BI-DIRECTIONAL NEURAL MACHINE TRANSLATION FOR ENGLISH - SIDAAMU AFOO

MASTER'S PROGRAM (MSc)

ANDU ALEMAYEHU

HAWASSA UNIVERSITY, HAWASSA, ETHIOPIA

MARCH, 2023

BI-DIRECTIONAL NEURAL MACHINE TRANSLATION FOR ENGLISH - SIDAAMU AFOO

ANDU ALEMAYEHU

ADVISOR

MICHAEL MELESE (Ph.D.)

THESIS SUBMITTED TO

HAWASSA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

INSTITUTE OF TECHNOLOGY, SCHOOL OF GRADUATE STUDIES, HAWASSA UNIVERSITY, HAWASSA, ETHIOPIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTERS OF SCIENCE IN COMPUTER SCIENCE

MARCH, 2023

Declaration

I hereby declare that this MSc Specialty or equivalent thesis is my original work and has not been presented for a degree in any other university, and all sources of the material used for this thesis/dissertation has been duly acknowledged.

Name: - Andu Alemayehu Chone

Signature: _____

This MSc Specialty or equivalent thesis has been submitted for examination with my approval as thesis advisor.

Name: - Michael Melese (Ph.D.)

Signature: _____

Place and Date of Submission: _____

SCHOOL OF GRADUATE STUDIES

HAWASSA UNIVERSITY

ADVISORS' APPROVAL SHEET

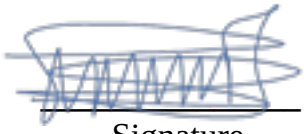
This is to certify that the thesis entitled "Bi-directional Neural Machine Translation for English - Sidaamu Afoo" was submitted in partial fulfillment of the requirements for the degree of Master's Degree with specialization in COMPUTER SCIENCE for, the Graduate Program of the Department of COMPUTER SCIENCE, and has been carried out by **Andu Alemayehu** (ID NO. GPCoSc/0002/12), is conducted under my supervision. Therefore, I recommend that the student has fulfilled the requirements and hence hereby can submit the thesis to the Department.

Michael Melese (Ph.D.)	_____	_____
Name of major advisor	Signature	Date
Girmay Tekle	_____	_____
Name of co-advisor	Signature	Date

SCHOOL OF GRADUATE STUDIES

HAWASSA UNIVERSITY EXAMINERS' APPROVAL SHEET

We, the undersigned, members of the Board of Examiners of the final open defense by **Andu Alemayehu Chone** have read and evaluated his/her thesis entitled "***Bi-Directional Neural Machine Translation for English – Sidaamu Afoo***", and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirements for the degree.

Michael Melese (Ph.D)		May 24,2023
Name of Major Advisor	Signature	Date
Girmay Tekle	_____	_____
Name of Co-Advisor	Signature	Date
Andargachew Mekonnen (Ph.D candidate)	_____	May 22, 2023
Name of Internal Examiner-I	Signature	Date
Teshager Kasu	_____	_____
Name of Internal Examiner-II	Signature	Date
Wondwossen Mulugeta (Ph.D)		May 22, 2023
Name of External examiner	Signature	Date
_____	_____	_____
SGS Approval	Signature	Date

Final approval and acceptance of the thesis is contingent upon the submission of the final copy of the thesis to the School of Graduate Studies (SGS) through the Department/School Graduate Committee (DGC/SGC) of the candidate's department.

Stamp of SGS Date: _____

Acknowledgment

Gratitude fills my heart, Dear Lord, for the bountiful blessings you have bestowed upon me. Your generosity has exceeded all my expectations, granting me a loving family, wise mentors, guiding teachers, and cherished friends who bring joy to my life with their caring words and actions.

Next, I would like to extend my sincerest thanks to my advisor, Dr. Michael Melese, for his unwavering support and guidance throughout my research journey. His insightful advice and motivational encouragement played a critical role in shaping the direction of my research. His expertise and support were truly invaluable and I am deepest grateful for all that he has done. Also, I would like to thank my co-advisor, Girmay Tekle, for his comments and encouragement to finish my research.

I am also grateful for the support provided by Sidama Region state, who provided financial support for this research project. Their contribution was critical in allowing me to complete my work.

Finally, I want to express my sincere gratitude to my family for their sold support and encouragement. Their faith in me has been a driving force and an inspiration. I especially want to extend my gratitude to my father, Alemayehu Chone Hakamo, for his wise parenting advice and financial assistance. Additionally, I want to thank my closest friends, Mekonen Moke and Asmelash Legese, for their invaluable feedback and support throughout my research journey

In conclusion, I am grateful to all those who have supported this research project and made it a success. Their contributions have been invaluable, and I am honored to acknowledge their support.

Contents

List of Tables

List of Figures

List of Algorithms

List of Acronyms

ANNs	Artificial neural networks
BLEU	Bilingual Evaluation Understudy
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	deep neural network
EBMT	Example-Based Machine Translation
GRU	Gated Recurrent Unit
HMT	Hybrid Machine Translation
IR	Information Retrieval
LSTM	Long short-term memory
MT	machine translation
NLP	Natural Language Processing
NMT	Neural Machine Translation
NNs	neural networks
POS	Part of Speech
RBMT	Rule-Based Machine Translation
RecNN	Recursive Neural Network
ReNN	Recursive Neural Network
RNN	Recurrent Neural Network
SMT	Statistical Machine Translation
SNRS	Sidama National Regional State
SOV	subject-object-verb
SVO	subject-verb-object

Abstract

Machine translation is one of the Natural Language Processing applications, which enables the translation of text from one natural language to another. This study aimed to design and develop a bidirectional English-Sidaamu Afoo neural machine translation system, as the need system has become increasingly important due to the growing number of language users, it needs to increase its presence on the web, For effective communication and information sharing, translation of various official documents, news articles, and other written texts in both languages is necessary and last to need integrating the other high-level NLP tools, but no prior solution in this area. Recently, Neural Machine Translation has emerged as a promising approach to machine translation, delivering state-of-the-art translation quality. Unlike traditional machine translation methods, NMT uses a single neural network that can be continuously fine-tuned to improve translation performance. This study aimed to develop a bidirectional Sidaamu Afoo-English machine translation system using deep learning techniques, specifically LSTM and Transformer models. In an attempt to do this study, due to un availability of parallel data for machine translation, we opted to collect parallel data from a religious domain, specifically from Bible and Sidaamu Afoo conversation. After gathering the data, experiments were conducted using 15,000 parallel sentences from different domains. To determine the optimal model, the efficiency in terms of training time, memory usage, and BLEU score was evaluated. The results showed that the Transformer model yielded the best results, with a BLEU score of 0.413 for Sidaamu Afoo to English translation and 0.465 for English to Sidaamu Afoo translation. Future work to enhance the performance of the system could include further research and the addition of more clean data and larger corpus sizes.

Keywords: *Machine Translation, Neural Machine translation, Sidaamu Afoo Machine Translation, bidirectional machine translation, Transformer model*

CHAPTER ONE

INTRODUCTION

1.1. Background of the study

A language is a system of communication that consists of a set of sounds and written symbols that people in a specific country or region use for talking or writing. Human communication primarily relies on a structured and conventional use of words that can be conveyed through speech, writing, or gesture [1]. Natural Language Processing (NLP) refers to a computer program's capability to comprehend human language in its spoken and written form, also known as natural language. NLP has been in existence for over half a century and is rooted in the field of linguistics [2]. NLP encompasses all the necessary components that enable a computer to comprehend and generate natural language. It is a subfield of Artificial Intelligence and linguistics that is dedicated to facilitating computers' understanding of human language [3].

Natural language processing has become a major research area in artificial intelligence and machine learning in recent years [1]. This has led to the development of various NLP applications, including machine translation, morphological analysis, syntax analysis, named entity recognition, text summarization, multilingual and cross-language information retrieval, speech recognition, and more [2]. Among these applications, Machine translation is a method to convert the source text from one natural language to another natural language with the help of computerized systems such as, without human intervention [3].

Machine translation, especially from Sidaamu Afoo to English languages, is crucial for various reasons. It facilitates global communication and collaboration by helping Sidaamu Afoo speakers communicate more effectively in English-speaking countries and vice versa. It also enables non-English speakers to access information in English, such as scientific research, news articles, and business reports, which would otherwise be inaccessible. Additionally, it fosters cultural exchange by allowing people to interact with different cultures and traditions without language barriers, contributing to a more diverse and inclusive society. Finally, machine translation helps businesses expand their operations globally, creating increased economic opportunities and growth.

Sidaamu Afoo is an Afro-Asiatic language, belonging to the Highland East Cushitic branch of the Cushitic family [4]. The Sidaama language, also known as Sidaamu Afoo, is spoken by the Sidaama people who reside in certain regions of southern Ethiopia. It is particularly prevalent in the densely populated Sidaama National Regional State. The language is not only a major means of communication for the local community, but it also holds official status as a working language in the Sidaama National Regional State (SNRS). It is worth noting that Sidaamu Afoo is one of the largest languages under the Cushitic language category, with a speaker base of over four million individuals. This places it as the third-largest language in the category, following Afan Oromo and Somali. The language has a rich cultural heritage and holds great importance for the Sidaama people and their identity [5].

Deep learning is a subfield of machine learning where concerned algorithms are inspired by the structure and function of the brain called artificial neural networks [6]. All the value today of deep learning is through supervised learning or learning from labeled data. Each algorithm in deep learning goes through the same process. A hierarchical series of nonlinear transformations can be applied to the input to produce a statistical model as the output.

DL in machine translation is a family of profound systems that can learn high-level features through several neural networks rather than one. The subfield of learning by machines is a modern approach to learning representations from information that emphasizes learning from successive layers of more deep observations [7]. The layer systems with a massive number of layers and hyperparameters in one(four) basic network models are; Unsupervised-pertained networks, CNN(Convolutional Neural Network), RNN(Recurrent Neural Network, and RecNN(Recursive Neural Network).

The past two years have seen a deep learning revolution bring rapid and dramatic change to the field of machine translation [8]. For users, new neural network-based models consistently deliver better quality translations than the previous generation of phrase-based systems. Researchers are excited about the new opportunities that Neural Machine Translation (NMT) offers. NMT allows for simplified training pipelines and the ability to train unified models directly from data, which represents a significant advancement beyond the limitations of Statistical Machine Translation (SMT). This promise has energized the research community, resulting in recent work that focuses almost exclusively on NMT and seemingly advances the state of the art every few months [11].

In general, NMT uses deep learning techniques to teach itself to translate text based on existing statistical models. It makes for faster translations than the statistical method and can create higher-quality output. NMT can use algorithms to learn linguistic rules on its own from statistical models [9]. The rule-based approach, however, demands a substantial human effort in terms of preparing the rules and linguistic resources like POS development, Syntactic Parsers, Morphological Generator, Bilingual Dictionaries, Transfer Rules, and Reordering Rules [10] [11]. So this all well not developed previously in Sidaamu Afoo and rule-based is not recommendable.

NMT offers several advantages when it comes to translating from English to under-resourced languages and reverse. These include improved accuracy, adaptability to new language pairs, faster translation, contextual understanding, and cost-effectiveness. NMT models are trained on large amounts of data, which allows them to learn complex patterns and dependencies between languages and produce more accurate translations. They are also able to understand the context of the text being translated, which makes them ideal for translating from English to Sidaamu Afoo and vice-versa. Therefore, this study shall better use Deep learning techniques.

1.2. Motivation

The internet has a disproportionate amount of content in English, despite only 20% of the world's population speaking English [7]. Machine translation (MT) tools can be used to translate text and audio from one language to another, but there has been little research on using MT for the Sidaamu Afoo language. Real-time translation applications are important for native speakers, and different types of MT tools are available, including text-to-text, text-to-speech, speech-to-text, and speech-to-speech [12]. Frist reason that motivated the researcher, Currently Sidaamu Afoo is the working language of the Sidama region, so any governmental document needs to be translated into Sidaamu Afoo. Secondly, Sidama Afoo is developing and researching language up to the university level, currently in Hawassa University in the Linguistic department and Hawassa Teacher Training College (TTC), so their students and researcher need to translate English articles, magazines, and books are into Sidaamu Afoo. thirdly Sidaamu Afoo speakers use different social media, so they need to share their cultural heritage and thought across the globe using the English language but some don’t know the English language, therefore it needs to be translated Sidama Afoo written idea into English language, and last, Text-to-text is the most common and important for everyday

digital communication. However, the researcher uses different social media and any posts available in the English language, this content can be translated into the other language but we cannot translate it to Sidaamu Afoo, the problem is an absence of a machine translation tool. Contrarily if translated Sidaamu Afoo source text is taken as another Cushitic family language like Afan Oromo, Somali Afii, and others, this was happening on Facebook media machine translation tool and those directed pressure of technology on language. In general, this research contributes to and facilitated the technology-unsupported language of Sidaamu Afoo to grow the technology-portable language. Therefore, in this study, the researcher has been motivated to develop a machine translation tool for English - Sidaamu Afoo.

1.3. Statement of the Problem

Sidaamu Afoo is a language with a large number of speakers with a lot of historical, cultural, and religious documents available in the Sidaamu Afoo languages [13] [14]. Although resources exist in both Sidaamu Afoo and English, they should be made accessible to speakers of other languages as well. Unfortunately, the absence of document translations has impeded effective communication across linguistic barriers. This underscores the importance of investing in language translation efforts to foster inclusivity and enable greater access to information for diverse communities. Machine translation systems for different language pairs have been done in past years. A majority of the research has focused on language pairs involving English and other languages [15]. For instance, Filipino-English, Myanmar-English, English-Amharic, and English-Afan Oromo [16]. However, no machine translation study was conducted on English-Sidaamu Afoo language pairs.

In today's Information age, a lot of written documents, publishers, textbooks, magazines, advertisements, and other material on the web are being produced in technologically supported and resourced languages such as English, European and Asian languages. In this studies and applications involving foreign languages have been conducted using various methodologies and approaches [17]. Conversely, research in the area of MT for Ethiopian languages, which are under-resourced as well as economically and technologically disadvantaged, has started very late [17]. In order to enable effective communication between technologically advanced and resource-rich languages such as English, and to make use of the information and documents produced in global languages, it is necessary to translate these documents into local languages like Sidaamu Afoo.

Machine translation is one of the NLP applications that facilitate communication between languages English, which is a resource fullness language, and the technologically unsupported language is Sidaamu Afoo.

As technology continues to advance, the world becomes more and more connected. Physical distance is no longer as big of a roadblock as it once was; allowing people to connect over several different social platforms. Yet, while distance may not divide us like it once did, the vast number of spoken languages continues to make universal communication difficult [18]. Language barriers are a common occurrence that hinders communication and collaboration all across the globe. And while translators have been effectively overcoming these barriers for centuries. Currently, the most effective manner by which technology can accomplish this task is through neural machine translation.

Moreover, other many contents are posted on different social media using the Sidaamu Afoo language, but that is not translated into another language because Sidaamu Afoo did not have a machine translation tool. However, some social media like Facebook try to make the wrong translation because take Afan Oromo or Somali Afii instead of Sidaamu Afoo posts/text. In other cases, content that is frequently changed or updated would demand high costs of human translation, and MT in Sidaamu Afoo might be the perfect solution.

In this era of multimedia and online communication, we want to be able to communicate with people all over the world without a little language difference. Machine translation steps online to help make languages understandable across the world [19]. The different researchers are focused on improving their machine translation systems to allow users to easily share their lives and have conversations across the world. In Ethiopia, most regional languages attempt to develop neural machine translations such as Afan Oromo [20], Amharic language [21], Tigrinya Language [22], and Somalia Afii, but none of Sidaamu Afoo.

To this end, this research work attempts to address the following research question;

1. Which deep learning model is effective for bidirectional English – Sidaamu Afoo translation?
2. Which higher parameter value sets are gives better results for English-Sidaamu Afoo language translation?

3. To what extent the proposed bidirectional English – Sidaamu Afoo translation work?

1.4. Objectives of Study

1.4.1. General objective

The main objective of this research is to develop and design a neural machine translation system that can translate between English and Sidaamu Afoo in both directions.

1.4.2. Specific objective

- ✓ To review literature to understand the state-of-the-art in machine translation, Sidaamu Afoo, English language and deep learning,
- ✓ To collect and prepare the parallel corpora on English – Sidaamu Afoo from a different source,
- ✓ To design a general architecture for English-Sidaamu Afoo machine translation,
- ✓ To develop an implementation for bidirectional English - Sidaamu Afoo machine translation,
- ✓ To test and evaluate the performance of a system using different methods, and
- ✓ To report the finding of the study for the upcoming research area.

1.5. Scope and Limitation of the study

The focus of this study is on the application of deep learning NMT methods for translating Sidaamu Afoo and English text. The corpus used for this study consists of 14,300 pairs of sentences, with 95.33% of the data coming from the religious domain and the remaining 4.67% (700 pairs of sentences) from Sidaamu Afoo conversations translated into English. English - Sidaamu Afoo machine translation system is designed which enables to translation of English texts into Sidaamu Afoo text and reverse up to sentence level based on the input provided which means word level, phrase level, and sentence level. Speech translation is not included in this study.

The limitation to conducting this research work is the absence of publicly available previously done research in this language on machine translation. The other is a preparation of data constraint,

which means a scarcity of standardized English – Sidaamu Afoo parallel corpus. This study is restricted to working with lengthy and complicated sentence structures, where functional translations using up to 28 words in Sidaamu Afoo and 21 words in English are effectively.

1.6. Methodology of study

The methodology of a research study refers to the systematic and theoretical analysis of research methods employed in a particular field of study. It involves defining a research design, conducting a literature review, selecting an appropriate research process and data collection methods, collecting and analyzing data, tools, and techniques, and evaluating the results clearly and concisely. The methodology is essential to ensure that the research is conducted rigorously and systematically and that the results obtained are valid and reliable.

1.6.1. Research design

An experimental research design was utilized in this study. This type of research design strictly adheres to the scientific method, which involves the formulation of a hypothesis, the manipulation of variables by the researcher, and the evaluation, calculation, and comparison of variables [26]. Different stages of this research are named as a discussion to achieve the research objective there are data collection and preparation, literature review, analyzing written documents, selecting the approach, preparing parallel corpus, training dataset, and evaluating and testing the proposed system. To achieve the objectives, the following steps are employed.

1.6.2. Literature Review

Related kinds of literature on the machine translation concept and the major processes of NLP application, reviewing previously implemented machine translation approaches and related works previously done on other languages of pairs. To understand how machine translation works for English to foreign language translation. In addition to this, both English and Sidaamu Afoo grammar books, articles, reports, and other related literature are reviewed.

1.6.3. Data collection

The collection of text data is an essential component in developing a machine translation model. Thus, the corpus used in this research was collected from online and hard copy religious documents, cultural documents from the Sidaama regional state tourism office, Sidama national state region constitution, and English Sidaamu Afoo language education materials. Eventually, all of this needed for parallel dataset preparation is because there is no well-organized and prepared standard dataset for the proposed model and English- Sidaamu Afoo language pairs. The parallel dataset preparation is needed to experiment with the model on English-Sidaamu Afoo language pairs in the model training and testing phases.

1.6.4. Tools and techniques

To develop a machine translation model using deep learning techniques and used different software packages. However, choosing a toolkit, a researcher needed to gain a general idea of the various open-source machine translation toolkits that are available at the time of writing. They are; TensorFlow is a large-scale, general-purpose, open-source machine-learning toolkit, for the implementation of these deep artificial [7]. TensorFlow is a freely available library designed for numerical computations and large-scale machine learning. Developed by Google Brain, TensorFlow simplifies the process of data acquisition, model training, prediction serving, and results improvement [27]. The library includes a comprehensive set of machine learning and deep learning models and algorithms. In addition, it has several libraries and open-source packages which are available to deal with a particular problem that will simplify the programming tasks. Such as PyTorch, and Keras is used to train and develop the machine translation model. The experiment conducted used an open-source neural machine translation toolkit called Kera's, PyTorch which is freely available in Kera's Repository with the implementation of LSTM type of RNN architecture with global attention using an encoder-decoder language model [17].

1.6.5. Experimental setting

After we finished collecting and preparing the dataset in a comfortable way for a machine, we saved separately all types that txt, CSV, and XSL unit datasets. we have shown samples in

Appendix I. To conduct our experiments, we used a Google collab space with 12 GB of RAM and a GPU-based computer, which was used to process with a short training time. To build our system, we used the Python programming language along with the Keras, TensorFlow, NumPy, and PyTorch libraries. We conducted the experiments using deep learning methods special LSTM and Transformer model.

1.6.6. Evaluation

Machine translation evaluation could be done by using manual or automatic evaluation methods. Manual evaluation gives a better result to measure the quality of machine translation and analyze the errors within the system output [16]. The most challenging issues in conducting a human evaluation of machine translation output are high costs and time consumption. Therefore, automatic methods like Bilingual Evaluation Understudy (BLEU) are proposed to measure the performance of machine translation. The researcher used BLEU score metrics to evaluate the performance of the model.

1.7. The Significance of the study

The main idea of language translation is to enable the exchange of information between two different language speakers at the level of speech or text. Then, the only choice to translate languages is human or machine-based translation. However, using a human expert is costly and difficult everywhere. Consequently, it is necessary to study machine translation to automate the mechanism of language translation. Therefore, the development of machine translation gives a great contribution to scarce resource language if the translation is done from resource-rich language to scarce resource language. Sidaamu Afoo is one of the largest spoken southern parts of Ethiopia special in the Sidaama region, Furthermore, most Sidaamu Afoo speakers do not well-understood the English language, so it is necessary to provide society with updated information about the world by translating from English sources.

Also, as Sidaamu Afoo is the official working language of the Sidaama regional state, applying machine translation to the translation of different educational books like elementary books or other materials can contribute to different government institutions like the human resource manual.

Moreover, using social media are transmitting their post to target languages; is necessary machine translation to help students on translating studying material to the target language to increase understanding. Sidaamu Afoo Machine translation is also useful when you have large amounts of user-generated content that needs to be translated quickly. For instance, machine translation makes it possible to quickly translate and review customer reviews, online comments, and social media posts.

The well-known approaches for cross-language information retrieval (CLIR) query translation and document translation to determine which is more effective and necessary one machine translation tool. A common strategy for addressing the issue of queries being in a different language than the documents is to utilize a machine translation (MT) system. This approach enables the reduction of information retrieval (IR) to a pseudo-monolingual level, which refers to the practice of conducting IR in a shared language. For Sidaamu Afoo language queries and English language documents, this means either translating the queries into English to match the language of the documents or translating the documents into Sidaamu Afoo to match the language of the queries.

In general machine translation target benefit students which Lower cost of translation, a businessman which the ability to bring a product to market faster, social media user which Terminology consistency, the government sector that delivers technology application, the private sector which gives different product platform delivery and researcher working in information retrieval.

1.8. Thesis Structure

The thesis is organized into Six different but interrelated chapters. Chapter one gives the overall introductory information of the argument. In this chapter, we try to clear up what are the research problems, the research question used to answer the relevance of the problem it is addressing the research problem, the purpose of a general and specific objective to solve the research problems stated and the general research methodology carried out to accomplish research objectives are. Chapter two is a literature review. This chapter describes the literature part and the source of the research area, morphology structure of the languages to learn from others' work, and to evidence, the hypothesis of the research, and different kinds of literature on the methodology and deep learning approaches. Chapter three is about the Related work. This chapter discusses in more

detail how related papers are carried out and more descriptions and the logical reason and relation with our work. A generally compare and contrast different work which contributes to our work or not. Chapter four is about designing the Sidama – English machine translation architecture. Step by step description of how the proposed solution addresses the problems stated in the research problem using the neural machine translation model is developed and how it works is detailed in this chapter. The fifth chapter presents the experimental results and evaluation of the Bidirectional machine translation. Finally, a conclusion and future work are presented.

CHAPTER TWO

LITERATURE REVIEW

2.1. Introduction

This chapter provides a comprehensive overview of machine translation, as well as discusses various machine translation approaches. Primarily, the first section discusses machine translation which includes, Statistical Machine Translation (SMT), Rule-Based Machine Translation (RBMT), Example-Based Machine Translation (EBMT), Hybrid Machine Translation (HMT), and Neural Machine Translation (NMT). Each approach is briefly described to give a comprehensive understanding of the available methods. And the second section discusses related work which is Ethiopian language with foreign language pairs, foreign language pairs, and Ethiopia language pairs, and finally, a summary of related work.

2.2. Overview of Machine Translation

As most of the world is online, the task of making data accessible and available to all is a challenge. A major challenge in making data accessible is the language barrier. There are a multitude of languages with different sentence structures and grammar. Machine Translation is generally translating phrases from one language to another with the help of a statistical engine like Google Translate [23]. The concept of machine translation is an automated translation, which means translation computed by a computer. Machine translation is a subfield of NLP, which uses a monolingual, bilingual, or parallel dataset and additional linguistic properties named corpus to construct language models used to translate text [7].

After more than sixty years of research in machine translation, the resulting technology has finally begun to be adopted for purposes beyond merely gifting. Many factors contributed to this change in the way Machine translation is perceived and used, including noticeable improvements in translation quality and increasing demand [24]. The most basic function of Machine Translation is a word-by-word translation where a word in the source language is replaced with the

corresponding word in another language. To investigate this function using several types of Machine Translation approaches.

2.2.1. Rule-Based Machine Translation

Rule-Based Machine Translation (RBMT) is a type of machine translation system that utilizes linguistic information about the source and target languages to translate text. This approach is also referred to as Knowledge-Based Machine Translation and the Classical Approach of MT. RBMT is based on the utilization of bilingual dictionaries and grammars which store information about the main semantic, morphological, and syntactic regularities of each language. This information is then used to translate text from the source language to the target language. In essence, RBMT works by following a set of predefined rules and utilizing linguistic information to determine the most appropriate translation. It is considered a more traditional approach to machine translation compared to more advanced systems such as neural machine translation. Despite its limitations, RBMT is still widely used in certain applications where the quality of the translation is of lesser importance and a faster translation process is required [25]. Having input sentences (in some source language), an RBMT system generates them to output sentences (in some target language) based on morphological, syntactic, and semantic analysis of both the source and the target languages involved in a concrete translation task.

2.2.1.1 Basic Principles of the RBMT Approach

RBMT methodology utilizes a comprehensive set of linguistic rules during three distinct phases of the translation process: analysis, transfer, and generation. This approach relies heavily on the application of syntax and semantic analysis, as well as syntax and semantic generation, to produce an accurate translation of the source text [26] [25]. The process of RBMT is divided into several steps, starting with the analysis phase, in which the source text is analyzed for its grammatical structure and semantic meaning. This information is then transferred to the next phase, the transfer phase, where the linguistic rules are applied to the source text to generate a target text that is structurally and semantically equivalent to the source text. Finally, in the generation phase, the target text is generated based on the results of the transfer phase. The target text is a complete and accurate representation of the source text, having undergone a thorough analysis and application

of linguistic rules during the entire RBMT process. In general, the RBMT methodology follows the steps outlined in Figure 2.1 provided, resulting in a high-quality, rule-based machine translation.

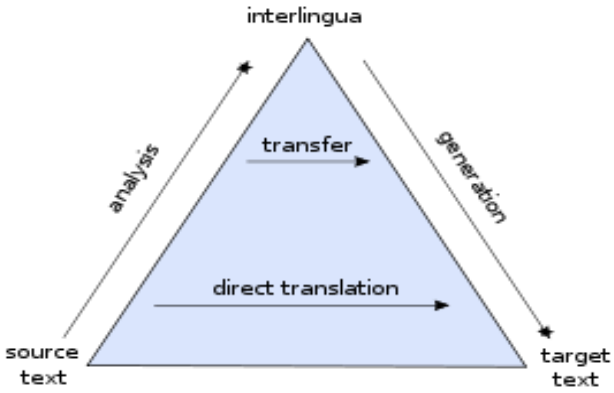


Figure 2. 1 steps of the RBMT process

2.2.1.2. Challenges of the RBMT Approach

The following are the shortcomings that are associated with the RBMT approach

- ✓ The limitations of current dictionaries include a shortage of high-quality options,
- ✓ high costs associated with creating new dictionaries,
- ✓ manual input required for linguistic information,
- ✓ difficulties handling complexities in large systems such as ambiguity and idiomatic expressions,
- ✓ inability to adapt to new domains, and costly and ineffective efforts to update existing systems through rule creation and lexicon extension.

2.2.2. Corpus-based Machine Translation Approach

Corpus-based machine translation is a type of machine translation that relies on statistical models built from large collections of bilingual text, called "parallel corpora". The approach uses patterns

in the parallel corpora to translate text, rather than relying on hand-written rules. This method is often used to build machine translation systems for low-resource languages or for domains where parallel corpora are not readily available. Examples of corpus-based machine translation systems include Google Translate and Microsoft Translator. Machine Translation based on Corpus, commonly referred to as Data-Driven Machine Translation, is a different approach to addressing the knowledge acquisition issue in Rule-Based Machine Translation. CBMT relies on bilingual parallel corpora to gather information for new translations. This method utilizes vast amounts of raw data, in the form of parallel text and translations, to obtain its knowledge. The corpus-based approach can be further divided into two sub-categories: Statistical Machine Translation and Example-based Machine Translation.

2.2.3. Statistical Machine Translation

SMT is a method of automatically translating text from one language to another using statistical models trained on large parallel text corpora. The goal of SMT is to translate text in a way that produces a high-quality result that is both fluent and preserves the meaning of the original text.

An example of SMT would be using an algorithm trained on bilingual sentence pairs to translate a sentence from French to English. The algorithm would use statistical models to predict the most likely English translation given the input French sentence [27].

2.2.3.1. Challenges of the SMT Approach

The following are the shortcomings that are associated with the SMT approach

- Corpus creation can be costly for users with limited resources
- The results are unexpected. Superficial fluency can be deceiving.
- SMT does not work well between languages that have significantly different word orders (e.g., Sidaamu Afoo and English languages).

2.2.4. Example-based Machine Translation

Machine translation based on examples (EBMT) is defined by its reliance on bilingual corpora consisting of parallel texts, where the key concept is translation through analogy. An EBMT

system takes in a set of sentences in the source language and their matching translations in the target language, with a one-to-one mapping [25]. Example-based machine translation (EBMT) utilizes a principle of translation by analogy and consists of four stages: acquiring examples, managing the example base, applying examples, and synthesizing translations. The system is trained through the use of example translations, where sentences of similar types from the source language are translated into the target language

2.2.4.1. Challenges of the EBMT approach

EBMT is an attractive approach to translation because it avoids the need for manually derived rules. However, it requires analysis and generation modules to produce the dependency trees needed for the examples database and for analyzing the sentence. Another problem with EBMT is computational efficiency, especially for large databases, although parallel computation techniques can be applied [25].

2.2.5. Hybrid Machine Translation

HMT is a type of MT that combines the strengths of both statistical and rule-based machine translation approaches. In this approach, the translation system first uses statistical models to generate a candidate translation and then uses a rule-based system to refine and improve the candidate translation. The combination of these two approaches results in improved translation quality compared to using only one of the approaches. Explain merely, the statistical models used in hybrid machine translation are trained on large parallel corpora of text in the source and target languages, and they learn to generate translations based on the patterns and relationships they observe in the training data. However, the statistical models are not able to incorporate linguistic knowledge and rules in the same way that rule-based systems can. Rule-based systems, on the other hand, rely on hand-crafted grammatical and linguistic rules to generate translations, but they can be less flexible and less able to handle novel language patterns and input. By combining the strengths of both statistical and rule-based systems, hybrid machine translation systems can generate translations that are both accurate and fluent, making them a popular choice for many applications that require high-quality translations [28].

