



DEBRE BERHAN UNIVERSITY
INSTITUTE OF TECHNOLOGY
COLLEGE OF COMPUTING
Department of Information Systems

***A Bi-Directional Ge'ez-Amharic Neural Machine Translation:
a Deep Learning Approach***



Amdework Asefa Belay

**A Thesis Submitted to Department of Information Systems of Debre
Berhan University in Partial Fulfillment for the Masters of Science
in Information Systems**

Advisor: *Wondwossen Mulugeta (PhD)*

Addis Ababa, Ethiopia

Debre Berhan, Ethiopia

July 2021

DEBRE BERHAN UNIVERSITY
INSTITUTE OF TECHNOLOGY
COLLEGE OF COMPUTING
Department of Information Systems

Amdework Asefa Belay

Advisor: *Wondwossen Mulugeta (PhD)*

This is to certify that the thesis prepared by Amdework Asefa, titled: *A Bi-Directional Ge'ez-Amharic Neural Machine Translation: a Deep Learning Approach* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Information Systems complies with the regulations of the University and meets the accepted standards concerning originality and quality.

Approved by Board of Examiners

	<u>Name</u>	<u>Signature</u>	<u>Date</u>
Advisor:	_____	_____	_____
Internal Examiner:	_____	_____	_____
External Examiner:	_____	_____	_____

© 2021

AMDEWORK ASEFA BELAY

ALL RIGHTS RESERVED

ACKNOWLEDGMENT

Above all, though I do not deserve it, I thank my God for the entire he has done for me and by his grace bringing me here. Next, I would like to express my deep-felt gratitude to my advisor, Dr. Wondwossen Mulugeta, for his nice advice and tireless support in helping me to do better work. Moreover, I would like to express my sincere gratitude to my brothers and sisters (friends) for their support and encouragement until I accomplished this study. Particularly, by helping Geez translations, evaluation and by collecting bilingual Ge'ez-Amharic parallel sentences, my childhood teacher, Merigieta Meseret Tsegaye (የንታየ/yenitaye), Marigeta Memhir Bekalu Menberu, my brothers D/n Mamas Tadesse, D/n Atsinaf G/Egziabhier and Woldeyesus Ayele, nothing will pay back you whatever I say for everything you have done for me. Just May the supreme God, graciously glorify you all. Finally, I would like to express my deepest gratitude to my mother, Abezu Chanie, who endured many hardships and trials to me to make me reach here, and my younger sister Tigza Assefa (I see her as my older sister), who always encouraged me to complete this research and for everything in my life.

ምንተኑ አዕሰዮ ለእግዚአብሔር በእንተ ኩሉ ዘገብረ ሊተ፡፡ መዝ ፻፲፮፡፲፪

Amdework Asefa



Date: July 2021

DEDICATION

This research work is dedicated to my mother Abezu Chanie, my sister Tigza Assefa and the Ethiopian Orthodox Church. Especially, dedicated for ቀንፋ ጩዎች አንበሳ ማሰሪያ ቅዱስ ገብርኤል ሰ/ት/ት ቤት (*k'enefa ch'ewochi ānibesa maserīya k'idusi gebiri 'ēli se/ti/ti bēti*).

ABSTRACT

Currently, due to globalization, our world is moving into one village and human languages are being transnational. So far, human interpreters have been resolving communication gaps between two people who speak different languages. However, since human translation is costly and inconvenient, many kinds of research are being done to resolve this problem with Machine Translation (MT) techniques. MT is a process of automatically translating text or speech from one human language to another by computers. Neural Machine Translation uses Artificial Neural Networks such as Transformers, which are the state of the art models that shows promising result over the previous MT models. Several ancient scripts written in the Ge'ez language that needs to be translated are available in Ethiopia and abroad. Currently, youth and researchers are interested to learn and involve in research areas of Ge'ez and Amharic manuscripts. This thesis, therefore, aims to demonstrate the capabilities of deep learning algorithms on MT tasks for those morphologically rich languages. A bi-directional text-based Ge'ez-Amharic MT was tested on two main different deep learning models viz. Seq2Seq with attention, and Transformer. A total of 20,745 parallel corpora was used for the experiment, from which the 13,787 parallel sentences were collected from former researchers and a new 6958 parallel corpus was prepared. In addition, a Ge'ez Latin numeric corpus having 3,078 parallel lines has been added to handle numeric translation. We conducted four experiments, and the transformer outperforms other techniques by scoring 22.9 and 29.7 BLEU scores from Ge'ez to Amharic and vice versa using 20,745 parallel corpora. The typical Seq2Seq model improves the BLEU score of the SMT model, obtained by previous researchers with BLEU scores of +0.65 and +0.79 that is 2.46% and 4.66% increment from Ge'ez to Amharic and from Amharic to Ge'ez using 13,833 parallel sentences. Doing further researches with clean larger corpus size and pre-trained models may improve the result we have reported in this work. However, we faced a scarcity of corpus and pre-trained models for Amharic and Ge'ez languages to get better results.

Keywords: Artificial Neural Network, Attention, BLEU, Seq2seq, Machine Translation, Neural Machine Translation, Transformer

Table of Contents

LIST OF TABLES	IV
LIST OF FIGURES	V
LIST OF ALGORITHMS	VI
LIST OF ACRONYMS	VII
CHAPTER ONE: INTRODUCTION.....	1
1.1 BACKGROUND	1
1.2 MOTIVATION	3
1.3 STATEMENT OF THE PROBLEM.....	4
1.4 RESEARCH QUESTIONS	6
1.5 THE OBJECTIVE OF THE STUDY	7
1.5.1 General Objective:.....	7
1.5.2 Specific Objectives:.....	7
1.6 RESEARCH METHODS	7
1.6.1 Literature Review.....	7
1.6.2 Corpus Preparation	8
1.6.3 Implementation Tools and Experiment	8
1.6.4 Evaluation producers.....	9
1.7 SCOPE AND DELIMITATIONS OF THE STUDY	9
1.8 SIGNIFICANCE OF THE STUDY	9
1.9 ORGANIZATION OF THIS WORK	9
CHAPTER TWO: LITERATURE REVIEW.....	10
2.1 OVERVIEW	10
2.2 NATURAL LANGUAGE PROCESSING.....	10
2.3 MACHINE TRANSLATION	10
2.3.1 History of Machine Translation	10
2.3.2 Approaches to Machine Translation.....	12
2.4 ARTIFICIAL NEURAL NETWORKS	24
2.4.1 Deep neural networks	25
2.4.2 Different types of Neural Networks.....	26
2.4.3 Word embedding	40
2.5 MACHINE TRANSLATION EVALUATION METRICS.....	44
The Bi-Lingual Evaluation Understudy (BLEU)	45
2.6 CHALLENGES TO MACHINE TRANSLATION.....	46
2.7 RELATED WORKS	46
Summary of Related Works.....	53
CHAPTER THREE: GE'EZ AND AMHARIC LANGUAGES	54
3.1 THE GE'EZ LANGUAGE.....	54
3.1.1 Ge'ez Script Arrangements (ኑባሬ ፊደል).....	54
3.1.2 Ge'ez Numerals (አኃዝ).....	55
3.1.3 Similar Letters (ተመካሳይያን).....	55
3.2 THE AMHARIC LANGUAGE.....	56
3.3 LINGUISTIC RELATIONSHIPS OF GE'EZ AND AMHARIC	57
3.3.1 Writing System	57
3.3.2 Syntactic Language Structure (Word Order).....	57

3.3.3	<i>Grammar structure of Ge'ez and Amharic (ሰዋሰው)</i>	58
3.4	MAJOR PARTS OF SPEECH	58
3.4.1	<i>Noun (ስም)</i>	58
3.4.2	<i>Adjective/ቅጽል</i>	59
3.4.3	<i>Verb (ግስ)</i>	59
3.4.4	<i>Adverb (ተውላክ ግስ)</i>	60
3.5	MINOR PARTS OF SPEECH.....	61
	<i>Punctuation Marks</i>	61
CHAPTER FOUR: RESEARCH METHODOLOGY		62
4.1	OVERVIEW	62
4.2	RESEARCH DESIGN.....	62
	<i>Design Science Research Methodology</i>	63
4.3	THE PROPOSED SYSTEM.....	65
4.4	PREPROCESSING	67
4.4.1	<i>Major Cleaning</i>	67
4.4.2	<i>Normalization</i>	67
4.4.3	<i>Tokenization</i>	68
4.4.4	<i>Padding</i>	69
4.5	INPUT EMBEDDING	70
4.6	THE ENCODER	70
4.7	THE DECODER.....	71
4.8	THE EVALUATION	73
CHAPTER FIVE: EXPERIMENT AND RESULT ANALYSIS		74
5.1	OVERVIEW	74
5.2	CORPUS PREPARATION	74
	<i>Challenges during corpus preparation</i>	76
5.3	EXPERIMENTAL SETUP.....	77
	<i>SW and HW Tools Used for Experiment</i>	77
5.4	EXPERIMENTS.....	78
5.5	LINGUIST EVALUATION	89
5.6	ANSWERING RESEARCH QUESTIONS	92
5.7	CHALLENGES OF GE'EZ AND AMHARIC DURING MACHINE TRANSLATION	93
CHAPTER SIX: CONCLUSION AND RECOMMENDATION.....		94
6.1	OVERVIEW	94
6.2	CONCLUSION	94
6.3	CONTRIBUTION OF THE STUDY	94
6.4	FUTURE WORKS AND SUGGESTIONS.....	95
REFERENCES		96
ANNEX		101

List of Tables

Table 2.1 General Comparison of Machine Translation Approaches.....	23
Table 2.2 BLEU scores of Morpheme Based Bi-directional Ge'ez-Amharic MT	52
Table 2.3 Comparison between related works on Ge'ez-Amharic MT	53
Table 3.1 similar letters in Ge'ez and Amharic	55
Table 3.2 Amharic Script (a) added script, (b) Derived script.....	56
Table 3.3 Root or main Verbs (አርአዕተ-ግስ) of Ge'ez.....	60
Table 5.1 Hardware and Software Requirements	77
Table 5.2 The different experiments, models, and corpus size used.....	78
Table 5.3 The length of sentence, token and type used for each experiment	79
Table 5.4 The proposed Models and their BLEU score result.....	88
Table 5.5 The four Experiments and their BLEU score result.....	88
Table 5.6 Parallel sentences prepared for manual evaluation (Translated by OpenNMT).....	90
Table 5.7 Parallel sentences prepared for manual evaluation (Translated by Transformer)	91
Table 5.8 Manual evaluation results compared with the BLEU score.....	91
Annex-Table A.1 Vocabulary and One-Hot encoding Vector.....	101
Annex-Table B.1 Word embedding of a sentence "አንተ ግን ያው አንተ ነህ"	101
Annex-Table C.1 Ge'ez Script Arrangements.....	101
Annex-Table C.2 Ge'ez and Amharic numerals.....	102

List of Figures

Figure 1.1 The relationship between AI, ML, and Deep Learning [10]	2
Figure 2.1 Architecture of RBMT Approach [4]	13
Figure 2.2 Bernard Vauquois' pyramid [4].....	14
Figure 2.3 Instance of example-based translation.....	16
Figure 2.4 the general model of ANN followed by its processing.....	24
Figure 2.5 Diagram of the Attention model shown in Bahdanau's paper [45].....	32
Figure 2.6 Architecture of Transformer Model adopted from [47]	34
Figure 2.7 a Transformer of 2 stacked encoders and decoders	36
Figure 2.8 Scaled dot product attention adopted from [47]	38
Figure 2.9 Multi-head self-attention adopted from [47]	38
Figure 4.1 Design Science Research Framework Adopted From [63]	62
Figure 4.2 DSR Methodology Process Model Adopted from [63]	64
Figure 4.3 Overall architecture of the proposed system	66
Figure 4.4 The detailed structure of Transformer's Encoder and Decoder architecture.....	72
Figure 5.1 Sample dataset from Ge'ez Amharic corpus in distinct files	76
Figure 5.2 Snapshot of OpenNMT (a) training, (b) loss, and (c) output.....	82
Figure 5.3 Hyper-parameters of the Transformer model	83
Figure 5.4 Results shown by the OpenNMT model with the new corpus	84
Figure 5.5 Output results shown by the Seq2Seq model with the new corpus	85
Figure 5.6 The numeric corpus	86
Figure 5.7 Google Translate is still suffering from mistranslating Ge'ez numeral translation .	87
Figure 5.8 Comparison of Deep Learning models	89
Annex-Figure D.1 Snapshot of fragment code of Sentencepiece tokenizer and detokenizer .	103
Annex-Figure D.2 A Snapshot for fragment code of preprocessing.....	103
Annex-Figure D.3 Snapshot of fragment code of Normalization.....	104
Annex-Figure E.1 A sample photo of the OpenNMT translation manual evaluation	105
Annex-Figure E.2 A sample photo of the Transformer translation manual evaluation	106

List of Algorithms

Algorithm 4.1 Algorithm for general cleaning (Preprocessing) of a corpus.....	67
Algorithm 4.2 Algorithm for Amharic text normalization.....	68
Algorithm 4.3 Amharic sub-word segmentation Algorithm	69

List of Acronyms

Adam: <i>Adaptive Moment Estimation</i>	IRSTLM: <i>Istituto per la Ricerca Scientifica e Tecnologica Language Modeling</i>
AI: <i>Artificial Intelligence</i>	LSTM: <i>Long Short Term Memory</i>
ALPAC: <i>Automatic Language Processing Advisory Committee</i>	METEOR: <i>Metric for Evaluation of Translation with Explicit Ordering</i>
ANN: <i>Artificial Neural Networks</i>	MLP: <i>Multi-Layer-Perceptron</i>
ATR: <i>Advanced Telecommunications Research</i>	MS: <i>Micro-Soft</i>
BERT: <i>Bidirectional Encoder Representation from Transformers</i>	MT: <i>Machine Translation</i>
BLEU: <i>BiLingual Evaluation Understudy</i>	NaN: <i>Not a Number</i>
BP: <i>BackPropagation</i>	NEC: <i>Nippon Electric Company</i>
BPE: <i>Byte Pair Encoding</i>	NIST: <i>National Institute of Standards & Technology</i>
BPTT: <i>Backpropagation Through Time</i>	NLP: <i>Natural Language Processing</i>
CAP: <i>Credit Assignment Path</i>	NMT: <i>Neural Machine Translation</i>
CBMT: <i>Corpus-Based Machine Translation</i>	NN: <i>Neural Network</i>
CBOW: <i>Continuous Bag of Words</i>	NVIDIA: <i>New Video Intelligence Automaton (Envidia -Spanish)</i>
CNN: <i>Convolutional Neural Network</i>	OCR: <i>Optical Character Recognition</i>
CPU: <i>Central Processing Unit</i>	OKI3: <i>Open Knowledge Initiative</i>
C-STAR: <i>Consortium for Speech Translation Advanced Research</i>	OpenNMT: <i>Open Neural Machine Translation</i>
DMT: <i>Direct Machine Translation</i>	ReLU: <i>Rectified Linear Unit</i>
EBMT: <i>Example-Based Machine Translation</i>	RNN: <i>Recurrent Neural Network</i>
EOTC: <i>Ethiopian Orthodox Tewahido Church</i>	RoBERTa: <i>Robustly Optimized BERT Pre-Training Approach</i>
FDRE: <i>Federal Democratic Republic of Ethiopia</i>	Seq2Seq: <i>Sequence 2(to) Sequence</i>
FNN: <i>FeedForward Neural Networks</i>	SGD: <i>Stochastic Gradient Descent</i>
GNMT: <i>Google's Neural Machine Translation</i>	SMT: <i>Statistical Machine Translation</i>
GPU: <i>Graphical Processing Unit</i>	SVO: <i>Subject-Verb-Object (SOV, VSO..)</i>
GRU: <i>Gated Recurrent Unit</i>	TAUM: <i>Troy Area United Ministries</i>
HICATS: <i>Hitachi Computer-Aided Translation System</i>	TPU: <i>Tensor Processing Unit</i>
IBM: <i>International Business Machines Information Interchange</i>	WMT: <i>Workshop on Machine Translations</i>
	Word2vec: <i>Word 2(to) Vector</i>

CHAPTER ONE: INTRODUCTION

1.1 Background

Machine Translation (MT) is an automatic translation of text or speech from one natural language to another, preserving the meaning of the input text, and producing fluent text in the target language [1]. MT is a relatively old task that has taken a long journey in research and development. Over the years, two major MT approaches have emerged; a rule-based approach and the corpus-based approach. In the rule-based approach, experts' knowledge about the source and the target language to develop syntactic, semantic, and morphological rules is required to achieve the translation. Whereas, under the corpus-based approach, there is a parallel *corpus*¹ built by human experts from where the knowledge is automatically extracted by analyzing translation examples (bitext)² [2]. Corpus-based MT includes Example-Based MT (EBMT), Statistical Machine Translation (SMT), Neural Machine Translation (NMT), and other Hybrid MTs [2, 3, 4]. These are discussed under section 2.4 in detail.

