0.4560 - val_loss: 4.3215 - val_accuracy: 0.4836
Epoch 9/60
5/5 [==============================] - ETA: 0s - loss: 3.3497 - accuracy: 0.4
927INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 34s 8s/step - loss: 3.3497 - accuracy:
0.4927 - val_loss: 4.3330 - val_accuracy: 0.4927
Epoch 10/60
```

```
5/5 [==============================] - ETA: 0s - loss: 3.2759 - accuracy: 0.5
059INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 33s 8s/step - loss: 3.2759 - accuracy:
0.5059 - val_loss: 4.3046 - val_accuracy: 0.5036
Epoch 11/60
5/5 [==============================] - ETA: 0s - loss: 3.2065 - accuracy: 0.5
103INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 32s 8s/step - loss: 3.2065 - accuracy:
0.5103 - val_loss: 4.3504 - val_accuracy: 0.5073
Epoch 12/60
5/5 [==============================] - ETA: 0s - loss: 3.1352 - accuracy: 0.5
123INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 33s 8s/step - loss: 3.1352 - accuracy:
0.5123 - val_loss: 4.4159 - val_accuracy: 0.5127
Epoch 13/60
5/5 [==============================] - ETA: 0s - loss: 3.0636 - accuracy: 0.5
135INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 36s 8s/step - loss: 3.0636 - accuracy:
0.5135 - val_loss: 4.4945 - val_accuracy: 0.5055
Epoch 14/60
5/5 [==============================] - ETA: 0s - loss: 2.9861 - accuracy: 0.5
154INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 36s 8s/step - loss: 2.9861 - accuracy:
0.5154 - val_loss: 4.5542 - val_accuracy: 0.5164
Epoch 15/60
5/5 [==============================] - ETA: 0s - loss: 2.9089 - accuracy: 0.5
180INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 34s 8s/step - loss: 2.9089 - accuracy:
0.5180 - val_loss: 4.6521 - val_accuracy: 0.5164
Epoch 16/60
5/5 [==============================] - ETA: 0s - loss: 2.8354 - accuracy: 0.5
192INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 41s 10s/step - loss: 2.8354 - accuracy
: 0.5192 - val_loss: 4.7596 - val_accuracy: 0.5182
Epoch 17/60
5/5 [==============================] - ETA: 0s - loss: 2.7777 - accuracy: 0.5
188INFO:tensorflow:Assets written to: model1\assets
```

```
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 169s 41s/step - loss: 2.7777 - accurac
y: 0.5188 - val_loss: 4.8050 - val_accuracy: 0.5200
Epoch 18/60
5/5 [==============================] - ETA: 0s - loss: 2.7247 - accuracy: 0.5
190INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 168s 41s/step - loss: 2.7247 - accurac
y: 0.5190 - val_loss: 4.8789 - val_accuracy: 0.5055
Epoch 19/60
5/5 [==============================] - ETA: 0s - loss: 2.6635 - accuracy: 0.5
248INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 127s 31s/step - loss: 2.6635 - accurac
y: 0.5248 - val_loss: 4.9839 - val_accuracy: 0.5073
Epoch 20/60
5/5 [==============================] - ETA: 0s - loss: 2.6049 - accuracy: 0.5
240INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 42s 10s/step - loss: 2.6049 - accuracy
: 0.5240 - val_loss: 4.9901 - val_accuracy: 0.5091
Epoch 21/60
5/5 [==============================] - ETA: 0s - loss: 2.5548 - accuracy: 0.5
269INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 39s 9s/step - loss: 2.5548 - accuracy:
0.5269 - val_loss: 5.1239 - val_accuracy: 0.5091
Epoch 22/60
5/5 [==============================] - ETA: 0s - loss: 2.5131 - accuracy: 0.5
323INFO:tensorflow:Assets written to: model1\assets
INFO:tensorflow:Assets written to: model1\assets
5/5 [==============================] - 46s 11s/step - loss: 2.5131 - accuracy
: 0.5323 - val_loss: 5.0906 - val_accuracy: 0.5055
```

Appendix D: Sample Tigrigna stop words

’ምበር	ስለዚ	ንሕና	እሞ
’ሞ	ስለዝበላ	ንሱ	እተን
’ቲ	ሽዑ	ንሳ	እቲ
’ታ	ቅድሚ	ንሳቶም	እታ
’ኳ	በለ	ንስኺ	እቶም
’ውን	በቲ	ንስኻ	እንተ
’ዚ	በዚ	ንስኻትኩም	እንተሎ
’የ	ብምባል	ንስኻትክን	እንተኾነ
’ዩ	ብተወሳኺ	ንዓይ	እንታይ
’ያ	ብኽመይ	ኢለ	እንከሎ
’ዮም	ብዘይ	ኢሉ	እኳ
’ዮን	ብዘይካ	ኢላ	እዋን
ልዕሊ	ብዙሕ	ኢልካ	እውን
ሒዙ	ብዛዕባ	ኢሎም	እዚ
ሒዛ	ብፍላይ	ኢና	እዛ