2.2.6. Neural Machine Translation

Recently, neural approaches to MT have enjoyed great popularity and success for a broad range of language pairs. Neural machine translation aims to model the full translation process using a single neural network model. A popular approach to modeling the task is the encoder-decoder framework, in which the source sentence is encoded into a vector, commonly called a context vector, and a decoder generates a translation from this vector [29].

An artificial neural network is used in the NMT method to forecast the likelihood of a set of words, generally modeling full sentences in a single integrated model [18]. ANNs also referred to as neural networks (NNs), are computational systems that draw inspiration from the biological neural networks of the human brain. ANNs consist of a network of interconnected units or nodes called artificial neurons, which are modeled loosely after the neurons in the human brain. These neurons are usually organized into layers, and each layer may perform distinct transformations on the input data. The signals flow from the input layer through the intermediate layers to the output layer, possibly passing through the layers multiple times [36]. There are several types of artificial neural networks used in computational models, but the basic type of ANN is described below.

2.2.6.1. Feed Forward Neural Network (FFNN)

The information in the neural network travels in one direction and is the purest form of an Artificial Neural Network. This kind of neural network can have hidden layers and data enter through input nodes and exit through output nodes. Classifying activation function is used in this neural network. There is no backpropagation, and only the front-propagated wave is allowed [30]. The feed-forward network does not have any feedback loops. The backpropagation algorithm can be employed to update the weight values and minimize prediction errors [38]. These neural networks are easier to maintain and are highly responsive to noisy data. Feedforward neural networks are used in various applications, including but not limited to, data compression, pattern recognition, speech recognition, and computer vision.

2.2.6.2. Recurrent Neural Network (RNN)

Recurrent neural networks are yet another variation of feed-forward networks. Here each of the neurons present in the hidden layers receives an input with a specific delay in time. The RNN

mainly accesses the preceding info of existing iterations. For example, to guess the succeeding word in any sentence, one must know the words that were previously used [31]. The principle of a Recurrent Neural Network is to feedback the output of a layer back to the input again. This principle helps to predict the outcome of the layer. In the Computation process, each neuron will act as a memory cell. The data to be used later will be remembered and work for the next step will go on in the process. The prediction will improve by error correction. In error correction, some changes are made to create the right prediction output. The learning rate is the rate of how fast the network can make the correct prediction from the wrong prediction [30]. There are many applications of RNN, and one of them is machine translation. However, it has two problems of standard RNNs troubling the training of the model [32].

1. The problem of Vanishing Gradient

RNNs offers the ability to model time-dependent and sequential data problems, such as machine translation, text generation, and stock market prediction [33]. However, training RNNs can be challenging due to the issue of vanishing gradients. This occurs when gradients become too small, leading to insignificant parameter updates and making it difficult to learn long data sequences. As a result, RNNs suffer from the problem of vanishing gradients, which hinders their effectiveness

2. The problem of Exploding Gradient

An Exploding Gradient is a phenomenon that occurs during neural network training when the slope grows exponentially rather than decaying. This problem arises due to the accumulation of large error gradients, which lead to significant updates in the weights of the neural network model during training. Gradient problems, including the issue of Exploding Gradients, can cause poor accuracy, reduced performance, and longer training times.

- Long Short-Term Memory (LSTM)

LSTM is a type of Recurrent Neural Network (RNN) architecture. It was designed to address the problem of vanishing gradients that is commonly faced by traditional RNNs. LSTM networks use a memory cell that can retain information for a long time, along with different gates that control the flow of information into and out of the memory cell. The gates in an LSTM network, such as the input gate, forget gate, and output gate, help selectively retain or discard information based on

its importance. This allows LSTMs to handle long-term dependencies and model complex sequences of data more effectively than traditional RNNs [34]. LSTMs have been successfully applied in various tasks, such as speech recognition, language modeling, machine translation, and video analysis, among others.

2.2.6.3. Convolutional Neural Network (CNN)

In this type of neural network, Learnable biases and weights are given to the neurons initially. Image processing and signal processing are some of their applications in the computer vision field [30]. CNN is acceptable for computer vision applications for flag and picture handling which returns over OpenCV [7].

2.2.6.4. Recursive Neural Network (ReNN)

A ReNN is a form of deep NN constructed by recurring application of the same weight set via a structure to make a structured summary; or scalar prediction of variable input structures, via topological ordering of a given structure [7]. ReNN is a coordinating machine-learning framework for processing structured area information (i.e., protein topologies, web pages for hypertext markup language (HTML), deoxyribonucleic acid regulatory networks, NLP parse trees, image analysis, etc.)

2.2.6.5. Transformer Model

Transformer is a type of DL model introduced in 2017 in a paper called "Attention is All You Need" [35]. It is used primarily in NLP tasks such as machine translation, sentiment analysis, and text classification. The Transformer model utilizes self-attention mechanisms to process input sequences and generate output sequences [36]. Unlike RNNs, which use sequential processing, the Transformer can process all parts of the input in parallel, which makes it more efficient and allows it to handle longer sequences. An example of using a Transformer model would be to translate sentences from one language to another. The model would take in a sentence in the source language and generate an equivalent sentence in the target language.

2.3. Related work

Related works cover mainly machine translation with its approaches, procedures, methodologies, and techniques researchers used. To the best of our knowledge, there has been no research conducted on the Sidaamu Afoo language on machine translation. While this study is the first, different literature in local and foreign languages is reviewed. This review covers the machine translation system for the machine translation done in the pair of Ethiopian and foreign languages, foreign language pairs, and Ethiopian language pairs.

2.3.1. Machine Translation between Ethiopian and foreign languages

The study of Amharic-Arabic Neural Machine Translation was developed using Attention-based Encoder-Decoder architecture which is adapted from the open-source OpenNMT system [37]. In this research, work used two special kinds of Recurrent Neural Networks (RNN) LSTM and GRU, which are capable of learning long-term dependencies. An OpenNMT system which is an open-source toolkit for NMT is used to construct the NMT model and translate the text from Amharic to the Arabic language. By modifying the existing monolingual Arabic and its corresponding Amharic translations available on Tanzile, they have created a small parallel text corpus. The corpus is comprised of Quranic text, which includes 114 chapters, 6,236 verses, and 157,935 words. Preprocessing of both Amharic and Arabic scripts has a great impact on the performance of the NMT system. Sentences are split and aligned manually and then all punctuation marks are removed from texts. After extensive experiments, the maximum source and target sequence length are set to 44, the maximum batch size for training and validation is set to 80 and 40 respectively, and the learning rate to 0.001 with Adam optimization for both LSTM and GRU RNN type. The remaining parameters are used as default. The system saves the model for each of the 10,000 training samples and then computes the accuracy and perplexity of each model 10 times. With a BLEU score of 12%, 11%, and 6%, respectively, LSTM-based OpenNMT surpasses GRU-based OpenNMT and Google Translation systems in the evaluation of experiments using NMT models based on LSTM and GRU.

In addition to Amharic-Arabic, a unidirectional English to Dawurootsuwa MT model by using a neural network approach [38], free available tools such as Python, Keras toolkit, and Tensor Flow

are used to implement the models [38]. The researcher developed the model using NN approaches, Under RNN, an LSTM, and GRU model types. A parallel corpus, which consists of 20,345 pairs of sentences is prepared from different sources and classified as a 90% training set and a 10% test set. A recurrent Neural Network model with 22 input nodes and 27 output nodes is developed and implemented using the Keras toolkit of Python programing language and the Adam algorithm. They contain four results-based automatic (BLEU) scores and manual evaluation (Arithmetic Mean Value) techniques with a hidden layer size of 2. In a simple RNN model, the BLEU score is 0.5187 with a learning rate of 0.002 and the AMV result is 0.60914. In embedding the RNN model the BLEU score is 0.5245 with a learning rate of 0.003 and the AMV result is 0.60914. In the bidirectional RNN model, the BLEU score is 0.5452 with a learning rate of 0.004 and the AMV result is 0.60914. Finally, in the encoder-decoder model, the BLEU score is 0.555 with a learning rate of 0.005 and the AMV result is 0.60914. And after a 0.005 learning rate, there is similar scores were recorded with the maximum threshold epochs of 100. From the result, concluded that the encoder-decoder model with a BLEU score of 0.555 is good accuracy achieved compared to the rest model and less achieved comparatively from the AMV result.

Lastly, the Development of bidirectional machine translation of English to Afan Oromo using Hybrid techniques conducted a research study that involved four experiments in the field of machine translation [2]. The primary focus of the research was to explore the effectiveness of SMT in translating between Oromiffa and English, two languages with distinct sentence structures. The first experiment involved an Oromiffa-English SMT and yielded a BLEU score of 41.50%, indicating a relatively high level of accuracy in translating from Oromiffa to English. The second experiment, an English-Oromiffa SMT, produced a BLEU score of 32.39%. For the third and fourth experiments, the researchers implemented a hybrid approach, incorporating elements of both the Oromiffa-English and English-Oromiffa translations. This approach yielded a BLEU score of 37.41% and 52.02% respectively, demonstrating the potential for a combined approach to result in improved translation accuracy. The researchers used a corpus of 3000 parallel sentences, with 2700 sentences utilized for training and 300 sentences reserved for testing the system. To overcome the challenge of the different sentence structures in the two languages, the researchers employed syntactic reordering to align the structures of the source and target language.

2.3.2. Machine translation for non-Ethiopian Languages

The development of an RNN model for English to Yoruba MT was done by eight different researchers in 2020 [39]. The research developed a machine translation tool for English to Yoruba a using recurrent neural network model. The researchers obtained Parallel corpus from the English and Yoruba bible corpus. The developed model was tested and evaluated by using both manual and automatic evaluation techniques. Automatic evaluation of the developed system gives a percentage BLEU score of 54.8 when tested on a dataset of 588 sentences and 57.3 when tested on a data set of 1000 sentences. The average BLEU score obtained from the two datasets is 56%. The research reveals that BLEU scores up to 50 and above generally reflect good and fluent translations. Results from manual evaluation by ten human evaluators show that the system is adequate and fluent. Also, results from the automatic evaluation show that the developed model has decent and good translation as well as higher accuracy because it has a better correlation with human judgment.

2.3.3. Machine translation of Ethiopian languages

In 2017 a study was conducted to investigate Amharic-Tigrigna Machine Translation using a Hybrid Approach [40]. The study consisted of two experiments. The first experiment was a straightforward Amharic to Tigrigna Statistical Machine Translation (SMT), which resulted in a BLEU score of 7.02%. The second experiment, using the hybrid approach, had a BLEU score of 17.47%. The researchers utilized a total of 1,582 parallel sentences in both Amharic and Tigrigna for their experiments, with 90% of the sentences being used for training and 10% for testing the system. Despite having similar sentence structures, the two languages differ at the phrase level, hence the researchers implemented reordering rules at this level. This study only focused on reordering rules to address the structural differences, and the researchers suggest that further exploration of rules, particularly in terms of morphological rules, should be conducted.

Table 2. 1 summary of related work

Authors	Title	Methods	Data size	Bleu score	Remark as Reviewer
Ibrahim Gashaw and H L Shashirekha	Amharic to Arabic using Attention-based Encoder-Decoder architecture	LSTM	10,000	12 %	Small data size and no self-attention
Elias Asefa and Hussien Seid in 2021	unidirectional English to Dawurootsuwa machine translation model by using a neural network approach	encoder-decoder model	20,345	55.5%	Long dependency problem
In 2013, Jabesa Daba and Yaregal Assabie	bidirectional machine translation of English to Afan Oromo using Hybrid techniques	SMT and rule-based approaches(reordering)	3000	Oromiffa-English (37.41%) & English-Oromiffa (52.02%)	No long dependency learns but reordering is effective
eight different researchers in 2020	a recurrent neural network model for English to Yoruba machine translation	RRN	5000	57.3%	Small data size and simple RNN
In 2017, Gebremariam	Amharic-Tigrigna Machine Translation using a Hybrid Approach	SMT and Reordering Rule	1,582	17.47%	Small data size

2.3.4. Summary of related work

This work summarizes the findings of various machine translation research studies and provides a list of the authors, the title of each study, the methods used, data size, Bleu scores, and the identified gaps in each study. The first study, conducted utilized an LSTM-based Attention-based Encoder-Decoder architecture for translating from Amharic to Arabic. The data size was 10,000 and the resulting Bleu score was 12%. The researcher noted as a remark the small data size and no self-attention. The second study employed an encoder-decoder model for English to Dawurootsuwa translation. The data size was 20,345 and the Bleu score was 55.5%. The third study utilized a hybrid technique combining SMT and rule-based approaches for translating from English to Afan Oromo. The data size was 3000, and the Bleu scores were 37.41% for Oromiffa-English and 52.02% for English-Oromiffa. The noted gap was no long dependency learns but reordering is effective. The fourth study, involving eight different researchers, used a recurrent neural network model for English-to-Yoruba machine translation. The data size was 5000, and the resulting Bleu score was 57.3%. The identified for further were small data size and simple RNN. The last review study is a Hybrid Approach combining SMT and reordering rules for translating from Amharic to Tigrigna. The data size was 1,582, and the Bleu score was 17.47%. The gap further mentioned was the small data size.

CHAPTER THREE

SIDAAMU AFOO AND ENGLISH LANGUAGES

3.1. Introduction

In this chapter, we will delve into the fundamentals of Sidaamu Afoo and English language. We will start with an introduction to the writing system and then move on to the structure of a sentence. Additionally, we will cover the usage of articles and punctuation marks. We will also explore the different parts of speech, including nouns, verbs, adverbs, adjectives, and prepositions and last syntax and semantics. By the end of this chapter, you will have a comprehensive understanding of the key components of Sidaamu Afoo and English language, which will enable you to communicate effectively using this language.

3.2. Overview of the English Language

English is a widely spoken language that originated in England and is now spoken as a first language by over 400 million people around the world [41]. It is also used as a second language by millions more. English is an Indo-European language belonging to the Germanic branch of languages, and it is closely related to other Germanic languages such as German and Dutch [42]. English has a rich history that spans over 1,400 years, and it has evolved significantly over time. Old English, which was spoken from the 5th century to the 11th century, was heavily influenced by Germanic languages such as Old Norse and Anglo-Saxon [42]. Middle English, which was spoken from the 11th century to the 15th century, saw the introduction of French and Latin influences due to the Norman Conquest of England in 1066 [43].

Modern English, which began in the 15th century, has been influenced by many other languages, including Greek, Italian, Spanish, and Arabic [44]. Today, English is the primary language of international business, science, technology, and entertainment. It is also the official language of many countries, including the United States, Canada, Australia, and the United Kingdom [45].

English is a highly versatile language, with a large vocabulary and complex grammar rules [45]. It is known for its flexibility and adaptability, and it continues to evolve and change over time.

3.2.1. English Language Writing System

English orthography is the alphabetic spelling system used by the English language [46]. English orthography uses a set of rules that governs how speech is represented in writing. English has relatively complicated spelling rules because of the complex history of the English language. Most sounds in English can be spelled in more than one way and many spellings can be pronounced in more than one way [46]. The English language contains 24 to 27 (depending on dialect) separate consonant phonemes and between fourteen to twenty vowels and diphthongs [47]. However, English only uses the twenty-six letters of the Latin alphabet.

3.2.2. Syntax of English Language

Syntax refers to the set of rules that govern the structure of sentences in a language. In English, syntax is concerned with how words, phrases, and clauses are arranged to form meaningful sentences. English syntax is based on a subject-verb-object (SVO) word order, meaning that most sentences in English follow the pattern of having a subject, a verb, and an object [48]. For example, "The cat (subject) sat (verb) on the mat (object)". However, English syntax is also flexible, and sentences can be structured in different ways to convey different meanings or emphasize different parts of the sentence. For example, in the sentence "On the mat sat the cat," the object is moved to the beginning for emphasis. Syntax also involves the use of grammatical structures such as tense, mood, and voice, which are used to indicate the time, attitude, and perspective of the sentence [49].

3.2.3. Semantics of English Language

The semantics of the English language refers to the study of the meaning of words, phrases, sentences, and texts in the English language. It deals with how words are used to convey meaning and how the meaning of words can change in different contexts [50]. Semantics is a branch of linguistics that explores the relationship between language and meaning. It involves the study of the structure of language, as well as the ways in which words and phrases are used to convey meaning. In English, words can have multiple meanings depending on the context in which they

are used. For example, the word "bank" can refer to a financial institution or to the side of a river. Understanding the nuances of word meanings and how they are used in different contexts is crucial for effective communication in English. It more described major types of semantic relationships on table 3.1 [51] .

Table 3. 1 semantic relationship of English language

Semantic Relationship	**Explanation**	**Examples**
Categories	Words can be categorised into groups (fruits, emotions, buildings)	Fruit —> Banana, Apple, Watermelon Emotions —> Sadness Joy Anger Anger —> furiousness angst
Antonyms	A pair of words with opposite meanings	fast/slow , big/small, right/wrong sleep/wake , wife/husband
Synonyms	Words that have the same or similar meaning to each other. Some words will have multiple meanings (homonyms) so they may have different sets of synonyms depending on the sense of the word	fair/just/objective/ impartial/unbiased leap/spring/bound/hop/ bounce debate/discuss/confer about/talk over/talk through/thrash
Connotations	Though synonyms have similar literal meanings they can differ in their inferred meaning or their connotation.	Compare connotations of: young vs. childlike, customer vs. consumer, lovely vs. knockout ,grab vs. snatch
Homophones	Words that sound the same but are spelt differently (have a different meaning)	suite vs. sweet, throne vs. thrown, serial vs. cereal
Homographs	Words that are spelt the same but sound different (have a different meaning)	tear (drop) vs. tear (the paper)

Homonyms	Words with multiple meanings, that are spelt and sound the same	bear(animal) vs. bear (withstand) bank (of river) vs. bank (of finance)

3.2.4. Articles of English Language

Articles are words that help to identify a noun used in a sentence is specific or unspecific. In English, there are two types of articles; which are Definite article and Indefinite article [52]. Definite Articles is the word “The” It restricts the noun & narrows it down to one specific thing. It’s used with singular, plural, or uncountable nouns. Indefinite articles are not referring to a specific thing but to a general idea. Indefinite articles are two words “a”, “an”.

3.3. Overview of the Sidama Language

Sidaamu Afoo is a Highland East Cushitic language spoken in the south-central part of the Federal Democratic Republic of Ethiopia [4]. The Sidaama language, a member of the Afro-Asiatic Cushitic family, is primarily spoken by the Sidaama people residing in the highly populated Sidaama National Regional State and which current estimated population of the major Sidama about eight million people [5] [13]. Sidaama is the official working language for the Sidama National Regional State (SNRS) and is also referred to as Sidaamu Afoo, meaning 'mouth of Sidama,' as it represents the people [53]. The language and people are referred to as 'Sidaama'. However, nowadays, the people are known as Sidaama, and the language as Sidaamu Afoo, which literally translates to 'Sidaamas' Mouth'. Therefore, in this term paper, it is preferred to use Sidaamu Afoo to refer to the language and Sidaama to refer to the people and the area [54].

The Sidama language has several dialects, including the Kambaata, Hadiya, and Alaba dialects, among others [55]. It has its unique writing system, known as the Sidama script, which was developed in the 1990s [4]. The language has a complex system of vowels and consonants, with a total of 34 consonants and 5 vowel phonemes. Sidama is a tonal language, which means that the

pitch or tone of a word can change its meaning. It has four tones, including high, mid, low, and falling tones. The grammar of the Sidama language is characterized by subject-object-verb word order. It also has a rich system of noun classes, with a total of 18 noun classes, each with its own set of prefixes and suffixes. Sidama is an important language in Ethiopia, as it is spoken by over 3 million people.

3.3.1. Sidaamu Afoo Writing System

According to a literacy campaign in the 1970s Sidama had adopted traditional Ethiopian script. However, when the present government allows and encourages the use of vernacular languages for administration and primary education purposes in their respective regions (where they are spoken), Sidaama has adopted the Latin script [53]. The Sidaama script which is called Fidalla has 34 symbols [4]. Among these 27 of them are single digit and the remaining 7 are two-digit letters. The 26 single-digit letters are those found in Latin script as found in English and the seven two-digit letters are "ch, dh, ny, ph, sh, ts, and zh". In Fidalla, like in English both capital letters and small letters are used. The small letters are called Shimmaadda Fidalla and the capital Jajjaba Fidalla.

Table 3. 2 Sidaamu Afoo letters

Letters			
Number	**Capital**	**Small**	**Type**
1	A	a	Vowel
2	B	b	Consonant
3	C	c	Consonant
4	D	d	Consonant
5	E	e	Vowel
6	F	m	Consonant
7	G	g	Consonant
8	H	h	Consonant
9	I	i	Vowel
10	J	j	Consonant
11	K	k	Consonant
12	L	l	Consonant
13	M	f	Consonant

14	N	n	Consonant
15	O	o	Vowel
16	P	p	Consonant
17	Q	q	Consonant
18	R	r	Consonant
19	S	s	Consonant
20	T	t	Consonant
21	U	u	Vowel
22	W	w	Consonant
23	V	v	Consonant
24	X	x	Consonant
25	Y	y	Consonant
26	Z	z	Consonant
27	CH	ch	Double consonant
28	DH	dh	Double consonant
29	NY	ny	Double consonant
30	PH	ph	Double consonant
31	SH	sh	Double consonant
32	TS	ts	Double consonant
33	ZH	zh	Double consonant
34	‘	‘	Partial Vowel

3.3.2. Syntax of Sidaamu Afoo

Sidaamu Afoo and English have differences in their syntactic structure. In the language of Sidaamu Afoo, the way sentences are structured is different from English. Instead of following the subject-verb-object (SVO) structure commonly used in English, Sidaamu Afoo follows the subject-object-verb (SOV) structure. This means that in a Sidaamu Afoo sentence, the subject comes first, followed by the object and then the verb.

For example, consider the sentence *"**Abebe waa agi***" in Sidaamu Afoo. In this sentence, **"Abebe"** is the subject, **"waa"** is the object, and **"agi"** is the verb. If we translate this sentence into English, it becomes *"**Abebe drank water,**"* where **"Abebe"** is still the subject, **"drank"** is the verb, and **"water"** is the object. It's important to note that not all languages follow the same sentence

structure, and Sidaamu Afoo 's use of the SOV structure is just one of many variations found in human languages.

3.3.3. Semantics of Sidaamu Afoo

Like any language, Sidaamu Afoo has its own semantics, which refers to the study of the meaning of words, phrases, sentences, and texts in the language. The semantics of Sidaamu Afoo deals with how words are used to convey meaning and how the meaning of words can change in different contexts. It explores the relationship between language and meaning, and how different linguistic structures are used to express different meanings. Sidaamu Afoo, like many other languages, has its own unique set of words, phrases, and idioms that are used to convey meaning. Understanding the nuances of word meanings and how they are used in different contexts is crucial for effective communication in Sidaamu Afoo. It described more about semantic pattern of Sidaamu Afoo in bellow table 3.3 [4].

Table 3. 3 Semantic pattern of Sidaamu Afoo

Semantic pattern	**Explanation**	**Example**
Synonyms	Words that have the same or similar meaning to each other.	keena/bika- ‘measure’ jajja/jiro/- ‘wealth’ woxe/woomasha-‘money’
Location noun	a set of locational nouns that can be used to express various Path Conformations	duumba ‘behindness’ ba’xee “back”, aana ‘top’ gura ‘left’, iima ‘aboveness’ woro “lowerness/belowness” mereero “betweenness/middle/center” alba “beforeness/front”
Path Verb	Path is the site occupied by the Figure rather than the path followed by the Figure	ful- ‘to exit; ascend’ dirr- ‘to descend’,e’i ‘to enter’ sa’e ‘to pass (over, by, across)’

		iill- ‘to arrive’,tais- ‘to cross’ hig- ‘to return’
State change	It has a large class of state-change verbs, especially those that concern feelings and posture changes	hur- ‘to become cured’ qiid- ‘to become cold’ bag- ‘to become tired of’ dimb- ‘to get drunk’ damuu’m- ‘get a headache’ iibb- ‘to become hot’
Habitual Action	A clause with a verb in the imperfect aspect can be ambiguous between the habitual interpretation	duucha wote ‘all the time, every day’ wo’man’ka wote ‘always’ wo’ma-nka yanna/duucha-nka yanna ‘always’ barru barr-u-nni ‘every day’ duunchanka barra ‘every day’ hashsha hashsha [evening evening] ‘every evening’

3.3.4. Articles

The English language is structured from two types of articles known as definite articles and indefinite articles. According to [4], In the Sidaamu Afoo language, the current study reveals that noun referents are treated as neutral concerning definiteness. Unlike English, Sidaama Afoo does not have articles that are written before nouns to indicate definiteness. However, the Sidaama Afoo language does have a way of expressing the definiteness of noun referents through the use of suffixes such as **“-nni, -ho, -tte”.** These suffixes are added to common nouns to indicate the definite referent of the noun in the context of motion or possession. It is worth noting that while these suffixes can express definiteness, the language does not make a morphological distinction between definite and indefinite noun referents in other contexts.

3.3.5. Punctuation Marks

The use of punctuation marks in Sidaamu Afoo and English languages is largely similar, as they serve the same purpose in both languages [56]. The exception to this is the apostrophe mark (‘), which is used differently in the two languages. In English, the apostrophe is employed to indicate possessiveness, plural forms, and contractions. However, in Sidaamu Afoo, the apostrophe is used to denote a glottalized sound that is represented by the letters " /’l/, /’m/, /’n/, /’r/". Additionally, the apostrophe is also used in Sidaamu Afoo to distinguish between words that contain two vowels that are pronounced differently. For example, "saa" means "cow," while "sa’a" means "to pass."

3.4. Major parts of speech in both Language

In this section, we will explore how the different parts of speech in Sidaamu Afoo can be characterized based on their morphosyntactic properties. Previous studies have identified five fundamental word classes in the Sidaama language, namely, noun, adposition, adverb, conjunction, and verb [7]. However, each of these classes can be further subdivided into other sub-classes. For example, the noun class includes proper nouns, common nouns, and pronouns, while prepositions and postpositions are sub-classes of adpositions. These sub-classes can themselves be further divided into additional sub-classes, with the subdivision process potentially continuing iteratively based on the specific objectives and scope of the investigation.

3.4.1. Noun

A noun is a term used to refer to a person, place, object, or concept. Examples of nouns in English include "home", "dog", "city", and "love". Similarly, in Sidaamu Afoo, the words "mine", "woshicho", "quchuma", and "baxa" can be used to represent similar meanings. Nouns can be classified in different ways, depending on the specific dimensions of distinctions that are being considered. The four most common classifications of nouns are: common vs. proper nouns, feminine vs. masculine nouns, countable vs. uncountable nouns, and inherently vs. optionally possessed nouns.

3.4.2. Verb

A verb is a linguistic unit that conveys an action or a state of existence. For instance, "run", "eat", "sleep", and "is" are all verbs. In the Sidaamu Afoo language, verbs are identical to their English counterparts and are used to denote actions, states of existence, and events that take place within temporal boundaries. Verbs indicate the actions performed by the subject in a sentence and can be classified into several categories, including transitive, intransitive, modals, and auxiliary verbs. Transitive verbs transfer a message to a complement or an object, while intransitive verbs do not transfer a message to a complement and therefore do not have an object or complement.

Example Sidaamu Afoo Verbs:

- doda = run
- adha = marry
- abba = bring
- amo daa = come
- amadisiisa = fix

3.4.3. Pronoun

A pronoun is a linguistic unit that replaces a noun in a sentence. Examples of pronouns include "he", "she", "they", and "it". Similar to nouns, pronouns have gender and number. For instance, "ise/isi" refers to "she/he", with "ise" being feminine (singular) and "isi" being masculine (singular), while "insa" refers to "they" (plural) and can be either masculine or feminine. Pronouns can be classified based on their functions and meanings in a sentence, such as personal pronouns, possessive pronouns, demonstrative pronouns, relative pronouns, or reciprocal pronouns. For example, "Baqali minira hari" means "Bekele is going home," and "Isi minira hari" means "He goes home."

some examples of Sidaamu Afoo Pronouns:

- ane/ani = I

- ati = you (singular)
- ki’ne = you (plural)

3.4.4. Adverb

An adverb is a linguistic unit that provides further detail or modifies a verb, an adjective, or another adverb in a sentence. Adverbs can describe various aspects such as time, place, manner, frequency, and more. Examples of adverbs include "quickly", "loudly", "very", and "often". In the Sidaamu Afoo language, adverbs usually come before the verbs they modify. For instance, in the sentence "baqali cance coyrano," which means "Bekele speaks loudly," the word "cance" meaning "loudly" is an adverb that modifies the verb "coyrano" meaning "speak". Similarly, in the sentence "baqali bero dayino," which means "Bekele came yesterday," the word "bero" meaning "yesterday" is an adverb that modifies the verb "dayino" meaning "came."

3.4.5. Preposition

A preposition is a word that shows the relationship between a noun or pronoun and other words in a sentence. Examples include "in", "on", "at", and "to". Words that can have full meaning only in combination with some other words like nouns, adjectives, or verbs are termed prepositions/postpositions. They do not take any affix and belong to the closed part of speech. Prepositions or postpositions may precede or follow the category to which they add syntactic meaning. Let us see the following sentences: ‘baqali woomasha saaxinete gido maaxi’ - ‘bekele hide the money in the box. gido-‘in the’ in this sentence is postposition.

CHAPTER FOUR

SYSTEM DESIGN AND METHODOLOGY

4.1. Introduction

In this chapter, the proposed bi-directional English–Sidaamu Afoo NMT system will thoroughly be discussed. The discussion includes the research design and architecture of the system that includes the schematic representation of the system with its functional components, their input, and output. In addition, the experimental model that includes the component on which the experimentation is conducted, the software tools used for each component of the system, and the data used for the experimentation of the research will be discussed. Finally, the process of the experimentation and flow of the model will be explained.

4.2. Research methodology

The general research methodology used in this study is an experimental approach. Experimental research design is a type of research methodology used to study cause-and-effect relationships between variables [57]. It involves manipulating one or more independent variables and measuring their effects on a dependent variable while controlling for the effects of extraneous variables. Experimental designs typically sometimes involve randomly assigning participants to different groups, such as a treatment group and a control group, and often use double-blind procedures to minimize bias [58]. Experimental research designs are commonly used in many fields to test hypotheses and determine cause-and-effect relationships between variables. Overall, the experimental research design in neural machine translation involves a systematic and iterative process of selecting, training, evaluating, analyzing, and refining a neural machine translation model to achieve high-quality translations.

4.2.1 Research Question

The first step in experimental research design is to define a clear research question or hypothesis to investigate. Problem definition, shares best practices for defining the problem. How the right set up is often more important than the choice of algorithm. A defined problem illustrated as a research question that is answered by investigating the research experiments depends on the hypothesized model of deep learning.

4.2.2 Corpus Collecting

Once the research question is defined, researchers need to choose a corpus (a collection of texts) for training and evaluating their neural machine translation model. Preparing the training and testing data is a core part of machine learning. It's an active not passive part of machine learning research and is one of the most powerful variables to create high-quality machine learning systems. A corpus known as a "parallel corpus" contains both the original writings in the language L1 and their translations into a variety of languages L2. Parallel corpora frequently only contain information from two languages [59]. To clarify, parallel corpora can include multiple languages and consist of texts written in two or more of them. In our case, we used a parallel corpus to initiate the translation process.

4.2.2.1 Construction of Sidamic-English parallel text corpus

Developing a parallel corpus is very challenging and needs a high cost of human expertise. MT can produce high-quality translation results based on a massive amount of aligned parallel text corpora in both the source and target languages. MT systems need resources that can provide an interpretation/suggestion of the source text and a translation hypothesis. Parallel the corpus consists of parallel text that can promptly locate all the occurrences of one language expression to another language expression and is one of the significant resources that could be utilized for MT tasks [37].