SMT is an MT paradigm where translations are generated based on statistical models, whose parameters are derived from the analysis of bilingual text corpora. It is focusing on finding the translation with the highest probability of occurrence of words based on existing translations. Google launched Google Translate based on SMT in 2006. However, later on, after 10 years of research, Google substitute SMT with NMT to increase fluency and accuracy since SMT had poor grammatical accuracy [5].

Neural Machine Translation on the other hand is the newest approach to machine translation and is based on Artificial Neural Networks that consist of nodes conceptually modeled after the human brain. The complex and dynamic nature of networks in NMT allows the model to guess more educated and appropriate target text about the context. NMT systems iteratively learn and adjust weights to provide the best output than SMT but require a lot of processing power [6].

¹ A pair of equivalent source sentences and target sentences

² Refers to the corpora or the training data used to build a translation model

Deep Learning (also known as Deep Neural Network) is a sub-set of machine learning methods based on Artificial Neural Networks that imitates the workings of the human brain in processing data and creating patterns for use in decision-making. These networks are capable of learning unsupervised from data that is unstructured or unlabeled. [7]. Earlier versions of NNs such as the first perceptron were shallow, composed of one input and one output layer, and have at the most one hidden layer in between. Deep Learning is the name used for “stacked neural networks”; that is, networks composed of several layers. According to [8, 6, 9] More than or equal to four layers (including input and output) qualifies as “deep” learning. The relationship between AI, MT, and Deep Learning [10] is depicted in Figure 1.1.

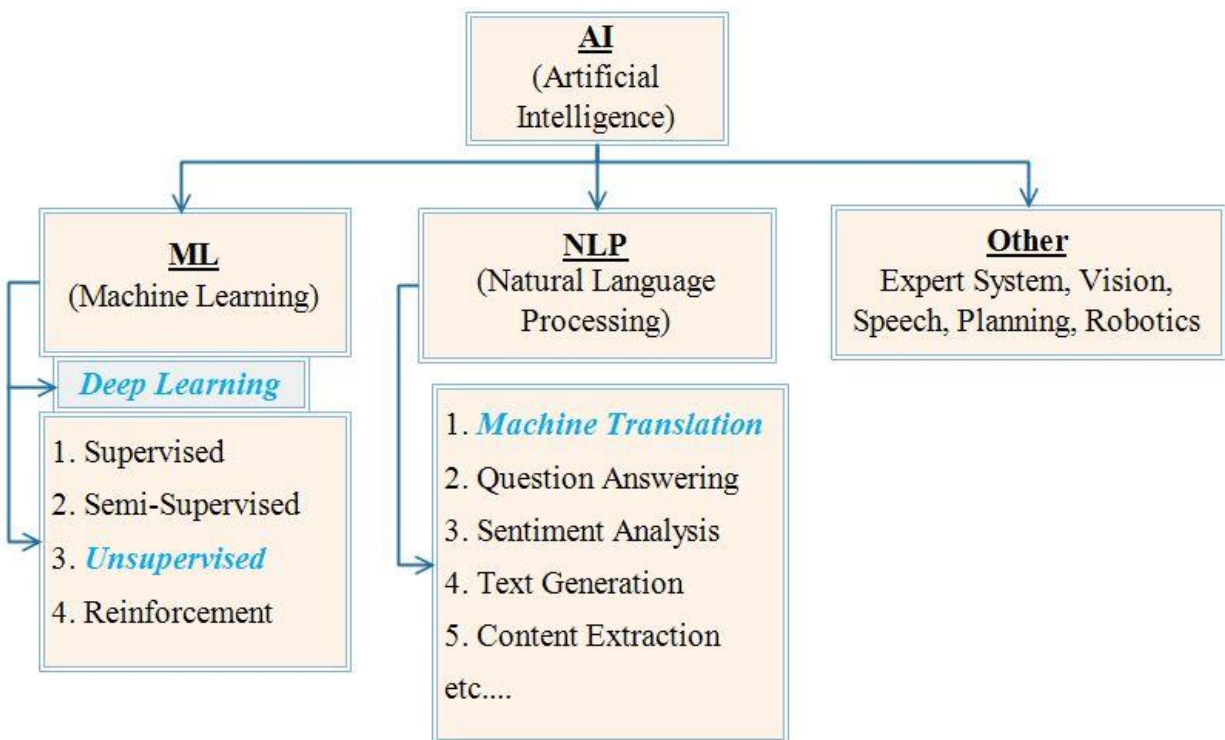


Figure 1.1 The relationship between AI, ML, and Deep Learning [10]

As shown in Figure 1.1 ML and NLP are branches of AI. Moreover, Deep learning is a subset of ML and MT is one of the major tasks of NLP. Hence, we will use an unsupervised Deep Learning approach to Machine Translation. This study is focusing on the state-of-the-art NMT with a deep learning structure for Ge’ez and Amharic languages.

Ge’ez is an ancient Semitic language with its own script that is originated around the 5th century BC [11]. The language began in the northern part of Ethiopia, in Eritrea and Tigray. It has been

the official language of Ethiopia for thousands of years [12]. As a result, many Ethiopian stories, civilizations, philosophies, religion, medicine, engineering, astrology, education, crafts, arts, and many other things that need to be translated are recorded. At present, the language has become so flagged and it is now only used as liturgical (praise and thanksgiving) language in the Ethiopian and Eritrean Orthodox Churches. Although it is a language that could be lost at all, the Ethiopian Orthodox Church has preserved it and deserves credit for it [11, 13]. Ge'ez uses the 'Abugida'³ writing system from left to right (former was an *abjad*⁴ system, from right-to-left, just like Arabic).

Amharic, on the other hand, is the current official working language of the Federal Democratic Republic of Ethiopia and is estimated to be spoken by well over 100 millions people as a first or second language that is 83.3% of the population and over 3 million people outside Ethiopia including USA, Israel, Egypt, Canada, etc. [14]. Amharic is the second most spoken Semitic language in the world (after Arabic). Today it is the largest language in Ethiopia and one of the five largest languages on the African continent. Following the Constitution drafted in 1993, Ethiopia is divided into nine (now ten)⁶ independent regions and two chartered cities, each with its own regional working language. Amharic is the working language of different regional states including Amhara regional state, Addis Ababa and Southern Nations, Nationalities, and people's regional States. Due to these reasons, creating an intelligent machine translation system for these languages will help millions of people.

1.2 Motivation

A lot of ancient scripts and documents, written in Ge'ez are available in Ethiopia and abroad. These documents are written in different fields of areas such as religion, medicine, engineering, astrology, education, and many more. The language attracts the attention of many researchers to carry out critical analysis on socio-cultural, political, astronomical, and historical aspects of Ethiopia's past. Currently, the youth and researchers are interested to learn the Ge'ez language and

³ (known as *alphasyllabary*, *neosyllabary* or *pseudo-alphabet*) is a writing system that is neither a syllabic nor an alphabetic script, but somewhere in between. Consonant-vowel sequences are written as a unit, each based on the consonant letter and vowel notation is secondary. E.g. ቀ (Fa). Adopted from አቡጊዳ (a-bu-gi-da), the name of its own script, based on the Ge'ez alphabet order (አበገደ) similar with Greek alphabet order (Α, Β, Γ, Δ).

⁴ A type of writing system in which (in contrast to true alphabets) each symbol or glyph stands for a consonant.

⁵ <https://www.press.et/english/?p=2654> accessed: Mar 2019

⁶ The Sidama region has been created in June 2020 and makes ten regions.

have good motivations to involve in research areas of Ge'ez manuscripts. As of Wendy Laura 2017 [15], about 7 Ethiopian and more than 25 abroad universities give Ge'ez language as either a department or a course and more universities are launching Ge'ez department in Ethiopia. Hence, this language is rising again and needs more attention. There should be a good translation system between Ge'ez and other languages [2, 16]. Google Translate supports more than 108 world-spoken languages including Amharic that have sufficient resources such as wiki and news. However, languages such as Ge'ez (with no speakers and online resources) and Afaan Oromo (has just 772 articles in its Wikipedia) which have not an adequate resource for translation purposes are not supported until 2021⁷.

That is why this study is initiated to explore the possibility of developing deep learning models that can produce fluent and natural-sounding translation between Ge'ez and Amharic languages with the available bilingual corpora using NMT techniques.

1.3 Statement of the Problem

Various studies have shown that hundreds of thousands of Ge'ez books are available in Ethiopia and abroad. It is believed that these books, both domestically and abroad, have contributed greatly to modern civilization [12]. To gain this wisdom and knowledge, these Ge'ez books and scripts must be digitized and translated into different languages. For these efforts, there are many digitization projects, which can electronically digitize these contents. Then they can be used for developing machine translation models. Particularly, if they are translated into Amharic they will have a valuable contribution to Ethiopians, in holding the country's values. Thus these contents can also be translated to languages like English using the available Google Translate to make them more accessible to the world. Moreover, students and researchers in different fields will gain the advantage of exploring the knowledge documented using the language. Even though it is time-consuming, Ge'ez is currently being translated into Amharic and other languages manually by linguists, mainly by the Orthodox Tewahido Scholars [2, 17].

Nevertheless, as we have already noted, manual translation is difficult to do widely and easily because of these reasons. The language is known only by some people and or priests, human

⁷ <https://www.wired.com/story/google-translate-wikipedia-siri-widely-spoken-languages-cant-translate/> accessed May 26, 2021

translation takes a very long time, editing and evaluation is too costly (in terms of money, time, and raw materials like paper, vellum, and ink), and translation errors cannot be handled easily. This can be very difficult to translate, evaluate and deliver a translation in a short period.

In the past few decades, there have been many attempts for solving machine Translation problems. While most of the works focused on resourced languages. Less-resourced and morphologically rich languages such as Ge'ez get limited attention due to a lack of data. Some studies [2, 16, 17] have been tried to develop a Machine Translation for Ge'ez and Amharic in different approaches such as Rule-Based, SMT, and hybrid. The rule-based and hybrid machine translations are based highly on word-based translation. In addition, the researchers said that there was a challenge because of the morphological richness of the languages [17]. Beyond this, the researchers faced Syntactic, Semantics, and Pragmatics transformation challenges, misworded (Wrong word choices), disordered grammatical organization of with Subject Verb Object (SVO), Verb Subject Object (VSO), and Subject Object Verb (SOV), as well, generating unknown words, name translation problems, and so on. Moreover, according to Tadesse [2], different alignments were handled such as one-to-one, one-to-many, and many-to-one alignments; however, many-to-many alignment and derivational and Inflectional morphemes are not handled [2]. Hence, we try to overcome these problems in this study [2, 16].

In the grammatical organization; Ge'ez follows somewhat free word order structure *SVO*, *VSO*, *SOV*, and *OVS* Part of Speech Tags. Here are some examples:

SVO: “ውእቱ መጽአ ኅበ ቤቱ (wi'itu mets'i'ā hābe bētu. and “እግዚአብሔር ነበረ ለሙሴ (igizī'ābiḥēri nebebo lemusē)”.

VSO: “መጽአ ውእቱ ኅበ ቤቱ (mets'i'ā wi'itu hābe bētu)” and “ነበረ እግዚአብሔር ለሙሴ (nebebo igizī'ābiḥēri lemusē)”.

SOV: “ውእቱ ኅበ ቤቱ መጽአ” (wi'itu hābe bētu mets'a'ā) and “እግዚአብሔር ለሙሴ ነበረ” (igizī'ābiḥēri lemusē nebebo). Though their word order is different the above three PoSs have the same meaning as “እሱ ወደ ቤቱ መጣ (isu wede bētu met'a)” meaning “He came home” and “እግዚአብሔር ሙሴን ተናገረው (igizī'ābiḥēr musēn tenagerew)” meaning, “God spoke to Moses” respectively. These Parts of

Speeches (PoS) were not handled in previous works. Only one of the three PoSs was applied in one research [2].

Furthermore, different words like “ሰላሊ, [se’ali]” and “ሰአሊ, [se’ali]” have the same sound but different meanings with “to *draw*” and “to *beg*” respectively. Again “ዓመት (*A’met*) and “አመት (*Amet*) have the same sound but different meaning, which means “*Year*” and “*housemaid*” respectively. This type of syntaxes needs a context-based translation just like Google Neural Machine Translation (GNMT). The previous researchers strongly recommended extending their research using a pure and larger corpus size and various domains of contents in addition to the religious one [2, 17].

Even though there are attempts in both languages to deal with the aforementioned problem, almost all of the works focused on exhaustive word and morpheme-based translations using rule-based or Statistical Machine Translations such as phrase-based with very limited corpus, which are not mature enough to be used. All the above researches on Ge’ez and Amharic [2, 16, 17] are conducted with SMT and faced those problems. Therefore, to address and overcome those problems we will try to use more training data, by developing a better model with Deep Neural Networks. NMT can handle alignments and word ordering by itself via Attention mechanisms.

In recent times Neural Machine Translation models that learned from large, unstructured data using a language-independent architecture achieved the state-of-the-art result in Machine Translation. However, such approaches need to learn sub-word tokenizer models and pre-trained models such as BERT. While resourceful languages such as English benefited from this technique, languages such as Amharic and Ge’ez are barely represented due to a lack of learned sub-word segmenter such as BPE, WordPeice, and SentencePeice.

1.4 Research Questions

Lastly, this study tries to answer and address the following research questions:-

- ❖ How effective are the Transformer and the Seq2Seq models in MT for low-resourced languages such as Ge’ez and Amharic?
- ❖ Which NMT algorithm from the Transformer and Seq2Seq is better for low resource languages like Ge’ez and Amharic?
- ❖ What are the main challenges of translation between Ge’ez and Amharic?

1.5 The objective of the study

1.5.1 General Objective:

The main objective of this work is to explore and experimentally demonstrate deep learning-based bi-directional Ge'ez-Amharic Neural Machine Translation.

1.5.2 Specific Objectives:

To achieve the specified general objective, the following specific objectives will be applied.

- ❖ Investigate literature related to Deep learning-based Machine Translation tasks.
- ❖ Explore orthographic and phonetic characteristics of both Ge'ez and Amharic languages.
- ❖ Prepare a parallel bi-lingual corpus for both Ge'ez and Amharic languages.
- ❖ Do basic cleaning on the bilingual corpus.
- ❖ Apply Amharic normalization.
- ❖ Use tokenization.
- ❖ Apply padding to shorter sentences to make them equal to the longest one.
- ❖ Design a bi-directional Ge'ez-Amharic translation model.
- ❖ Conduct experiments with different deep learning models
- ❖ Evaluate the performance of translation in both manual and automatic evaluation methods.
- ❖ Select the best performer model
- ❖ Explore challenges of Ge'ez-Amharic Machine Translation.
- ❖ Suggest Future works and recommendations.

1.6 Research Methods

To achieve the general and specific objectives of the study and to answer the research questions, the Design Science research methodology will be used as a general research methodology and the following subtopics will also be used as an additional research methodology.

1.6.1 Literature Review

Performing the literature survey provides knowledge about the state-of-the-art models in the current research areas. It will help us to understand the existing approaches, techniques, and tools. It provides a comprehensive interpretation of the problem domain, to design an effective Ge'ez-Amharic MT. A vast review of literature in the area of MT with a special focus on Deep Learning

approaches and algorithms will be done. Many peer-reviewed publications including books, articles, journals, and other scholarly publications will be reviewed. Moreover, discussions with the linguist of Ge'ez and Amharic will also be done.

1.6.2 Corpus Preparation

We will use an existing domain-specific bilingual corpus having 13,787 parallel sentences prepared by the former researcher [2] from the bible, Mass, praise of Mary, and other Ethiopian Orthodox Church's faith books. We will also collect and prepare additional corpus in addition to the existing one.

An experiment will be conducted with the formerly available corpus to determine whether the new NMT is effective for low-resourced languages and to be comparable with previous SMT results reported by [2]. However, The NMT will also be checked with both the existing corpora and with the existing plus the newly added corpus. In addition, parallel Ge'ez-Amharic numeral datasets will be added to handle numeric translation.

1.6.3 Implementation Tools and Experiment

Python will be used as a tool to develop models with Pytorch and OpenNMT will also be used for confirmation. Python supports extensive collections of special libraries for implementing deep-learning methodologies. Particularly offers a neater and faster way to build highly performing algorithms.

The model for Ge'ez-Amharic Machine Translation will be made with Pytorch and experimented using the available corpus. We will also try our dataset on OpenNMT to ensure whether our result is reliable or not. OpenNMT is an open-source ecosystem for Neural Machine Translation and neural sequence learning [18]. We will train the proposed model with two established NMT architectures, namely, Attention Based Seq2Seq NMT and the Transformer [19].

In addition, draw.io, the online diagrams tools , and Edraw max 7.9 will be used to draw diagrams of this study. MS Word, PPT, and notepad++ will be used for writing a report, presentation slides, and text processing such as corpus preparation respectively, and browsers such as Opera, Google Chrom, and MS Edge will also be used for running Google's Co-laboratory scripts.

⁸ <https://app.diagrams.net/> accessed from sept 2020 – Jul 2021

1.6.4 Evaluation producers

To evaluate the translation quality of the proposed Ge'ez-Amharic NMT system and to be comparable with previous works we will use a BLEU (Bilingual Evaluation Understudy) score and Human (Manual) evaluation system (rating scale).

1.7 Scope and Delimitations of the study

This study focuses on Deep Learning NMT approaches for Ge'ez and Amharic *text* Translation. An 85.45% of the corpus (17724 pairs of sentences), used for this study, is from the religious domain and the remaining 14.56% (3021 pairs of sentences) is from Ge'ez teaching books.

The delimitation of this study on the other hand is, it does not include Amharic Romanization and post-editing techniques. The lack of conversation bilingual corpus is also another issue.

1.8 Significance of the study

The outcome of this study can help researchers to investigate old literature written in Ge'ez since many of ancient civilization pioneer literature such as Medicine, Astronomy, and calendar (like Bahire Hasab), Archeology, Engineering, and ...so on has been written in Ge'ez. The study can keep Ge'ez alive by raising and extending its age. As well, this work can help the upcoming youths who are interested to study the Ge'ez and Amharic languages. Moreover, it helps people to find out the true history of Ethiopia.

1.9 Organization of this work

This document is organized into six chapters. The **first chapter** presented the statement of the problem, objective, scope, methodology, Significant, scope, and limitations of the study. The **second chapter** deals with the literature review of the science behind the MT process, approaches, evaluation techniques, and related works. The **third chapter** discusses the Ge'ez and Amharic Languages, their Lexical, Morphological, and Syntactic characteristics. The **fourth chapter** presents the methodology used for this study, the proposed MT architecture, and its components in detail. The **fifth chapter** presents the dataset, implementation, result, and analysis of the system. In the **last chapter**, we put our conclusion and possible recommendation for future studies based on the findings of this work.

CHAPTER TWO: LITERATURE REVIEW

2.1 Overview

This chapter discusses Natural Language Processing (NLP), Machine Translation (MT), and history of machine translation, approaches to machine translation (rule-based and corpus-based machine translations), Artificial Neural Networks (ANN), machine translation evaluation metrics, challenges to MT, and related works.

2.2 Natural Language Processing

Different scholars have defined Natural Language Processing (NLP) in different ways. From those definitions, we selected the two more generalized ones. Andreas Kaufmann [20] defined it as “*NLP is a subdomain of artificial intelligence and as an interdisciplinary field; it is combining linguistic knowledge with computer science.*” Baysolow II, Taweh [21] also said, “*NLP is a subfield of computer science that is focused on allowing computers to understand language in a ‘natural’ way, as humans do*”. Natural Language Processing shows the capability of computers to understand and processes human languages focusing on the interactions between human and computer via communication or language in a natural way. NLP includes tasks such as machine translation, automatic text summarization, named entity recognition, sentiment analysis, speech recognition, Question and answering, PoS tagging, and so on [16].

2.3 Machine Translation

Machine translation is the process of translating a text or speech in a given or input language to a target or output language using automated computers [1]. The story of Machine Translation begins in the 1940s, which was considered as the pioneers and stages of MT.

2.3.1 History of Machine Translation

Endeavor to do machine translation systems started just as soon as computers came into existence. According to Mohamed Amine Chérâgui [22], the history of machine translation started in 1948 and classified into five periods and narrated as follows: -

First period (1948-1960): The beginning: In this era, four different researchers showed a good pioneer to MT. In 1949, Warren Weaver adopted the term computer translation in his Memorandum and proposed the first ideas using computers in translation. In 1954 by a group of researchers from Georgetown University in collaboration with IBM, the first very basic automatic translator was developed, which translates about sixty Russian sentences into English. The authors claimed that within three to five years, machine translation would not be a problem. In the same year (in 1954): Victor Yngve published the first journal on MT, entitled “*Mechanical translation devoted to the translation of languages by the aid of machines*”. MT research programs pop up in Japan and Russia (1955), and in London (1956) when the first MT conference was held.

Second Period (1960-1966) Parsing and disillusionment: In the early 1960s (*Tesnière stratificationnelle Lamb*) there were parsers developed from different types of grammars, such as grammar and dependency grammar. In February 1961, there were series of weekly lectures organized by David G. Hays at the Rand Corporation in Los Angeles, and computational linguistics was born. In 1964, ALPAC (Automatic Language Processing Advisory Committee) was created with the American government to study the perspectives and the chances of machine translation. In 1966, ALPAC published its famous report and it concluded that its works on machine translation are just wasting of time and money. The conclusion of this rapport hurt MT's search for several years.

Third period (1966-1980): New birth and hope: The project named REVERSO by a group of Russian researchers was started in 1970. In the same year Peter Toma, a member of a group search for Georgetown at that time, developed *SYSTRANI* (Russian-English). In 1976, at the University of Montreal, a group of researchers under the direction of Alai Colmerauer created the system *Weather* (the machine translation to weather forecasts for the public) in the project TAUM. Lastly, *Atlas2* was created in 1978 by the Japanese firm *Fujitsu*, this translator was based on rules and able to translate from Korean to Japanese and vice versa.

Fourth Period (1980-1990): Japanese invaders: The Japanese Company *Sharp* markets its Automatic translator (English - Japanese) in 1982. This translator was based on rules and an approach to translation transfer. In 1983: the NEC (Nippon Electric Company) develops its translation system based on an algorithm called *Pivot* by using Interlingua. OKI3 (Open

Knowledge Initiative) at Massachusetts Institute of Technology, also developed a system named *Pensee* in 1986, which is a translator (Japanese-English) based on rules. In 1986, the group *Hitachi* developed its translation system (Japanese- English) based on rules (which is an approach taken by transfer), and christened HICATS (Hitachi Computer-Aided Translation System).

Fifth Period (since 1990): The Web and the new vague of translators: The project named C-STAR (Consortium for Speech Translation Advanced Research) made the first demonstrations trilingual transatlantic in January 1993. It was a project on the machine translation of the parole in the field of tourism (dialogue client travel agent), by videoconference. This system dealt with three languages (English, German, and Japanese) the company Softissimo starts marketing the translator REVERSO in 1998. In 2000, an example-based MT started with Japanese- English and Chinese - English. The system was done by a Japanese laboratory ATR.

2.3.2 Approaches to Machine Translation

MT has taken a long journey in research and development and it is a relatively old task. Over the years, two major approaches emerged, a *rule-based approach* and the *corpus-based approach*. In the rule-based approach, experts' knowledge about the source and the target language is required to develop syntactic, semantic, and morphological rules for translation. Whereas, under the corpus-based approach, there is a parallel corpus built by human experts from where the knowledge is automatically extracted by analyzing translation examples (bixtext) [2]. Corpus-based MT includes Example-Based MT (EBMT), Statistical Machine Translation (SMT), Neural Machine Translation (NMT), and other Hybrid MTs [3, 4, 23].

i. Rule-Based Machine Translation Approach

RBMT was the first commercial and practical approach to machine translation, developed several decades ago. It is also known as Knowledge-Based Machine Translation or Classical Approach of MT. It works based on a manually determined set of rules encoded by linguistic experts. RBMT relies on a very large number of bilingual dictionaries for each language pair, and with countless linguistic rules such as rules for syntactic analysis, lexical transfer, syntactic generation, morphology, lexical rules, etc. The rules attempt to define correspondences between the structure of the source language and the target language. Representation should be unambiguous lexically and structurally. It consists of a collection of rules called grammar rules and lexicon to process the

rules [2, 17]. The goal of RBMT is to convert the source language structure to the target language structure by preserving the meaning of source language texts and generating equivalent target-language texts [24].

The benefit of RBMT is that without a need for huge bilingual corpora, a good engine can translate a wide range of texts, not like in SMT or NMT, which require large bilingual corpora. However, RBMT is time-consuming and labor-intensive to develop a system and it may take several years for one language pair.

Additionally, In RBMT when facing real-life texts, like metaphorical or slang texts, human-encoded rules are limited and unable to cover all possible linguistic phenomena. This may result in poor translation quality. For this reason, RBMT has completely been replaced by SMT or hybrid systems; however, it is still being used for less common language pairs wherein there are not sufficient corpora to train Statistical or Neural Machine Translation engines. The steps in RBMT are depicted in Figure 2.1 below.

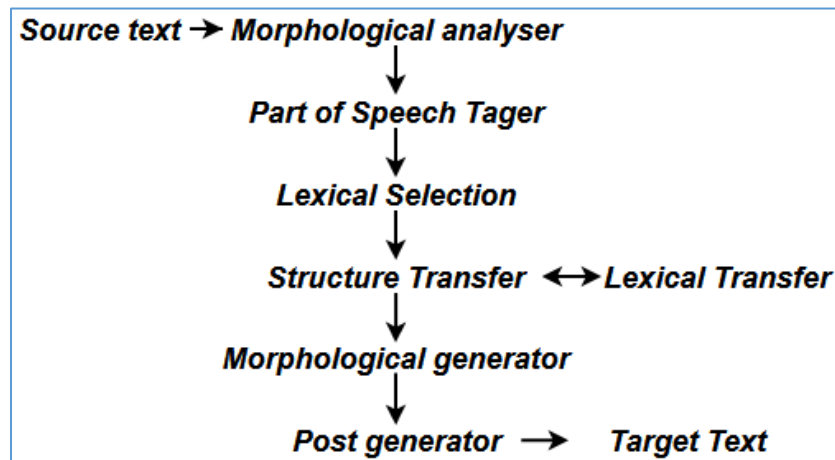


Figure 2.1 Architecture of RBMT Approach [4]

Sub Approaches of RBMT

There have been three basic approaches under rule-based machine translation; the Direct, Transfer-Based, and Interlingua Machine Translation Approaches [4]. They differ in the depth of analysis of the source language and the extent to which they attempt to reach a language-independent representation of meaning or intent between the source and target languages. The Vauquois Triangle illustrates these levels of analysis and shows their dissimilarities as depicted in Fig. 2.2.

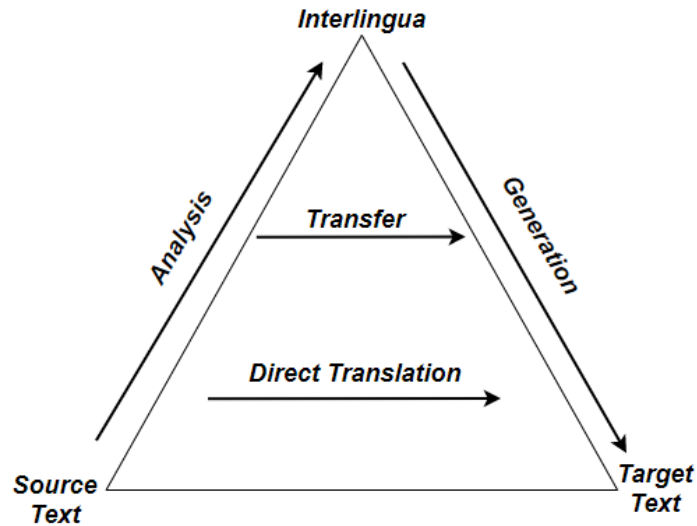


Figure 2.2 Bernard Vauquois' pyramid [4].

a) **The direct Machine Translation (DMT) Approach** Starts with the shallowest level at the bottom of the pyramid as shown in Figure 2.2. It works by translating Words of Source Language without passing through an additional (intermediary) representation. The analysis of Source Language texts is oriented to only one Target Language [4, 24].

b) **The transfer-based Machine Translation Approach** is the second generation of Rule-Based machine translation (between the 1960s and 1980s) after Direct Machine Translation. In this approach, the source language is transformed into a less language-specific representation called *abstract*. Then for the target language, an equivalent representation (with the same level of abstraction) will be generated using bilingual dictionaries and grammar rules. Transfer-based machine translation creates a translation from an intermediate representation that simulates the meaning of the original sentence. Unlike Interlingua MT, the Transfer model depends partially on the language pair involved in the translation [4, 24].

c) **Interlingua Machine Translation Approach** is the third generation of Rule-Based machine translation that is an inherent part of a branch called *Inter-Linguistics*⁹. In Interlingua, the source language is transformed into an auxiliary or intermediary language (i.e. a “language-neutral” representation) that is independent of any language. Later, the translated verse for the target language is generated from the Interlingua. One of the main advantages of Interlingua MT is that it can support a large number of target languages to be turned into [4, 24].

⁹ From two Latin words, *Inter* and *Lingua*, which means *intermediary* and *language*

ii. *Corpus-Based Machine Translation Approaches*

Corpus-based machine translation (also known as “data-driven” or Empirical machine translation) is another recent approach for machine translation that started to overcome the problem of rule-based machine translation (a knowledge acquisition problem, to meet the wide variety and time-changing characteristics of the real text). As its name implies, Corpus-Based Machine Translation (CBMT) uses, a huge amount of corpora, (mostly a bilingual parallel corpus) to obtain knowledge for new entry translation. The bilingual parallel corpus contains text and its translations. The translation knowledge is acquired from these corpora. The corpus-based approach is further classified into three main sub approaches; Example-based Machine Translation, Statistical Machine Translation, and Neural Machine Translation Approaches [4].

A. *Example-Based Machine Translation (EBMT) Approach*

Example-Based Machine Translation is also known as “memory-based”, “case-based”, “experience-guided”, “example-guided inference”, or “analogy-based” translation [17]. It works translation by recalling or finding analogous examples of the language pairs from premade corpora. EBMT translates a source sentence by imitating the translation of the matched examples of a similar sentence already in the corpora. These similar sentences (examples) are used to translate a similar type of sentence stored in a database from the source language to the target language. In EBMT Analogy of text, the *similarity* is the key; hence, the concept is “Translation by Analogy”. In EBMT the basic principle is that, if a formerly translated phrase occurs, a similar translation is likely to be correct once again. There are four stages in EBMT [16]

- ❖ ***Example acquisition***: is about how to acquire examples from the parallel bilingual corpus.
- ❖ ***Example base management***: - is about how examples are stored and maintained.
- ❖ ***Example application***: - concerns itself with how examples are used to facilitate translation, which involves the decomposition of an input sentence into examples and the conversion of source texts into target texts in terms of an existing translation.
- ❖ ***Target sentence synthesis***: - is to compose a target sentence by putting the converted examples into a smoothly readable order, aiming at enhancing the readability of the target sentence after conversion.

Example-Based Machine translation works by calculating the distance between the input sentence and the analogy stored in the database or corpora. The smaller the distance is more similar to the input. When a new source sentence is entered for translation, the examples are retrieved to find similar ones in the source; afterward, the target sentence is generated by imitating the translation of the matched examples. As shown below in Figure 2.3, based on the first (i) examples, the second (ii) translation can be done [2].