Sidamic-English parallel text corpora was not develop previously so no parallel corpus for MT tasks. we have constructed some parallel text corpus from religious documents by downloading different sources like bible live[60] and others from Sidama-English day-to-day conversation. The

organization of bible text is categorized into verses (sequence of sentences and phrases). A sample verse in Sidaamu Afoo and its equivalent translation in English is shown in Appendix I. To teach the Bidirectional Sidaamu Afoo - English Language from the religious domain, 14300 Sidamic-English parallel sentences corpora have been prepared. The total size of all corpus is 4.13Mb and the number of parallel sentences is 22925. Day-to-day conversation of Sidaamu Afoo to English Conversation preparing manual hardcopy and its total size of the corpus is 20Kb and the number of parallel sentences is 700. We have used it for training mixed corpus that is religious domain and Conversation domain of Sidaamu Afoo to the English language which is the size of a total corpus 2.13Mb and the number of the total parallel sentence is 15000.

We were using many techniques and steps for constructing the parallel corpus

- ✓ Scraping from website

Web scraping is the process of extracting data or information from websites. It involves using software tools to automate the retrieval of data from web pages and then storing that data in a structured format, such as a spreadsheet or database. Web scraping is typically done using programming languages like Python or R, and it involves using techniques like parsing HTML and CSS, simulating user interaction with web pages, and interacting with APIs. Beautiful Soup is a Python library used for web scraping. Here is a pseudo-code of how we could use Beautiful Soup for web scraping:

- ✓ Scraping

Algorithm 1: Algorithm of scraping raw data

	***Input**:*	*Website Url*
	***Output**:*	*Raw data*
1	***Start***	*Import the necessary libraries*
2		*Use the requests library to send a GET request to the website we want to scrape*
3		*Create a Beautiful Soup object from the response content*

4		*Find the HTML elements you want to scrape*
5		*Iterate through the elements you found and extract the data you want*
	End	

Algorithm 2: Algorithm of Aligned to parallel corpus

	***Input**:*	*Downloaded sentence*
	***Output**:*	*Parallel corpus*
1	***Start***	*Take English sentence*
2		*Take Sidaamu Afoo meaning of English sentence*
3		*Aligning to paired sentence*
4		*Check the wrong paired*
5		*Remove the wrong pared sentence*
6		*Remove very long sentences*
5		*Save Parallel corpus*
	End	

4.3 Design of proposed Architecture

Deep learning-based English - Sidaamu Afoo is a translation system where a given English text is translated into an equivalent Sidaamu Afoo sentence through two layers which are Encoder and Decoder. An input English sentence is preprocessed before performing one-hot representation. Word embedding can be formed from one-hot representation and it is given to the Encoder layer

as input. This encoder representation is provided to the decoder, which uses the input to generate an equivalent Sidaamu Afoo translation.

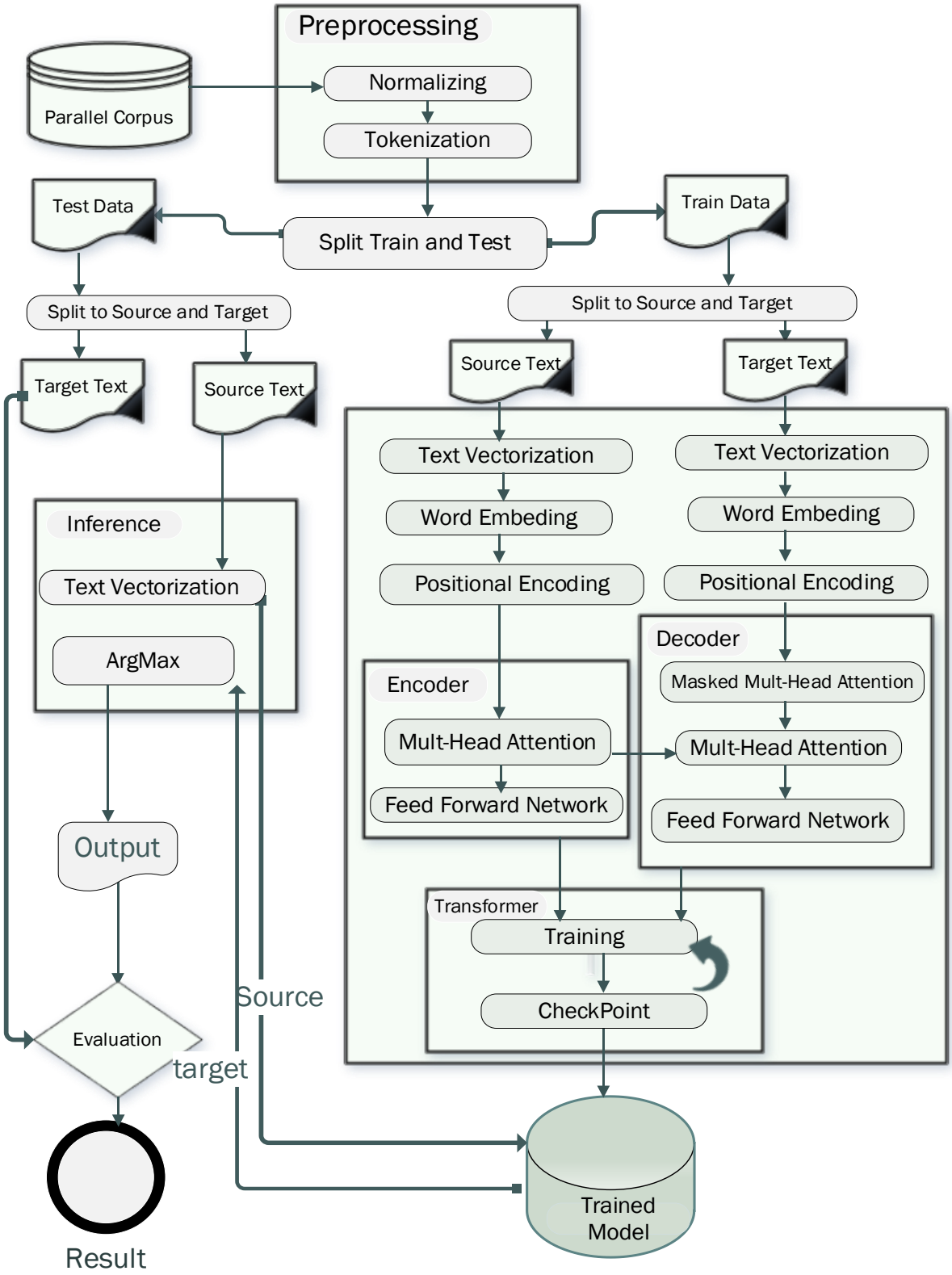


Figure 4. 1 Generally architecture of the Proposed System

The model is explained as a start from corpus preparation up to evaluation and testing of the model. This is the start of the parallel corpus as taking the first step of data preprocessing and on this point processed data two ways that are data normalization and tokenization. Secondly, those cleaned data are split into two parts which are test sets and training sets and using the separated data for different purposes. So, we have used for training the model using training corpus which source text and target text, then thirdly those data change into text vector and encoded to positional encoding each of source texts and target texts, then after as taken input for encoder model and decoder model respectively. Lastly, the model trains the data using check point by epoch and saved it as a trained model. A part of the test set is to use the evaluation of the model that is split into source text and target text, a source text input into the transformer model, and the model responses to the target text and final evaluated the previously referenced target text with model responded target.

4.3.1 Text Preprocessing

The training and testing corpus for this study was gathered from various sources that contain parallel texts (in English and Sidaamu Afoo) in the religious domain, i.e., bible texts. Preprocessing is the process of getting the input sentence into a format that the machine training plan to use. Tokenization, normalization, and the reduction of large sentences are all components of the preprocessing stage.

4.3.1.1 Normalization

In machine learning and deep learning, normalization is a method of data preparation that is widely used to convert between different data formats and comparably organize data without losing information or distorting differences in the ranges of values. Normalization is also required for some algorithms to model the data correctly. For deep learning, not all datasets must be standardized. It is only required when the ranges of characteristics are different [61].

- ✓ Normalizing

Algorithm 3: The algorithm creates a Cleaned corpus

	Input:	*Raw Corpus*
	Output:	*Cleaned Corpus*
1	***Start***	*Load raw corpus*
2		*Splitting corpus into a sentence*
3		*The tokenizing sentence into a word*
4		*Removing the nonprintable character*
5		*Convert into lowercase*
6		*Replacing the Sidaamu Afoo Character ‘, “, “” with other*
7		*Removing punctuation Character*
8		*Again, re-backing **step 5** replaced the character*
9		*Appending all sentences and resulting clean corpus*
	End	

Once we have read the data, we will keep the first examples for faster training. In case we want to develop a higher-performance model we need to use the full data set. Then we must clean the data by removing capital letters and punctuation.

4.3.1.2 Tokenization

Tokenization involves breaking down the text into a series of meaningful pieces, referred to as tokens, which serve as the fundamental units of natural language. Tokens can be comprised of words, letters, or sub-words. Tokenization is hence extensively applicable. classified into three types – word, character, and sub-word (n-gram characters) tokenization. The most popular tokenization algorithm is called Word Tokenization, and it divides the text into individual words depending on a specific delimiter. Various word-level tokens are created depending on the delimiters. Tokenization is a typical task in natural language processing. It's an essential step in

both conventional NLP techniques like the Count Vectorizer and advanced DL based frameworks like Transformers [62].

- ✓ Tokenization and vocabulary creation

Algorithm 4: The algorithm of creates Vocabulary

	Input:	*Cleaned Corpus*
	Output:	*Vocabulary*
1	***Start***	*Load Cleaned corpus*
2		*Shuffling the loaded data*
3		*Splitting into train, and test.*
4		*Tokenizing each train and test*
5		*Identifying unique words and save in the vocabulary*
6		*Create vocabulary size*
	End	

This algorithm gives the following results

- ✓ Maximum length Sidaama Afoo sentence: 21
- ✓ Maximum length English Language sentence: 28
- ✓ Vocabulary size of Sidaama Afoo: 24155
- ✓ Vocabulary size of English: 9133

From the previous code, we have a maximum length of 24155 words for Sidaamu Afoo sentences and 9133 words for English**.** Here we can see the advantage of using an encoder-decoder model, previously we had the limitation of working with equal-length sentences, so we needed to apply padding to the English sentences up to 28, now it is half**.** Consequently, and more importantly, it also reduces the number of LSTM time steps, reducing computation needs and complexity. We apply padding to make the maximum length of the sentences in each language equal. Now that we have the data ready let's build the model.

4.3.2 One hot encoding

Deep learning algorithms can only understand numerical data, not text. One hot encoding is the process of converting categorical data into numerical form to feed into these algorithms, leading to improved predictions and classification accuracy. This encoding technique creates a new binary feature for each possible category and assigns a value of 1 to the feature that corresponds to the original category. This type of encoding is a commonly used method for preprocessing categorical features in machine learning models and is applied to the integer representation of the data [63]. The process involves removing the integer-encoded variable and introducing a new binary variable for each unique integer value. It starts with a column of categorical data that has been label encoded, then it splits that column into multiple columns by replacing the numbers with either 1s or 0s based on the value of each column. Although this method can be effective for ordinal data, it can cause problems with predictions and produce poor results if the input data doesn't have any order or ranking for the category value [64].

4.3.3 Text Vectorization

Text vectorization is the process of converting text data into a numerical representation. The conventional method of transforming input data from its natural format (such as text) into real number vectors is employed as it is the format that deep learning models can accommodate. To create using simple techniques which involve three operations. First, the input text is tokenized that A sentence is represented as a list of its constituent words, and it’s done for all the input sentences. The second step involves creating a vocabulary by selecting all the tokenized words and unique words, which are then sorted alphabetically. After that, a sparse matrix is created for the input, based on the frequency of words in the vocabulary. Each row in the sparse matrix represents a sentence vector, with the number of columns equal to the vocabulary size.

4.3.4 Word Embedding

Word embeddings extract features from the text, enabling the utilization of these features in machine-learning models to work with text data [65]. Word embeddings are a crucial aspect of NLP (Natural Language Processing) and machine learning when working with text data. The goal

of word embeddings is to represent words in a way that captures their meaning and relationship with other words in a dense, continuous vector representation. By converting words into numerical vectors, we can feed these representations into machine learning models, allowing them to work with the text data and perform various NLP tasks such as sentiment analysis, text classification, and language translation, to name a few.

In other words, word embeddings are a way of transforming text into a numerical format that can be easily understood and processed by machine learning algorithms [66]. This representation captures the semantic meaning of the words, making it easier for the models to understand the context and relationships between words in the text data. A word vector is a numerical representation of a word in a lower-dimensional space, where words with similar meanings have similar representations and approximate meanings. Each word vector is comprised of fifty unique features, each represented by one of the fifty values in the vector. Anything that connects words is referred to as a feature. For instance, woman, girl, queen, etc... Each word vector has values corresponding to these features. To conclude, word embeddings play a significant role in making NLP and machine learning with text data possible, by representing words in a meaningful, numerical format that enables the efficient processing of text data by machine learning algorithms.

- An Objective of word embeddings

  - ✓ To decrease dimensionality
  - ✓ forecast the words around a word using its meaning
  - ✓ Inter-word semantics need to be captured.

- How Do Word Embeddings Work?

  - ✓ They provide data for models of machine learning.
  - ✓ Take the words,
  - ✓ express them numerically,
  - ✓ and then use them for training or inference

We experimented with word embeddings of varying dimensionalities as this parameter affects the model's accuracy. To determine the optimal dimensionality, we trained LSTM and transformer models using different deep-learning approaches with different dimensionalities. Specifically, the

LSTM model was trained with a dimensionality of 512, while the transformer model was trained with a dimensionality of 256.

4.3.5 Positional encoding

Positional encoding is a method used in transformer models to encode the position or location of an entity in a sequence. This is important because the position of an item in the sequence can impact the meaning and context of the data. For example, in natural language processing, the order of words can greatly impact the meaning of a sentence. To assign a unique representation to each position in the sequence, a positional encoding is added to the embedding of the item at that position. This ensures that each position has its distinct representation, allowing the model to differentiate between items in different positions [67]. There are several reasons why using only the index value to represent the position of an item is not sufficient in transformer models. Firstly, the index value can grow very large in magnitude for long sequences, making it difficult to effectively train the model. Secondly, normalizing the index value to lie between 0 and 1 can cause issues when dealing with variable-length sequences. This is because each sequence would be normalized differently, leading to the inconsistent representation of the positions.

Transformers employ an efficient positional encoding method, where each position in the sequence is mapped to a vector. This results in a matrix output from the positional encoding layer, with each row in the matrix representing an encoded object in the sequence along with its positional information [68]. The illustration below depicts an example of a matrix that only contains positional information. In conclusion, positional encoding is a crucial aspect of transformer models, providing a unique representation for each position in a sequence. By using positional encoding, the model can effectively capture the meaning and context of the data, regardless of the length of the sequence or the size of the index value.

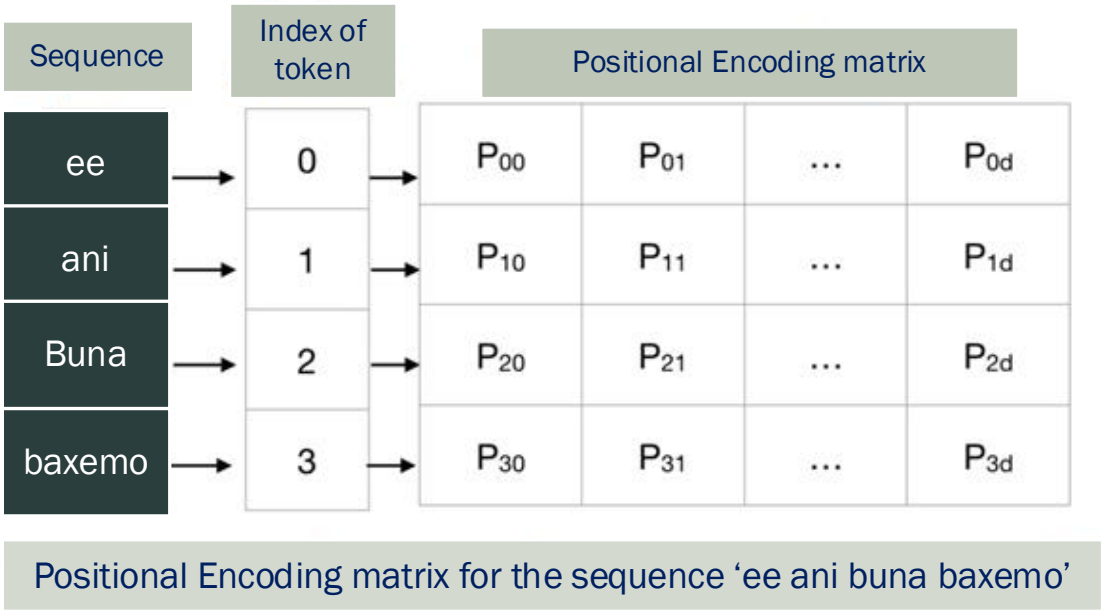


Figure 4. 2 positional Encoding

In the case of a transformer, Attention layers see their input as a set of vectors, with no sequential order. This model also doesn't contain any recurrent or convolutional layers. Because of this a "positional encoding" is added to give the model some information about the relative position of the tokens in the sentence. After obtaining the embedding vector, the positional encoding vector is added to it. These embeddings are a representation of a token in a d-dimensional space, where tokens with analogous meanings are situated in closer proximity to each other. Although embeddings represent the meaning of tokens in a sentence, they do not capture the relative position of tokens. By including positional encoding, tokens can be mapped in a d-dimensional space based on both their meaning and their position within the sentence. This results in tokens with similar meanings and positions being represented closer to each other in the space.

4.3.6 Encoder-Decoder

The Encoder-Decoder is a type of neural network introduced in 2014 and widely used in various projects, especially in language translation systems [54]. It consists of two separate neural networks, where the first network takes a sentence as input and outputs a series of numbers, and the second network takes this numerical representation as input and produces the translated sentence. The first network is known as the encoder and the second network is called the decoder.

The Encoder-Decoder is a popular neural network architecture that has been widely used in various projects, especially in language translation systems [69]. It is a fundamental cornerstone in this field, providing a way to translate sentences from one language to another. The architecture consists of two separate neural networks, which work together to produce the translation.

The first network, known as the encoder, takes a sentence as input and outputs a numerical representation of that sentence. This numerical representation is then used as input for the second network, the decoder. The decoder takes the numerical representation as input and produces the target language sentence. The encoder-decoder architecture is designed to handle the challenges of language translation, such as preserving the meaning and structure of the input sentence while converting it into another language.

The encoder network processes the input sentence word by word, converting each word into a numerical representation that captures its meaning and context. This numerical representation is often called a vector, and it is created using a technique called word embedding. The encoder then combines these word vectors into a single numerical representation of the input sentence, known as the encoder vector.

The decoder network takes the encoder vector as input and generates the target language sentence, one word at a time. It uses the encoder vector to understand the context of the input sentence and then generates the appropriate words in the target language, preserving the meaning and structure of the input sentence [70].

For example, if we have the Sidama sentence "ee, ani buna baxeemo" and it's English translation "yes, I like coffee," we can see the encoding and decoding process in the encoder-decoder architecture. The input sentence is first encoded into a numerical representation by the encoder, and this representation is then used as input for the decoder, which produces the target language sentence.

In conclusion, the Encoder-Decoder architecture is a powerful tool for language translation. Converting sentences into numerical representations provides a way to preserve the meaning and structure of the input sentence while translating it into another language. The use of two separate neural networks, the encoder, and decoder, enables the architecture to handle the complexities of language translation, making it a valuable tool in this field.

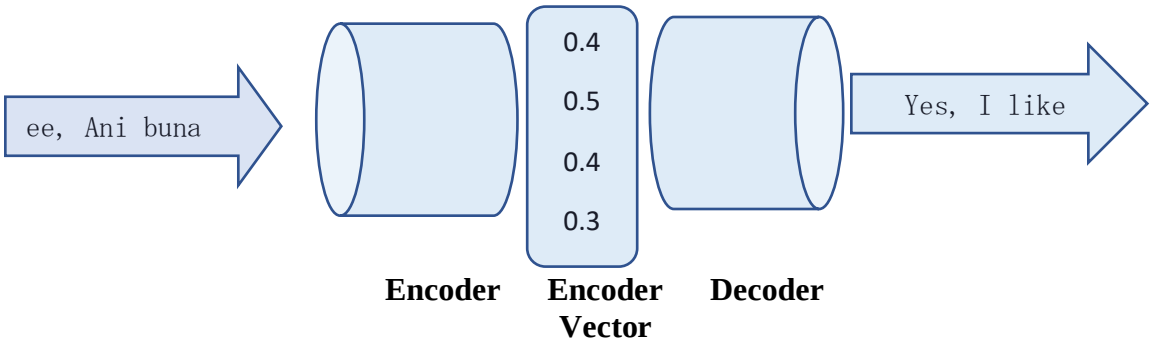


Figure 4. 3 encoder-decoder architecture

4.3.7 Training and Inference

Deep learning operates in two distinct phases: training and inference. The training phase involves feeding a curated dataset to the deep learning model so that it can learn the patterns and relationships in the data. This learning process enables the model to understand the type of data it will be analyzing in the future. In the inference phase, the model can make predictions based on live data and produce results that can be acted upon. This is when the DNN applies its learned knowledge to real-world data and makes predictions about what that data represents. Deep learning inference refers to the process of utilizing a trained DNN model to make predictions on new, unseen data.

During the deep learning training process, the DNN already performs inference as it attempts to predict the output for each input it receives. Therefore, deploying a trained DNN for inference can be straightforward. One approach could be to simply copy the trained DNN and use it as-is for making predictions on new data. This highlights the importance of the training phase in deep learning as it enables the DNN to accurately predict new data in the inference phase. In essence, the training phase is when the DNN is taught how to analyze data and make predictions, while the inference phase is when the model uses that knowledge to make predictions on new, unseen data. The goal of deep learning is to achieve accurate predictions in the inference phase, which is only possible if the model has been adequately trained during the training phase. Training a deep learning model involves a trial-and-error process until the network can accurately make predictions based on the desired outcome. The deep learning training process is how a DNN sifts through existing data and makes conclusions about what the data represents.

Whenever the DNN makes an incorrect conclusion, the result is fed back into the system, allowing it to learn from its mistakes and improve over time. This continual feedback loop updates the connections between the artificial neurons, which increases the accuracy of the model's predictions in the future. As the DNN is presented with new and potentially more complex data, it should be able to analyze and categorize the information accurately. Through this process of trial-and-error, the DNN continually improves its ability to make accurate predictions. Ultimately, it will continue to learn from its encounters and become more intuitive over time [71].

4.3.8 Evaluation

Model evaluation is a crucial step in deep learning, which involves assessing the performance of a deep learning model using various evaluation metrics. These metrics provide insight into the strengths and weaknesses of the model and help in understanding its efficacy. The results of model evaluation play a significant role in both the initial research phase, where different models are compared, and in monitoring the performance of a deployed model [72]. There are several evaluation metrics commonly used for measuring the performance of deep learning models, especially in the case of classification tasks. Various evaluation metrics such as accuracy, precision, confusion matrix, log-loss, and bleu score can be utilized. For this specific case, accuracy, log-loss, and bleu score are employed as the evaluation metrics.

Accuracy measures the number of correct predictions made by the model, relative to the total number of predictions. Log-loss, on the other hand, is a measure of the confidence of the model's predictions, with lower log-loss indicating higher confidence. Bleu score is a commonly used evaluation metric in the field of NLP and measures the similarity between a predicted sequence and the reference sequence. These metrics, when used together, provide a comprehensive evaluation of the model's performance, and help in fine-tuning the model to improve its accuracy and efficiency.

4.3.9 Checkpoint

In deep learning, a checkpoint is a snapshot of a model's parameters at a specific point in time during training. Checkpoints are useful because they allow you to save the model's progress and resume training from where you left off, even if the training process is interrupted or stopped. In

the context of the Transformer model, checkpoints can be used to save the model's parameters at specific intervals during training. This is useful because the Transformer model can be quite large and training can take a long time, so it's important to be able to save the model's progress and resume training from where you left off. To implement checkpoints in the Transformer model, it can use a library like TensorFlow or PyTorch, which provide built-in support for saving and loading model checkpoints. In TensorFlow, for example, we used the tf.train.Checkpoint class to save and restore the model's parameters.

4.4 LSTM Model

The LSTM is a type of recurrent neural network designed to handle long-term dependencies in data [73]. This is achieved by allowing information to persist, or be remembered, within the network. In contrast to traditional RNNs, which can struggle with the vanishing gradient problem, LSTMs are capable of handling long-term dependencies by allowing the network to remember previous information and use it for processing the current input.

The LSTM network consists of three main components: the input gate, the forget gate, and the output gate. The input gate regulates the information that is allowed to enter the cell state of the LSTM, while the forget gate determines which information should be forgotten. The output gate, on the other hand, decides which information should be passed on to the output layer. These three gates work together to regulate the flow of information within the network, allowing it to learn and retain long-term dependencies. LSTMs are an advanced type of RNN that is well-suited to handling long-term dependencies in data. They can persist information and regulate the flow of information within the network, making them a powerful tool in various applications such as natural language processing and machine translation.

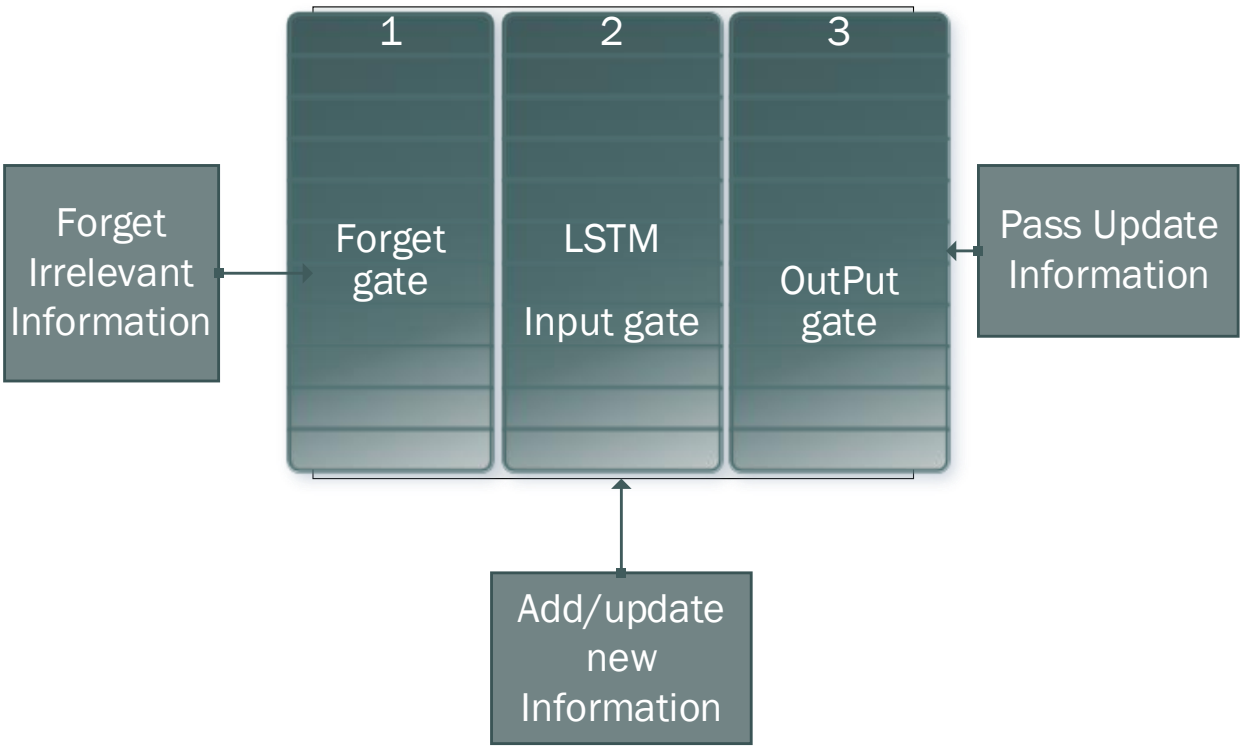


Figure 4. 4 components of LSTM models

These three parts of an LSTM cell are known as gates. The first part is called **Forget gate, the** second part is known as **the Input gate** and the last one is **the Output gate**. Just like a simple RNN, an LSTM also has a hidden state where $\mathbf{H_{(t-1)}}$ represents the hidden state of the previous timestamp and $\mathbf{H_t}$ is the hidden state of the current timestamp. In addition to that LSTM also have a cell state represented by $\mathbf{C_{(t-1)}}$ and $\mathbf{C_{(t)}}$ for the previous and current timestamp respectively. Here the hidden state is known as short-term memory and the cell state is known as long-term memory. Refer to the bellow general diagram of the LSTM model.

4.4.1 Forget Gate

In a cell of the LSTM network, the first step is to decide whether we should keep the information from the previous timestamp or forget it. Here is the equation for the forgetting gate.

$$f_t = \sigma\left(X_t * U_f + H_{t-1} * W_f\right) \text{ ---------- 4.1}$$

We tried to understand and define the equation, here

- ✓ $\boldsymbol{X_t}$ - input to the current timestamp
- ✓ $\boldsymbol{U_f}$ - weight associated with the input
- ✓ $\boldsymbol{H_{t-1}}$ - The hidden state of the previous timestamp
- ✓ $\boldsymbol{W_f}$ - It is the weight matrix associated with the hidden state

Later, a sigmoid function is applied to it. That will make f_t a number between 0 and 1. This f_t is later multiplied with the cell state of the previous timestamp as shown below.

$$C_{t-1} * f_t = 0 \quad \text{if } f_t = 0 \text{ (}\textit{Forget everything}\text{)} \text{ ----------4.2}$$

$$C_{t-1} * f_t = C_{t-1} \quad \text{if } f_t = 1 \text{ (}\textit{Forget nothing}\text{)} \text{ ---------4.3}$$

If f_t is 0 then the network will forget everything and if the value of f_t is 1 it will forget nothing.

4.4.2 Input Gate

The Input Gate in LSTMs is responsible for determining the relevance of the new input data and deciding the amount of information that should be added to the memory cell. Its function is expressed through the following equation:

$$i_t = \sigma(X_t * U_i + H_{t-1} * W_i) \text{ ----------4.4}$$

Here,

- ✓ $\boldsymbol{X_t}$ - Input at the current timestamp t
- ✓ $\boldsymbol{U_i}$ - weight matrix of input
- ✓ $\boldsymbol{H_{t-1}}$ - A hidden state at the previous timestamp
- ✓ $\boldsymbol{W_i}$ - Weight matrix of input associated with hidden state

Again, it needs to apply the sigmoid function over it. As a result, the value of i at timestamp *t* will be between 0 and 1.

4.4.3 New information

$$N_t = \tanh\left(X_t * U_c + H_{t-1} * W_c\right)\text{----------}4.5$$

Now the new information that needed to be passed to the cell state is a function of a hidden state at the previous timestamp t-1 and input x at timestamp t. The tanh activation function is used in this step. As a result, the new information value will always fall between -1 and 1. If the new information value (Nt) is negative, it is subtracted from the cell state. Conversely, if the value is positive, the information is added to the cell state at the current timestamp. However, the Nt won't be added directly to the cell state. Here comes the updated equation

$$C_t = f_t * C_{t-1} + i_t * N_t \textit{(Updating cell state)}\text{ ---------}4.6$$

Here, C_{t-1} is the cell state at the current timestamp and the others are the values we have calculated previously.

4.4.4 Output Gate

Here is the equation of the Output gate, which is pretty similar to the two previous gates.

$$O_t = \sigma\left(X_t * U_o + H_{t-1} * W_o\right)\text{ ---------- }4.7$$

Its value will also lie between 0 and 1 because of this sigmoid function. Now to calculate the current hidden state we will use O_t and tanh of the updated cell state. As shown below.

$$H_t = O_t * tanh\left(C_t\right)\text{ ---------- }4.8$$

It turns out that the hidden state is a function of long-term memory (C_t) and the current output. If you need to take the output of the current timestamp just apply the SoftMax activation on hidden state H_t

$$Output = Softmax(H_t)\text{----------}4.9$$

Here the token with the maximum score in the output is the prediction.

This is the More intuitive diagram of the LSTM network.

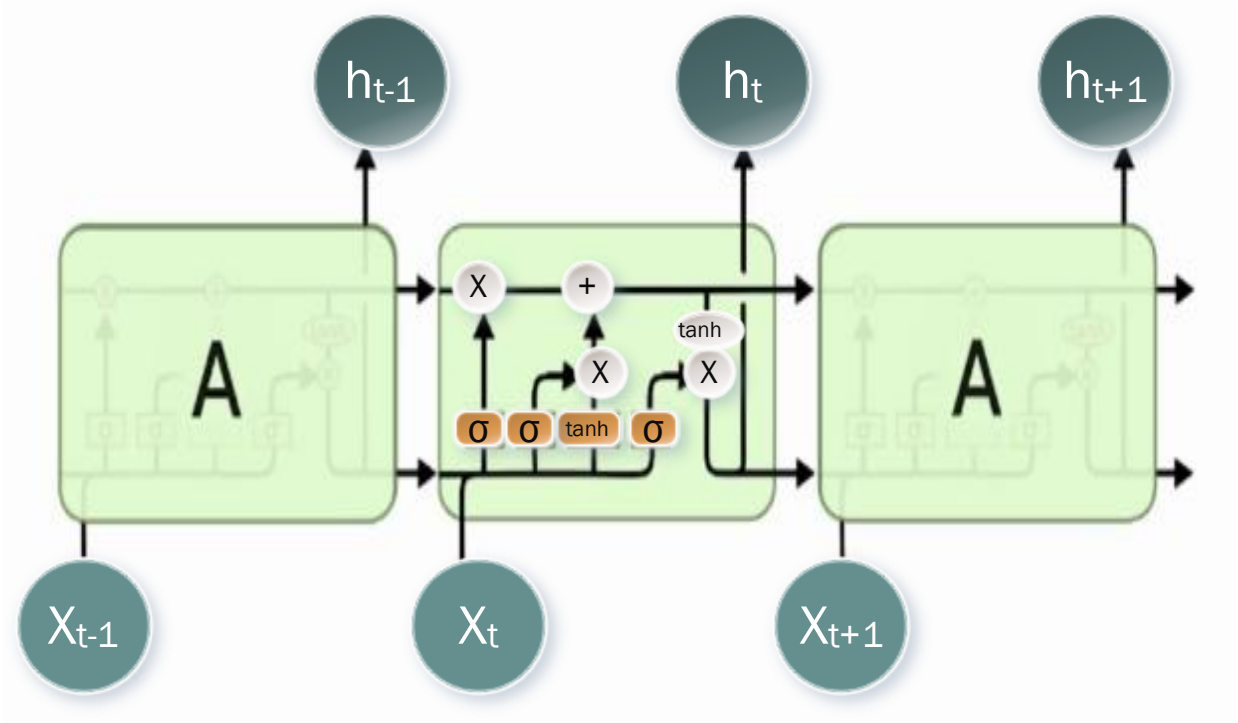


Figure 4. 5 Lstm general diagram

We have split the model into two parts**,** first, we have an **encoder** that inputs the source sentence and produces a hidden vector. To build the encoder, a two-step process is employed. Firstly, an Embedding layer is utilized to convert the words into vectors. Secondly, an RNN is used to calculate the hidden state, with LSTM being the specific RNN utilized. The output of the encoder is subsequently used as input for the decoder. In the decoder, a similar approach is taken, employing an LSTM layer along with a dense layer that predicts the target word.

To do experiments of LSTM it going to through many steps:

1. Pre-processing

We have discussed understanding of what is Normalizing and tokenization in the above topic on preprocessing, but on this, we will see how work in a real experiment using an algorithmic way as well as possible we will see on keras tools which using do Python programming

2. Model Development

In the following section, we will create the model and explain each layer as we add our python code in part of the appendix.

- ✓ Encoder

As depicted in the model image, the initial layer to be established is the embedding layer. To accomplish this, we need to include an Input Layer first, where the sole parameter to be taken into account is "shape," denoting the maximum length of Sidama Afoo sentences, which is 28 in our scenario.

We will then connect it to the embedding layer and what embedding we have seen previously topic, here the parameters to take into account is 'input dim' which is the length of the Sidaamu Afoo vocabulary, and 'output dim' which is the shape of the embedding vector. This layer will convert any of the Sidama Afoo words into a vector of the shape of the output dimension.

The concept behind this is to extract the meaning of the word in a form of a spatial representation where each dimension will be a characteristic defining the word. a world will be converted into a vector of shape 512. The higher the output dimension the more semantic meaning you can extract from each word, but also the higher the calculations required and the processing time.

Next, we will add the LSTM layer of size 64. Even though each time step of the LSTM outputs a hidden vector, we will focus our attention on the last one, therefore the parameter return sequence is 'False'. We will see how the LSTM layer works with return sequences=True for the decoder.

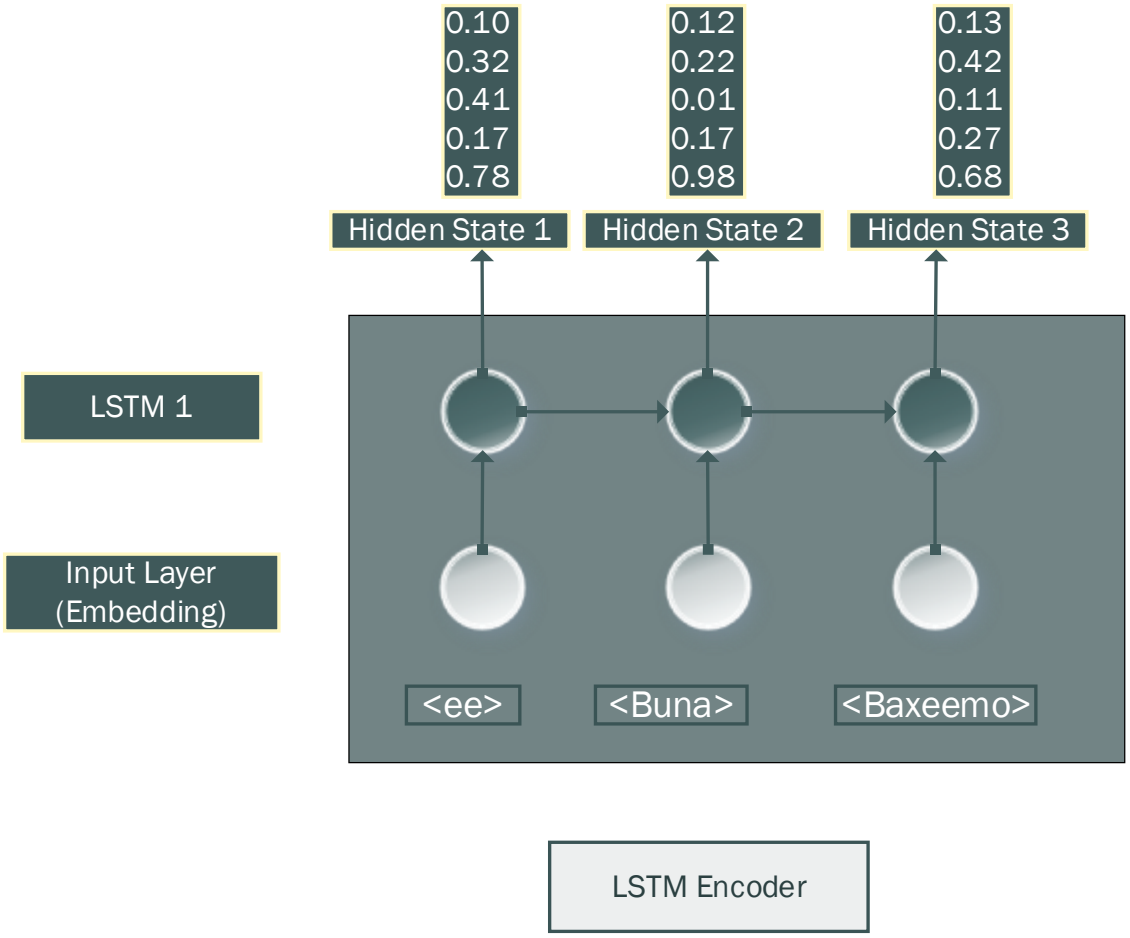


Figure 4. 6 Lstm encoder diagram

✓ Decoder

The output of the encoder layer will be the hidden state of the last time step. We will then need to feed this vector into the decoder. Let's look more precisely at the decoder part and understand how it works.

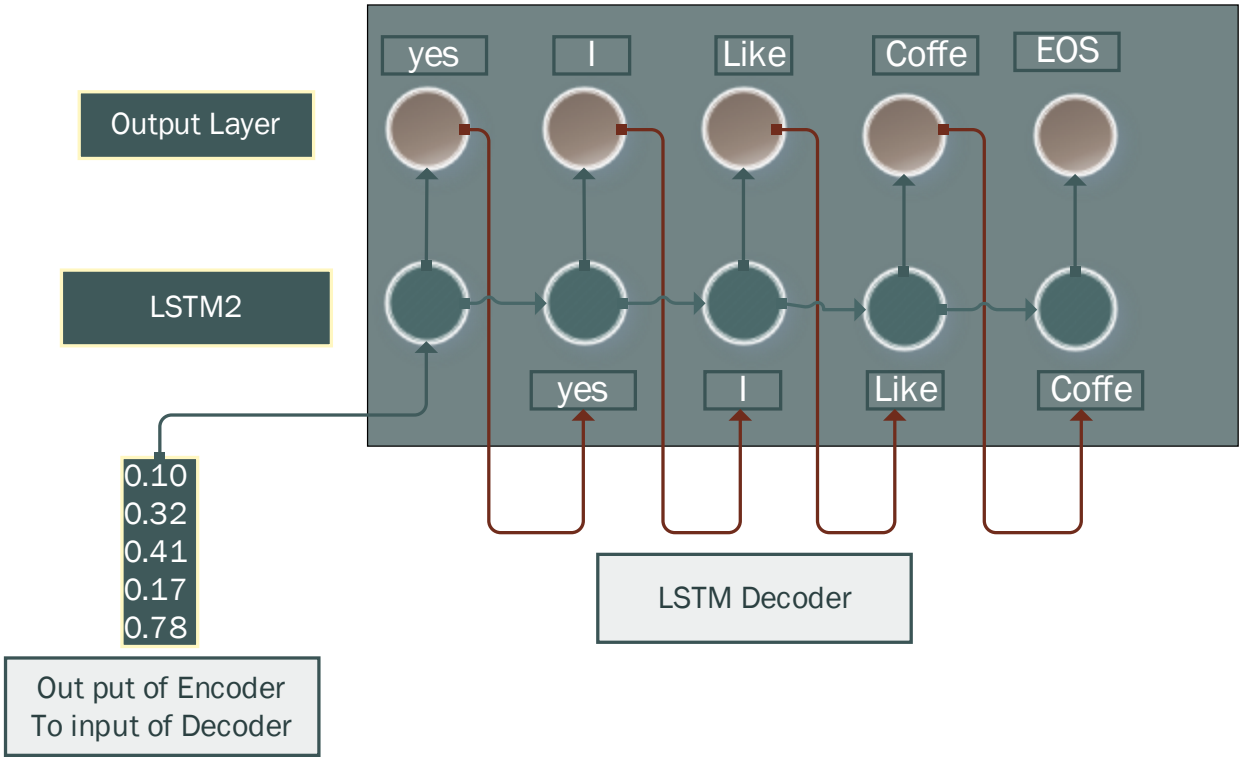


Figure 4. 7 LSTM decoder diagram

As we can see in the image the hidden vector is repeated n times, so each time step of the LSTM **receives the same vector**. To have this same vector for every time step we need to use the layer Repeat Vector, as its name implies its role is to repeat the vector it is receiving, the only parameter we need to define is n, the number of repetitions. This number is equal to the number of time steps of the decoder part, in other words, the maximum English sentence length is 21.

Once we have the input ready, we will continue with the decoder. This is also built with an LSTM layer, the difference is the parameter return sequences, which in this case is 'True'**.** a parameter for Previously, in the encoder portion, only one vector was expected at the final time step, while all other vectors were disregarded. However, now we expect an output vector at each time step, which enables the Dense layer to generate predictions. The final step is to predict the translated word. For this, we need to use a Dense Layer. The parameter we need to define is the number of units, this number of units is the shape of the output vector and it needs to be the same as the length of the English vocabulary. because The vector will be all values close to zero, except one of the units that will be close to 1**.** We then need to map the index of the unit that outputs a 1 with a dictionary where we map each unit to a word. For example, if the input is the word and the output is a vector where

all are zeros and then the unit some number is 1, we map this index against the dictionary containing the English words and we get the value target word.

We have just seen how to apply the Dense layer and predict one word, but how do we predict the whole sentence? Because we are using return sequence=True, to predict words one at a time, the LSTM layer generates a vector at each time step. As a result, we must apply the Dense layer described earlier to each time step's output. To do this, Keras has developed a specific layer called Time Distributed, which applies the same Dense layer to every time step.

Lastly, we stack the layers to create the model and add a function loss. Once we define the model, then a need to train it. When the model is trained, we can make our first translation. You will also find the function 'logits_to_sentence' that maps the output of the dense layer with the English vocabulary.

As generally, an encoder-decoder structure allows for **a** different input and output sequence length. First, we use an Embedding layer to create a spatial representation of the word and feed it into an LSTM layer that outputs a hidden vector, because we just focus on the output of the last time ste**p,** we use return sequences=False. This output vector needs to be repeated the same number of times as the number of time steps in the decoder part, for that we use the Repeat Vector layer. The decoder will be built with the LSTM layer and the parameter return sequences=True, so each output of the time steps is used by the Dense layer.

4.5 Transformer Model

The attention mechanism has been a significant contributing factor to the advancement of neural machine translation systems. One notable model that leverages this concept is the Transformer, which has proven to be faster in training compared to traditional models such as Google Neural Machine Translation. Its success can be attributed to its ability to parallelize computations, making it a highly efficient model for various tasks [74]. The research paper titled "Attention Is All You Need" proposed a novel network architecture called the Transformer, which is based entirely on attention mechanisms and eliminates the need for recurrence and convolutions [35]. The paper

demonstrated that these models are superior in quality to traditional models and can be parallelized more easily, resulting in significantly faster training times. The experiments conducted on two machine translation tasks supported these findings [35] [75]. Using the transformer model on top of higher translation quality, it requires less computation to train and is a much better fit for modern machine learning hardware, speeding up training by up to an order of magnitude.

We have illustrated below how we apply the Transformer to machine translation. In machine translation neural networks, there is usually an encoder that reads the input sentence and produces a representation of it, followed by a decoder that generates the output sentence word by word while utilizing the encoder's generated representation. The Transformer architecture, on the other hand, begins by generating initial embeddings and positional encoding for each word. These embeddings are then represented and passed to the subsequent step. After generating the initial embeddings, the Transformer utilizes self-attention to combine information from all the other words in the sentence, creating a new representation for each word that takes into account the entire context. These representations are then passed to the feed-forward network. This process is repeated multiple times in parallel for all words, resulting in the creation of new representations. The decoder follows a similar approach, but, attends all words of final representation generated from the encoder, instead of generating one word at a time.

- Step to step explanation of how to work transformer model

The Transformer model is a popular neural network architecture used for machine translation. Here is a step-by-step overview of how the Transformer model works for machine translation:

Step 1. Input Embedding: The input sentence in the source language is first converted into a sequence of embeddings, where each word is represented as a vector. The embedding layer maps the input words to a continuous vector space, allowing the model to capture the semantic meaning of each word.

Step 2. Positional Encoding: Since the Transformer model does not use recurrent layers, it needs a way to encode the position of each word in the input sequence. Positional encoding is added to the input embeddings to provide the model with information about the position of each word in the sequence.

Step 3. Encoder Layers: The input embeddings with positional encoding are then passed through a stack of encoder layers. Each encoder layer consists of a multi-head self-attention mechanism and a position-wise feedforward neural network. The self-attention mechanism allows the model to attend to different parts of the input sequence, while the feedforward network applies non-linear transformations to the input embeddings.

Step 4. Decoder Layers: The decoder takes the output from the encoder layers and generates the output sentence in the target language. Similar to the encoder layers, the decoder consists of a stack of multi-head attention and feedforward neural network layers. The decoder also includes an additional attention mechanism that attends to the output of the encoder layers to incorporate information from the input sentence.

Step 5. Output Embedding and Linear Layer: The output from the decoder layers is then passed through an output embedding layer and a linear layer to generate the final translation.

Step 6. Training: During training, the model is given pairs of source and target sentences, and the parameters of the model are adjusted to minimize the difference between the predicted and actual target sentences.

Step 7. Inference: During inference, the model takes a source sentence as input and generates the corresponding target sentence using the learned parameters.

Overall, the Transformer model works by encoding the input sentence into a sequence of embeddings, processing the embeddings through a stack of encoder and decoder layers, and generating the output sentence in the target language.

We have to create a translator that uses transformers to convert English to Sidaamu Afoo and revers. So, if we look at it as a left box, our network takes as input a Sidaamu Afoo sentence and returns an English sentence or reverses the language.

Figure 4. 8 transformer input-output diagram

Previously, we have observed the existence of an encoding component and a decoding component, as well as followed the connections between these two components.

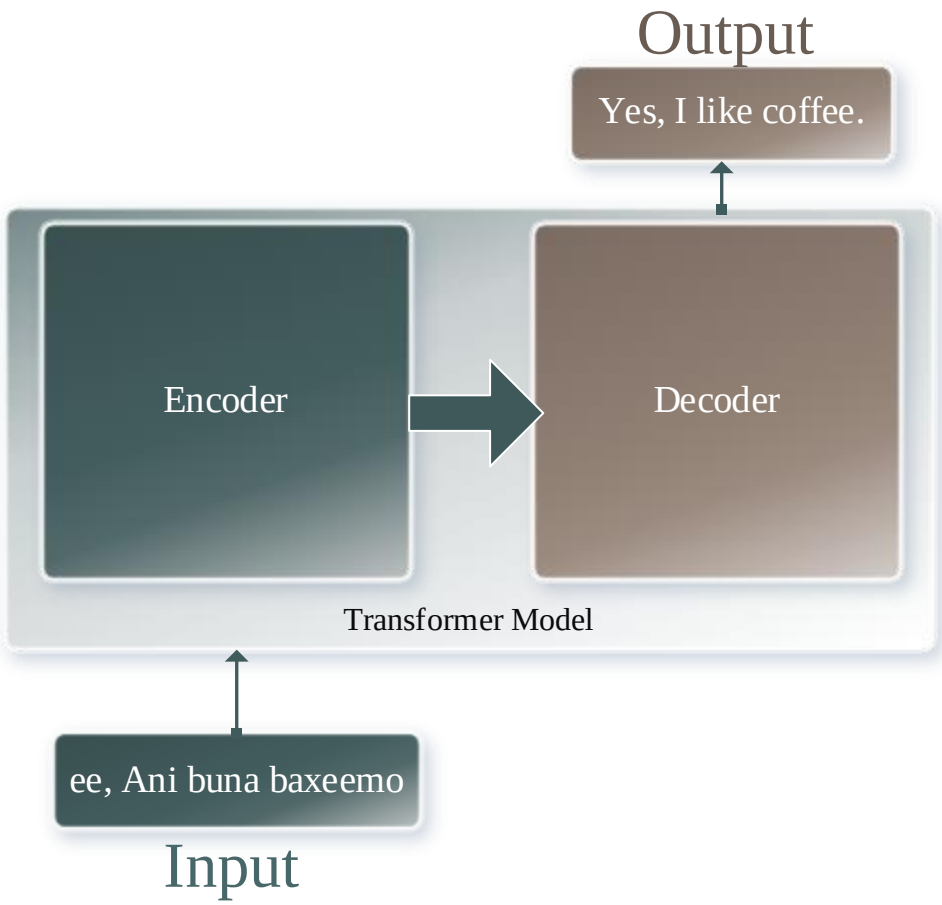


Figure 4. 9 Transformer encoder-decoder components

4.5.1 Stack of the encoder and Decoder in the Transformer

The encoding component consists of multiple encoders stacked on top of each other, with the current example having six of them. The same number of decoders is used in the decoding component. There is no set number for the number of encoders and decoders used, and experimentation with different arrangements is possible

The multiple stacks of encoder-decoder are used for machine translation by processing the input sentence in multiple layers, with each layer capturing a different level of abstraction or meaning. The encoder-decoder architecture consists of two main components: an encoder and a decoder. The encoder takes the input sentence and generates a fixed-length vector representation of the sentence, which contains all the information necessary to translate the sentence into the target language. The decoder then takes this vector representation and generates the output sentence word by word. However, a single layer of encoder-decoder may not be sufficient to capture all the nuances of the input sentence, especially for longer sentences. Therefore, multiple stacks of encoder-decoder layers are used to improve the translation quality. Each additional layer in the encoder-decoder stack refines the representation of the input sentence, allowing the model to capture more complex relationships between words and phrases. Similarly, each additional layer in the decoder stack refines the generated output sentence, allowing the model to produce more accurate translations. Overall, the multiple stacks of encoder-decoder allow the machine translation model to capture more complex relationships and nuances in the input sentence, resulting in higher translation quality.

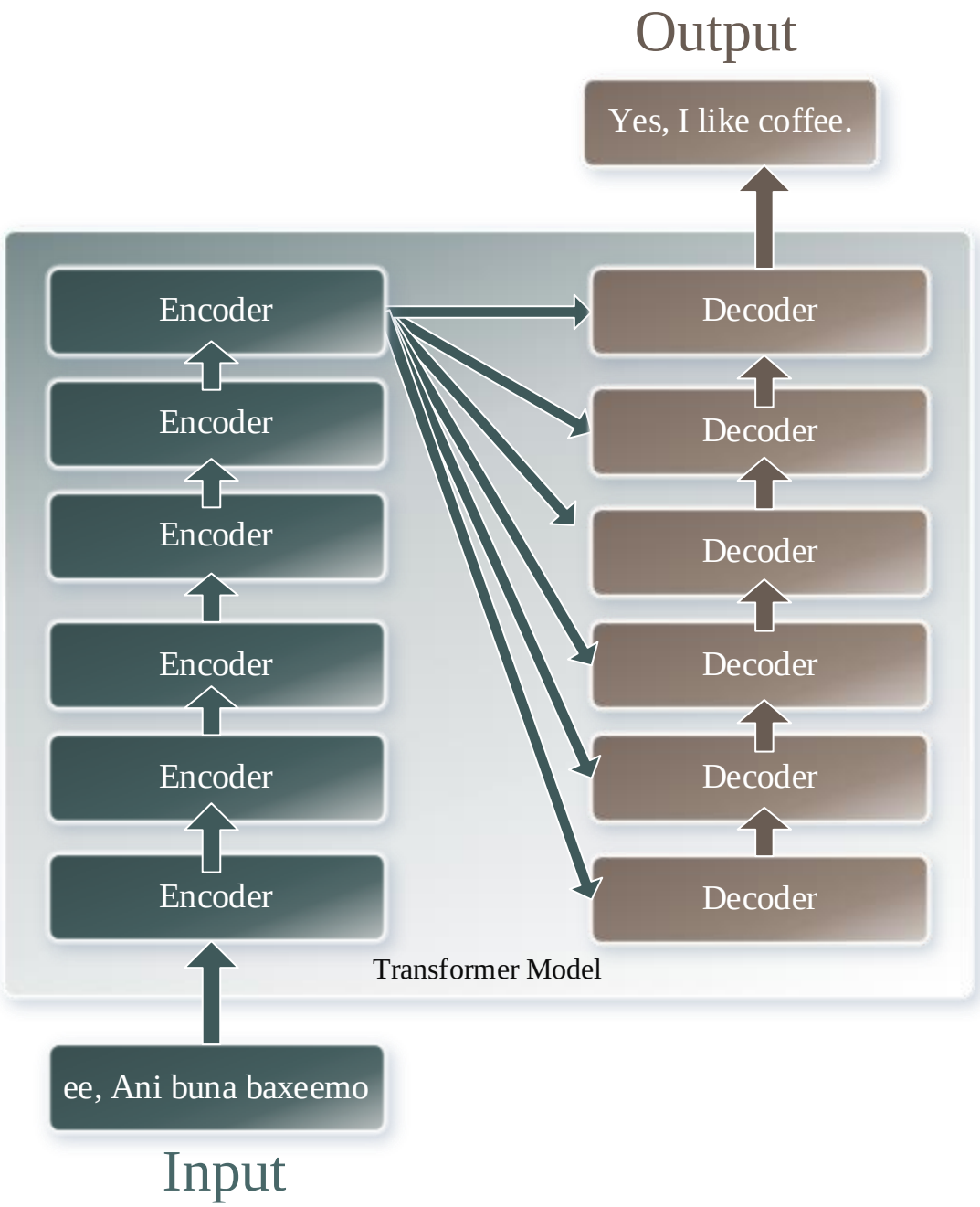


Figure 4. 10 Transformer encoder-decoder stack

4.5.2 The Encoder-Decoder Sub-Layers in Transformer

The encoders, are all identical in structure, consisting of two sub-layers each, but they do not share the same weights.

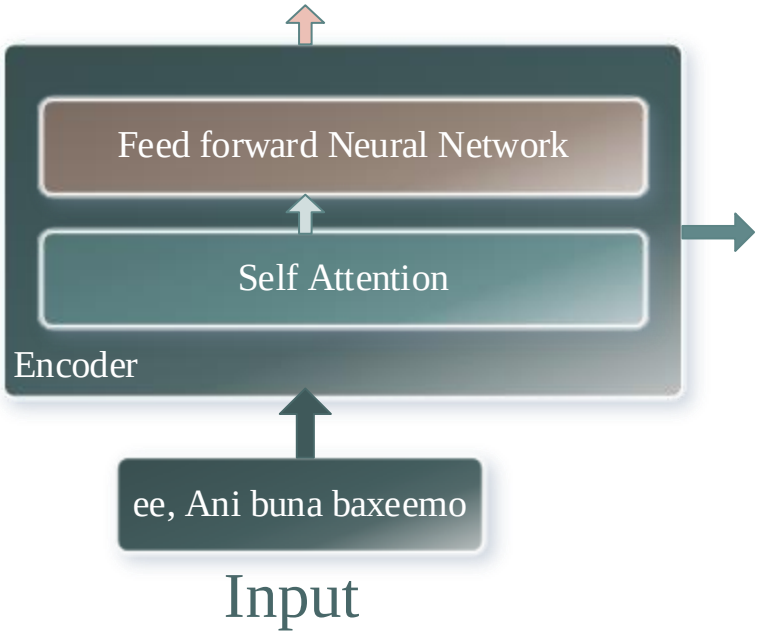


Figure 4. 11 Transformer encoder-decoder sublayer

The encoding component first processes the inputs through a self-attention layer, allowing the encoder to examine other words in the source sentence as it encodes a specific word. The output of the self-attention layer is then fed into a feed-forward neural network, which is independently applied to each position. The decoder, like the encoder, also has both self-attention and feed-forward layers, but it also includes an additional attention layer that helps it concentrate on relevant parts of the input sentence.