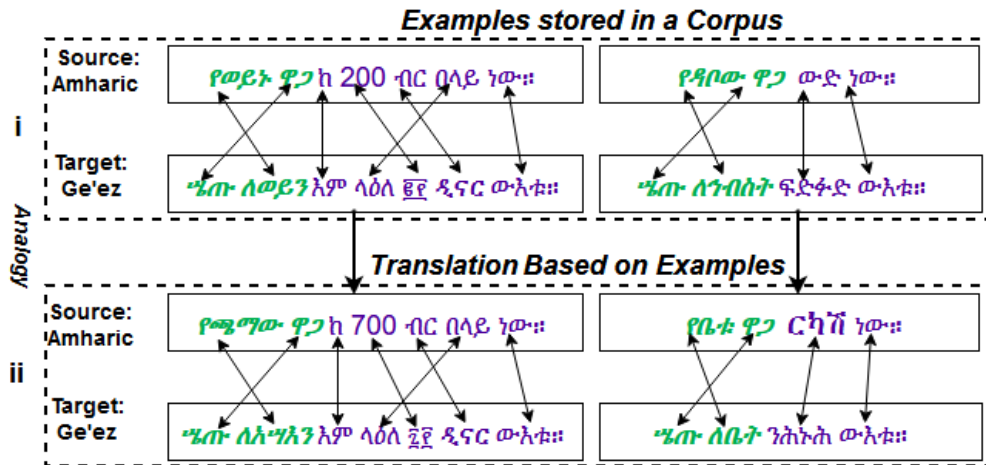


Figure 2.3 Instance of example-based translation

B. Statistical Machine Translation Approach

Statistical Machine Translation is an empiricist approach, which works by using a very large volume of bilingual corpora to train the translation engine. However, some studies corroborate SMT as one of the paradigms of EBMT [24]. In this, statistical methods such as n-gram based SMT and Occurrence-based SMT are applied to generate a translated version using bilingual corpora.

The first model of SMT was based on Bayes Theorem and proposed by Brown et al [25]. The probability that every sentence in one language has a possible translation of any sentence in the other [4]. SMT requires a large amount of monolingual and bilingual data. The monolingual corpus is required to estimate the right word orders that the target language should look like and the bilingual is an aligned sentence, used to build the translation model training and decoding purpose that determine the word (phrase) alignment between the two aligned sentences [17]. SMT in general is focusing on finding the translation with the highest probability of being the best. However, balancing Adequacy (how faithful is the translation to the source) and Fluency (how

natural is the translation) should be under-considered. In SMT, the quality metrics for the source and target language is measured via a product of faithfulness and fluency [2]

$$\mathbf{Best\ translation\ } \hat{T} = \mathit{argmax}_T = \mathbf{Faithfulness}(T, S) \mathbf{fluency}(T) \quad (2.1)$$

Where in that a source language sentence S may translate into any target language Sentence T. It is based on statistical searching of the most likely translation from a huge bilingual corpus. For every pair of strings (A, G) ‘A’ is given and a probability P(G|A) is assigned to produce ‘G’ as its translation [17]. The entire formula can be written as equation 2.2 based on Bayes' theorem.

$$P(A|G) = \frac{P(G|A)P(A)}{P(G)} \quad (2.2)$$

Where P(A) is the probability of the *language* model, and P(G|A) is the probability of the *translation* model. A sentence is translated based on the probability distribution of P(A|G) in which a string ‘A’ in the target language (for instance, Amharic) is the translation of a string ‘G’ in the given source language (for example, Ge’ez). A sentence, which comes, as the translation (ê), is the one that has the highest probability with each sentence in ‘A’ is a translation of ‘G’ with this probability. In mathematical terms [17], P(G) is fixed, the maximization of P(A|G) denoted by ê is thus equivalent to maximization of P(G|A)P(A) and it gives.

$$\hat{e} = \mathit{argmax}_A P(G|A)P(A) \quad (2.3)$$

Where P(G|A): The *Translation* model that provides the probabilities of possible translation pairs of the source sentence A given the translated sentence G. The P(A) is the *Language* model that provides a probability to each unit of text. In addition, *argmax_A* is the Search algorithm in decoder that searches for the best translation from the given all possible translations based on the probability estimates P(A|G) and P(A) and performs the actual translation. Based on this concept there are three components of SMT in its general architecture namely, the *Language Model*, the *Translation model*, and the *Decoder*. [16].

The Language model: determines the probability of a sequence of words or ensuring that words come in the right order, i.e., Subject-Object-Verb (SOV). The target Language Model is produced using a monolingual corpus where reasonable sequences of words are given high probabilities and

senseless ones are given low probabilities. It generally reflects how frequently a string of words occurs as a sentence.

The translation model: looks for statistical correlations between input texts and translations. That is the bilingual relationship between the source and target strings of corresponding parallel corpora. It then shows how likely a given source text is mapped to a translation by generating confidence scores. However, the translation engine itself has no notion of rules or grammar.

There are three categories of translation models: - word-based models, phrase-based models, and syntax-based models [17]. In the *word-based* translation model, the objective is to discover the word-to-word translational correspondences in a bilingual corpus. Here the fundamental unit is a word. It handles translation and alignment at the word level. However, compound words, idioms, and homonyms create complexity. In the *phrase-based* model, the fundamental unit is a *phrase* or sequence of words. A sequence of words in the source and the target language should be developed. These multi-word segments of words are called *blocks* or *phrases*, which are not linguistic phrases like noun phrases but phrases found using statistical methods from the corpus. Decoding is done based on the vector of features with matching values for the language sequence pair. It translates any contiguous sequence of words and reduces restrictions produced by word-based translation. The phrase-based model is the widely adopted method among all the proposed approaches in SMT. In the *syntax-based* model, the fundamental unit is the translation rule. It is based on the idea of translating syntactic units, rather than single words (as in word-based MT), or strings of words (as in phrase-based MT). Translation rule consists of a sequence of words and variables in the source language, a syntax tree in the target language (having words or variables at leaves), and a vector of feature values that describes the likelihood of the language pairs.

The Decoder on the other hand uses a searching algorithm to determine the most probable translation among all possible translations. Most decoders in the SMT are based on the best-first search (E.g. A*). Some examples of decoders are the Beam search algorithm, Greedy decoder, stack-decoding algorithm... and so forth.

C. Neural Machine Translation (NMT) Approach

In the last few years, a new machine translation paradigm has emerged in the MT era, the Neural Machine Translation (NMT), recently proposed by Kalchbrenner and Blunsom [26] also called

data-driven or, less often, corpus-driven machine translation. NMT is the newest approach to machine translation and displaces its corpus-based predecessor, statistical machine translation [6, 27]. It is trained on large pairs of source and target language sentences (corpora), containing huge translation memories of hundreds of thousands or even millions of translation units. It was introduced as a promising approach with the potential of addressing many shortcomings of traditional machine translation systems. The training of the models is similar to phrase-based models, but uses a completely different computational approach: *neural networks* (discussed in section 2.5), a set of algorithms, modeled loosely after the human brain that is designed to recognize patterns [28].

The nodes can hold single words, phrases, or longer segments and relate to each other in a web of complex relationships based on bilingual texts used to train the system. The artificial neural networks are used to predict the likelihood of a sequence of words, typically modeling entire sentences in a single integrated model. The complex and dynamic nature of such networks allows the formation of significantly more educated guesses about the context and therefore the meaning of any word to be translated [6, 29].

In 2014, sequence-to-sequence models were introduced opening new possibilities for neural networks in NLP. Before the seq2seq models, the network needed a way to transform the sequence input into computer-ready numbers (one-hot encoding vector, and embedding) shown in [Annex A Table A.1](#). With seq2seq, the possibility of training a network with input and output sequences became possible [27].

An NMT system is a neural network that directly models the conditional probability $P(y|x)$ of translating a source sentence, x_1, \dots, x_n , to a target sentence, y_1, \dots, y_m . A regular form of NMT has two components: an *encoder*, which calculates a representation for each input sentence, and a *decoder* that generates one target word at a time and hence decomposes the conditional probability as the following equation [19].

$$\log P(y|x) = b + \sum_{j=1}^m \log(y_j | y < j, s) \quad (2.4)$$

NMT systems continuously learn and adjust to provide the best output. [30].

This research is mainly focused on and works with this approach to develop a model for Ge'ez-Amharic Machine Translation and there are different levels of NMT, such as Word Level, sub-

word level, morpheme level, character level, phoneme level, and so on. In this study, the sub-word level is used as it is more efficient and effective in real-world applications and suited to NNs.

The main advantage of NMT is that a single system can be trained directly on the source and target text, no longer requiring the pipeline of specialized systems used in statistical machine learning. This is called the End-to-end model (no pipeline of specific tasks) as only one model is required for the translation. In NMT, word alignment, the cornerstone of SMT, is no longer needed as an input of the system. Current Neural MT engines can extract word alignment from the attention weights [31].

On the other hand, the major problem with neural networks occurs if the training data is unbalanced, the model cannot learn from the rare samples as well biased to frequent ones. This problem is known as the rare word problem. Besides, the limited vocabulary is due to computational constraints. These models are trained with, say, 50,000 word-vocabularies, and any longer words are broken up into word pieces. This is a real problem for deployments with large numbers of brand names or large specialized vocabulary. Neural systems are very hard to debug than SMTs and the errors produced by neural systems are sometimes quite capricious. The other disadvantage is that the biasing of the model towards frequent words on the corpus. A recent paper proposes a solution using a post-processing step to translate the rare words with a dictionary [32]. In Addition, it requires a large bilingual corpus like SMT, a very high resource (time and processing power), that needs a special type of processing units like GPU and TPU which is not easily able to run on the regular CPU processors. However, some generous companies such as Google and NVIDIA thanks to them freely provide these resources.

Challenges of NMT

Many scholars have accused that NMT systems lack robustness, especially when input sentences contain rare words and it is more sensitive than SMT to noises. Even though accuracy and speed are essential, these issues have stuck NMT to be used in practical deployments and services. According to Koehn et.al [33], and many other researchers Neural Machine Translation has the following six challenges.

Domain mismatch

Neural Machine Translation systems have a lower quality out of domain, to the point that they completely sacrifice adequacy for the sake of fluency. Large amounts of training data are only available out of domain, but it is expected to have robust performance.

Amount of training data

Neural Machine Translation requires a very huge amount of training data than SMT to be more fluent. NMT systems have a sharper learning curve for the amount of training data, resulting from poor quality in low-resource settings, but better performance in high resource settings. However, it is tough to get a training corpus with the sizes of millions of words.

Rare words

Although NMT systems outperform SMT systems on the translation of very infrequent words, but still challenging for it too. NMT models perform particularly poorly on rare words like in highly inflected categories (such as verbs). This issue was addressed by Google's researcher [32].

Long sentences

NMT systems had lower translation quality on very long sentences but do comparably better up to a sentence length of about 60 words. The introduction of the attention model remedied this problem somewhat, but not fixed the problem completely until the transformer models come into existence. While overall NMT was better than SMT, the SMT system outperformed NMT on sentences of length 60 and higher. Quality for the two systems is relatively close, except for the very long sentences (80 and more tokens). However, now different Transformer pre-trained embedding models such as Bidirectional Encoder Representations from Transformers (BERT) resolved this issue¹⁰ [32].

Word alignment

Attention can do some word alignment especially in attention-based NMTs; however, this attention model for NMT does not always fulfill the role of a word alignment model, but may dramatically diverge. The attention model was the imposition of an alignment of the output words to the input words. However, it may nearly fail to handle word alignment for some NMT models. To solve the alignment problem of any attention mechanism Li, Xintong, et al. [34] found that *Alignment by Explicit Alignment Model* and *Alignment by Prediction Difference*, which is agnostic to specific NMT models. The researcher said the prediction difference is better for understanding

¹⁰ <https://github.com/UKPLab/sentence-transformers/issues/364> accessed April 2021

and visualizes NMT from word alignment. Again the Transformer models are advisable than the standard attentions with SeqSeq models on handling alignments.

Beam search

In many cases, poor translations are found beyond an optimal beam size setting. Optimal beam sizes are between 30 -50 in many cases (sometimes up to 200), however, quality is still drop in larger beams and NMT only improves translation quality for narrow beams and is bad when exposed to larger search space.

Even though these six issues are the main challenges of NMT, in 2016 Google [28] presented Google's Neural Machine Translation (GNMT) system, which attempts to address these issues, particularly the rare word problems, and most of these problems are not issues anymore.

Hardware and software requirements of NMT

Choosing hardware is just as crucial as the software and algorithms because the training of neural networks consists of a large number of matrix multiplications, which needs parallel computing power. The usual CPUs cannot process this much computation in a short time. It may take months or even years to train the model. To minimize the training time processing units with parallel computing power such as Graphics Processing Units (GPUs) and or Tensorflow Processing Unit (TPU) are required. Google and NVIDIA made the current advances in artificial neural networks possible by providing GPUs and TPUs. GPUs are constructed specifically for large-scale parallel computing and matrix multiplications.

D. Hybrid Machine Translation Approaches

The above individual approaches have their own shortcomings, and many hybrid machine translation approaches have been proposed. The three main categories of hybrid systems are:

- ❖ Rule-based engines using statistical translation for post-processing and cleanup,
- ❖ Statistical systems guided by rule-based engines.
- ❖ Either of the above with some input from the Neural Machine Translation system.

In the first case, the text is translated first by an RBMT engine. This translation is then processed by an SMT engine, which corrects any errors made. In the second one, the RBMT engine does not translate the text but supports the SMT engine by inserting metadata (e.g. noun, verb, adjective, present or past tense, etc.). The last one is either of the two is supported by NMT. A summary of all approaches to machine translation is shown below in table 2.1.

Table 2.1 General Comparison of Machine Translation Approaches

MT	Advantages	Drawbacks
Rule-Based Machine Translation (RBMT)	<ul style="list-style-type: none"> ❖ Fine for the translation of small content volumes ❖ Provides consistent translation quality for short sentences and a fixed set of terminology data. ❖ A good engine can translate a wide range of texts (Domain-independent) ❖ Only the dictionary, no bilingual text required. ❖ Total control (a possible new rule for every situation) ❖ Reusability (existing rules of languages can be transferred when paired with new languages) 	<ul style="list-style-type: none"> ❖ Requires extensive proofreading & experts ❖ heavy dependence on lexicons and rules ❖ rules of language change through time and need to be constantly updated ❖ Time-consuming and labor-intensive, may take several years for one language pair. ❖ human-encoded rules are unable to cover all possible linguistic phenomena ❖ Conflicts between existing rules may lead to poor translation ❖ Don't deal well with slang or metaphorical texts. ❖ Requires good dictionaries ❖ Lack of fluency [1] ❖ The more the rules the harder to deal with the system
Example-Based Machine Translation (EBMT)	<ul style="list-style-type: none"> ❖ It May result in high-quality translation when highly similar examples are found. 	<ul style="list-style-type: none"> ❖ A large set of high-quality training data is required. ❖ When there is no similar example found, the translation quality may be very low.
Statistical Machine Translation (SMT)	<ul style="list-style-type: none"> ❖ Make most sense when needed to translate in high volumes, such as technical manuals. ❖ Training data is widely available on the Internet ❖ Eliminates the need to handcraft a translation engine for each language pair and create linguistic rule sets, as is the case with RBMT ❖ Requires less virtual space than other prior models of MT 	<ul style="list-style-type: none"> ❖ Requires very large, well-organized, and high-quality bilingual and monolingual corpora for each language pair. ❖ Fail when presented with texts that are not similar to material in the training corpora. ❖ Unable to translate idioms and marketing material
Hybrid Machine Translation (HMT)	<ul style="list-style-type: none"> ❖ Better translation quality ❖ Combination advantage of two or more MTs 	<ul style="list-style-type: none"> ❖ Need for extensive editing. ❖ More Complex work than single MTs ❖ Human translators will be required
Neural Machine Translation (NMT)	<ul style="list-style-type: none"> ❖ The most advanced option ❖ Iteratively learn and adjust weights to provide the best output ❖ Provides a single system that can be trained to decipher the source and target text. ❖ Provides translations that are much more fluent and readable than other MTs. 	<ul style="list-style-type: none"> ❖ A very large set of high-quality training data is required. ❖ Training models for NMT is an expensive affair ❖ Require a lot of processing power ❖ Encounters difficulties when faced with highly technical language, or the use of rare words and proper nouns. ❖ More sensitive to corpus quality and hyperparameters than SMT

All Corpus-based MTs require adequate and clean bilingual corpora. The more the data, the improved the quality translation.

2.4 Artificial Neural Networks

Artificial Neural Network (ANN) simply called Neural Network (NN) is an efficient computing system whose structure is derived from the analogy of biological neural networks. It is composed of thousands of units called artificial neurons¹¹, which are the fundamental piece of deep learning algorithms [6]. Each neuron takes inputs from numerous other neurons, multiplies them by assigned weights, adds them, applies an activation function, and passes the sum to one or more neurons. Neural Machine Translation uses these artificial neural networks. Figure 2.4 shows the general model of neural networks and their process.

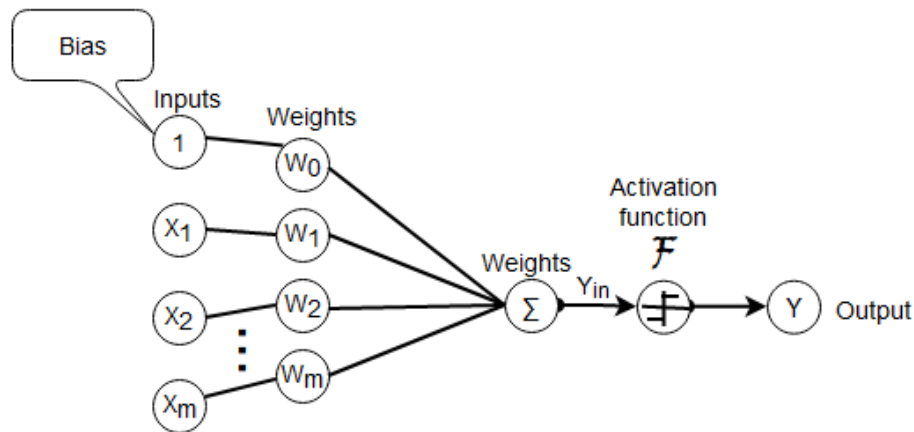


Figure 2.4 the general model of ANN followed by its processing¹²

For Figure 2.4 the net input of the artificial neural network can be calculated as:

$$y_{in} = x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 \dots x_m \cdot w_m + b \cdot w_0 \text{ (} b \text{ is always 1)}$$

$$\text{i. e., Net input } y_{in} = \sum_i^m x_i \cdot w_i + b \cdot w_0 \quad (2.5)$$

This output can be calculated by applying the activation function \mathcal{F} over the net input. $\gamma = \mathcal{F}(y_{in})$
ANNs have different components such as neural unit (neuron), Learning Weights (weight matrices or Interconnections), Layers (The input, the hidden, and the output layers), Bias, Activation Functions (threshold or transformation), and Loss functions, and so on¹³.

¹¹ Named after the neurons in a biological brain

¹² <https://wiki.pathmind.com/neural-network> Accessed on Jan 2021

¹³ <https://otexts.com/fpp2/nnetar.html>, <https://wiki.pathmind.com/neural-network> Accessed on Jan 2021, https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_basic_concepts.htm Jan 2021

2.4.1 Deep neural networks

The deep neural network is also known as *deep neural learning* or *deep learning* is a subset of machine learning methods in artificial intelligence (AI) that imitates the workings of the human brain based on artificial neural networks that are capable of learning unsupervised from data that is unstructured or unlabeled. [7].

Even though there is no common universal convention about the number of layers to be called deep learning, it is distinguished from the more common one single-hidden-layer neural networks by its *depth*; that is, the number of node layers via which data must pass in a multistep process of pattern recognition. However, representations are usually deep (hence the buzzword deep learning): they are not created in a moment phenomenon, but stages from other shallower representations or layers. These layers may usually contain hundreds of neural units and the number of connections ranges in the thousands. Hence, it is important to raise questions like “*What is the minimum number of layers in a deep neural network?*” or “*At which depth level does Shallow Learning end, and Deep Learning begin?*”

Most researchers in the field agreed that deep learning has multiple nonlinear layers. Mikel L. Forcada [6] and Hinton et al. [35] though most of the earlier versions of neural networks such as the first perceptron are shallow, which are composed of one input, one output, and at most one hidden layer. The researchers said there for *more than three* layers with input and output layers is called “*deep*” learning. So deep is not just arbitrary and a buzzword to make algorithms seem like for the down and too hard to understand. It is a strongly defined term with the meaning of *more than one hidden layer*. In deep learning networks, each layer of nodes trains on a distinct set of features based on the previous layer output. The more layers the neural nets have, the more complex the features the nodes can recognize. A Multi-Layer-Perceptron (MLP) with four or more layers (including input and output) is called a Deep Neural Network [6, 9, 29]. In addition to its name, a deep neural net has *three levels of depth* (*Deep, very deep, and extremely deep*)

Deep: According to Hinton et.al [35] one of the earliest deep neural networks has *three* densely connected hidden layers. After fine-tuning, a network with three hidden layers forms a very good generative model of machine translation tasks.

Very deep: According to Schmidhuber [36] considers Depth of Credit Assignment Paths (CAPs)¹⁴ > 10 to be very deep learning.¹⁴ Whereas Simonyan et.al said, a *very deep* neural network has at least 16 hidden layers [37].

Extremely Deep: He et al. In 2016, the *extremely deep* residual networks consist of 50 up to 1000+ hidden layers [38]. Again, Schmidhuber [36] said both Feedforward Neural Networks (FNNs) (acyclic) and recurrent neural networks (RNNs) (cyclic) have won competitions. In a sense, RNNs are the deepest of all NNs¹⁵ they are general computers more powerful than FNNs, and can in principle create and process memories of arbitrary sequences of input patterns

As discussed above, though there is no universal agreement upon the threshold of depth separating shallow learning from deep learning, most researchers said a deep neural network shall call *deep* if the net has more or equal to 2 hidden layers. Hence, in this study, we used this concept to call deep learning. A network with two or more hidden layers is a *deep neural network* or *deep learning*. The more hidden layers the more the model learns better and predicts correct translation.

2.4.2 Different types of Neural Networks

There are numerous types of neural networks, such as Feedforward Neural Networks (FNN), Back Propagation (BP), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Attention and Transformers.

The *Feedforward Neural Networks* are also known as *multilayer perceptron* or *deep feedforward networks* where the connections between nodes or layers do not form a cycle. FNNs were the first type of ANN invented and are simpler than recurrent neural networks, made from FNNs. They are called feedforward because information only travels forward in the network (no back loops), from input nodes via hidden nodes if have, to output nodes¹⁶.

Back Propagation short for "*backward propagation of errors*" is the process of updating the weights and biases of the neurons based on the error at the output. Which is the method of fine-tuning the weights of a neural net based on the error rate obtained in the former output (in iteration). Proper tuning of the weights allows the model to reduce error rates and to make it reliable by

¹⁴ A chain of transformations from input to output is a Credit Assignment Path or CAP. For a feedforward neural network, the depth of the CAPs, and thus the depth of the network, is the number of hidden layers plus one.

¹⁵ Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun *Deep Residual Learning for Image Recognition* (page 4)

¹⁶ Feedforward Neural Networks. Brilliant.org. Retrieved 11:35, June 3, 2020, from <https://brilliant.org/wiki/feedforward-neural-networks/> by John McGonagle, José Alonso García, Saruque Mollick

increasing its generalization. It is a standard method of training artificial neural networks. This method helps to calculate the gradient of a loss function for all the weights in the network.

Convolutional Neural Network (CNN) or (ConvNet) is a deep neural network for images modeled after the human visual cortex and originally employed in the field of computer vision [14]. This Learning algorithm takes in an input image, assign weights and biases to various objects in the image, and makes an explicit assumption to classify one from the other. Unlike regular ANN and deep neural nets, CNNs are based on having the neurons arranged in 3D (width, depth, and height)¹⁷ [37].

Now a day CNNs are used on different NLP applications such as Machine Translation. The Facebook AI research published a paper that shows CNNs are better than RNN architectures for Machine Translation. CNN-based MT architectures work similarly to CNN on images. In which sentences are treated as 1D images. These models are faster and have achieved higher results than RNN models for translation tasks because of parallelization [39]. Other NNs are discussed below.

(a) Recurrent Neural Network (RNN):

RNNs are a generalization of feed-forward neural networks that has an internal memory with loops designed specifically to deal with textual data like machine translation, text summarization, sentiment classification, image captioning...focusing on *temporal dependency*, or *dependencies over time*¹⁸. The output from the previous step is fed as input to the current step. This means the current output depends not only on the current input but also on past inputs, (the result is dependent on previous n time steps). They have a shared weight and a *memory*¹⁹ that takes information from prior inputs to influence the current input and output. RNNs are universal also known as Turing complete and used for mapping inputs to outputs of varying types, lengths and are fairly generalized in their application. RNNs take two inputs at each time step; an input (in the case of the encoder, one word from the input sentence), and a hidden state. The next recurrent neuron takes the second input vector and hidden state 1 to create the output of state 1. This is continuously done in a Recurrent Neuron. A *recurrent neuron* is a single neuron that stores the state of a previous input and combines it with the current input by preserving its relationship [40]. Recurrent neural networks power **Backpropagation Through Time (BPTT)** algorithm to determine the gradients,

¹⁷ From [DesireCourse.Net] Udemy - Tensorflow and Keras For Neural Networks and Deep Learning

¹⁸ <http://karpathy.github.io/2015/05/21/rnn-effectiveness/> Accessed on Jan 2021

¹⁹ Memory: is an internal state of RNN used to process sequences of inputs. Not in feed-forward neural networks.

which is somewhat different from traditional backpropagation as it is specific to sequence data. In BPTT, the model trains itself by calculating errors from its output layer to its input layer.

RNNs suffer from two problems, known as *exploding gradients* and *vanishing gradients* [41]. These issues are caused by the size of *the gradient*²⁰, which is the slope of the loss function along the error curve.

The *vanishing gradient*: problem happens when the gradient is too small, it continues to become smaller, updating the weight parameters until they become zero or insignificant. This makes the learning of long data sequences difficult or the algorithm is no longer learning. For instance in a sentence like “The man who ate my Injera has white hair”, the description of *white hair* is for the man and not the Injera. Hence, this is a long dependency. If the error were back propagated in this case, it would need to apply the chain rule. After applying the chain rule and if any one of the gradients approached 0, all the gradients would rush to zero exponentially fast due to the multiplication. Such states would no longer help the network to learn anything.

Exploding gradients: On the other hand occur when the gradient is too large, creating an unstable model. In this case, the model weights will grow too large, and they will eventually be represented as NaN. The vanishing gradient problem is far more threatening as compared to the exploding gradient problem. Why because the Vanishing gradient problem is more concerning is that an exploding gradient problem can be easily solved by clipping the gradients at a predefined threshold value. To handle the *vanishing gradient* problem, other variants of RNN such as the LSTM and the GRU are created.

(b) Long Short Term Memories:

LSTMs are a special kind of RNNs and are capable of learning *long-term dependencies* by remembering information for longer periods, that is their default behavior. RNNs suffer from *vanishing gradient* problems when they are asked to handle *long-term dependencies*. For example in a sentence, “I have been staying in the Amhara region for the last 6 years. I can speak _____fluently”

The word it predicts will depend on the previous few words in context. Here it needs the context of Amhara to predict the missed word in blank space, and the most suitable answer to this sentence is “Amharic.” In other words, the gap between the relevant information and the point where it is needed may have become very large. Vanishing and exploding gradients make RNNs unusable.

²⁰ The gradients carry information used in the RNN

LSTMs were then introduced by Hochreiter & Schmidhuber [42] in 1997 to overcome this problem by explicitly introducing a memory unit, called the *cell* into the network. They work very well on many different problems and are still widely using.

A common LSTM unit is composed of a *cell*, an *input gate*, an *output gate* and a *forget gate*²¹. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell [43]. The *Cell State Vector (memory cell)* represents the memory of the LSTM and it changes the forgetting of old memory (forget gate) and the addition of new memory (input gate). The *Forget Gate Control* (decide) what information to throw away from the cell state (memory) and Decides how much of the past info it should remember. It looks at h_{t-1} and x_t , and outputs a number between 0 and 1 for each number in the cell state C_{t-1} . A 1 represents completely keep this while a 0 represents completely get rid of this.

$$\text{Forget layer----- } f_t = \sigma(w_f \cdot [h_{t-1}, x_t] + b_f) \quad (2.6)$$

The *Input Gate (Update)* controls what new information is added to the cell state from the current input and decides how much of this unit is added to the current state. This has two parts. First, the input gate layer (a sigmoid layer), decides which values it will update and the next is a tanh layer creates a vector of new candidate values, \hat{C}_{t-1} that could be added to the state and combine these two to create an update to the state.

$$i_t = \sigma(w_i \cdot [h_{t-1}, x_t] + b_i) \quad (2.7)$$

$$\bar{C} = \tanh(w_C \cdot [h_{t-1}, x_t] + b_C) \quad (2.8)$$

Updating the old cell state, C_{t-1} , into the new cell state C_t is just by multiplying the old state, C_{t-1} by f_t , and adding $i_t \times \hat{C}_t$ and forgetting the things it decided to forget earlier. This is the new candidate value, scaled by how much it decided to update each state value.

$$\text{Updating the old state cell----- } C_t = f_t * C_{t-1} + i_t * \bar{C}_t \quad (2.9)$$

The *Output Gate* on the other hand conditionally decides what to output from the memory. First, it runs a sigmoid layer, which decides what parts of the cell state it is going to output. Then, it put the cell state through tanh (to push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that it only outputs the parts it decided to.

$$\begin{aligned} \text{output Gate----- } o_t &= \sigma(w_o [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned} \quad (2.10)$$

²¹ <http://colah.github.io/posts/2015-08-Understanding-LSTMs/> Accessed on Feb 2021

(c) GRU (Gated Recurrent Unit):

It is a variant of LSTMs but is simpler in its structure and is easier to train. It combines the ‘forget’ and ‘input’ gates into a single *update gate*²². It also merges the *cell state* and *hidden state* then makes some other changes. These gates have their own sets of weights that are adaptively updated in the learning phase. GRU has two gates, the *reset*, and the *update gate*.

The *Update gate* helps the model to determine how much of the past information (from previous time steps) needs to be passed along to the future. That is powerful because the model can decide to copy all the information from the past and eliminate the risk of the vanishing gradient problem.

$$z_t = \sigma(w^{(z)}X_z + u^{(z)}h_{t-1}) \quad (2.11)$$

The *Reset gate* is used to decide how much of the past information to forget. The formula is the same as the update gate. The difference is in their weights and the gate’s usage

$$r_t = \sigma(w^{(r)}X_z + u^{(r)}h_{t-1}) \quad (2.12)$$

The *Current memory content* will determine what to remove from the previous time steps

$$h'_t = \tanh(wx_t + r_t \odot uh_{t-1}) \quad (2.13)$$

The *Final memory at the current time step* holds information for the current unit and passes it down to the network.

(d) Encoder-Decoder (Sequence-to-Sequence Models)

One of the older and more established versions of NMT is the Encoder-Decoder structure. This architecture is composed of two recurrent neural networks (Mostly LSTMs and GRUs) used together in tandem to create a translation model Easier. RNN Encoder-Decoder has been proposed by Cho et al [44] and Sutskever et al. [27]. A sequence-to-sequence (Seq2Seq) model aims to map a fixed-length input with a fixed-length output where the length of the input and output may differ. In the Encoder-Decoder structure, an encoder reads the input sentence, a sequence of vectors $x = (x_1, \dots, hT_x)$, into a vector c [45]. The most common approach is to use an RNN such that

$$h_t = f(x_t, h_{t-1}) \text{ And } c = q(\{h_1, \dots, hT_x\}) \quad (2.14)$$

Where $h_t \in \mathbb{R}^n$ is a hidden state at time t , and c is a vector generated from the sequence of the hidden states. f and q are some nonlinear functions. Sutskever et al. [27] used an LSTM as f and $q(\{h_1, \dots, hT\}) = h_T$, for example.

For translating a sentence “ዘሬ ምን የምትሰራው አለህ?” (*Zarē min yemitiserawi ālehi?*) From Amharic to Ge’ez has an input of 4 words and an output of 5 words “የም ምንት ውእቱ ሀለክ ዘትገብሮ” (*yomi*

²² <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be> Accessed on Feb 2021

miniti wi'itu hāloke zetigebiro) meaning “*What are you doing today?*” A regular LSTM cannot be used to map each word from the Amharic sentence to the Ge’ez sentence.

This is why the Seq2Seq model is used to address problems like that one. Generally, the encoder encodes the input sequence to an internal representation called *context vector*, which is used by the decoder to generate the output sequence. The lengths of input and output sequences can be different, as there is no explicit one-to-one relation between the input and output sequences.