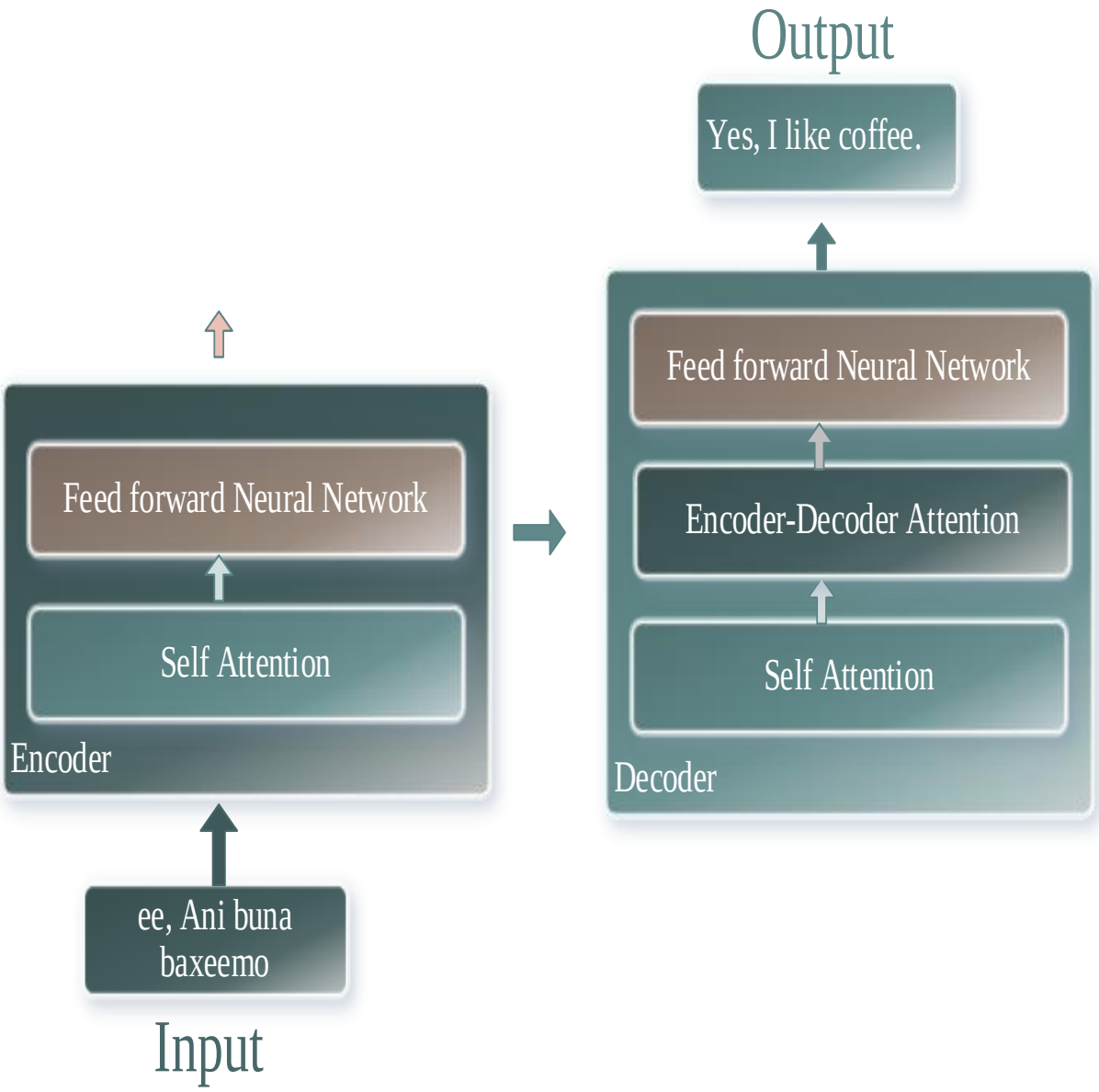


Figure 4. 12 self-attention and feed-forward neural network layer diagram

4.5.3 The general architecture of a transformer

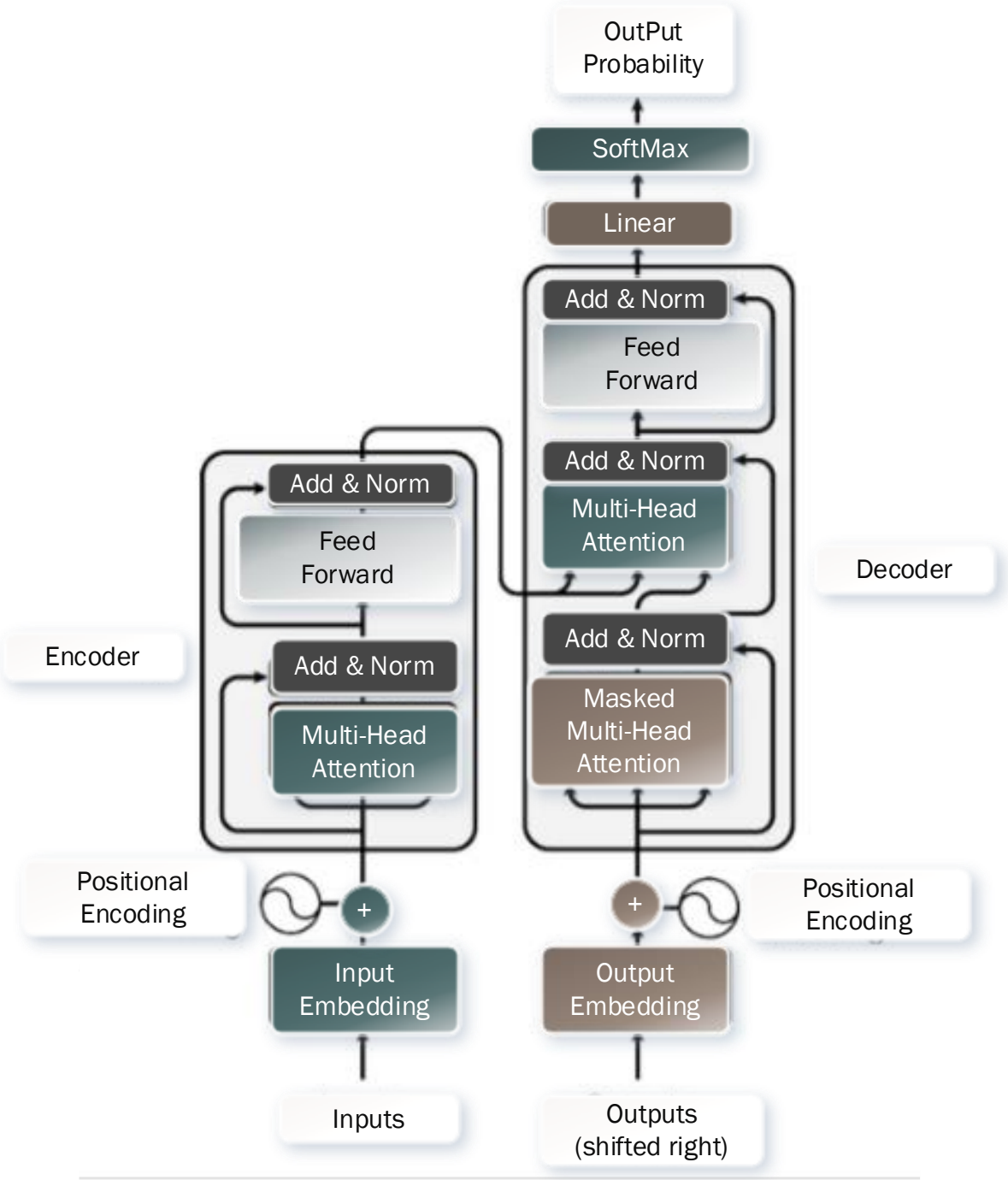


Figure 4. 13 general architecture of the Transformer model

Let us train a model using a transformer through many steps which are Preprocessing, vectorization, embedding, positional encoding, encoder-decoder phase, training using transformer parameters then the last evaluation. This processing step is usually called **tokenization** and normalization it's the first of the steps before we feed the input into the model. These are the same as those we used in the LSTM model.

After Preprocessing such as tokenizing and normalization data and load data that normalized data eventually, we create Text vectorization because we can't Usually, the text is transformed into sequences of token IDs, which function as indices into an embedding. This vectorization technique converts the text content to numerical feature vectors. Bag of Words takes a document from a corpus and converts it into a numeric vector by mapping each document word to a feature vector for the machine learning model.

In this approach of text vectorization, we perform two operations.

- ✓ Tokenization
- ✓ Vectors Creation

Once tokenization is complete, we can extract all the unique words from the corpus, which represents the collection of tokens obtained from all the sentences and is used to create a bag of words.

4.5.4 Create vectors for each Sentence

Here the vector size for a particular sentence is equal to the number of unique words present in the corpus. For each sentence, we will fill each entry of a vector with the corresponding word frequency in a particular sentence. The vector size is represented by the M×N matrix that which M is a row of a matrix which is the number of total sentences and N is the column which is the total number of unique tokens.

Algorithm 5: Algorithm of creates text Vector

***Input**: Corpus*

	Output:	*Text Vector*
1	***Start***	*Load Corpus*
2		*Tokenizing a corpus*
3		*Extracting a unique token*
4		*Sorting tokens by alphabetic*
5		*Give integer value to token //indexing*
6		*Calculate the total number of tokens //N number*
7		*Calculate the total number of sentences// M number*
8		*Create vector by M×N*
	End	

A Snippet code of the text Vectorization example we have used shows our experiment on Sidaamu Afoo language text

```
from sklearn.feature_extraction.text import CountVectorizer
corups=["ayyarete albillite hiittoti hakko","hiitee ayaanna ayiriinsaani hakko",
"maaraati koonne ayaana ayiriinsaanihu","hakkunni gedensaanni kaaliiqi musera",
"hakkunni gedensaanni kaaliiqi iyyaasura"]
countvector=CountVectorizer()
x=countvector.fit_transform(corups)
vector=x.toarray()
print(vector)
```

```
[[1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0]
 [0 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0]
 [0 1 0 0 1 0 0 0 0 0 0 0 0 1 1 0]
 [0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1]
 [0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0]]
```

Thirdly word embedding is like the same method s Lstm but using techniques is different so many times 512-word dimensionality using in the transformer The Embedded layer in Keras is suitable for neural networks working with text data. To use this layer, the input data must be encoded as integers, with each word assigned a unique integer. The Tokenizer API, which is also provided with Keras, can perform this data preparation step. During initialization, the Embedding layer is given random weights and will learn an embedding for each word in the training dataset. This layer is typically defined as the first hidden layer of a network.

It must specify 3 arguments:

- **input dim**: This is the size of the vocabulary in the text data. For example, if our data is integer encoded to values between 0-35922, then the size of the vocabulary would be 35923 words.
- **output dim**: This is the size of the vector space in which words will be embedded. It defines the size of the output vectors from this layer for each word. For example, it could be 512 or 1024, or even larger. Test different values for our problem.
- **input length**: This is the length of input sequences, as we would define for any input layer of Kera's model. For example, if all of our input documents are comprised of 47 words, this would be 47.

For example, below we define an Embedding layer with a vocabulary of 35923 (e.g. integer encoded-words from 0 to 35922, inclusive), a vector space of 512 dimensions in which words will be embedded, and input documents that have 47 words each.

```
import numpy as np
model = tf.keras.Sequential()
model.add(tf.keras.layers.Embedding(35923, 512, input_length=47))
# The model will take as input an integer matrix of size (batch,input_length),
# should be no larger than 35923 (vocabulary size).
# Now model.output_shape is (None, 47, 512), where `None` is the batch dimension.
input_array = np.random.randint(35923, size=(512, 47))
model.compile('rmsprop', 'mse')
output_array = model.predict(input_array)
print(output_array.shape)
print(model.summary())
```

```
(512, 47, 512)
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (None, 47, 512)           18392576

=================================================================
Total params: 18,392,576
Trainable params: 18,392,576
Non-trainable params: 0
_________________________________________________________________
None
```

Then afterword embedding is many procedures in transformers so we would be applying positionally encoding if we know about the concept of those in the above portion

---

Algorithm 6: The algorithm creates a positional encoding value

---

	Input:	*Embedding vector*
	Output:	*Positional encoded value*
1	***Start***	*Take word embedding vector*
2		*Take the index of the word*
3		*Take the length of the sequence*
4		*Take a length of vocabulary*

5	*Calculate a two times value of zero/one then divided by dimensionality*
6	*Calculate a vocabulary power of **step 5** value*
7	*Calculate the position value of the word by **step 6** value*
8	*Calculate a cosine/sine value of **step 7** value*
9	*Give **step 8** value to word mapping as positional encoding*

End

A model contains no recurrence and no convolution, for the model to make use of the order of the sequence, we have injected some information about the relative or absolute position of the tokens in the sequence. To enhance the network's ability to capture the sequence information, we incorporate "positional encodings" into the input embeddings at the bottom of both the encoder and decoder stacks. The positional encodings have the same model dimension as the embeddings, enabling the two to be combined by addition. The attention mechanism paper typically employs sine and cosine functions of varying frequencies for the positional encodings. Therefore, in our work, we adopt the formula from the attention mechanism paper and utilize sine and cosine functions of various frequencies.

$P_{E\ (pos,\ 2i)} = sin\left(\frac{pos}{1000^{2i/d_{model}}}\right)$----------4.10 [35]

$P_{E\ (pos,\ 2i+1)} = cos\left(\frac{pos}{1000^{2i/d_{model}}}\right)$----------4.11 [35]

where:

pos - the position

i - the dimension

d_{model} - input and output dimensionality

where *pos* is the position and *i* is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset *k*, *PEpos+k* can be represented as a linear function of *PEpos*. We conducted an experiment where we used learned positional embeddings [9], and discovered that the two variations produced results that were almost identical(see figure 4.14 row (E)). We chose the sinusoidal version because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

To understand positional encoding, consider an input sequence of length L and the task of finding the position of the k^{th} object in this sequence. This positional encoding can be calculated using a combination of sine and cosine functions with varying frequencies.

$P_{E\ (pos,\ 2i)} = sin\left(\frac{pos}{n^{2i/d_{model}}}\right)$----------4.12

$P_{E\ (pos,\ 2i+1)} = cos\left(\frac{pos}{n^{2i/d_{model}}}\right)$----------4.13

where:

The position of an object in the input sequence is denoted as "Pos", with a value of $0 \leq k < L/2$. The dimension of the output embedding space is represented as "d". The positional function "P(k, j)" maps a position "k" in the input sequence to the index (k, j) in the positional matrix." The position of an object in the input sequence is denoted by 'Pos' with a value of $0 \leq k < L/2$. The dimension of the output embedding space is represented by 'd'. The 'Position function (P(k, j))' maps the position 'k' in the input sequence to the index (k, j) of the positional matrix. A user-defined scalar 'n' is set to 10,000 by the authors of "Attention is All You Need." 'i' is used to map to the column indices $0 \leq i < d/2$, with a single value of i mapping to both sine and cosine functions.

The expression shows that even positions correspond to the sine function and odd positions correspond to the cosine function."

Example

To understand the above expression, let's take an example of the phrase 'ee ani buna baxeemo', with n=100 and d=4. The following table shows the positional encoding matrix for this phrase. The positional encoding matrix would be the same for any 4-letter phrase with n=100 and d=4.

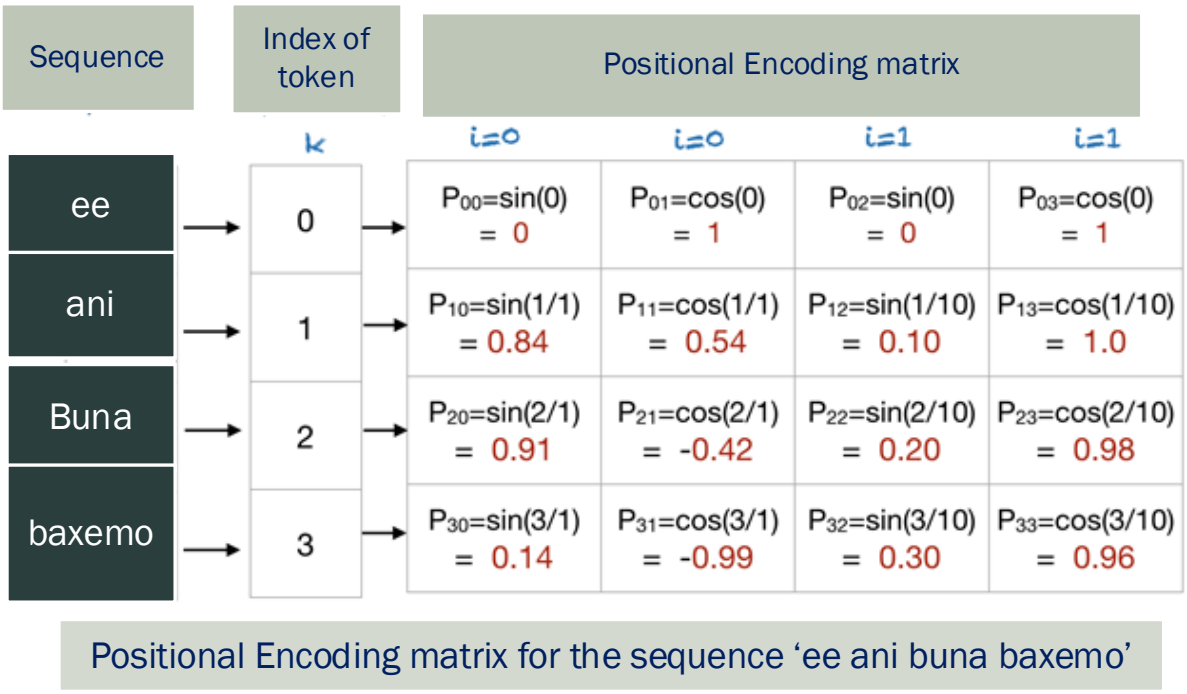


Figure 4. 14 positional encoding matrix diagram

A simplified Python code implementation of positional encoding using NumPy is presented below for ease of understanding.

```python
#positional encoding matrix
import numpy as np
import matplotlib.pyplot as plt

def getPositionEncoding(seq_len, d, n=10000):
    P = np.zeros((seq_len, d))
    for k in range(seq_len):
        for i in np.arange(int(d/2)):
            denominator = np.power(n, 2*i/d)
            P[k, 2*i] = np.sin(k/denominator)
            P[k, 2*i+1] = np.cos(k/denominator)
    return P

P = getPositionEncoding(seq_len=4, d=512, n=100)
print(P)
```

```
[[ 0.          1.          0.         ...  1.          0.
   1.        ]
 [ 0.84147098  0.54030231  0.8317052  ...  0.99994627  0.01018134
   0.99994817]
 [ 0.90929743 -0.41614684  0.92355454 ...  0.99978509  0.02036163
   0.99979268]
 [ 0.14112001 -0.9899925   0.19384206 ...  0.99951647  0.0305398
   0.99953355]]
```

we have finished all prep processing, text vectorization, word embedding, and positional encoding, fourthly pass to the next encoder decoder states reminding us previous all steps applying to training data both source and target corpus. So, we have applied the source vector through the encoder part and the target vector in the decoder part.

The encoder stack is taking source vectors firstly applying self-attention and secondly feed-forward networks then after those data pass to decoder stacks as input. And Decoder stack takes the target vector and applies it through masked self-attention, self-attention from encoder representation, and feed neural network. All about those we have discased in bellow formula, algorithms, and snippet code

4.5.5 Encoder

The Attention is All You Need model's encoder comprises N=6 identical layers that are stacked on top of each other. Each layer contains two sub-layers, the first being a multi-head self-attention mechanism, and the second being a simple, position-wise fully connected feed-forward network. To ensure that these sub-layers are effective, a residual connection is established around each of them, followed by layer normalization. This approach implies that the output of each sub-layer is LayerNorm(x + Sublayer(x)), with Sublayer(x) representing the function implemented by the sub-layer itself. To support these residual connections, all sub-layers in the model and the embedding layers generate outputs with a dmodel dimension of 512.

4.5.6 Decoder

According to "Attention is All You Need," the decoder comprises N=6 identical layers, just like the encoder. In addition to the two sub-layers present in each encoder layer, the decoder includes a third sub-layer that conducts multi-head attention over the encoder stack's output. Like the encoder, residual connections and layer normalization are employed around each of the sub-layers. To prevent positions from attending to future positions, the self-attention sub-layer in the decoder stack is modified. This masking, along with the offset of the output embeddings by one position, guarantees that the predictions for a particular position (i) can only depend on the known outputs of positions preceding it. As previously stated, the encoder and decoder stacks begin to process once positional encoding enters the residual connection.

4.5.7 Residual Connections

Similar to ResNet architecture, the Transformer adds to each output’s sub-layer the input that entered this sub-layer before processing (It is represented by “Add & Norm” in black in a general diagram of the transformer). The idea behind the residual connection is to make optimization easier. It preserves the information before each operation to enable faster learning in the back-propagation phase that compares the model’s output with the target we want to achieve.

Since we want to update the weights correctly along the way, the input before each sub-layer enables the back-propagation procedure to update the weights easily and efficiently. Over time, the

mathematical operations performed in earlier stages can cause the layers to blur, while the network quickly learns how to update the last layers nearer to the output. If an error occurs at some point in the process, attempting to address the complex output may not be the best approach, and adding the initial input can aid in weight learning. It is a kind of intermediate stage of input-output comparison after each action that prevents creating a monstrous output at the end of the decoding procedure. Without this operation, it will be hard for the attention mechanism to find out how to change each sub-layer in the learning step to get better results.

4.5.8 Fully Connected Feed-Forward Network in the Transformer

Both, the encoder and the decoder have two fully-connected layers followed by an activation function (ReLU). The goal of those layers is to enrich the learning process and to enable it to learn new dependencies on the sequence by itself.

4.5.9 Self-attention

The attention mechanism was introduced to improve the performance of the encoder-decoder model for machine translation. The attention mechanism was designed to enable the decoder to flexibly utilize the most pertinent portions of the input sequence. This is accomplished by a weighted combination of all encoded input vectors, with the most relevant vectors being assigned the highest weights. The principle of self-attention is to check how each word in a sentence is related to the other words. To check the similarity between two words that are represented by vectors, the attention uses the dot product operation.

Algorithm 7: An algorithm of self-attention

	Input:	*Input vector of sentences*
	Output:	*Self-attention value*
1	***Start***	*Take the input vector for input word one*
2		*Initialize weights #for key, query, value*

3	*Calculate the key, query, and value #input multiplied by the weight*
4	*Calculate attention scores for Input one*
5	*Take the SoftMax across these attention scores*
6	*The SoftMax attention scores for each input are multiplied by its corresponding value*
7	*Sum weighted values #Take all **steps 6** & sum them element-wise*
8	*Take the **step 7** value as the result vector to input one*
9	*Repeat **steps 4–7** to Input two & more*

End

Now, we will go over the formula of the Scaled Dot-Product Attention that is computed according to the following equation.

$$Attention(Q, K, V) = softMax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad \text{------ 4.14}$$

where Q, K, and V are the concatenation of query, key, and value vectors.

- Algorithm Step 2: initialized weights

Every input must have three representations that are called **key, query,** and **value**. let's take that we want these representations to have a dimension of three. Let's take every input has a dimension of four, each set of weights must have a shape of 4×3.To obtain these representations, every input is multiplied with a set of weights for **keys**, a set of weights for, and a set of weights for **values**. In a neural network setting, these weights are usually small numbers, initialized randomly using an appropriate random distribution like Gaussian distributions. This initialization is done once before training.

- Algorithm Step 3: Derive key, query, and value

Now that we have the three sets of weights, let's obtain the **key**, **query**, and **value** representations for every input. In practice, a *bias vector* may be added to the product of matrix multiplication.

- Algorithm **Step 4: Calculate attention scores for Input one**

The fourth step is calculating the scores. The score will be determined for each word of the input sentence against the initial word for which we are calculating the self-attention from the corpus. To calculate the score, we use will dot product between the query vector word one and the key vector of the word that we score. For instance, for scoring the word two while calculating the self-attention for word one we will dot product query one with key one. To obtain *attention scores*, we start by taking a dot product between Input and one **query** with all **keys** including itself.

- Algorithm Step 5: Calculate SoftMax

Take the SoftMax across these attention scores of input one. Previously as described in a formula, we will divide the score by the square root of the dimensions of the key vector ($\mathbf{d_k}$). accordingly, "attention all need "paper the dimension of the key vector is 64, so that will be 8. The reason behind that is if the dot products become large, this causes some self-attention scores to be very small after we apply the SoftMax function. In the equation of the self-attention above, dk represents the dimension of the queries and keys (64 dimensions) and is used to normalize the scores. Then, the divided scores are transferred to the SoftMax function. This function calculates the probabilities and forces the scores to be positive and summed up to one.

- Algorithm Step 6: Multiply scores with values

The soft maxed attention scores for each input one are multiplied by their corresponding value. This results in three *alignment vectors*.

- Algorithm Step 7: Sum weighted values to get Output 1

Take all the weighted values and sum them element-wise the resulting vector is Output one, which is based on the query representation from Input one interacting with all other keys, including itself. Generally, the dimension of the **query** and **key** must always be the same because of the dot product score function. However, the dimension of **value** may be different from the **query** and **key**. The resulting output will consequently follow the dimension of **value**.

4.5.10 Multi-headed attention

The attention paper proposed another type of attention mechanism called multi-headed attention. Below is the step-by-step process to calculate multi-headed self-attention:

Step 1, Take each word of the input sentence and generate the embedding from it.

Step 2, In this mechanism, we created *h* (h = 8) different attention heads, each head has different weight matrices (W(Q), W(K), W(V))

Step 3, In this step, we multiply the input matrix with each of the weight matrices (W^Q, W^K, W^V) to produce the key, value, and query matrices for each attention head.

Step 4, Now, we apply the attention mechanism to these query, key, and value matrices, this gives us an output matrix from each attention head.

Step 5, In this step, we concatenate the output matrix obtained from each attention head and dot product with the weight W_O to generate the output of the multi-headed attention layer.

Mathematically multi-head attention can be represented by:

$$MultiHead(Q,K,V) = concat(head_1, head_2 \dots head_n)W_0$$

$$where, head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Lastly, we already we have finished the encoder-decoder phase so, we would be continuing to train using hyperparameters, optimizers, and many more techniques. so, we did train results and discussion in chapter five.

CHAPTER FIVE

EXPERIMENTAL RESULTS AND DISCUSSION

5.1. Introduction

Based on the design of the model, English – Sidaamu Afoo bidirectional machine translation is experimented with using a deep learning method special transformer-based model. This Chapter evaluates its performance by conducting mainly depends on two experimental models using an LSTM and a transformer model.

5.2. Corpus Preparation

This deep learning approach requires a bilingual parallel corpus. For this study, we collected parallel documents in English and Sidaamu Afoo from various sources such as Sidaamu Afoo to English Conversations, the Holy Bible, and regional constitution documents. The parallel corpus consists of texts translated into both English and Sidaamu Afoo, aligned at the sentence level. After tokenizing, true casing, and cleaning the collected English-Sidaamu Afoo parallel corpus, we obtained exactly 15,000 parallel sentences. These parallel corpora will be used in different models, split into training and test sets."

5.3. Experimental setting

In this study, an investigation depends on two main deep learning models which are LSTM and Transformer., it investigated using three data splitting techniques that are taken a different number of tests and training data those Experiment one is 30% test by 70% training, experiments two 20% test by 80% training, and experiment three 10% test by 90% training. In addition, using different deepening higher parameters that are taking different dropout rate values which are 0.0, 0.2, 0.4, and 0.5 a changing dropout value has experimented with four times in each model (LSTM and Transformer model) which total of eight experiments. It also uses batch three batch sizes 32,64, and 128 for each model experimented then a total of six experiments, and for use embedding dimension three values 256, 512, and 1024 for both models, a total is six experiments. And lastly,

it proposed was bidirectional machine translation so each model investigated by two experiments Sidama Afoo to English and vice versa, and also the transformer model the same on tested in both directions, a total is four experiments. Overall, twenty-nine experiments were evaluated and compared in this study and finally selected from all the best-performed experiments to both models which are LSTM two experiments that Sidaamu Afoo to English and. vice versa and also a transformer.

5.4. Experiment parameters

Parameters are physical properties whose values determine the characteristics or behavior of something [76]. Therefore, in our experiment, we utilized many high parameters, which are listed and defined as follows.

Number of Epoch: - it is the number of times a learning algorithm sees the complete dataset. Sometimes, just increasing the number of epochs for model training delivers better results, although this comes at the cost of increased computation and training time [72]. So, when we used a different number of epochs depending on the model that to get a good result.

- Impacts of epoch on deep learning model

In deep learning, epoch refers to the number of times that the entire training dataset is passed through the neural network during the training process. The number of epochs is an important hyperparameter that can impact the model's performance and generalization ability. Increasing the number of epochs can sometimes lead to better performance on the training data, as the model has more opportunities to learn from the data. However, increasing the number of epochs can also lead to overfitting, where the model becomes too specialized to the training data and does not generalize well to new data. On the other hand, decreasing the number of epochs can result in underfitting, where the model does not learn the underlying patterns in the data well enough to make accurate predictions. Therefore, it's important to find the right balance between the number of epochs and the model's performance on the training and validation data. This can be done through techniques such as early stopping, where the training is stopped before overfitting occurs, or by monitoring the validation loss to determine the optimal number of epochs for the model.

Embedding dimension: - Choosing the optimal embedding dimension is largely dependent on factors such as available computing resources. Word embeddings are a method of representing individual words of a language or domain as real-valued vectors in a lower-dimensional space. 512-dimensional embedding used for training. We can have chosen a different dimension, but it didn't give better results.

- Impacts of Embedding dimension on deep learning model

In deep learning, embedding dimensions refer to the size of the vector space in which the input data is represented. The size of the embedding dimensions is an important hyperparameter that can impact the performance of the deep learning model. Increasing the size of embedding dimensions can result in a model that is more expressive and can capture more complex relationships between the input and output data. However, increasing the size of the embedding dimensions can also lead to overfitting, where the model becomes too specialized to the training data and does not generalize well to new data. On the other hand, decreasing the size of embedding dimensions can result in underfitting, where the model does not learn the underlying patterns in the data well enough to make accurate predictions. Therefore, it's important to find the right balance between the size of embedding dimensions and the model's performance on the training and validation data. This can be done through techniques such as grid search or randomized search, where different combinations of hyperparameters are tested to find the optimal size of embedding dimensions for the model.

Batch size: - A moderate batch size can improve the learning process of the model by reducing the variance of the parameter updates and enabling more efficient use of hardware resources. The batch size refers to the number of training examples used in a single forward and backward pass. A batch size of 32 or 64, using more time of irrespective dataset size and the number of samples, will deliver a smooth learning curve in most cases Even in scenarios where your hardware environment has large RAM to accommodate a bigger batch size, it recommends staying with a batch size of 32 or 64[72]. Accordingly, with this information, we have compromised batch size with our model that most of our training is used 64 batch sizes.

- Impacts of Batch size on deep learning model

In summary, the batch size hyperparameter determines the number of samples that are processed in each training iteration, and it can impact the computational resources required, the generalization performance, the noise reduction, and the learning dynamics of the deep learning model. A larger batch size may result in faster convergence and a more stable gradient estimate, but it may require more memory and computational resources. A smaller batch size may allow the model to generalize better to new data and learn more fine-grained details, but it may be slower and require less memory and computational resources. The optimal batch size depends on the specific problem and the available computational resources, and it requires careful experimentation and analysis.

Learning rate: - It is a hyper-parameter that regulates how our neural network's weights are applied concerning the loss gradient. It specifies how rapidly a neural network updates previously learned knowledge. A desired learning rate is both high enough to allow for quick training and low enough to ensure that the network converges to something useful [77].

Dropout rate: - Dropout is a regularization technique that is commonly used in neural network models to prevent overfitting. It works by randomly "dropping out" (or setting to zero) a certain number of output units in a layer during training. The dropout rate is the fraction of units that are dropped out and is typically set between 0.2 and 0.5. In the case of the Transformer model, dropout can be applied to the input embeddings, the self-attention layers, or the feed-forward layers. Applying dropout to the self-attention layers can help prevent the model from over-relying on any one part of the input while applying dropout to the feed-forward layers can help prevent overfitting by reducing the capacity of the model.

The advantages of using dropout in Transformer models include:

- ✓ Improved generalization: by randomly dropping out units during training, dropout forces the model to learn multiple independent representations of the input, which can help it generalize better to unseen data.
- ✓ Reduced overfitting: by reducing the capacity of the model, dropout can help prevent overfitting by preventing the model from adapting too closely to the training data.

- ✓ Better performance: by reducing overfitting and improving generalization, dropout can result in improved performance on test and validation sets.

It's important to note that dropout is not a must-have technique, it's something that we could use depending on our data and the model.

- Impacts of Dropout rate on deep learning model

In deep learning, dropout rate refers to the proportion of nodes in a neural network that are randomly "dropped out" or ignored during each training iteration. The dropout rate is an important hyperparameter that can impact the performance of the deep learning model. Increasing the dropout rate can help prevent overfitting, as it forces the neural network to learn more robust and generalized features. However, increasing the dropout rate too much can also lead to underfitting, where the model does not learn enough important features from the data. On the other hand, decreasing the dropout rate can result in a model that is more specialized to the training data and may not generalize well to new data.

Units: - 1024 Units here means In an LSTM unit, will have a 1024-neurons or cells, and the number of units is set depending upon how complicated we expect the corpus to be which thinks lots of nouns, pronouns, and references that need to be resolved over a long sentence length. At each time step, when we input a word to the encoder a different implementation could be doing this exercise with characters instead of words, we would generate a 1024-dimensional vector representation of the input word.

5.5. Experiments

5.5.1. Experiment I

Experiment one is the English to Sidaamu Afoo translation and the Sidaamu Afoo to English translation was performed using an LSTM model.

5.5.1.1. Training the system

Kera's which is a freely available phyton Library is used to implement the model that both directions, English to Sidaamu Afoo and vice versa, by using similar and the same number of

English-Sidaamu Afoo parallel sentences. In the training, the process includes the following procedures and parameters. First, we used different data splitting techniques that are taken a different number of test and training data those 30% test by 70% training, 20% test by 80% training, and 10% test by 90% training. It has used those data split on the LSTM model and the last split of 10% test by 90% training giving a good result of the accuracy, loss rate, and bleu score. As a researcher, we have split in those considering the data size which means our data set is 15000 sentence parallel corpora but it has used the LSTM model with a limited memory or RAM size in LSTM model which is 12 RAM, GPU graphics, and 78.19 GB Disk storage permitted in google collab but in using those computing power LSTM didn't run and crashing when we have used beyond total data. So, in such case, we were pressured to use 15000 of the total data, we prepared around 22000 parallel sentences. more result description shows below in Table 5.1. in this experiment the checked different number of splitting techniques to select good data split in our study, but we have used common higher parameters their dimensionality, epoch, batch, and learning rate there are 512, 30, 64, and 0.001 respectively. This means taking the splitting ratio we have using the common value of a higher parameter.

Table 5. 1 Data splitting ratio table

Data Splitting	Train data	Test data	Accuracy	Loss	Bleu score
30% by 70% split	10500	4500	43.72%	2.95	0.097
20% by 80% split	12000	3000	47.22%	2.41	0.136
10% by 90% split	13500	15000	51.51%	1.96	0.186

More representation of data splitting by graphical chart.

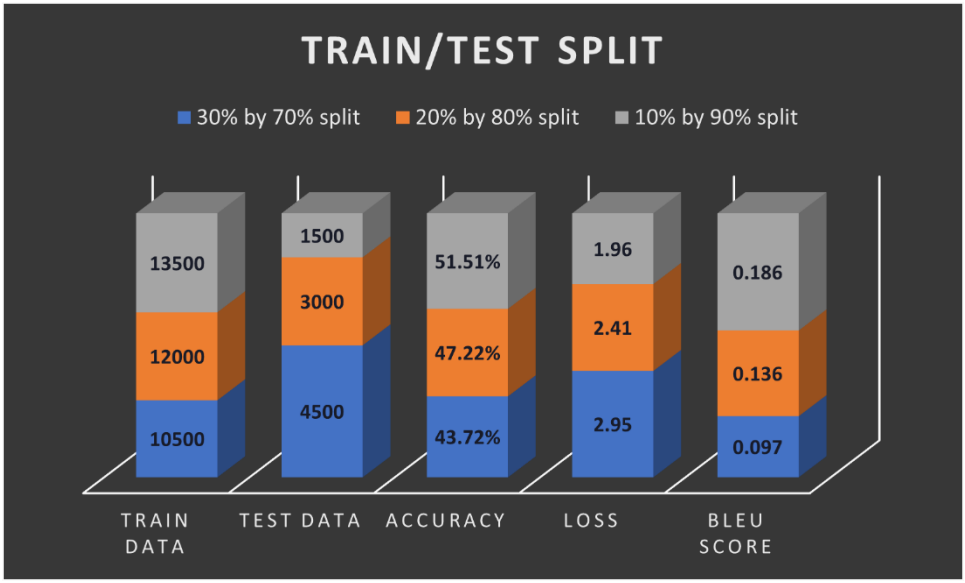


Figure 5. 1 Data splitting graphical representation

5.5.1.2.Parameter Selection for Lstm model

In our experimental investigation, we used our parallel corpus from Appendix I to train and test the model. Using our training data, we run several trials on different parameters to get the desired outcome. In those LSTM model experiments using the initial value of different parameters to select the desired outcome. So firstly, we started to select a batch size other are using common or initial values their embedding layer, epoch, learning rate, and train plus test data size which 512, 10, 0.001, and 15000 respectively. We had done those experiments with batch size inverse portion with the good result which means batch size increased from 32 to 64 to 128 while loss values also increase which 4.721, 5.06, and 5.356 respectively it summarizing the result of accuracy is low. Because our experiment gave a direct proportion with batch size and training loss. We have More described in Table 5.2 and figure 5.2

Batch	Training loss	Accuracy (%)	Time(sec)
32	4.71	37.2	146
64	5.06	36.0	134

128	5.37	36.7	128

Table 5. 2 batch selection table

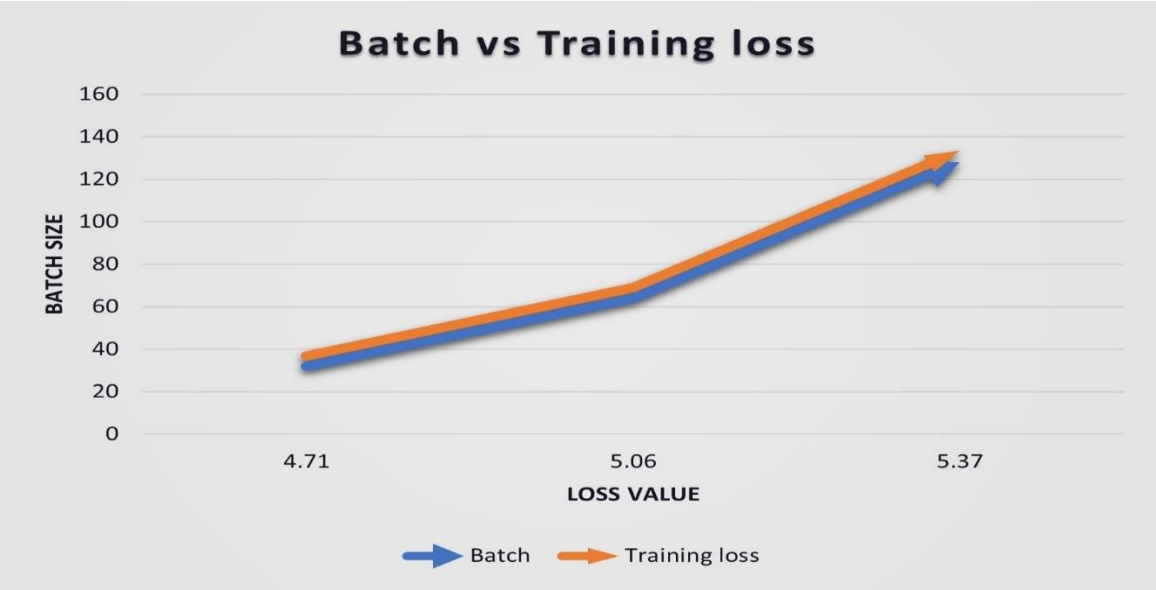


Figure 5. 2 batch selection graph

We chose the good result embedding dimension next of batch selection. The embedding dimension is 512 value LSTM model higher parameter selected. We have conducted tests utilizing both training loss and accuracy values to choose good results on depends on the training data. We used 64 batch sizes, a learning rate of 0.001, and Adam optimizers. The experiments were then run throughout 10 epochs. And we have checked different embedding dimensions that are 256, 512, and 1024 finally, the results listed are as follows. These results are barely different and nearly the same value.

Embedding dimension	Training loss	accuracy
256	5.05	38.07%
512	3.25	52.08%
1024	4.86	45.2%

Table 5. 3 loss and accuracy values for selecting the embedding dimension

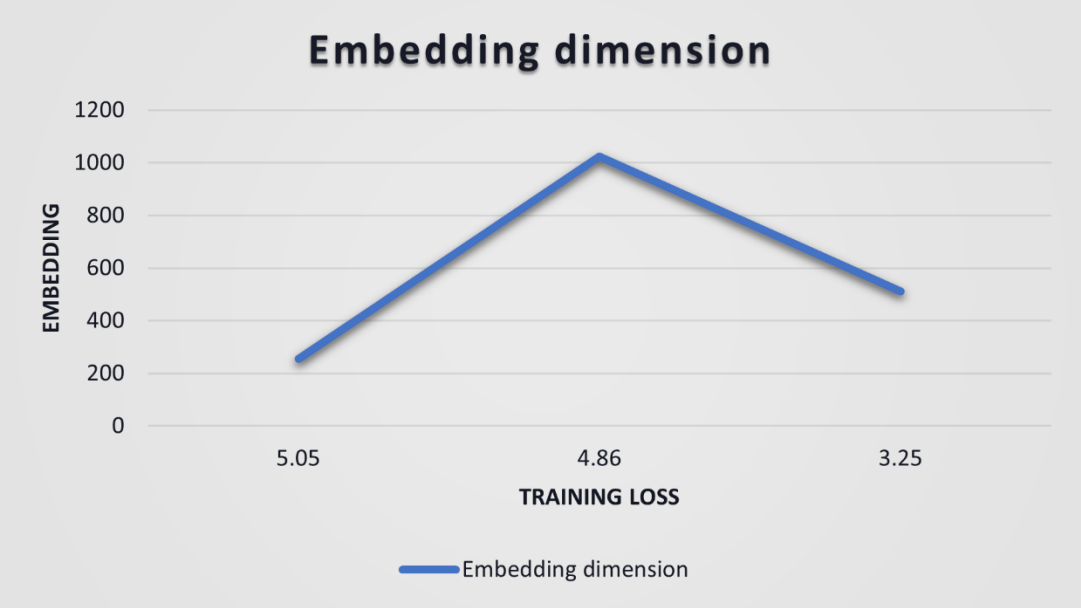


Figure 5. 3 embedding selection graph

We chose the good result epoch on thirdly. The epochs that we chose were 100 based on the result. We have conducted tests utilizing both training loss and accuracy values to choose good results on depends on the training data. We used 64 batch sizes, a learning rate of 0.001, and Adam optimizers. The experiments were then run for 100 epochs. And finally, the results listed are as follows. These results are barely different and nearly the same. The loss level is the same for all of the tests below, decreasing from a greater loss level to a lower loss level.

epoch	Training loss	accuracy
10	5.05	37.07%
20	4.25	38.08%
30	2.86	46.2%
40	1.90	65.2%
50	1.13	79.01%
60	0.75	86.3%
70	0.41	92.2%
80	0.25	96.4%
90	0.14	98.4%
100	0.068	99.2%

Table 5. 4 loss and accuracy value for selecting the Epoch

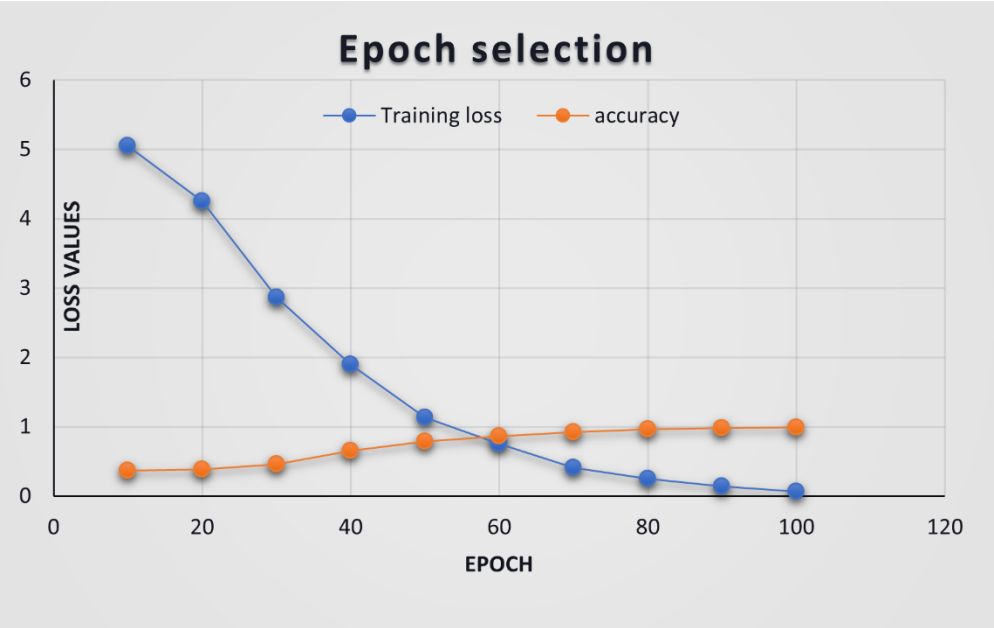


Figure 5. 4 Epoch vs training loss and accuracy value graphical representation.

The best results from the investigated experiments are those with 512 embedding dimensions and 100 epochs. Therefore, we have selected epoch 100 and embedding dimension 512. The next step is to select the dropout rate. Dropout is a regularization technique for reducing overfitting in neural networks, which randomly "drops out" (or sets to zero) a certain number of output units in a layer during training. The dropout rate is the fraction of units to drop out, typically set between 0.2 and 0.5. Based on our investigation, we compared the top trials with dropout rates of 0.0, 0.2, 0.4, and 0.5, learning rates of 0.001, and batch sizes of 64, and 100 epochs. Table 4.2 shows that we experienced a minor loss with a dropout rate of default values of 0.0 and 0.2.

Embedding dimension	**epoch**	**batch**	**Dropout rate**	**Training loss**
512	100	64	0.0(default)	0.86
512	100	64	0.2	0.89
512	100	64	0.4	0.90
512	100	64	0.5	0.92

Table 5. 5 Training Loss for selecting Dropout rate

After determining the learning space, we started with default values and gradually decreased them. We observed that the resulting losses were 0.92, 0.90, 0.89, and 0.86 for values of 0.5, 0.4, 0.2, and 0.0 respectively. Ultimately, we found that a learning rate of 0.0, or the default value produced the best results. As seen in Figure 5.5, we presented a graph of the loss level for each learning rate.

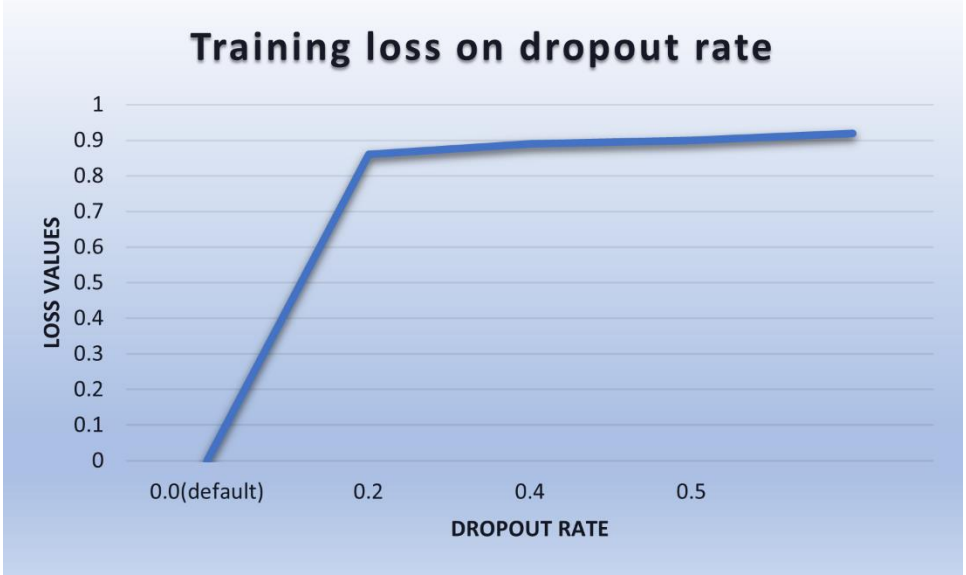


Figure 5. 5 dropout selection graph

5.5.1.3.Summary of an evaluated parameter of LSTM

Table 5. 6 Summary of LSTM parameter selection table

Parameter	**Checked value**	**Training loss**	**Selected one**
Batch	32	4.711	✓
	64	5.06	
	128	5.37	
Embedding dimension	256	5.05	
	512	3.25	✓

	1024	4.86	
Dropout rate	0.0	3.67	
	0.2	3.67	✓
	0.4	3.74	
	0.5	3.8	
Epochs	10	5.05	
	30	4.25	
	50	1.13	
	70	0.41	
	100	0.065	✓

5.5.1.4.Summary of best-Selected parameter LSTM

Table 5. 7 Selected parameter table for LSTM model

Parameter	Selected value	Training loss
Batch	32	4.711
Embedding dimension	512	3.25
Dropout rate	0.2	3.67
Epoch	100	0.065

Before moving on to other model experiments, we observed the effects of the LSTM model on our study. The BLEU score we obtained was 0.217, which was 21.7% when the best parameters were selected. We obtained this result from testing the English to Sidama translation, however, when we evaluated the reverse translation, the results were not as satisfactory. Specifically, we achieved a BLEU score of only 0.107 for the Sidama to English translation, indicating a lower level of performance compared to the English to Sidama translation. In our initial investigation, we tried the LSTM model on the English-Sidama language corpus and obtained this result, but it did not meet our expectations as researchers. Therefore, we moved on to testing other parameter selections using the Transformers model on the English-Sidama corpus. Our next experiments will focus on using the Transformer model.

5.5.2. Experiment II

The second set of experiments involved using a Transformer model to translate between English and Sidaamu Afoo. This was done by training the model on a dataset of English sentences and their corresponding Sidaamu Afoo translations and then using the trained model to generate translations in both directions. The goal of these experiments was to see how well the model could handle the task of translating between the two languages and to identify any potential challenges or limitations in the process. The results of these experiments will be used to improve the overall performance of the model and to develop new techniques for handling machine translation tasks more continuously.

5.5.2.1. Training the system for Transformer model

In this training session, we used the Keras library in Python to create a model that can translate between English and Sidaamu Afoo. The model employed similar parameters, computational resources, and techniques as the Long Short-Term Memory (LSTM) model. However, for the final training of the Transformer model, we used a different number of parallel corpora. To compare the performance of the LSTM model and the Transformer model, we used the same size of parallel corpus for both. Due to the high computational power required by the LSTM model for larger

parallel corpora, so we have limited the number of parallel data used and only employed around 15000 from the prepared corpus in our experiments with the LSTM model. After training the system using 15,000 parallel corpus data on the Transformer model, we observed good performance in terms of computational power, execution time, and BLEU scores metrics. The training process included the following procedures and parameters.

5.5.2.2. Parameter Selection on transformer model

In our experimental investigation, we used a parallel corpus from Appendix II to train and test the model. We ran multiple trials using different parameters on our training data to achieve the desired outcome. We investigated new parameter selections because the Transformer model utilizes different learning techniques and characteristics. As researchers, we selected the parameters again after conducting an experimental investigation to determine the best results. In this experiment, we used initial values for various parameters to select the desired outcome. Firstly, we started by selecting the batch size. The other parameters, such as the embedding layer, epoch, dropout rate, and train plus test data size, were set to common or initial values of 256, 10, 0.5, and 15000 respectively.

Batch	Embedding Dimension	Accuracy (%)	Training loss
32	256	41.8	1.38
64	256	52.2	1.04
128	256	57.5	0.84

Table 5. 8 selection of batch size

The table above shows the results of an experiment that varied the batch size and embedding dimension of a model, and the effect on the accuracy and training loss. The results of an experiment were analyzed to determine the effect of changing the batch size and embedding dimension on a model's accuracy and training loss. Three different combinations of batch size and embedding

dimension were tested: 32 and 256, 64 and 256, and 128 and 256. The first combination, with a batch size of 32 and an embedding dimension of 256, resulted in an accuracy of 41.8% and a training loss of 1.38. The second combination, with a batch size of 64 and an embedding dimension of 256, resulted in an accuracy of 52.2% and a training loss of 1.04. The third combination, with a batch size of 128 and an embedding dimension of 256, resulted in an accuracy of 57.5% and a training loss of 0.84. Overall, as the batch size increased and the embedding dimension remained constant, the accuracy of the model improved and the training loss decreased. More described below in figure 5.6 graphical representation of training loss decreased when the batch size increase

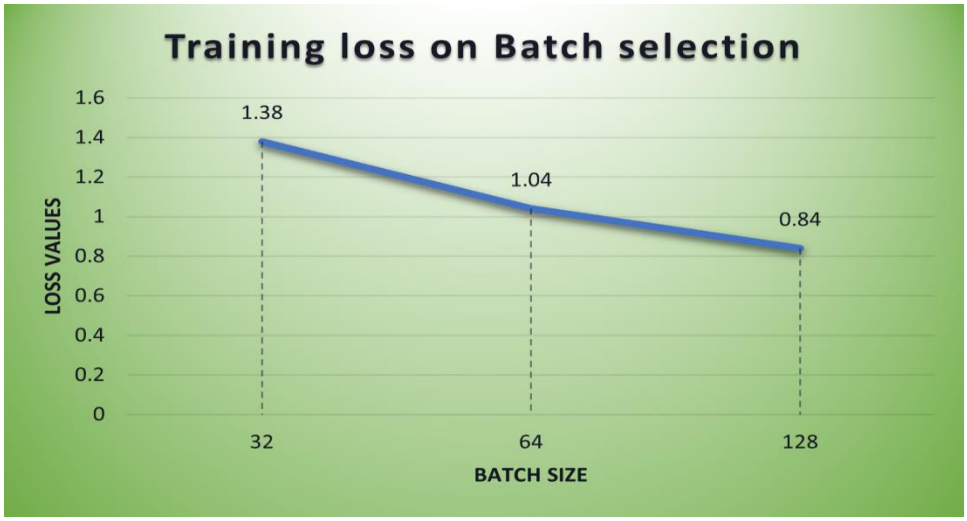


Figure 5. 6 batch size selection chart

Secondly, we first selected the batch size and then determined the optimal embedding dimension for our Transformer model. We experimented with embedding dimensions 256, 512, and 1024, aiming to identify the optimal parameter for our model. To evaluate the results, we used both accuracy and training loss as metrics for selecting the best configuration for our training data. The experiments were run with a batch size of 128, a dropout of 0.5, and for 10 epochs. The results of our experiments are summarized in Table 5.7.

Embedding dimension	**Batch size**	**Accuracy (%)**	**Training loss**
256	128	57.5	0.844
512	128	36.2	1.44

1024	128	31.6	1.64

Table 5. 9 Embedding dimension selection

The results of a study were analyzed to determine the effect of changing the embedding dimension and batch size on a model's accuracy and training loss. Three different combinations of embedding dimension and batch size were tested: 256 and 128, 512 and 128, and 1024 and 128. The first combination, with an embedding dimension of 256 and a batch size of 128, resulted in an accuracy of 57.5% and a training loss of 0.844. The second combination, with an embedding dimension of 512 and a batch size of 128, resulted in an accuracy of 36.2% and a training loss of 1.44. The third combination, with an embedding dimension of 1024 and a batch size of 128, resulted in an accuracy of 31.6% and a training loss of 1.64. Overall, as the embedding dimension increased and the batch size remained constant, the accuracy of the model decreased and the training loss value increased so the performance of decrease, more described in the graphical representation in figure 5.7 below.

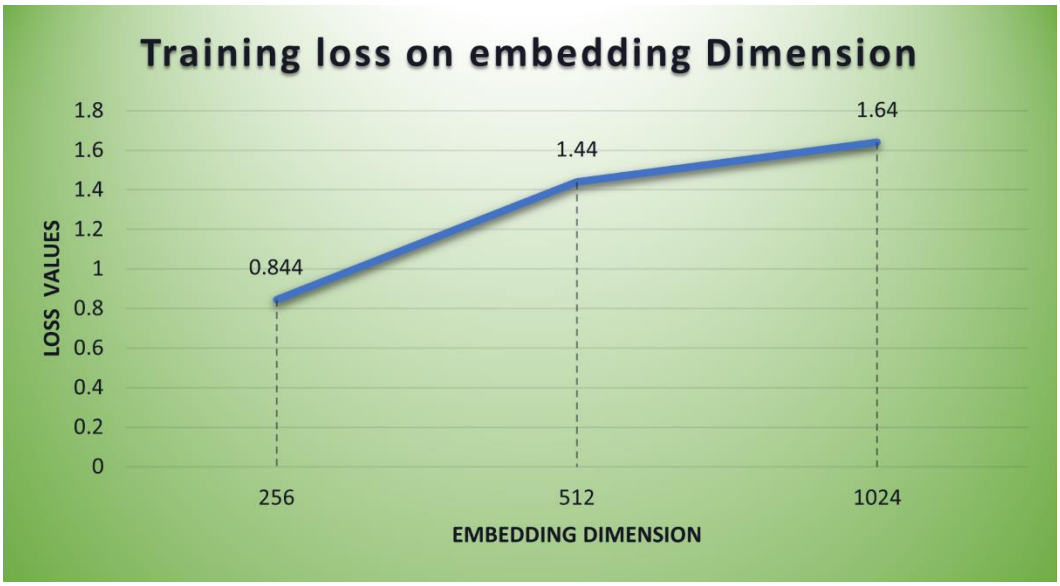


Figure 5. 7. Embedding dimension selection

After conducting two experiments, we found that the best results were obtained with an embedding dimension of 256, and a batch size of 128. The next step in our process was to select the appropriate dropout rate. In the transformer model investigation, we compared the results of several trials with

different dropout rates, including 0.0, 0.2, 0.4, and 0.5. The experiments were run with an embedding dimension of 256, a batch size of 128, and 10 epochs. As shown in Table 5.8, we found that a dropout rate of 0.0 resulted in the least amount of loss."

Embedding dimension	epoch	batch	Dropout rate	Training loss
256	10	128	0.0(default)	0.41
256	10	128	0.2	0.57
256	10	128	0.4	0.70
256	10	128	0.5	0.84

Table 5. 10 Selecting Dropout rate on a transformer

After experimenting with different dropout rates, we began with the default value and gradually decreased it. The resulting losses for dropout rates of 0.5, 0.4, 0.2, and 0.0 were 0.84, 0.70, 0.57, and 0.41 respectively. Our analysis revealed that the best results were achieved with a dropout rate of 0.0, or the default value. As shown in Table 5.8, we have provided a graph illustrating below figure 5.8, the loss levels for each dropout rate.

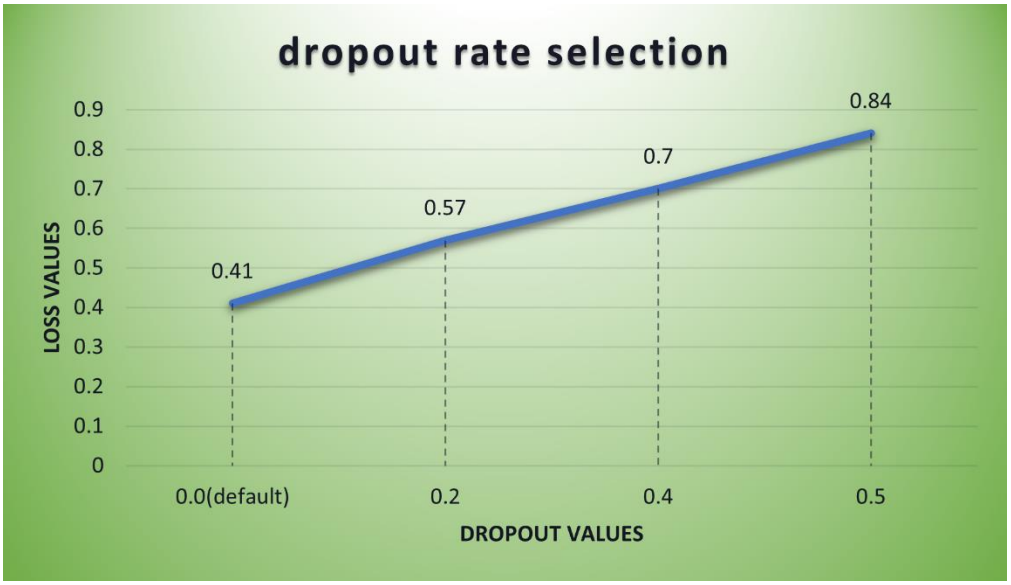


Figure 5. 8 transformer dropout rate selection

A dropout value of 0.0 means that no units are dropped out during training, which means that dropout is not applied to the model. In general, it's not recommended to use a dropout rate of 0.0 during training as it can lead to overfitting. Overfitting occurs when a model is too complex and adapts too closely to the training data, making it perform poorly on unseen data. Dropout is a regularization technique that is used to prevent overfitting by randomly dropping out units during training. By reducing the capacity of the model and forcing it to learn multiple independent representations of the input, dropout helps the model generalize better to unseen data.

In cases where a model is underfitting, which is the opposite of overfitting, the dropout rate could be set to 0.0 or close to 0.0 to increase the capacity of the model and let it adapt more closely to the training data. In summary, a dropout rate of 0.0 is not recommended during training as it can lead to overfitting and poor generalization performance on unseen data. It's important to find the optimal dropout rate for our specific model and dataset through experimentation, eventually, we have selected a 0.2 dropout for our training model.

Lastly, we experimented to determine the optimal number of epochs by evaluating both training loss and accuracy. The parameters used were 128 batch sizes, a dropout rate of 0.2, and 256 embedding dimensions. The experiments were run for a total of 50 epochs, and the results are presented in Table 5.9. The results show a continuous improvement with an increasing number of epochs.

epoch	Training loss	Accuracy (%)
10	0.49	69.1
20	0.204	93.4
30	0.045	97.3
40	0.038	98.01
50	0.037	98.02

Table 5. 11 Transformer epoch selection

The above table of training performance metrics for a transformer model is the last higher parameter. As the training progresses, we can see that the loss function is decreasing and the accuracy is increasing. This suggests that the model is learning and improving over time. In

particular, we can see that the accuracy of the model increases from 69.1% to 98.02% for 50 training epochs.

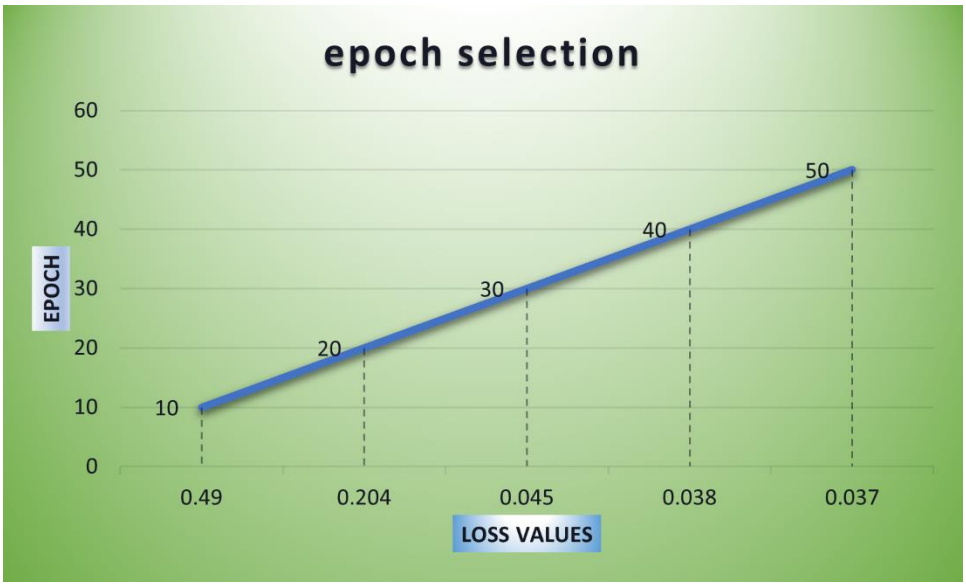


Figure 5. 9 transformer epoch decide chart

5.5.2.3.Summary of an evaluated parameter of Transformer model

Table 5. 12 Summary of Transformer model parameter selection table

Parameter	**Checked value**	**Training loss**	**Selected one**
Batch	32	1.38	
	64	1.04	
	128	0.84	✓
Embedding dimension	256	0.844	✓
	512	1.44	

	1024	1.64	
Dropout rate	0.0	0.41	
	0.2	0.57	✓
	0.4	0.70	
	0.5	0.84	
Epochs	10	0.49	
	20	0.204	
	30	0.045	
	40	0.038	
	50	0.037	✓

5.5.2.4.Summary of best-Selected parameter for Transformer model

Table 5. 13 Selected parameter table for transformer model

Parameter	Selected value	Training loss
Batch	128	0.84
Embedding dimension	256	0.844
Dropout rate	0.2	0.57
Epoch	50	0.037

5.6.Experimental result of testing

5.6.1. Result of Test Set on Experiment I

In our initial studies, we employed LSTM as the primary method for machine translation. We used performance metrics such as testing time and BLEU score to evaluate the model's effectiveness over 100 epochs. The results of these metrics showed that it took 13.4 minutes for the model to test for sentence-based translation from Sidama Afoo to English and 11.7 minutes for English to Sidama Afoo. To measure and test the performance of the system, we used a dataset of 15000 parallel English-Sidaamu Afoo sentences. The primary focus of our evaluation was the translation accuracy of the system when translating between these two languages. We employed the BLEU score methodology to analyze the results of the translation process. The BLEU score measures the similarity between the predicted text and the reference text. The results recorded from this methodology showed 21.7% for English to Sidaamu Afoo translation and 10.7% for Sidaamu Afoo to English translation, indicating that there is room for improvement in the system's performance.