(e) The Attention Mechanism:

The main weakness of a fixed-length *context vector* (thought vector) of Seq2Seq design is the incapability of remembering *longer* sequences (no > 20-time steps)²³ because only the last hidden state of the encoder RNN is used as the context vector for the decoder. Often it will forget the earlier parts of the sequence once it has processed the entire sequence. The attention mechanism came to resolve this problem and was first proposed by Bahdanau et al [45] in 2015. It is just selectively concentrating on a few relevant things. Hence, this process of searching for a set of positions in the encoders’ hidden states, where the most relevant information is available is named an *Attention*. Therefore, *Attention* is memory through time.²⁴

In the traditional Seq2Seq model, outputs (Y_i) of the Encoder at each time step²⁵ all the intermediate *states* of the encoder are discarded and use only its final state vector to initialize the decoder. This technique works well for smaller sequences, though as the length of the sequence increases, a single vector becomes trouble and it gets very hard to summarize long sequences into a single vector. As well, the performance of the system decreases drastically as the size of the sequence increases. The Attention Mechanism directly addresses this issue as it recalls and utilizes all the hidden states of the input sequence during the decoding process by creating a unique *mapping* between each time step of the decoder output to all the *hidden states* of the encoder. Therefore, in general, the central idea behind Attention is to utilize all the *intermediate states of the encoder* to construct the context vectors required by the decoder to generate the output sequence. This means, for each output of the decoder, it has access to all input sequences and can selectively pick out a specific element from that sequence to produce the output. In other words,

²³ <https://theaisummer.com/attention/> Accessed on Feb 2021

²⁴ www.limetorrents.info/ DeepMind’s deep learning videos 2020 with UCL, Lecture: Attention and Memory in Deep Learning, Alex Graves

²⁵ <https://towardsdatascience.com/intuitive-understanding-of-attention-mechanism-in-deep-learning-6c9482aecf4f> Accessed on Feb 2021

the Attention mechanism has an infinite reference window to reference from. It is developed to learn word mappings through Gradient Descent and Back-propagation. Bahdanau et al [45] stated that in a decoder, each conditional probability:

$$p(y_i|y_1, \dots, y_{i-1}, x) = g(y_{i-1}, s_i, c_i) \quad (2.15)$$

Where s_i is an RNN hidden state for the time i , computed by $s_i = f(s_{i-1}, y_{i-1}, c_i)$. Unlike in the traditional encoder-decoder approach, here the probability is conditioned on a distinct context vector c_i for each target word y_i . This is depicted in the following figure (Figure 2.5).

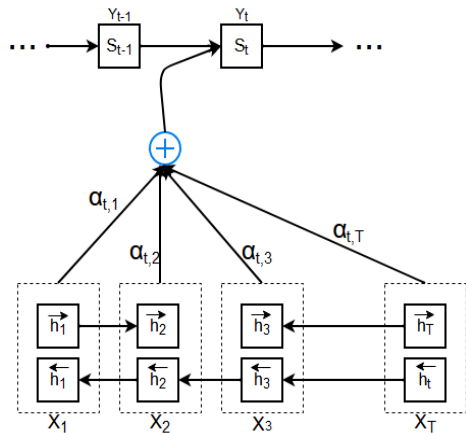


Figure 2.5 Diagram of the Attention model shown in Bahdanau's paper [45]

The embedding of all the words in the input (represented by hidden states) while creating the context vector is done by simply taking a weighted sum of the hidden states. A feed-forward neural network learns the weights and the context vector c_i for the output word y_i is generated using the weighted sum of the annotations: as shown in equation 2.16.

Where c is a weighted sum of the encoder-hidden states, α_{ij}

$$C_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \quad (2.16)$$

is the amount of attention the i^{th} output should pay to the j^{th}

input and h_j is the encoder state for the j^{th} input.

The weight α_{ij} of each annotation h_j is computed by a Softmax function given by the equation:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \quad (2.17)$$

Where $e_{ij} = \alpha(S_{i-1}, h_j)$ and α are alignment models which score how well the inputs around position j and the output at position i match, and s_{i-1} is the hidden state from the previous time step [46]. An attention model differs from a classic Seq2Seq model in two main ways. First, the encoder passes all the *hidden states* to the decoder instead of passing the last hidden state of the encoding step. Second, the decoder focus on the parts of the input that are relevant to the decoding time step, based on these three steps:

1. Look at the set of encoder hidden states it received, since each encoder hidden state is associated with a word in the input sentence
2. It gives each hidden state a score

3. Multiplies each hidden state by its Softmaxed score, thus accepting hidden states with high scores, and dropping out hidden states with low scores.

(f) *The Transformer:*

The Google-led team first introduced the transformer architecture in 2017 in a paper titled “Attention Is All You Need” [47], which is currently the state-of-the-art methodology on Machine Translation and even on other non-NLP fields such as computer vision [14]. It is built on the top of the attention mechanism, essentially a stack of *encoder* and *decoder* layers (based on the paper, six of them are on top of each other). Both the Encoder and Decoder are composed of modules that can be stacked on top of each other multiple times, which is labeled by N_x in Figure 2.6. Those modules consist mainly of Multi-Head Attention and Feed Forward layers.

The goal of the *transformer* is to change the sequential nature of the encoder to parallelization. Quoting from [47] “*The Transformer is the first transduction²⁶ model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution*”. Hence, the idea behind Transformer is to handle the dependencies between input and output with attention completely. Which is that the word in each position flows through its path in the encoder. A transformer implements end-to-end training, similar to the sequential encoder-decoder architecture, which has both encoder and decoder however, the transformer encoder-decoder elements are self-attention mechanisms rather than RNN or CNN. Besides, the transformer adds positional encoding to capture the relationship between consecutive words within a sentence. All sub-modules of the transformer are discussed below and Figure 2.6 shows the overall architecture.

²⁶ Here, “transduction” means the conversion of input sequences into output sequences.

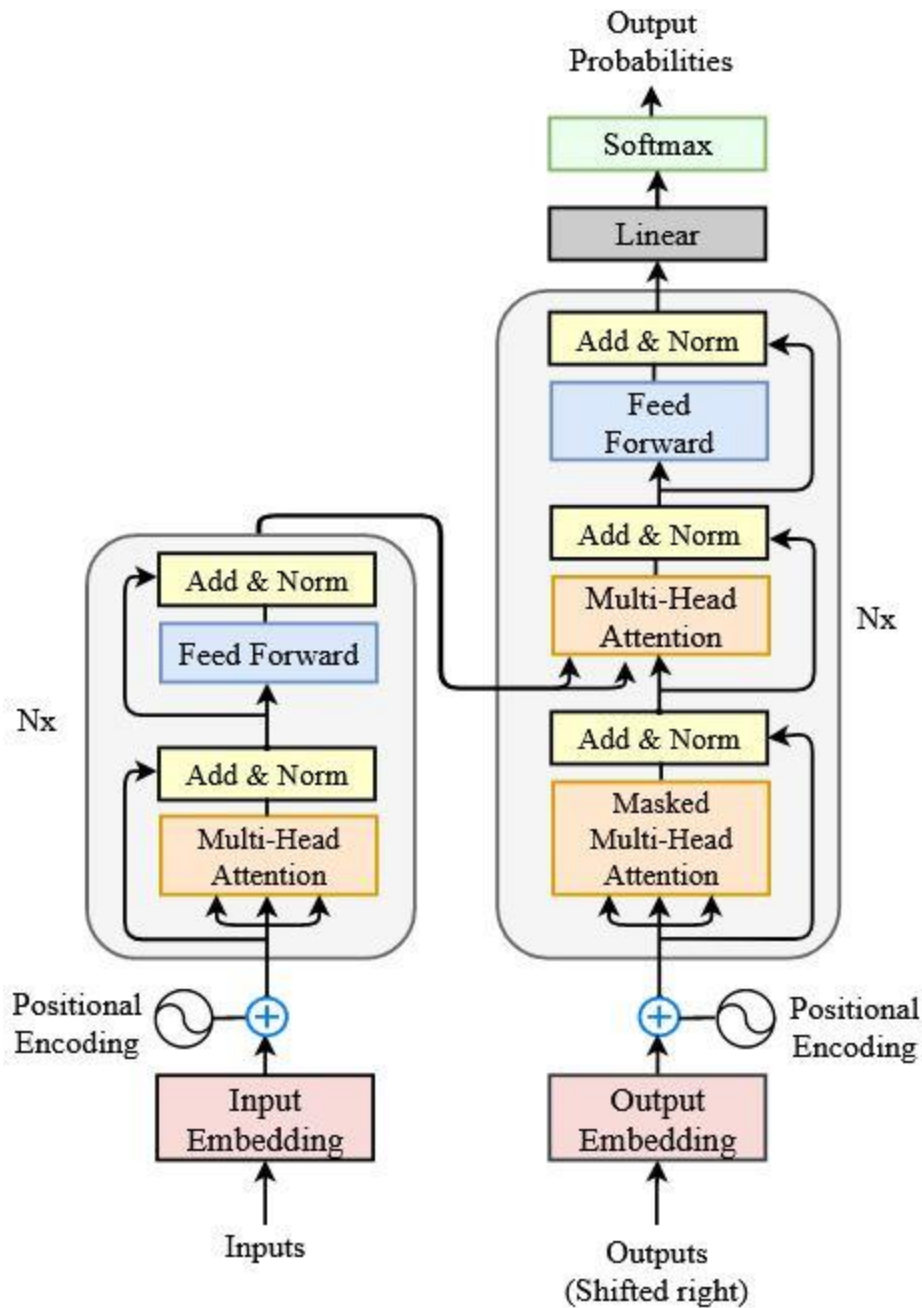


Figure 2.6 Architecture of Transformer Model adopted from [47]

a) Input

The Transformer takes a sequence of words as input, which are presented to the network as vectors. It uses usually a vocabulary (dictionary), in which each word is assigned a unique index. The index can be represented as a so-called one-hot-Encoding vector, which is predominantly made up of zeros, with a single “one” value at the correct location of a word.

b) Input or output embedding

Word embedding reduces the dimensionality of the one-hot encoded vectors by multiplying them with a so-called “embedding matrix”. The resulting vectors are called word embedding. The size of the word embedding in the original paper is 512” [47]. From pre-trained language models, such as word2vec, BERT, and RoBERTa, the transformer architecture gets additional information about the input with the help of embedding layer

c) Positional encoding (PoE)

Positional encoding is a set of small constants, which are added to the word-embedding vector before the first self-attention layer. It is a supplement to the transformer besides the embedding. Since the transformer does not have recurrence and convolution, it adds positional encoding into embedding instead. Positional information enhances the transformer representation to capture the relationship between the token in the sequence by considering word order. Both the embedding and positional encoding have the same dimension d , which lets them sum the result. As per [47] sine and cosine functions are used to calculate positional embedding for the different positions:

$$\begin{aligned} PE_{pos,2i} &= \sin(pos/10000^{\frac{2i}{D}}) \\ PE_{pos,2i+1} &= \cos(pos/10000^{\frac{2i}{D}}) \end{aligned} \tag{2.18}$$

Where pos and D represent the *position* and the *input dimension* respectively. PoE will generate a 512 dimension vector for each position, as well, the even and odd dimensions use *sin* and *cos* functions respectively. The value of PoE range between -1 and 1.

d) The Encoder

Encoders of the transformer are built on top of the self-attention. The job of the encoder layers is to map all input sequences into an abstract continuous representation that grasps the learned information for that entire sequence. The encoders are all identical in structure but they do not share the same weights. Each encoder in the stack has two main layers: (i) A multi-head self-attention Layer, and (ii) A position-wise fully connected feed-forward network. There are also residual connections around each of the two sublayers followed by a layer normalization. After embedding the words in the input sequence, each of them flows through each of the self-attention and the feed-forward layers of the encoder.

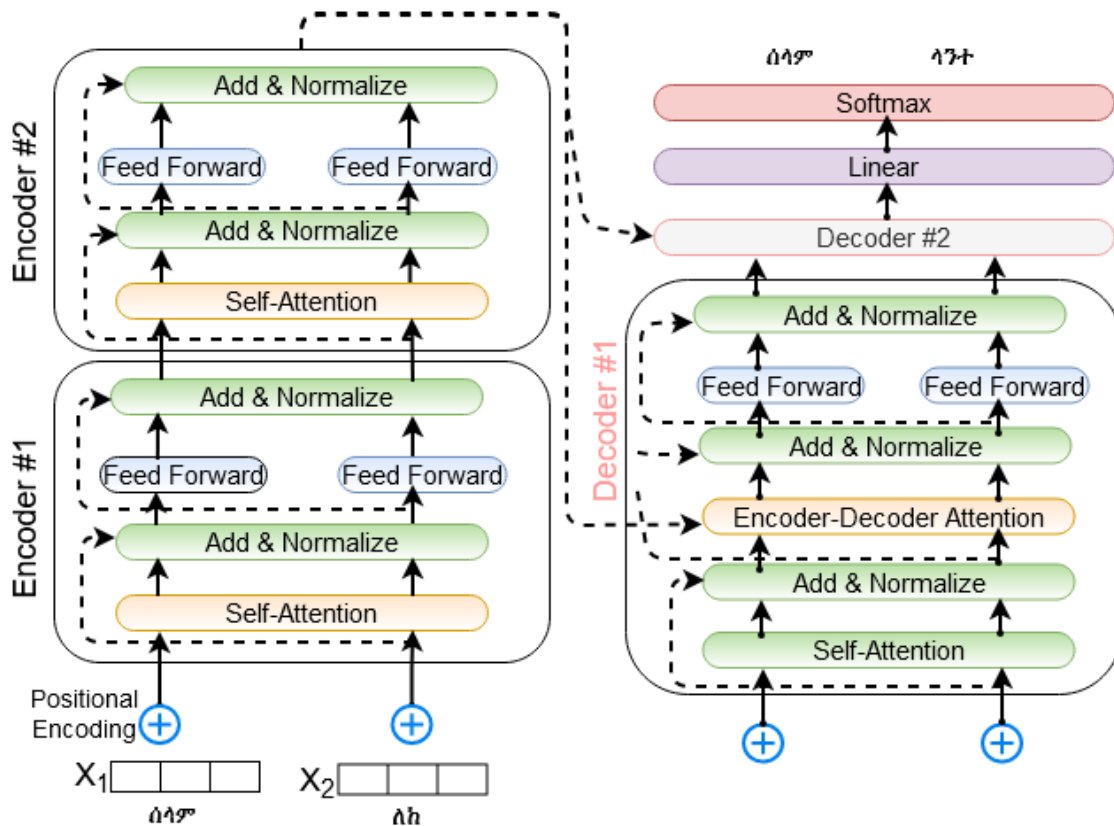


Figure 2.7 a Transformer of 2 stacked encoders and decoders²⁷

The Multi-Head Self Attention

Multi-headed attention in the encoder applies a specific attention mechanism called self-attention. Self-attention allows the model to associate each word in the input, with other words. There are dependencies between unique paths of words in the self-attention layer. However, the feed-forward layers do not have those dependencies, and thus the several paths can be executed in parallel. Self-attention is a layer that receives input and helps the encoder look at other words in the input sentence. It accepts a sequence of vectors and a results sequence of vectors. The outputs of the self-attention layer are fed to a feed-forward neural network. Here are some steps to calculate self-attention.

²⁷ <https://jalammr.github.io/illustrated-transformer/> Accessed on Feb 2021

Step-1: Create three vectors from the embedding of each word, named a *Query (Q)* vector, a *Key (K)* vector, and a *Value (V)* vectors²⁸ for each word. These vectors are created by multiplying the embedding by three matrices that were trained during the training process.

These new vectors have a smaller dimension than the embedding vector that is the *Q, K, V* vectors, and the embedding vector has a size of 64 and 512 respectively.

Step-2: Calculate a scoring matrix, which determines how much focus should a word be put on other words of the input sentence. The higher the score the more focus. The *Score matrix = the dot product of the Q vector and the K vectors*. Hence, if we are applying the self-attention for the word in position 1, the first score would be $q_1.k_1$ and the second score would be $q_1.k_2$... and so forth.

Step-3: Scaling Down the Attention Scores by divide the scores by the square root of the dimension of the key vectors (8 in the paper [47])²⁹. This leads to having more stable gradients as multiplying values can have exploding effects.

Step-4: Softmax of the Scaled Scores to get the attention weights. Softmax normalizes the scores between 0 and 1. By doing a Softmax the higher scores get high attention, and lower scores get lower. This allows the model to be more confident about which words to attend more.

Step-5: Multiply Softmax score with Value vector to get an output vector (and to sum them up). The intuition here is to keep words with the higher Softmax scores and to drown out the irrelevant words with lower scores (by multiplying them by tiny numbers like 0.001, for instance).

Step-6: Sum up the weighted value vectors. This produces the output of the self-attention layer at this position (for the first word). Then the output of that will be feed into a linear layer to process. The final equation for the scaled dot product attention is shown on Eq 2.19 and its figure is depicted in Figure 2.8

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.19)$$

²⁸ The *query, key* and *value* concept come from retrieval systems. For example, when a *query* is given to a Youtube, it will map the query to a set of *keys* (video title, description etc.) associated with candidate videos in the database, and then displays the best-matched videos (*values*).

²⁹ The square root of the dimension 64. There could be other possible values, but this is the default.