Table 5. 14 Summary test result in experiment I

Translation direction	**Training data**	**Test data**	**BLEU score**	**Testing time (minute)**
English language to sidamu Afoo	13500	1500	0.217	11.7
Sidamu afoo to the English language	13500	1500	0.107	13.4

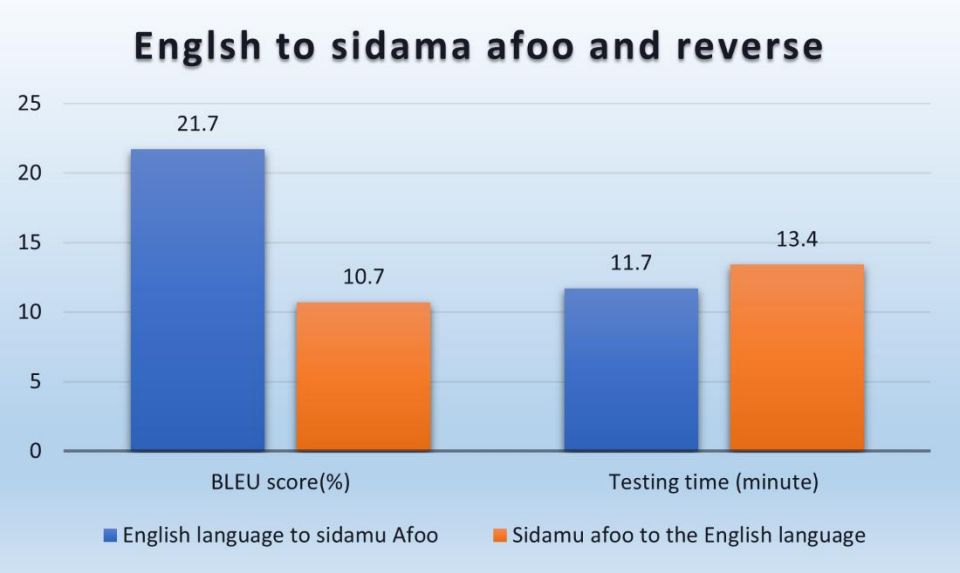


Figure 5. 10 LSTM test result graph

5.6.2. Result of Test Set on Experiment II

After conducting our experiment, we evaluated the performance of the Transformer model by measuring its ability to accurately translate text from one language to another. In particular, we tested the model's ability to translate from English to Sidama and Sidama to English. When we evaluated the English to Sidama translation, we used a metric called the BLEU score, which measures the similarity between the machine's translation and a human-generated reference translation. We found that the BLEU score for the English to Sidama translation was 0.465, or 46.5% when using the optimal parameters. This means that the model was able to generate translations that were 46.5% similar to a human-generated reference translation. However, when we reversed the translation direction and evaluated the Sidama to English translation, the results were not as favorable. Specifically, the BLEU score for this direction was only 0.413, indicating a lower level of performance compared to the English to Sidama translation. This suggests that the model may have struggled to accurately translate text from Sidama to English.

In a subsequent investigation, we applied the Transformer model to an English-Sidama language corpus and were pleased with the results, meeting our expectations as researchers. This suggests that the model may perform well when trained on a large dataset of text in both languages, rather than just one direction of translation. Overall, our study provides insight into the performance of

the Transformer model when used for machine translation, and highlights the importance to improve the performance of translation.

Below, you will find a table comparing the bidirectional machine translation model on Sidaamu Afoo with the English language. We have attempted to provide further explanation. The BLEU score column represents the BLEU score, a common evaluation metric for machine translation that compares the candidate translation to a set of reference translations. A higher BLEU score indicates a better match between the candidate translation and reference translations. The Testing time (minute) column represents the time in a minute it took to test the model.

Translation direction	Training data	Test data	BLEU score	Testing time (minute)
English language to sidamu Afoo	13500	1500	0.465	10.11
Sidamu afoo to the English language	13500	1500	0.413	8.52

Table 5. 15 transformer a test result table

The table above compares the performance of a bidirectional machine translation model between the English language and Sidaamu Afoo. The table includes four metrics for evaluation: BLEU score, and testing time. For the translation from English to Sidaamu Afoo, the model achieved a BLEU score of 46.5%. The total testing time for this direction was 10.11 minutes. For the reverse translation, from Sidaamu Afoo to English, the test showed the BLEU score was lower at 41.3%. The testing time for this direction was faster, taking only 8.52 minutes. We have provided a clear explanation in below figure 5.10

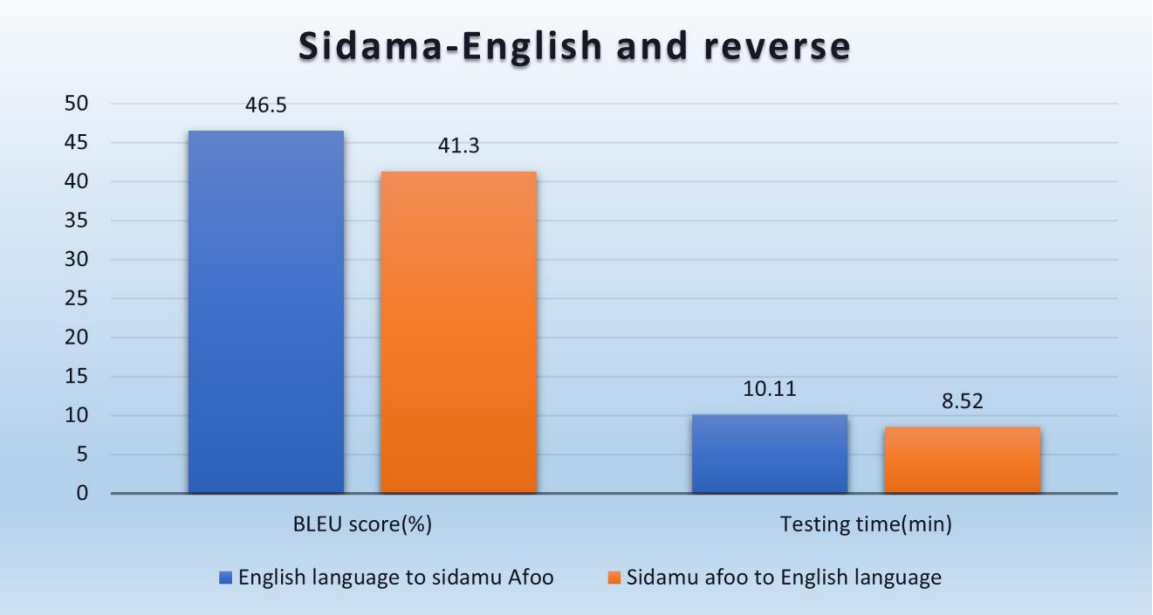


Figure 5. 11 bidirectional transformer result chart

5.7. Evaluation

Machine Translation Evaluation is the process of assessing the quality of a machine translation system's output. The goal of machine translation evaluation is to determine how well the system can accurately translate text from one language to another. The evaluation process helps to identify strengths and weaknesses in the machine translation system and provides a basis for improving its performance. Machine Translation Evaluation is crucial for the development of high-quality machine translation models and plays a key role in ensuring their effective deployment and use. Evaluating and comparing the performance of different machine translation models, such as LSTM and Transformer, on the task of translating from Sidama to English and vice versa can provide insight into their strengths and weaknesses. After conducting the evaluation, it reports the results to provide a comprehensive understanding of the performance of each model. These results can then be used to make informed decisions about which model to use for a specific translation task or to guide the development of future machine translation systems. In our experiment, we got a good result on the transformer model. Table 5.11 provides a more comprehensive explanation of the comparison between the LSTM model and the Transformer model for bidirectional machine translation between Sidaamu Afoo and the English language and vice versa.

	Sidamu Afoo to the English Language			English Language to Sidamu Afoo		
	Loss value	**Bleu Score**	**Time taken**	**Loss value**	**Bleu Score**	**Time taken**
LSTM model	0.312	0.107	18.4	0.0685	0.217	16.7
Transformer model	0.0445	0.413	9.52	0.0379	0.465	11.11

Table 5. 16 Lstm vs transformer model result

- ✓ Translation of Sidaamu Afoo to the English Language

A comparison was made between two models, an LSTM model, and a Transformer model, in terms of their ability to translate the language Sidaamu Afoo to English. The evaluation criteria were the loss value, Bleu Score, and time taken to complete the translation. The Transformer model performed better with a lower loss value (0.0445) and a higher Bleu Score (0.413), and also completed the task faster (9.52 minutes) than the LSTM model (loss value 0.312, Bleu Score 0.107, time taken 18.4 minutes). The results are as follows in figure 5.11.

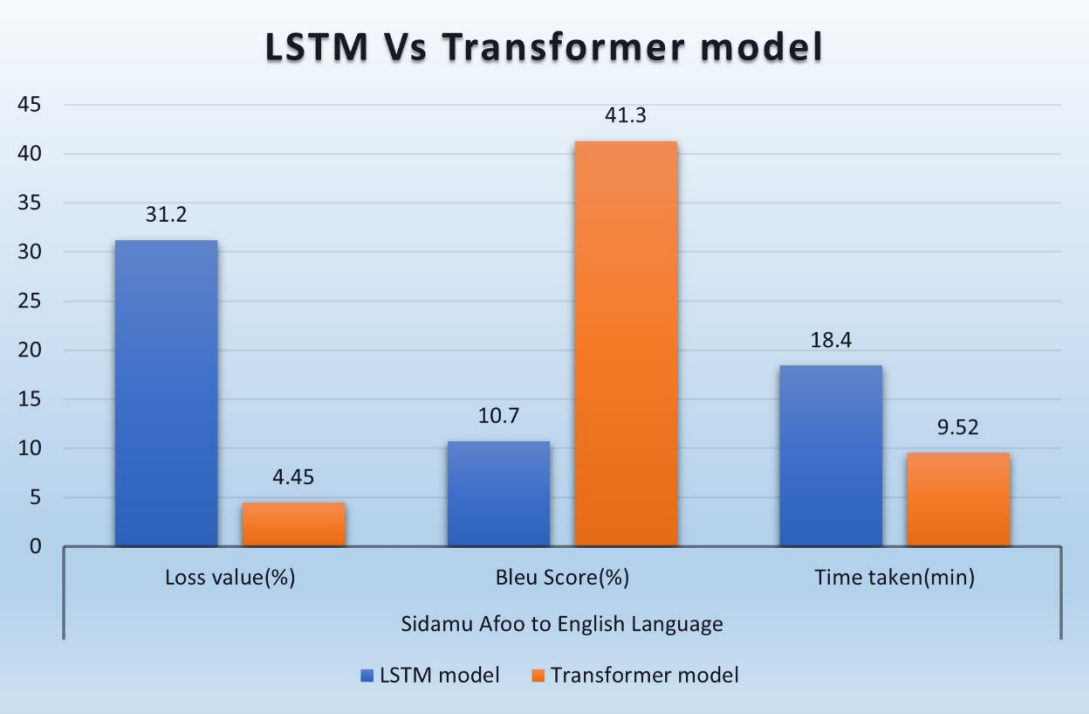


Figure 5. 12 Lstm vs Transformer chart of Sidaamu Afoo to English

- ✓ Translation from the English Language to Sidaamu Afoo

It can be seen that both the LSTM and Transformer models were able to translate from English to Sidaamu Afoo, with the Transformer model performing better with a lower loss value (0.0379) and a higher Bleu Score (0.465), completing the task in 10.11 seconds compared to the LSTM model's loss value of 0.0685, Bleu Score of 0.217, and time taken of 11.7 seconds." It was described below in figure 5.12.

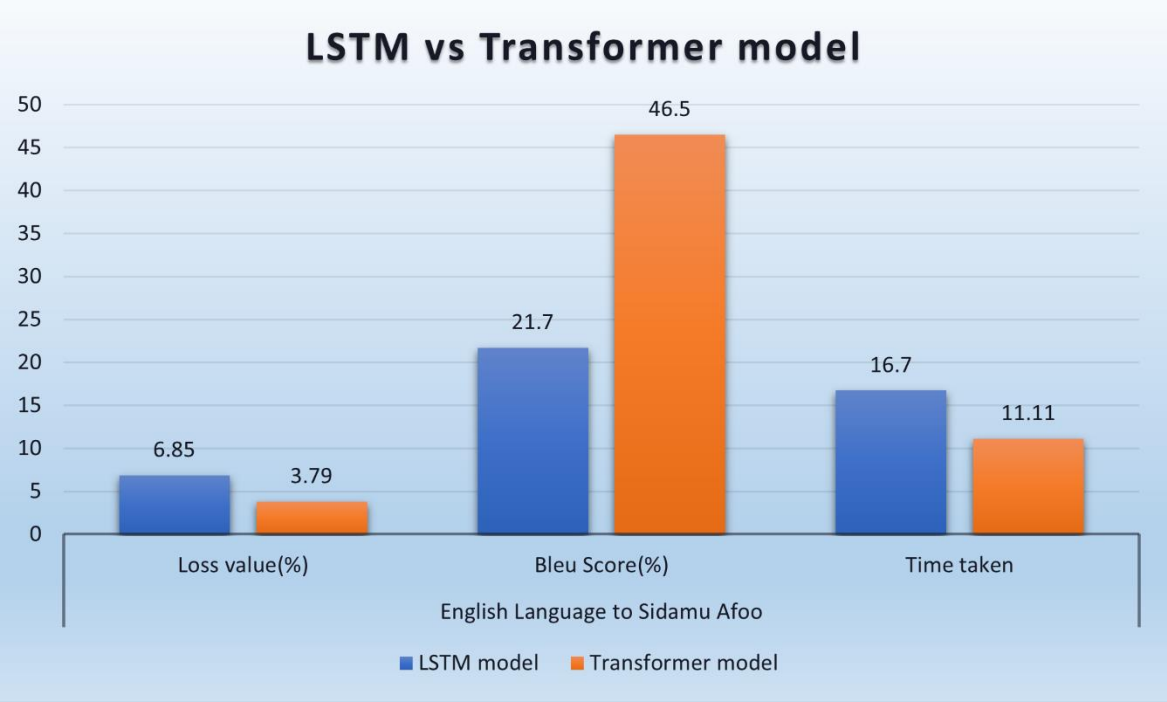


Figure 5. 13 Lstm vs Transformer chart of English to Sidaamu Afoo

- ✓ Sample Tested value and result

The reason why the latest improved experiment of the transformer model was chosen for this test value is due to its high accuracy results. The utilization of the transformer model in both directions resulted in significantly better outcomes, leading us to highly recommend and adopt its use. This is the primary factor behind our decision to use this test value. The sample was listed and evaluated using the BLEU score, as presented in Table 5.12 and Table 5.13 below both English to Sidaamu Afoo and reverse respectively.

Table 5. 17 English to Sidaamu Afoo sample test

no	Type of sentence	English to Sidaamu Afoo Translation	Bleu Score
1	***Source***	*for he is rightly instructed his god teaches him*	
	Translated	*[start] [UNK] maganisi gara ikkino daafira [UNK] [end]*	0.50
	Actual	*[start] loosi'rannohura maganisi gara ikkino coye kulannosi [end]*	
2	***Source***	*is that your sister*	
	Translated	*[start] hatti rodookiti [end]*	0.30
	Actual	*[start] hatti ate rodooti [end]*	
3	***Source***	*son of shallum son of zadok son of ahitub*	
	Translated	*[start] hiliqiyaasi shaluumiho shaluumi saadooqiho [end]*	1.00
	Actual	*[start] hiliqiyaasi shaluumiho shaluumi saadooqiho [end]*	
4	***Source***	*hear o lord and be merciful to me o lord be my helper*	
	Translated	*[start] kaaliiqa ballo eeggatena gatisie kaaliiqa rakke kaalie [end]*	0.4285
	Actual	*[start] kaaliiqa macciishshie mararie kaaliiqa ballo kaa'lie [end]*	
5	***Source***	*bless the lord o my soul and forget not all his benefits*	
	Translated	*[start] kaaliiqa ballo annikki [UNK] lubboya bagamannokkihu baxillisino hegerehona [end]*	0.125

	Actual	*[start] lubbo'ya kaaliiqa galati galatasino habbooti [end]*	

Table 5. 18 Sidaamu Afoo to English sample test

no	Type of sentence	Sidamu Afoo to English Language Translation	Bleu Score
1	***Source***	magano iima gordoho ayirri ayirrinyikkino gobba baalate leello	
	Translated	[start] be exalted o god above the heavens let your	0.513
	Actual	[start] be exalted o god above the heavens let your glory be over all the earth [end]	
2	***Source***	woyyannonkeha doodhino dancha coye mitteenni murro	
	Translated	[start] but for the sake of the elect whom he	0.00
	Actual	[start] let us choose what is right let us know among ourselves what is good [end]	
3	***Source***	hakkiicho ofolte iso agartu	
	Translated	[start] then they sat down and kept watch over him	0.894
	Actual	[start] then they sat down and kept watch over him there [end]	
4	***Source***	kaaiy baaloo	
	Translated	[start] please get up [end]	1.00
	Actual	[start] get up please [end]	
5	***Source***	source: hashsha wi'linanni galliro nafa soodanno woyite hagiirre afi'nanni	
	Translated	[start] but in night o darkness and yet in the	0.238

	Actual	[start] weeping may tarry for the night but joy comes with the morning [end]	

5.8. Discussion of results

The primary goal of this research was to explore the use of deep learning techniques for Sidama-English bidirectional machine translation. To achieve this objective, we focused on the design and implementation of machine translation models that could effectively translate between the two languages at the sentence level. A key aspect of this research was evaluating the performance of these models using a commonly used metric called the BLEU score, which measures the similarity between a machine-generated translation and a human-generated reference translation.

The study was conducted by performing a series of experiments, in which different machine translation models were trained and tested using a dataset of Sidama and English sentences. To achieve the highest possible BLEU score, we proposed the use of transformer-based models. These models are efficient with minimal computational resources and have been found to produce superior BLEU score results as compared to other models. We tested our hypothesis by implementing transformer-based sentence-level machine translation models for Sidama-English and English-Sidama and evaluating their performance using the BLEU score.

Our findings showed that transformer-based models are effective for Sidama-English bidirectional machine translation at the sentence level. The results of our tests demonstrate that the transformer-based approach can achieve the highest BLEU scores and that it is a promising method for machine translation tasks. The transformer model is the superior choice when considering memory usage, as it does not consume as much memory as RNN (specifically LSTM) models. This is a significant drawback of RNN models, particularly LSTM, as they require a large amount of RAM. In our experiments, we found that using 12 GB of RAM did not allow us to run our LSTM models at their optimal performance levels. As a result, we had to decrease the parallel corpus we used for training by 15000 to make the LSTM models run. Another option would be to increase the RAM size in Google Collab, but this would require a financial investment in dollars.

The advantage of the Transformer model is its ability to effectively capture the interdependence of words across the entire sentence, including those at the beginning and end of long sentences, more so than other sentence structures. This is because the model takes into account the context of every word within the sentence, which is determined by its word order. Although the model is more context-dependent, it tends to perform better on shorter sentences than on longer ones.

In our comparison of the LSTM and Transformer models, we found that the Transformer model did not have any memory constraints and performed better than LSTM models in terms of BLEU score and training time. The Transformer model demonstrated better performance not only in terms of memory usage but also in terms of training duration and BLEU score. As a result, the Transformer model was chosen as the preferred model for Sidama-English bidirectional machine translation in this study. The researcher stated that before this study, there had been no previous research on machine translation (MT) between the Sidama Afoo and English language pair. However, our study focused on the development of a bidirectional MT system for Sidama-English. As there was no existing parallel corpus available for this language pair, a parallel dataset was collected and used in the execution of our experiments.

5.9.Answer the Research Questions

At the beginning of this work, we formulated three research questions to be answered after the experiment, hence, here is their answer with the findings of the study. first research Question formulated as which deep learning model is effective for bidirectional English – Sidaamu Afoo translation, based on the experiments, we have investigated two deep learning models that are used for bidirectional English-Sidaamu Afoo machine translation. these two models are Transformer and LSTM. Each is those models demonstrates the used parallel prepared corpus by changing parameter sets many times. After the experiment, the first research question could be answered as the Transformer model performs better in both directions. So, our research confirmed that the transformer model is well-suited for this task and was able to provide an effective solution.

The second question was about which higher parameter values are gives better results for English-Sidaamu Afoo language translation. As seen previously chapter five experimental setting portion described different higher parameter sets used in the experiment for both models, there are dropout rate, embedding dimension, and batch size. On each parameter are used different values on both

models and translation directions. After investigation, the research answered the different parameter value sets, as listed above in table 5.7, 0.2 dropouts out rate, 512 embedding dimensions, and 32 batch sizes are LSTM model for both directions and table 5.13, 0.2 dropout out rate, 256 embedding dimension and 128 batch sizes are Transformer model for both directions.

The third question was about what extent of translation work, which means what the values measured in the assessment of the performance of the proposed bidirectional English-Sidaamu Afoo translation. Our research evaluated the performance of the transformer model and found that it performed well in both directions of translation. In detail our research answered a translation from English to Sidaamu Afoo, the model achieved a BLEU score of 46.5%. The total testing time for this direction was 10.11 minutes. For the reverse translation, from Sidaamu Afoo to English, the test showed the BLEU score was lower at 41.3%. The testing time for this direction was faster, taking only 8.52 minutes.

CHAPTER SIX

CONCLUSION AND RECOMMENDATION

6.1. Introduction

This section presents the results of the research and provides recommendations for anyone or any organization interested in working on machine translation (MT) tasks involving the languages of English and Sidaamu Afoo. The study aimed to evaluate the performance of bidirectional MT between these languages using deep learning techniques. Through a series of experiments, we tested different machine translation models and evaluated their performance using the BLEU score. The research found that transformer-based models were the most effective for Sidama-English bidirectional machine translation at the sentence level. The results of our tests demonstrate that the transformer-based approach can achieve the highest BLEU scores and is a promising method for machine translation tasks. Additionally, the study highlighted the challenges faced when working with a low-resource language pair such as Sidama-English and the importance of collecting a parallel dataset for training machine translation models. Based on these findings, we recommend that future research in this area should consider using transformer-based models and prioritize the collection of a parallel dataset for training.

6.2. Conclusion

The central focus of our research is to address the limitation of relying solely on a human translation by developing and evaluating an automated bidirectional machine translation system between the English and Sidaamu Afoo language pair using deep neural network techniques. The use of machine translation has the potential to greatly improve the efficiency and accuracy of translation between these languages, which is particularly important for low-resource languages such as Sidaamu Afoo.

To achieve this goal, we collected a dataset of 15000 parallel corpora from various sources such as religious texts, publicly available governmental data, for example, constitutions of the Sidama region, and conversational books. This data was used to train and test the machine translation

models in our study. The collected parallel corpus was crucial in our experiment as it was not available previously. We used different neural network-based models like LSTM and Transformer to test the performance of the machine translation.

The results of this research will provide valuable insights into the feasibility and effectiveness of using machine translation for this low-resource language pair. Our findings can inform future research in this area, particularly in the development of machine translation models for low-resource languages and the importance of collecting a parallel dataset for training. Additionally, our study has the potential to contribute to the field of machine translation by providing a deep approach for bidirectional machine translation between English and Sidaamu Afoo.

6.3. Contribution

Our research made significant contributions towards answering three main questions related to the design and performance evaluation of a machine translation tool for the English to Sidaamu Afoo language. Before, researchers hadn't tried using a machine translation tool for Sidaamu Afoo with any other language, so we encountered difficulties when attempting to use such tools in this low-resource language. We investigated the latest deep learning model, which is the Transformer model, and found that they were effective in machine translation for low-resource languages like Sidaamu Afoo when paired with resource-rich languages like English. However, we faced challenges in preparing a parallel corpus for this language, which involved compiling 700 sentences of conversation and 14,300 sentences from the religious domain, totaling 15,000 sentences overall. This process took a considerable amount of time. In addition to these major contributions, our research also made a noteworthy contribution in the preparation of standard and novel parallel corpora from scratch. This is just the first of many contributions that our research will make to this study.

6.4. Recommendation

Our study aims to utilize machine translation exclusively for sentence-to-sentence translation; thus, we suggest future research to integrate with speech-to-speech machine translation between these language pairs. Furthermore, we recommend that future research incorporates additional datasets to further enhance the quality of translation. This would lead to more accurate and fluent

translations. Additionally, testing the systems on TPU-based computers is suggested as this would significantly reduce the training time required for the systems. By doing so, the training process would be more efficient and result in improved performance.

Finally, we recommend leveraging the technology that has been implemented to expand the scope of machine translation to include additional language pairs. This will allow for the seamless exchange of information and ideas across multiple cultures and borders. Utilizing this technology will open up new avenues for communication and collaboration, helping to break down language barriers and facilitate cross-cultural understanding. With the increasing importance of international trade and the globalization of commerce, machine translation has become a vital tool for bridging linguistic and cultural differences. By exploring other language pairs, we can further enhance the potential of this technology and drive its widespread adoption.

Reference

[1] P. M. Mah, I. Skalna, and J. Muzam, "Natural Language Processing and Artificial Intelligence for Enterprise Management in the Era of Industry 4.0," *Appl. Sci.*, vol. 12, no. 18, 2022, doi: 10.3390/app12189207.

[2] H. Approach and J. Daba, "Addis Ababa University School of Graduate Studies College of Natural Sciences Department of Computer Science," no. November, 2013.

[3] S. P. Singh, A. Kumar, H. Darbari, and S. Jain, "MACHINE TRANSLATION USING DEEP LEARNING : AN OVERVIEW," no. April 2019, 2017, doi: 10.1109/COMPTELIX.2017.8003957.

[4] K. Kawachi, *A GRAMMAR OF SIDAAMA (SIDAMO), A CUSHITIC LANGUAGE OF ETHIOPIA by Kazuhiro Kawachi*. 2007.

[5] D. L. Lalego, "College of Natural Sciences Department of Computer Science Development of Morphological Analyzer for Sidaamu Afoo Desta Legese Lalego Advisor : Yaregal Assabie (PhD)," no. June, 2020.

[6] J. Díaz-Ramírez, "Machine Learning and Deep Learning," *Ingeniare*, vol. 29, no. 2, pp. 182–183, 2021, doi: 10.4067/S0718-33052021000200180.

[7] D. S. Kebede, "Hybrid Artificial Neural Machine Translation using Deep Learning Techniques English-to-Afaan Oromoo," no. January, 2019.

[8] F. Hieber *et al.*, "Sockeye: A Toolkit for Neural Machine Translation," pp. 1–18, 2017, [Online]. Available: http://arxiv.org/abs/1712.05690

[9] O. Login, "ULG ' S LANGUAGE SOLUTIONS BLOG STATISTICAL VS . NEURAL MACHINE TRANSLATION," 2021.

[10] S. Sreelekha, P. Bhattacharyya, and D. Malathi, "Statistical Vs Rule Based Machine Translation ; A Case Study on Indian Statistical vs . Rule-Based Machine Translation : A Comparative Study on Indian Languages," no. August 2017, 2021, doi: 10.1007/978-981-10-5520-1.

[11] S. Tesfaye, "A Hybrid approach for Machine Translation from Ge'ez to Amharic language," *Kekuatan Huk. Lemb. Jaminan Fidusia Sebagai Hak Kebendaan*, vol. 21, no. 2, 2020.

[12] R. Madhavan, "Machine Translation – 14 Current Applications and Services."

[13] K. K. Lagide, "A Book Review: The History and Culture of Sidama Nation," pp. 1–17, 2012.

[14] L. Studies, "Journalism and Communication School of Journalism and Communication ' Qeexaala ' as Cultural Communication : The case of Sidaama People By : Mesay Bogale," 2016.

[15] W. Workineh, "Attention-based Amharic-to-Wolaita Neural Machine Translation," no.

June, pp. 2–4, 2012.

[16] G. Tulu, “Bidirectional Amharic-Afaan Oromo Machine Translation Using Hybrid Approach,” 2020.

[17] A. Lambebo, “Multilingual Neural Machine Translation for Low Resourced Languages : Ometo-English”.

[18] L. Benkova and L. Benko, “Neural Machine Translation as a Novel Approach to Machine Translation,” *Divai 2020 13Th Int. Sci. Conf. Distance Learn. Appl. Informatics*, no. October, pp. 499–508, 2020, [Online]. Available: https://www.webofscience.com/wos/alldb/full-record/WOS:000675450500045

[19] P. M. Raffaele Pascale, “Google vs. Facebook Machine Translation.”

[20] S. Chala, B. Debisa, and A. Diriba, “Crowdsourcing Parallel Corpus for English – Oromo Neural Machine Translation using Community Engagement Platform”.

[21] A. M. Gezmu, “Neural Machine Translation for Amharic-English Translation,” vol. 1, no. Icaart, pp. 526–532, 2021, doi: 10.5220/0010383905260532.

[22] 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.

[23] D. Khurana, A. Koli, K. Khatter, and S. Singh, “Natural Language Processing : State of The Art , Current Trends and Challenges Natural Language Processing : State of The Art , Current Trends and Challenges Department of Computer Science and Engineering Manav Rachna International University , Faridabad-,” no. August 2017, 2018.

[24] A. D. M. Fabio Alves , Adriana Siliva, *and intercultural studies Book of abstracts*.

[25] M. D. Okpor, “Machine Translation Approaches : Issues and Challenges,” vol. 11, no. 5, pp. 159–165, 2014.

[26] G. Valunaite Oleskeviciene and J. Sliogeriene, “Research methodology,” *Numanities - Arts Humanit. Prog.*, vol. 13, pp. 39–52, 2020, doi: 10.1007/978-3-030-37727-4_2.

[27] Z. Xiaojun, “Philipp Koehn: Statistical Machine Translation.,” *Appl. Linguist.*, vol. 32, no. 3, pp. 359–362, 2011, doi: 10.1093/applin/amr017.

[28] H. Karlbom, “Hybrid Machine Translation: Choosing the best translation with Support Vector Machines,” no. September, 2016.

[29] S. W. Guangul, “The effects of segmentation techniques in digital image based identification of ethiopian paper currency,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 12, no. 3, pp. 1106–1110, 2018, doi: 10.11591/ijeecs.v12.i3.pp1106-1110.

[30] G. Di Franco and M. Santurro, “Machine learning, artificial neural networks and social research,” *Qual. Quant.*, vol. 55, no. 3, pp. 1007–1025, 2021, doi: 10.1007/s11135-020-01037-y.

[31] M. Aggarwal and M. N. Murty, “Deep Learning,” *SpringerBriefs in Applied Sciences and*

Technology, 2021.

[32] Wang Tingwu, “Contents 1. Why do we need Recurrent Neural Network?,” p. 41, [Online]. Available: https://www.cs.toronto.edu/~tingwuwang/rnn_tutorial.pdf

[33] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” *30th Int. Conf. Mach. Learn. ICML 2013*, no. PART 3, pp. 2347–2355, 2013.

[34] A. Sherstinsky, “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network,” *Phys. D Nonlinear Phenom.*, vol. 404, no. March, pp. 1–43, 2020, doi: 10.1016/j.physd.2019.132306.

[35] A. Vaswani *et al.*, “Attention is all you need,” *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017.

[36] A. Gillioz, J. Casas, E. Mugellini, and O. A. Khaled, “Overview of the Transformer-based Models for NLP Tasks,” *Proc. 2020 Fed. Conf. Comput. Sci. Inf. Syst. FedCSIS 2020*, vol. 21, pp. 179–183, 2020, doi: 10.15439/2020F20.

[37] S. Ibrahim , Gashaw H L, “Amharic-Arabic neural machine translation”.

[38] E. Asefa and H. Seid, “Developing English to Dawurootsuwa machine Translation Model Using RNN,” vol. 69, no. 6, pp. 50–56, 2021, doi: 10.14445/22312803/IJCTT-V69I6P108.

[39] A. Esan *et al.*, “Development of a Recurrent Neural Network Model for English to Yorùbá Machine Translation,” vol. 11, no. 5, pp. 602–609, 2020.

[40] A. Gebremariam, “College of Natural Science Amharic-to-Tigrigna Machine Translation Using Hybrid Approach College of Natural Sciences Akubazgi Gebremariam Advisor : Yaregal Assabie (PhD),” 2017.

[41] I. English, “Should English be the world language?”, [Online]. Available: http://www.debate.org/opinions/should-english-be-the-world-language

[42] S. Wright, *Language choices: Political and economic factors in three European states*. 2016. doi: 10.1007/978-1-137-32505-1.

[43] D. Crystal, “Discovering Literature : Medieval Middle English The Canterbury Tales by Geoffrey Chaucer William Caxton ’ s illustrated second edition of The Canterbury Tales Grammatical change in Middle English,” pp. 1–7, 2018.

[44] R. Willard, “A Middle English Grammar,” *Am. Speech*, vol. 14, no. 4, p. 296, 1939, doi: 10.2307/451630.

[45] N. M. Rayevska, “MODERN”.

[46] L.-T. Cheng, “The Evolution of English Writing System: A Review of Spelling Reform,” *Command Shakespear.*, pp. 1–19, 2007, [Online]. Available: http://1066andallthat.com/english_modern/shakespeare_english_01.asp

[47] L. J. Brinton and D. M. Brinton, “The linguistic structure of modern English,” *Linguist. Struct. Mod. English*, pp. 1–426, 2010, doi: 10.1075/z.156.

[48] A. Introduction, *ANALYSING*.

[49] R. Erlinda, “Chapter 8,” *Linguist. English Lang. Teach. Oleh Rita Erlinda, ISBN 978-602-8887-07-6 | CHAPTER 8-English Syntax*, pp. 130–154, 2012, [Online]. Available: http://fpik.bunghatta.ac.id/files/downloads/E-book/Linguistics for English Language Teaching/chapter_8-english_syntax.pdf

[50] Ji. Feist, “Semantic Structure in English,” *John Benjamins Publ. Co.*, 2016, [Online]. Available: https://escholarship.org/uc/item/5g15p348

[51] “it focus vocabsemantic.”

[52] D. Article and I. Article, “What is an Article ? Article Rules”.

[53] I. Xamiso, *coll. - Sidaamu - Amaarunna - Ingilizete Qaalla Borraasincho. ሲዳምኛ- አማርኛ-እንግሊዝኛ መዝገበ ቃላት. Sidaama - Amharic - English dictionary-Sidama Information and Culture Department (2007).pdf*.

[54] S. Belayneh, “College of Humanities , Language Studies , Journalism and Communication Department of Linguistics , Post Graduate Program Assignment for the Course Advances in Phonetics and Phonology (Ling 8020) The Syllable Structure of Sidaamu Afoo By : Submitted to ,” 2018.

[55] S. Sidamo and I. N. T. H. E. United, “Language & culture,” no. 812, 1960.

[56] Latamo yohannes, “Grade 5-8 Sidaamu Afoo Scope & Sequence”.

[57] D. Ulmer *et al.*, “Experimental Standards for Deep Learning in Natural Language Processing Research,” pp. 2673–2692, 2022, [Online]. Available: http://arxiv.org/abs/2204.06251

[58] D. J. Harland, “An Introduction to Experimental Research An Introduction to Exploratory Research,” *Nursing (Lond).*, vol. 4, no. 3, p. 6, 2015.

[59] J. Sinclair, “Parallel corpora.” http://www.ilc.cnr.it/EAGLES96/corpustyp/node20.html#:~:text=A parallel corpus is a,however%2C exist in several languages.

[60] “English Standard Version® - Hear the Word Audio Bible.” https://live.bible.is/bible/EN1ESV/MAT/1

[61] J. Sola and J. Sevilla, “Importance of input data normalization for the application of neural networks to complex industrial problems,” *IEEE Trans. Nucl. Sci.*, vol. 44, no. 3 PART 3, pp. 1464–1468, 1997, doi: 10.1109/23.589532.

[62] A. Pai, “tokenization,” *June 21st, 2022*. https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

[63] K. Potdar, T. S., and C. D., “A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers,” *Int. J. Comput. Appl.*, vol. 175, no. 4, pp. 7–9, 2017, doi: 10.5120/ijca2017915495.

[64] J. T. Hancock and T. M. Khoshgoftaar, “Survey on categorical data for neural networks,”

J. Big Data, vol. 7, no. 1, 2020, doi: 10.1186/s40537-020-00305-w.

[65] F. Almeida and G. Xexéo, "Word Embeddings: A Survey," no. 1991, 2019, [Online]. Available: http://arxiv.org/abs/1901.09069

[66] C. Wang, P. Nulty, and D. Lillis, "A Comparative Study on Word Embeddings in Deep Learning for Text Classification," *ACM Int. Conf. Proceeding Ser.*, no. June 2021, pp. 37–46, 2020, doi: 10.1145/3443279.3443304.

[67] M. Saeed, "A Gentle Introduction to Positional Encoding In Transformer Models." https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/#:~:text=Positional encoding describes the location,item's position in transformer models.

[68] A. Liutkus, O. Cífka, S.-L. Wu, U. Şimşekli, Y.-H. Yang, and G. Richard, "Relative Positional Encoding for Transformers with Linear Complexity," 2021, [Online]. Available: http://arxiv.org/abs/2105.08399

[69] K. Aitken, V. V. Ramasesh, Y. Cao, and N. Maheswaranathan, "Understanding How Encoder-Decoder Architectures Attend," *Adv. Neural Inf. Process. Syst.*, vol. 27, no. NeurIPS, pp. 22184–22195, 2021.

[70] 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.

[71] "Deep Learning Training vs. Inference." https://www.xilinx.com/applications/ai-inference/difference-between-deep-learning-training-and-inference.html#:~:text=Deep learning training is when,based on the desired outcome.

[72] J. Moolayil, *Learn Keras for Deep Neural Networks: A Fast-Track Approach to Modern Deep Learning with Python*. 2019.

[73] 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.

[74] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *AI Open*, vol. 3, no. 1, pp. 111–132, 2022, doi: 10.1016/j.aiopen.2022.10.001.

[75] G. Li and Y. Li, "Residual Tree Aggregation of Layers for Neural Machine Translation," 2018.

[76] *MERRIAM-WEBSTER DICTIONARIES MERRIAM-WEBSTER'S UNABRIDGED DICTIONAR*. [Online]. Available: https://www.merriam-webster.com/dictionary/parameter

[77] I. Educative, *Learning Rate in Machine learning*. 2023. [Online]. Available: https://www.educative.io/answers/learning-rate-in-machine-learning

Appendixes

Appendix I: Sample parallel corpora

Sidaama Language Sentences	English Language Sentences
Qoqqowu Mootimma Su'ma	Nomenclature of the Regional State
Sidaamu Dagoomu Qoqqowi Qasi'ro	Territorial Boundaries of the Sidaama National Region
insa kayinni iso maaeelsite oso'litu	and they laughed at him
iseno mitiimmakki maati yite xa'mitusi	she said to him speak
xa ka'e kawiinni ha'nona yii	rise let us go from here
kuri baalunku su'mi borreessame no	these were all mentioned by name
lamalayina mite uyisi’e	Give me seventy one
Baandeeranna Sumuda	Flag and Emblem
bala tugannoehu mittu diinino dino	there is neither adversary nor misfortune
qaaqqohono su'ma yesuusa yee fushshi	and he called his name jesus
keere hosini	good evening
arro	hot dry weather
maganinke ninke daafira gaaramannonke yuumm	our god will fight for us
hakkunni gedensaanni fulle rooma ha'nummo	and so we came to rome
lemina mite deerati	twenty one level
mootichuno ki'ne baalunku ledo hee'ro	the lord be with you all
daawiti kayinni lamentera miliqe gati	but david evaded him twice
kaaliiqi nohira waalcho gobbaanni cufi	and the lord shut him in
shoyilina mite osoo eli	He born the fourty one children
leyina mite hige sa’e	He pas the sixty one time
tubalqayinira naima yinanni rodoo noosi	the sister of tubal cain was naamah
kunninni kainohunni lowo geeshsha yaaddu	and they were in terrible distress
mannu baalunku wolqiweelinohu iseraatiwe yiisi	and the people were faint
ninke mimmitto	one another
soodoo jirte	morning rituals
bisooki haashi'ri	take shower
insano naaziretete yesuusa hasi'neemmo yitu	they answered him jesus of nazareth
dicallachishannohena horontanni yaaddooti waajj	do not fear or be dismayed
iseno hayixe bayidhe katama higgu	then she went into the city
go'rota	are you thirsty
go'rinoni	are you thirsty
hudi'roto	are you hungry
xa ani reyammora hasi'roottoni yiisi	here i am i will die
kunnira lamunku mitteenni iyyaarko hadhu	so they came to jericho

asaheeli kayinni xaano shorra diagurino	but he refused to turn aside
insano katama qiishidhe qiniitaabbe heedhu	so they built and prospered
bisoo'ne haashi'xe	take shower
uudano uudi'ri	get dressed
uudano uudi'xe	get dressed
go'roto	are you thirsty
ayiddaannise hoolinose daafira kaaliiqi maaranno	and the lord will forgive her
kunni barrinni shiishino tima ittinoote	no leavened bread shall be eaten
kunni garinni baatto beekke guddu	so they finished dividing the land
Baandeerunna sumudu tittirsha	The Region shall have emblem
lainohunni seerunni murranni	The flag and emblem particulars shall be determined by la

Appendix II: Sample Implementation Code

Transformer model implementation python code

```
import pathlib
import random
import string
import re
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import TextVectorization
```

```
from pickle import dump
from unicodedata import normalize
from numpy import array

# load doc into memory
def load_doc(filename):
  # open the file as read only
  file = open(filename, mode='rt', encoding='utf-8')
  # read all text
  text = file.read()
```

```
  text = file.read()
  # close the file
  file.close()
  return text

# split a loaded document into sentences

# save a list of clean sentences to file
def save_clean_data(sentences, filename):
  dump(sentences, open(filename, 'wb'))
  print('Saved: %s' % filename)
def trial(d):
  lines = doc.strip().split('\n')
  text_pairs=[]
  for line in lines:
    sida, eng = line.split("\t")
    sida = "[start] " + sida + " [end]"
    text_pairs.append((eng, sida))
  return text_pairs
def to_pairs(doc):
  lines = doc.strip().split('\n')
```

```
[ ]    pairs = [line.split('\t') for line in  lines]

       return pairs

     # clean a list of lines
     def clean_pairs(lines):
       cleaned = list()
       # prepare regex for char filtering
       re_print = re.compile('[^%s]' % re.escape(string.printable))
       # prepare translation table for removing punctuation
       table = str.maketrans('', '', string.punctuation)
       for pair in lines:
         clean_pair = list()
         for line in pair:
           # normalize unicode characters
           line = normalize('NFD', line).encode('ascii', 'ignore')
           line = line.decode('UTF-8')
           # tokenize on white space
           line = line.split()
           # convert to lowercase
           line = [word.lower() for word in line]
           # remove punctuation from each token
           #line = [word.translate(table) for word in line]
           # remove non-printable chars form each token
           line = [re_print.sub('', w) for w in line]
```

```
[ ]        line = [re_print.sub('', w) for w in line]
           # remove tokens with numbers in them
           line = [word for word in line if word.isalpha()]
           # store as string
           clean_pair.append(' '.join(line))
         cleaned.append(clean_pair)
       return array(cleaned)



     # load dataset
     # from google.colab import files
     # uploaded = files.upload()
     filename = '/content/drive/MyDrive/Colab Notebook for NMT/Corpus/LastDataForTransformer.txt'
     doc = load_doc(filename)
     # split into english-german pairs
     pairs = to_pairs(doc)
     # clean sentences
     clean_pairs = clean_pairs(pairs)
     # save clean pairs to file
     save_clean_data(clean_pairs, '/content/drive/MyDrive/Colab Notebook For NMT/Model/Transformer/Batch/english-sidama.pkl')
     # spot check
     text_pairs = trial(doc)
     for _ in range(5):
         print(random.choice(text_pairs))
```

```
[ ]  for _ in range(5):
         print(random.choice(text_pairs))
```

```
Saved: /content/drive/MyDrive/Colab Notebook For NMT/Model/Transformer/Batch/english-sidama.pkl
('the sons of elkanah amasai and ahimoth', '[start] hiliqaana amaasayinna ahimooti ili [end]')
('of the sons of harim eliezer isshijah malchijah shemaiah shimeon', '[start] hariimi sirchinni elieezeeri yishiya malkiya shemaiya simiooni [en
('with god we shall do valiantly it is he who will tread down our foes', "[start] maganu ledonke hee'riro qeelleemmo isi diinnanke lekkasira wor
('let them praise your great and awesome name holy is he', "[start] wo'munku lowohanna ayirrado su'masi galato isi qullaawaho [end]")
('then job answered the lord and said', '[start] hakkunni gedensaanni iyyoobi kaaliiqira [end]')
```

```
[ ]  random.shuffle(text_pairs)
     num_val_samples = int(0.10 * len(text_pairs))
     num_train_samples = len(text_pairs) - 2 * num_val_samples
     train_pairs = text_pairs[:num_train_samples]
     val_pairs = text_pairs[num_train_samples : num_train_samples + num_val_samples]
     test_pairs = text_pairs[num_train_samples + num_val_samples :]

     len_tex_pairs = len(text_pairs)
     len_train_pairs = len(train_pairs)
     len_val_pair = len(val_pairs)
     len_test_pair = len(test_pairs)

     print(f"{len(text_pairs)} total pairs")
     print(f"{len(train_pairs)} training pairs")
     print(f"{len(val_pairs)} validation pairs")
```

Appendix III: Training output

```
epochs = 50  # This should be at least 30 for convergence

transformer.summary()
transformer.compile(
    "rmsprop", loss="sparse_categorical_crossentropy", metrics=["accuracy"]
)
transformer.fit(train_ds, epochs=epochs, validation_data=val_ds)
transformer.save("/content/drive/MyDrive/Colab Notebook For NMT/Model/Transformer/Batch")
```

```
Model: "transformer"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 encoder_inputs (InputLayer  [(None, None)]            0
 )
 positional_embedding (Posi  (None, None, 256)         3845120
 tionalEmbedding)
 transformer_encoder (Trans  (None, None, 256)         1315072
 formerEncoder)
 decoder_inputs (InputLayer  [(None, None)]            0
 )
 positional_embedding_1 (Po  (None, None, 256)         3845120
 sitionalEmbedding)
 transformer_decoder (Trans  (None, None, 256)         1839360
 formerDecoder)
 dropout_2 (Dropout)         (None, None, 256)         0
 dense_4 (Dense)             (None, None, 15000)       3855000
=================================================================
Total params: 13,974,828
Trainable params: 13,974,828
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
94/94 [==============================] - 20s 123ms/step - loss: 1.7428 - accuracy: 0.2862 - val_loss: 1.4099 - val_accuracy: 0.3763
Epoch 2/50
94/94 [==============================] - 11s 119ms/step - loss: 1.4706 - accuracy: 0.3610 - val_loss: 1.2685 - val_accuracy: 0.4297
Epoch 3/50
94/94 [==============================] - 11s 121ms/step - loss: 1.2687 - accuracy: 0.4128 - val_loss: 1.1528 - val_accuracy: 0.4591
Epoch 4/50
94/94 [==============================] - 11s 121ms/step - loss: 1.1119 - accuracy: 0.4527 - val_loss: 1.0883 - val_accuracy: 0.4804
Epoch 5/50
94/94 [==============================] - 11s 119ms/step - loss: 0.9694 - accuracy: 0.4953 - val_loss: 1.0591 - val_accuracy: 0.4797
Epoch 6/50
```

```
94/94 [==============================] - 11s 118ms/step - loss: 0.0362 - accuracy: 0.9801 - val_loss: 1.1024 - val_ac
Epoch 41/50
94/94 [==============================] - 11s 118ms/step - loss: 0.0413 - accuracy: 0.9762 - val_loss: 1.1033 - val_accuracy: 0.5805
Epoch 42/50
94/94 [==============================] - 11s 118ms/step - loss: 0.0505 - accuracy: 0.9718 - val_loss: 1.0931 - val_accuracy: 0.5874
Epoch 43/50
94/94 [==============================] - 11s 118ms/step - loss: 0.0297 - accuracy: 0.9839 - val_loss: 1.2489 - val_accuracy: 0.5616
Epoch 44/50
94/94 [==============================] - 11s 118ms/step - loss: 0.0407 - accuracy: 0.9771 - val_loss: 1.1060 - val_accuracy: 0.5877
Epoch 45/50
94/94 [==============================] - 11s 117ms/step - loss: 0.0314 - accuracy: 0.9836 - val_loss: 1.1157 - val_accuracy: 0.5869
Epoch 46/50
94/94 [==============================] - 11s 119ms/step - loss: 0.0352 - accuracy: 0.9814 - val_loss: 1.1309 - val_accuracy: 0.5851
Epoch 47/50
94/94 [==============================] - 11s 121ms/step - loss: 0.0374 - accuracy: 0.9797 - val_loss: 1.1071 - val_accuracy: 0.5895
Epoch 48/50
94/94 [==============================] - 11s 119ms/step - loss: 0.0346 - accuracy: 0.9812 - val_loss: 1.1233 - val_accuracy: 0.5901
Epoch 49/50
94/94 [==============================] - 11s 118ms/step - loss: 0.0379 - accuracy: 0.9800 - val_loss: 1.1131 - val_accuracy: 0.5869
Epoch 50/50
94/94 [==============================] - 11s 118ms/step - loss: 0.0376 - accuracy: 0.9802 - val_loss: 1.1171 - val_accuracy: 0.5847
WARNING:absl:Found untraced functions such as embedding_layer_call_fn, embedding_layer_call_and_return_conditional_losses, embedding_1_lay
/usr/local/lib/python3.8/dist-packages/keras/engine/functional.py:1384: CustomMaskWarning: Custom mask layers require a config and must ov
  layer_config = serialize_layer_fn(layer)
/usr/local/lib/python3.8/dist-packages/keras/saving/saved_model/layer_serialization.py:112: CustomMaskWarning: Custom mask layers require
  return generic_utils.serialize_keras_object(obj)
```

Appendix IV: Test Output of English-Sidaamu Afoo

```
0
source:  for he is rightly instructed his god teaches him
translated:  [start] [UNK] maganisi gara ikkino daafira [UNK] [end]
actual:  [start] loosi'rannohura maganisi gara ikkino coye kulannosi [end]

1
source:  is that your sister
translated:  [start] hatti rodookiti [end]
actual:  [start] hati ate rodooti [end]

2
source:  son of shallum son of zadok son of ahitub
translated:  [start] hiliqiyaasi shaluumiho shaluumi saadooqiho [end]
actual:  [start] hiliqiyaasi shaluumiho shaluumi saadooqiho [end]

3
source:  hear o lord and be merciful to me o lord be my helper
translated:  [start] kaaliiqa ballo eeggatena gatisie kaaliiqa rakke kaalie [end]
actual:  [start] kaaliiqa macciishshie mararie kaaliiqa ballo kaa'lie [end]

4
source:  and they had then a notorious prisoner called barabbas
translated:  [start] [UNK] beetti [UNK] insa [UNK] [UNK] seekkitine agadhe [end]
actual:  [start] hatte yannara barbaani yinannihu haxawarru manchi usurame no [end]
```

```
[ ]
5
source:  and the land of judah will become a terror to the egyptians
translated:  [start] yihudu manni gibitse waajjishiishanno [end]
actual:  [start] yihudu manni gibitse waajjishiishanno [end]

6
source:  bless the lord o my soul and forget not all his benefits
translated:  [start] kaaliiqa ballo annikki [UNK] lubboya bagamannokkihu baxillisino hegerehona [end]
actual:  [start] lubbo'ya kaaliiqa galati galatasino habbooti [end]

7
source:  with patience a ruler may be persuaded and a soft tongue will break a bone
translated:  [start] ane magansinannihura xaadu dukkanira gondooru taabootira [end]
actual:  [start] wodaninni coyi'ra gashshaancho ammansiissanno ga'labbichu hasaawi miqicho eanno [end]

8
source:  his left hand is under my head and his right hand embraces me
translated:  [start] guracho angasi [UNK] qiniitichotenni hanqafannoe [end]
actual:  [start] guracho angasi barkisannoe qiniitichotenni hanqafannoe [end]

9
source:  fifty one
translated:  [start] ontayinamite [end]
actual:  [start] ontayinamite [end]
```

Appendix V: Test Output Sidaamu Afoo to English

```
+ Code  + Text                                                    Connect
[ ]
source:  magano iima gordoho ayirri ayirrinyikkino gobba baalate leello
translated:  [start] be exalted o god above the heavens let your
actual:  [start] be exalted o god above the heavens let your glory be over all the earth [end]

1
source:  woyyannonkeha doodhino dancha coye mitteenni murro
translated:  [start] but for the sake of the elect whom he
actual:  [start] let us choose what is right let us know among ourselves what is good [end]

2
source:  hakkiicho ofolte iso agartu
translated:  [start] then they sat down and kept watch over him
actual:  [start] then they sat down and kept watch over him there [end]

3
source:  qullaawu maxaafira iso hexxannohu baalu disaalfatanno yine borreessinoonni
translated:  [start] therefore it is not in those who forsake him
actual:  [start] for the scripture says everyone who believes in him will not be put to shame [end]

4
source:  debiiri kunni albaanni qiriyaatisefeeri yinanni
translated:  [start] now the name of debir formerly was kiriathsepher [end]
actual:  [start] now the name of debir formerly was kiriathsepher [end]
```

```
5
source:   hashsha wi'linanni galliro nafa soodanno woyite hagiirre afi'nanni
translated:  [start] but in night o darkness and yet in the
actual:  [start]  weeping may tarry for the night but joy comes with the morning [end]

6
source:  kaaiy baaloo
translated:  [start] please get up [end]
actual:  [start] get up please [end]

7
source:  hatte hawarro yesuusi rosaanosi amme baara widoonni fullona yiinsa
translated:  [start] on that day when he had said to one
actual:  [start] on that day when evening had come he said to themlet us go across to the other side [end]

8
source:  saadate ledo foorannohu rumaminoha ikko
translated:  [start] cursed be anyone who lies with any kind of
actual:  [start] cursed be anyone who lies with any kind of animal [end]

9
source:  magani'ne kaaliiqi albaanni minshe nookkire ikke
translated:  [start] you shall be blameless before the lord your god
actual:  [start] you shall be blameless before the lord your god [end]
```

Appendix VI: Bleu Score result

English to Sidaamu Afoo translation Bleu Score metrics result

```
# evaluation
from nltk.translate.bleu_score import sentence_bleu
from nltk.translate.bleu_score import corpus_bleu
test_sida_texts = [pair[0] for pair in test_pairs]
test_eng_texts = [pair[1] for pair in test_pairs]


print(corpus_bleu(actual, predicted, weights=(1, 0, 0, 0)))
print(corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))
print(corpus_bleu(actual, predicted, weights=(0.3, 0.3, 0.3, 0)))
print(corpus_bleu(actual, predicted, weights=(0.25, 0.25, 0.25, 0.25)))
```

```
0.46541793560105493
0.3575720588445522
0.3384730904881418
0.2584689813124231
```

Sidaamu Afoo to English translation Bleu Score metrics result

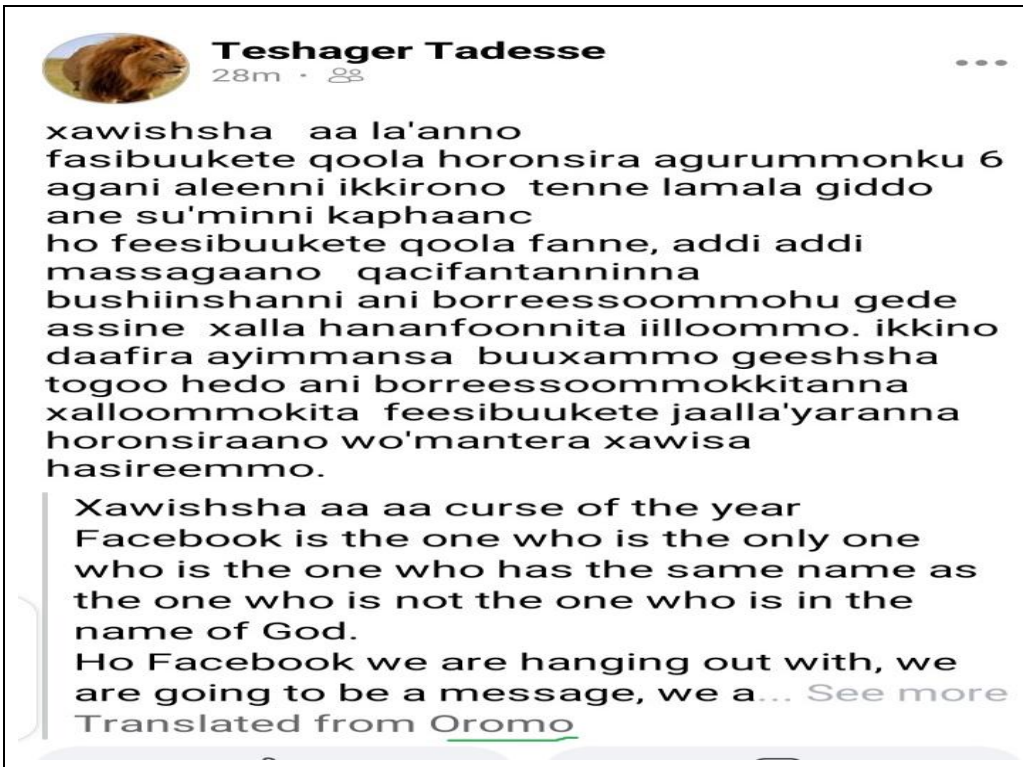
```
# evaluation
from nltk.translate.bleu_score import sentence_bleu
from nltk.translate.bleu_score import corpus_bleu
test_sida_texts = [pair[0] for pair in test_pairs]
test_eng_texts = [pair[1] for pair in test_pairs]

print(corpus_bleu(actual, predicted, weights=(1, 0, 0, 0)))
print(corpus_bleu(actual, predicted, weights=(0.5, 0.5, 0, 0)))
print(corpus_bleu(actual, predicted, weights=(0.3, 0.3, 0.3, 0)))
print(corpus_bleu(actual, predicted, weights=(0.25, 0.25, 0.25, 0.25)))
```

```
0.41337325428416805
0.360167899378609
0.35681342521433257
0.3182979991908172
```

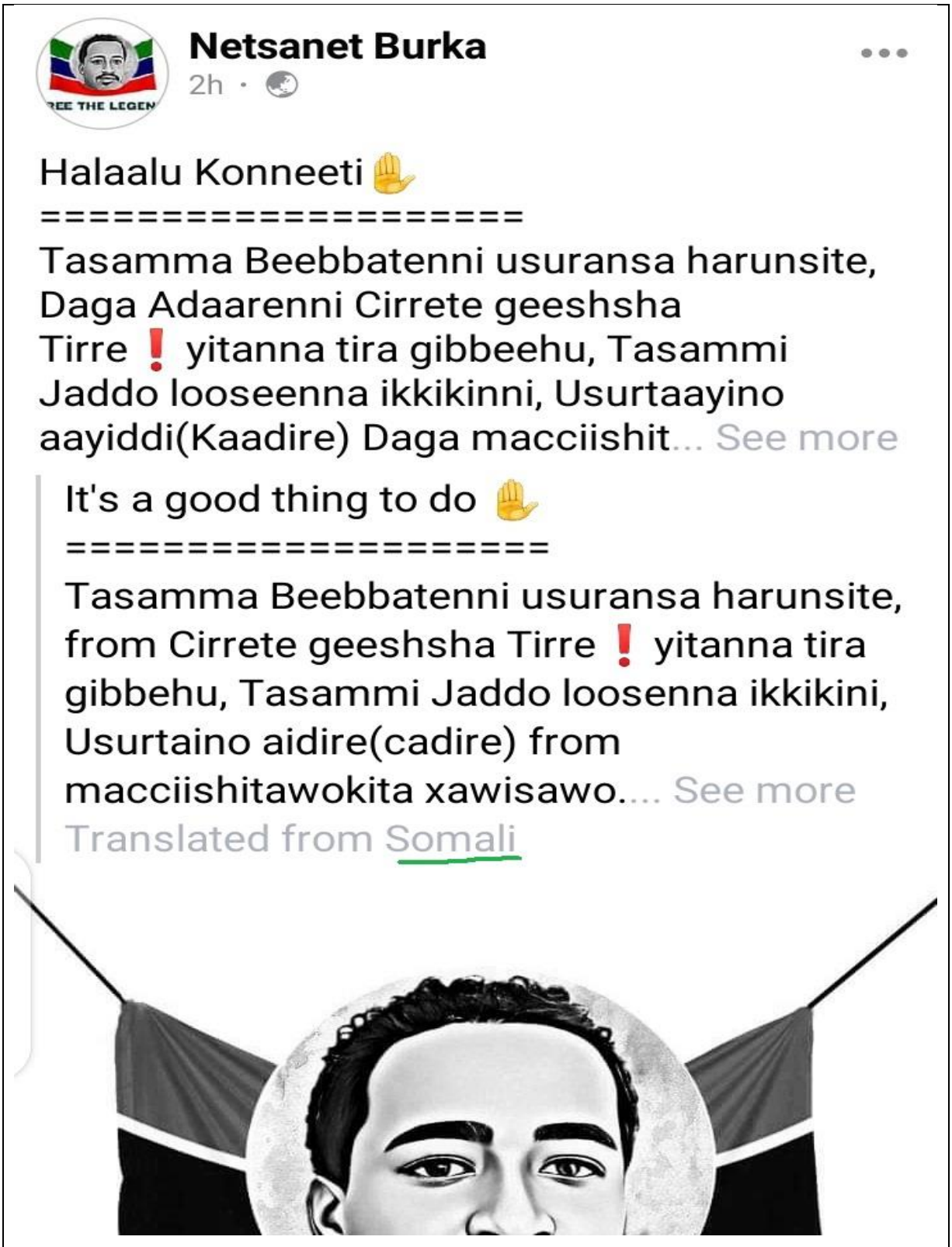
Appendix VII: Sample Survey of Facebook posts

A Sidaamu Afoo text translated as taken from another Cushitic language like Afan Oromo, Somalia Afi, Estonian, and many more

Sidaamu Afoo posts taken as Afan Oromo on Facebook language translation



Sidaamu Afoo posts taken as Somalia Afii on Facebook language translation



Sidaamu Afoo posts taken as the Estonian Language on Facebook language translation

SUSA Discussion Forum
Jo Yefkr Sew · 3d ·

FreeFree Ejjeetto
3d ·

Hiiqancho Oodo !!
XXXXXXX
Hawaass Quchchumi Gido Aaffamanohu Taabori Laynkkinna Geegoote Roosi Mini Roossano Techo Barra Saaddasa 18/4/2014 M.D **Sidaamu** Qoqqowi Gaashshano Kallancho P/r **Tessema Elias Sho'le** Mite Jadono Loo... See More

Hiqancho Oodo !!
XXXXXXX
Hawaass Quchchumi Gido Aaffamanohu Taabori Linkkinna Geegote Rose Mini Roseano Techo Barra Saaddasa 18/4/2014 M.D Sidaamu Qoqqowi Gaashshano Kallancho P/r Tessema Elia... See More
Translated from Estonian