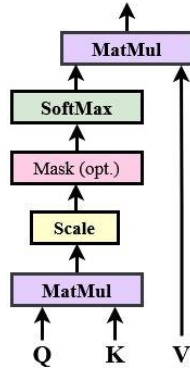


Figure 2.8 Scaled dot product attention adopted from [47]

The split vectors (Q, K, and V) go through the self-attention process individually. Each self-attention process is called *head*³⁰. The output vector of every head is concatenated with a single vector before going via the final linear layer multiplied by W^o . Multi-head can be represented as:

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_n)W^o \quad (2.20)$$

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

Where the matrices $W_i^Q, W_i^K,$ and W_i^V are trainable weights, and $Q, V,$ and K are $\in \mathbb{R}^{N \times d}$. Here N is the number of inputs tokens and d is the input dimension. This head learns something different and this gives the encoder more representation power.

The multi-headed attention on the other hand is a module in the transformer network that computes the attention weights for the input and produces an output vector with encoded information on how each word should attend to all other words in the sequence. It is depicted in Figure 2.9.

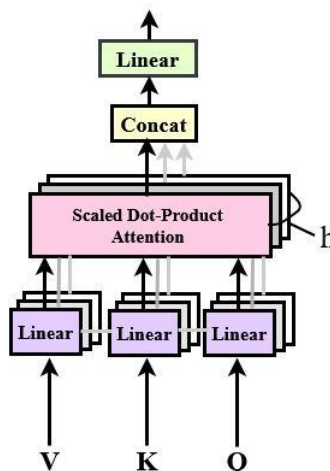


Figure 2.9 Multi-head self-attention adopted from [47]

³⁰ A process were a multi-head attention is named after.

The Pointwise fully connected Feed-Forward Networks

The multi-headed attention output vector is added to the original positional input embedding. This is called a residual connection. The output of the residual connection goes through a layer normalization. The normalized residual output gets projected through a pointwise feed-forward network for further processing. The pointwise feed-forward network is a couple of linear layers with a ReLU activation in between. The residual connections help the network train, by allowing gradients to flow through the networks directly. The layer normalizations are used to stabilize the network, which results in substantially reducing the training time necessary. The pointwise feedforward layer is used to project the attention outputs possibly giving it a richer representation.

e) The Decoder

The decoder is similar to the encoder, which has both self-attention layers and FNNs; however, the decoder adds a masked multi-head attention layer that helps the decoder focus on relevant parts of the input sentence. After the encoder maps an input sequence into an abstract continuous representation, the decoder then takes that continuous representation and generates a single output step by step. Hence, the job of the decoder is to display the translated text.

The inputs of the decoder go through embedding and positional encoding layers. Then the positional embedding is inputted to the first multi-head attention layer, which computes the attention scores for the input of the decoder.

Since the decoder is autoregressive and generates the sequence word by word, it needs to be prevented from looking to future tokens. For example, when computing attention scores on the word “*dehina*”, it should not have access to the word “*Negn*”,³¹ because, that word is a future word that was generated after. The word “*dehina*” should only have access to itself and the words before it. A method that prevents computing attention scores for future words is called *masking*. The mask is a matrix, which has the same size as the attention scores filled with values of 0’s and negative infinities. Therefore, Masking is a process of adding 0’s and negative infinities to attention scores and the future tokens are represented by zeros. Masked multi-head attention allows the model to attend only to the previous word and to prevent the decoder from looking at future tokens. The mask is added before calculating the Softmax, and after scaling the scores.

³¹ Have a meaning of “Fine” and “am” respectively. To say “Am fine” by omitting the subject “I”

The second multi-head attention layer is in charge to map the encoder output and the decoder input to decide which part of the encoder input is relevant to focus on. Thus, the outputs of the encoder (the query, and key) and decoder input are the results of the first multi-head attention layer. The output of the second multi-headed attention passes through a pointwise feedforward layer for further processing. The decoder stack outputs a vector of floats. Hence, how these vectors can turn into a word is a big question and that is the job of the final linear layer, which is followed by a Softmax Layer. The linear layer is a simple fully connected neural network that turns the vector produced by the stack of decoders, into a very larger score vector called *logits*³². The Softmax layer then turns those scores into probabilities (all positive, all add up to 1.0). The cell with the highest probability is selected, and the word associated with it is displayed as the output for this time step. This process is repeated until the decoder produces the last <EOS> token, which shows the end of the sequence. Similar to the encoder layer normalization is applied after each sublayer of residual connection.

2.4.3 Word embedding

Word embedding is a representation of words in the form of real-valued vectors, where words that have the same meaning have a similar representation. First, words are transformed into vocabulary (dictionary, a list of unique words with their corresponding indexes) then converted to one-hot encoding vector, finally to vectors of continuous real value numbers in a predefined vector space. It normally involves a mathematic embedding from a high-dimensional *sparse*³³ vector space (e.g., one-hot encoding vector space, in which each word takes a dimension) to a lower-dimensional *dense*³⁴ vector space. Hence, embedding is a low-dimensional vector that captures a lot of syntactic and semantic information of words and their relationships.

The word-embedding processes a text just like this: First, each word in the vocabulary is decoded in the one-hot encoding. E.g. in the sentence “አንተ ግን ያው አንተ ነህ (*ante gin yaw ante neh*)”, (meaning, “*but you are you*”), the vocabulary (or unique words) are (ነህ, አንተ, ግን, ያው). To create a vector that contains the encoding of the sentence, the one-hot vectors (sample shown in [Annex A](#) [i](#) table A.1) for each word should be concatenated. However, a one-hot encoded vector is *sparse* and inefficient. Hence, we move to the second step, encoding each word using a unique number.

³² A score for each token (unique word) in the vocabulary

³³ A vector where most indices are zero

³⁴ A vector where all elements are fully represented with numbers (no 0's)

Recalling the example above, assign 1 to “ከሆ”, 2 to “አንተ”, 3 to “ግን”, and 4 to “ያው” and so on. We can then encode the sentence “አንተ ግን ያው አንተ ከሆ” as a dense vector like [2, 3, 4, 2, 1]. This approach seems efficient because instead of a sparse vector, a *dense* one is used. However, there are still problems with this approach. The integer-encoding is arbitrary that does not capture any relationship between words and they can be challenging for a model to interpret.

Finally, an embedding is used as a dense vector of *floating-point* values that is a trainable parameter for the model. A sample embedding is depicted on [Annex B, table B.1](#) Word embedding is mostly 8-dimensional for small datasets, up to 1024-dimensions for large datasets, but embedding vectors of size 200 or 300 are usual. A larger-dimensional embedding can capture more relationships between words but takes more data to learn. Different word embedding models are commonly used in NMT and rely on deep learning techniques, such as Word2Vec, BERT, RoBERTa ... etc. Some of these are discussed below.

i. Word2Vec

Word2vec is a predictive model to efficiently create word embedding by using a two-layer neural network. It was first introduced by Mikolov et al. from Google in 2013 [48], which is the most popular word embedding model. It uses shallow neural networks to calculate a word embedding based on the context of the words. The objective function of Word2Vec causes the words that have a similar context to have similar embedding³⁵.

Word2vec is not a single algorithm but a combination of two techniques named Continuous bag of words (CBOW) and Skip-gram models. Both of these are shallow neural networks (have only one hidden layer) which map words to the target variable, which are also words. Both of these techniques learn weights that act as word vector representations.

The CBOW tends to predict the probability of a word given a context (single or a group of words). It uses continuous representations whose order is irrelevant. Instead of feeding n previous words into the model, the model receives a window of n words around the target word w_t at each time step t . Let V is the size of a vocabulary, N is embedding dimension, and $X(w_1, w_2 \dots w_i)$ and w_t are input and output respectively. Each input word w_i and output word w_t is represented as a one-hot vector x and y , based on vocabulary size V . The model starts to learn features by multiplying vector x and word embedding matrix W of size $V \times N$ to produce embedding vector of a given w_i .

³⁵ Sebastian Ruder, "On word embedding - Part 1". <http://ruder.io/word-embeddings-1/>, 2016. Accessed: Feb 2021

The embedding vector outputted from the hidden layer is the average of many contextual words to the target word. The multiplication of the hidden layer and word target matrix W' of size $N \times V$ to produce the one-hot encoded vector y .

The Skip-Gram model on the other hand turns the CBOW's model on its head. Instead of using the surrounding words (context) to predict the center word, it uses the center word to predict the context. In other words, the Skip-Gram predicts the context given words. Although the Skip-Gram and the CBOW models share similar but reverse algorithms, their differences often make one of them superior for a particular task. E.g, numerous context-target pairs are treated as a new observation in a Skip Gram model and are better for the larger data set and vice versa for CBOW.

ii. FastText

FastText, created and released by the AI Research lab of Facebook in 2016 that is a library for efficient learning of word representations (embedding) and sentences. It enables the creation of either unsupervised or supervised learning algorithms to get vector representations of words [49]. Though Word2Vec successfully handles the problem caused by a one-hot encoding vector, it has many limitations. The major problem is that rare (infrequent) words in the training dataset do not map to vectors. This leads fastText to be selected, in that it performs better than Word2Vec and allows rare words to be mapped to vector properly but it takes a longer time³⁶ to train than word2vec. Instead of inputting separate words into the NN (in word2vec), FastText breaks words into various n-grams (sub-words). For instance, the tri-grams for the word “*ቀደስከሙ*” (*kedeskmu*) is “*ቀደስ*”, “*ደስከ*”, and “*ስከሙ*” regardless of the starting and ending of a word boundary. Then the embedding vector for “*ቀደስከሙ*” will be the sum of all n-grams. After training the NN, there will be word embedding for all the n-grams given the training dataset. This lets the fastText to properly-represent rare words since it is very likely that some of their n-grams also appear in other words. In short, fastText is created to overcome the generalization of unknown words. The idea is similar to Word2Vec but the major amendment of fastText is it goes one level deeper to build word embedding.

FastText incorporates character n-grams into the CBOW model. Sub-word embedding uses the principles of morphology that usually improve the quality of representations of rare words. Instead of learning vectors for words directly, fastText represents each word as an n-gram of characters.

³⁶ Because number of n-grams > number of words

A CBOW model is trained to find out the embedding afterwords are represented using n-gram character. This model is a bag of words model with a sliding window over a word and no internal structure of the word is taken into account. The order of the n-grams does not matter as long as the characters are within that window. Every word is decomposed into its character n-grams N and every n-gram n is represented by a vector x_n . The word vector then is just the sum of the two vectors as shown in Equation 2.21.

$$v_w + \frac{1}{|N|} \sum_{n \in N} x_n \quad (2.21)$$

The set of n-grams (N) is limited to 3 to 6 characters that have two main advantages. First, as long as new words have identical characters as known ones, generalization is feasible. Second, less training data is required since more information can be extracted from every bit of text. That is why there are pre-trained fastText models than other embedding algorithms (294 languages in 2021)³⁷ [14].

Contextualized word embedding

A word can have different meanings in different contexts. However, most of the traditional feature-based word embedding techniques represent a word with different contexts (in different sequences) as one generalized representation. They lack contextualized representation then. To resolve this problem, numerous pre-trained models are created. E.g. Peter et al. [31] proposed ELMo word embedding to represent words as the entire input sequence. ELMo trained with large datasets on the bidirectional language model. Though, It is good for representing words from both directions, due to its shallow connection of independently trained language models, the output representations are not rich enough. Hence, researchers provide different alternatives for contextualized word embedding techniques. Most of the models are trained on the concept of transformer architecture. Here some common pre-trained embeddings such as BERT and RoBERTa are discussed.

iii. BERT

BERT stands for Bidirectional Encoder Representation from Transformer, which is a transformer-based pre-trained machine learning technique for NLP developed by Jacob Devlin et.al at Google in 2018 that is trained unsupervised on a large corpus [32].

BERT is used as a word and sentence representation technique by assessing words in a sentence from previous to next or vice versa that makes it a deep bidirectional pre-trained language model.

³⁷ <https://fasttext.cc/docs/en/pretrained-vectors.html> Accessed on Feb 2021

The BERT model is pre-trained to perform masked language modeling and next sentence prediction. BERT masked language model accepts a sequence of words as input and the input context is encoded by multi-head self-attention results a contextualized representation of each word. The principle of the masked language model is used to design BERT, which randomly masks some of the tokens from the given sequence. The masking aims to predict the masked word based on the context. For instance, if we have a sentence “በማለዳው ያላገኘሁህ የት ሄደህ ነው?” (*bemaledawi yalagenyehuhi yeti hēdehi newi?*) then “በማለዳው ያላገኘሁህ [MASK] ሄደህ ነው?” is given as input sequence to the model. The model predicts “yet” in Amharic “የት” as the replacement of the [MASK] word by considering both forward and backward context.

As well, BERT has two alternatives to generate language models. The pre-training and the fine-tuning approaches. The pre-training approach takes an unlabeled dataset, and all parameters are initialized from 0. The fine-tuning approach makes the BERT model task-specific which initializes the model parameters from the pre-trained model and all parameters are fine-tuned on labeled task-specific datasets. BERT achieves a state-of-the-art result in different tasks such as machine translation and question answering even though the above tasks are monolingual.

iv. RoBERTa

RoBERTa stands for Robustly Optimized BERT proposed by Liu et al. [50] with the pretraining Approach intending to improve the performance of Google’s BERT in different tasks by modifying hyperparameters of the model [32]. It modifies key hyperparameters of BERT, removes the next sentence prediction, and training with larger mini-batches and learning rates. From the modifications, RoBERTa permits training on longer sequences for more training time. RoBERTa has a similar architecture with BERT but uses a byte-level BPE as a tokenizer and a different pretraining structure. Moreover, RoBERTa allows dynamically change the masking strategy applied to the training set. The model scores almost competitive results on the downstream tasks with other contextualized word embedding techniques.

2.5 Machine Translation Evaluation Metrics

There are generally two types of MT evaluation metrics, Human and automatic MT evaluation metrics [63]. Human evaluation is done by a linguist that evaluates segments manually, which is intensive but expensive and time-consuming. Whereas Automatic evaluation of MT is the evaluation of translated contents using automated metrics such as BLEU, NIST, METEOR, and

so on. Automated metrics emerged to address the need for objective, consistent, quick, and affordable assessment of MT output as opposed to a human evaluation.

The most appropriate automatic metric for measuring one's MT system will depend on the language, content type, use case, and MT approach. However, nowadays, many machine translation works are evaluated with BLEU metric and to be comparable with them we will use it for this study too.

The Bi-Lingual Evaluation Understudy (BLEU)

BLEU is the most widely used metric for MT evaluation and was first proposed by Kishore Papineni, et al [51] in 2002. The idea behind BLEU is that “the closer the MT is to a linguist translation, the better it is”.

BLEU measures the overlap of unigrams (single words) and high-order n-grams between MT output and reference translations (Test sets). Its main component is n-gram precision, and to compute a modified precision score, p_n , for the complete test corpus, first compute the n-gram matches at a sentence level, then add the clipped n-gram counts for each candidate (C) sentence and divide by the number of candidate n-grams in the test corpus as shown in equation 2.22.

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n-gram \in C} Count_{clip}(n-gram)}{\sum_{C' \in \{Candidates\}} \sum_{n-gram \in C'} Coun(n-gram')} \quad (2.22)$$

Then, it will compute the *Brevity Penalty*, BP , to make the length of candidate translation match with the length of reference translations if the candidate translation is longer: It is computed as:

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases} \quad (2.23)$$

Where c is the length of the candidate translation and r is the reference corpus length. Finally, it will calculate the geometric average of the modified n-gram precisions, p_n , using n-grams up to length N and positive weights w_n will be summed to one.

$$BLEU = BP \cdot \exp \left(\sum_{n=1}^N w_n \log p_n \right) \quad (2.24)$$

Hence, At the corpus level, BLEU has been shown as a strong correlation with human evaluation. We will evaluate our models using BLEU to be comparable to previous works. Besides, we will do side-by-side human evaluation by linguistic raters, who evaluate and compare the quality of translations predicted by the proposed model.

2.6 Challenges to Machine Translation

MT in general is too challenging for many reasons such as collecting, preparing, and cleaning a very massive amount of corpus for low resource languages such as Ge'ez and Amharic. Particularly, many languages in the world have different lexical and morphological structures. These languages can use different structures for the same purpose and the same structure for different purposes. A word can have more than one meaning due to Semantic (out of context), Syntactic (in a sentence), and Pragmatic (situations and context) meanings, Technical Verbs, paragraphs with symbols and Equations, and Abbreviated Word are very difficult to translate. No direct equivalent word can be found for a particular word of one language in another. As well, numerous research works, frameworks, and tools for different approaches are being created and released, identifying and choosing the best, being familiar with that state-of-the-art tool and framework in a short time is also another challenge. Generally, MT is an extremely challenging task, mainly since natural languages are ambiguous, context-dependent, and ever-evolving [2].

2.7 Related Works

Earlier works on NMT, Ge'ez, and Amharic MT, such as thesis and other online publications are systematically reviewed to understand the domain and the advancements.

Machine Translation Systems for Non-Ethiopian Language Pairs (International Works)

- i. *Google's Neural Machine Translation system: Bridging the gap between human and machine translation.*

The publication by Yonghui Wu, Mike Schuster, et al [28] presented a work that initially started to solve the basic three problems of NMT. Slower training, ineffectiveness in dealing with rare words, and sometimes fail to translate all words in the source sentence. Their model contains a deep LSTM network with eight encoder and eight decoder layers to eliminate slow training. To address rare words, they used sub-word units or *wordpieces* for inputs and outputs. A beam search technique was used to enable the model to translate all of the provided inputs. Their model is a common sequence-to-sequence learning framework with attention. The model was evaluated with WMT En →Fr, En →De datasets, and many Google internal production datasets. On WMT En

→Fr and En →De, the training sets contain 36M and 5M sentence pairs respectively³⁸. In both cases, news test 2014 was used as the test sets to compare against previous works. In addition to WMT, the model was evaluated with some Google-internal datasets such as English ↔ French, English ↔ Spanish, and English ↔ Chinese. On WMT’14 English-to-French, the single model scores 38.95 BLEU, an improvement of 7.5 and 1.2 BLEU from a single model without an external alignment reported in [32, 52] respectively. Moreover, their models were completely self-contained. Likewise, on WMT’14 English-to-German, the single model scores 24.17 BLEU, which is 3.4 BLEU better than a previous competitive baseline. Finally, with a human side-by-side evaluation, the GNMT model reduces translation errors by an average of 60% compared to Google’s previous phrase-based translation system on the above pairs of languages.

ii. *Effective Approaches to Attention-based Neural Machine Translation*

A paper by Minh-Thang et al [19] studies two simple and effective classes of attentional mechanism [19]: a *global* approach that always attends to entire source words and a *local* one that only looks at a subset of source words at a time. They proved the success of both approaches on the WMT translation tasks between English ↔ German. The model achieved a significant gain of 5.0 BLEU points over non-attentional systems that already incorporate known techniques such as dropout. Their ensemble model using different attention architectures yields a new state-of-the-art result in the WMT’15 English to German translation task with 25.9 BLEU points, an improvement of 1.0 BLEU points over the existing best system backed by NMT, and an n-gram re-ranker.

Machine Translation Systems for Non-Ethiopian and Ethiopian language pairs

i. *Optimal Alignment for Bi-directional Afaan Oromo-English Statistical Machine Translation*

This is a study by Yitayew Solomon [53], which was aimed to explore the effect of word, phrase, and sentence level alignments on Bidirectional Afaan Oromo-English statistical machine translation. The corpora were collected from the Criminal code, FDRE constitution, Megleta Oromia and Holly Bible. A total of 6400 simple and complex sentences were used. The researcher used Mosses for the translation process, MGIZA++, *Anymalign*, and *hunalign* tools for word,

³⁸ The datasets contain more than 3 million and 500 thousand pairs of sentences for En→Fr and En→De respectively.

phrase, and sentence level alignments respectively, and IRSTLM for language modeling. From different experiments, the better performance of 47% and 27% BLEU score was scored from Afaan Oromo-English and English-Afaan Oromo translation, respectively. They said an average of 37% accuracy improvement was registered in their study. Concluded that, alignment has a great effect on the quality and accuracy of statistical machine translation between both language pairs.

ii. Bi-Directional English-Afaan Oromo Machine Translation Using Convolutional Neural Network

This one is a study by Arfaso Birhanu [54] which used a total of 5550 parallel sentences, collected from the Holy Bible, published conversational books, Ethiopian governmental constitutions (both regional and federal), Oromia regional revenue, and from Oromia health sectors. The researcher used 80% and 20% of the total dataset for training and testing respectively. Three experiments were conducted. The first was a word-based statistical approach that was used as a baseline, the second was with the RNN method and used as a competitive model and the last one was with convolutional neural networks for the bi-directional translation between Afaan Oromo and English languages. The Baseline (STM) model scored 20.51 and 19.86, The RNN based model scored 22.79 and 21.67, and The CNN-based model also scored 24.37 and 23.18 BLEU scores from English to Afaan Oromo and Afaan Oromo to English respectively. The CNN achieved 3.86 and 3.32 BLEU scores improvement on translation from English to Afaan Oromo and vice versa translation than baseline system. In addition, an improvement of 1.58 and 1.51 BLEU score on translation from English to Afaan Oromo and from Afaan Oromo to English translation respectively than the RNN approach. Even though the CNN is faster than RNN during training, both are getting in low-quality translation as the length of the sentence is grown.

iii. Amharic-Arabic Neural Machine Translation

Another work is done by Ibrahim Gashaw and HL Shashirekha [55]. The researcher used Two LSTM and GRU-based NMT models are developed using Attention-based Encoder-Decoder architecture, using an open-source OpenNMT system. The corpus were collected from Quran is available on Tanzile. They compared the LSTM and GRU-based NMT models and Google Translation system and found that LSTM based OpenNMT outperforms the other, with a BLEU score of 12%, 11%, and 6% for LSTM, GRU, and GNMT respectively.

Machine Translation System for Ethiopian Language pairs

i. Experimenting Statistical Machine Translation for Ethiopic Semitic Languages

Research by Michael Melese Woldeyohannis and Million Meshesha [56] was conducted with 25,470 parallel sentence corpus for both Amharic and Tigrigna. In addition, a separate language model consists of 36,989 sentences for Amharic language and 62,335 sentences for Tigrigna language at the word level. The corpus was mainly collected and prepared from the bible. The researchers prepared a word-word, word-morpheme, morpheme-word, morpheme-morpheme, morpheme-based, and word-based Amharic-Tigrigna and Tigrigna-Amharic parallel data. Using word and morpheme as a unit of the model, eight models have been constructed including word-word, word-morpheme, morpheme-word, and morpheme-morpheme for both Amharic-Tigrigna and Tigrigna-Amharic machine translations. The BLEU score of 6.65 and 8.25 from Tigrigna-Amharic and Amharic-Tigrigna respectively was recorded using a word-word unit. In addition, from Amharic-Tigrigna 13.49 and Tigrigna-Amharic 12.93 BLEU scores were recorded using morpheme as a unit. On the other hand, a BLEU score of 5.81 for Tigrigna-Amharic and 9.11 for Amharic-Tigrigna was achieved using a morpheme unit for Tigrigna and a word unit for Amharic. Moreover, using word unit for Tigrigna and morph unit for Amharic 10.71 for Tigrigna-Amharic and 9.09 BLEU score for Amharic-Tigrigna have been achieved. Finally, the researchers found and concluded as their work result shows that a 4.24% performance improvement was observed using morpheme-based translation over word-based translation from Amharic-Tigrigna translation.

ii. Amharic-Awngi Machine Translation An Experiment Using Statistical Approach

A work by Habtamu Mekonnen [57] based on SMT and the corpus was collected from Amharic texts, Mass Media Agency, and Bible. They used 3500 (1500 simple, 1000 compound, and 1000 complex sentences) and a maximum of 5000 sentences for each sentences type for training with a 10 fold cross-validation. For the language model, they used a minimum of 5700 and a maximum of 14491 monolingual sentences for the Awngi language. Moses for Mere Mortal for the translation process, MGIZA++ for alignment, and IRSTLM for language model were used. A 37% (1-gram scoring) and 17.26% (3-gram scoring) BLUE score was recorded using complex sentences.

iii. *Bidirectional Amharic-Afaan Oromo Machine Translation Using Hybrid Approach*

The study was done by Gelan Tulu Heyi [58]. The researcher collected and prepared the corpus from Fana Broadcasting Corporate News, Holy Bible. A 1402 Amharic-Afaan Oromo parallel sentences were used. Around 7.2% of the total parallel sentences, (i.e., 101 sentences) for testing and 93.8%, (i.e., 1301 parallel sentences) were used for training. Totally two experiments were conducted using two different approaches namely statistical and hybrid approaches. In the statistical approach, a BLEU score of 89.39% and 80.33% were achieved to translate Amharic to Afaan Oromo and vice versa respectively. In a hybrid approach, a BLEU score of 91.56% and 82.24% were achieved for Amharic to Afaan Oromo and Afaan Oromo to Amharic translation respectively. The result shows that the hybrid approach is slightly better than the statistical approach. The result recorded was somehow high because the test set taken was from the corpus itself.

Works on Ge'ez and Amharic Languages

This section discusses related works of machine translation specifically related to both Ge'ez and Amharic languages. Each work is discussed in detail as follows.

i. *Ge'ez to Amharic Automatic Machine Translation: A Statistical Approach*

Done by Dawit Mulugeta [17], which uses phrase-based Statistical Machine Translation to translate from Ge'ez to Amharic. The researcher used 12840 Ge'ez and Amharic parallel sentences from only religious sources. The corpora were prepared from the online available Old Testaments of Ge'ez and Amharic Bible. In addition to the bible, other religious sources like the Praises of St. Mary (Wedase Mariam), Arganon, and some editions of Hamer Magazine were also used. The dataset was split into 90% training and 10% for testing purposes. The software and tools used by the researcher were *Moses* for translation and modeling purposes, *IRSTLM* for language modeling, and *GIZA++* for word alignment. The researcher said that the SMT system does not perform well due to the limited size of the corpus, 12, 840 parallel bilingual sentences. A BLUE score obtained was 8.26% [17].

ii. *Morpheme Based Bi-directional Ge'ez-Amharic Machine Translation*

This work is conducted by Tadesse Kassa [2] and outperforms the word-based machine translation. Because there was trouble using word-based translation in SMT when translating between two morphologically rich languages like Ge'ez and Amharic. At the word level, it is difficult to manage many forms of a single word, not specific and lacks consistency. Whereas at the morpheme level sub-parts of words are specific, easy to manage, and have the consistency of form. The parallel corpus was prepared from the EOTC website and Old Testament of the Holy bible and anaphora or Kidassie (anaphora of Saint John Chrysostom, Saint Epiphanius, and Saint Athanasios), The rest of the bitext which includes seven days Wedase Marya, Anketse Berhan, yewedesewa mela'eket, Kidan and Liton were manually prepared. The morpheme-based aligned sentences were prepared using “*morfessor*”³⁹ and rule-based. Two bilingual files for each technique were prepared. The corpus contains 13,833 simple and complex spiritual sentences. The software and tools used by the researcher were: *Moses-Decoder* for translation setup, *SRILM* for language modeling, *GIZA++* for extracting word and morpheme alignments, *Morfessor* for segmentation of words, and *Pycharm* for python and shell scripting in Ubuntu 16.04 LTS operating system. Six experiments were conducted using word and morpheme as a translation unit. Using the word as a translation unit two experiments were conducted (Experiment on word-based translation from Amharic to Ge'ez and from Ge'ez to Amharic) and four experiments were conducted at morpheme level, (That is two experiments using unsupervised morpheme segmentation and the other two using rule-based segmentation). Finally, the best-performed unit was selected, which was a morpheme-based translation for bi-directional Ge'ez and Amharic MT. Hence, the experiment shows a better performance of 15.14% and 16.15% BLEU scores using morpheme-based from Ge'ez to Amharic and Amharic to Ge'ez translation, respectively. As compared to word-level translation, there is on average of 6.77% and 7.73% improvement from Ge'ez-Amharic and Amharic-Ge'ez respectively. Accordingly, the performance of rule-based morphological segmentation is better than unsupervised with an average BLEU score of 0.6% and 1.27% for Ge'ez to Amharic and Amharic to Ge'ez respectively. This is shown in table 2.2 below.

³⁹ *Morfessor* is a family of probabilistic machine learning methods for finding the morphological segmentation from raw text data.

Table 2.2 BLEU scores of Morpheme Based Bi-directional Ge'ez-Amharic MT

Types of an experiment conducted	Result of an experiment in BLEU from both directions		
		Ge'ez to Amharic	Amharic to Ge'ez
Word Based Translation		8.37%	8.42%
Morpheme Based Translation	Using Morfessor	14.54%	14.88%
	Using Rule-Based	15.14%	16.15%

iii. *A Hybrid Ge'ez to Amharic Machine Translation*

Conducted by Biruk Abel [16] composed of two main components a Rule-Based Ge'ez Corpus Preprocessor and a Baseline SMT. It uses a serial coupling of rule-based Ge'ez language word reordering followed by a standard SMT system. The Rule-Based Preprocessor takes the manually Part of Speech tagged Ge'ez corpus and produces another corpus that contains reordered Ge'ez sentences having a similar structure with that of Amharic sentences. It first reads all sentences from the input file and iterates through all sentences and it determines the PoS pattern and applies the corresponding reordering rule. After each sentence is processed the output corpus along with the Amharic corpus will be fed up as an input to the Baseline SMT. Then using the input corpora, the actual translation of Ge'ez sentence to Amharic sentences will be performed by the Decoder of the Baseline SMT by using the Language model of Amharic and Translation model. The researcher used two sets of corpora to test the proposed Hybrid Ge'ez to Amharic Machine Translation System and the Baseline SMT. The first set contains Ge'ez and Amharic corpus without any POS information that will be used as an input to the Baseline SMT without being fed to the hybrid one. The second set contains POS tagged Ge'ez corpus and an Amharic corpus without any POS information. The POS tagged Ge'ez corpus will first be preprocessed via the Rule-Based Ge'ez Corpus Preprocessor before being supplied as an input to the Baseline SMT. The Ge'ez corpus contains 976 sentences with 3010 words and the Amharic corpus contains the same number of sentences with 3174 words.

The researcher conducted two experiments. The first one was to test the Baseline SMT and the other was to test the proposed system. To test the Baseline SMT both Ge'ez and Amharic corpus without POS were used while to test the proposed system Ge'ez corpus with POS and Amharic corpus with no POS were used. Based on the test results the Baseline SMT scored a BLEU of 72% and the proposed system outscores it by 4% and scored 76% owing to the reordering rules applied

on Ge'ez corpus. This study showed a high result. However the researcher used a corpus with only simple sentences (maximum of 5 words in a sentence), no real-world translations focused, and the test set taken was from the corpus itself.

Summary of Related Works

All the related works, discussed above on both Ge'ez and Amharic languages are done either via Statistical Machine Translation or a hybrid methodology. However, there are many kinds of research done on Neural Machine Translation in different language pairs. The researchers mentioned above do their part and suggested many further works to be done. The summary of related works on both Amharic and Ge'ez is shown in table 2.3 underneath.

Table 2.3 Comparison between related works on Ge'ez-Amharic MT

No	Author (Year)	Title	Methodology & Algorithms	Size of Corpora	Result	Remark
1	Dawit Mulugeta (2015)	Ge'ez to Amharic Automatic Machine Translation: a Statistical Approach	SMT, Moses, GIZA++, IRSTLM	12,840 parallel bilingual Sentences	BLUE score 8.26 on 10F CV	Small data size. Low BLEU score
2	Tadesse Kassa (2018)	Morpheme-Based Bi-directional Ge'ez -Amharic Machine Translation	SMT & Rule Based, Mosses, MGIZA++, IRSTLM	13,833 simple and complex sentences	15.14% and 16.15% BLEU scores	The small and domain-specific dataset.
3	Biruk Abel (2018)	Ge'ez to Amharic Machine Translation	Rule-Based and SMT, Moses	976 parallel sentences	BLEU 76% (Questionable)	The test set taken was from the corpus itself. Only simple sentences with a max of 5 words. A static structure of sentences

CHAPTER THREE: GE'EZ AND AMHARIC LANGUAGES

This chapter will cover the lexical, syntactical, grammatical structures, linguistic relationships, and writing systems of both Ge'ez and Amharic languages.

3.1 The Ge'ez Language

Ge'ez, also known as Ethiopic, is an ancient Semitic language of Ethiopia (including Eritrea). Ethiopia is one of the oldest and historic countries in the world having its own characters called *ፊደል (Fidel)* or *ሆኔያት (hohieyat)* and numbers called *አኃዝ (Ahaz)*. The earlier and the current civilization, bravery, religion, culture, and history of Ethiopia were known throughout the world via the scripts on stone, vellums, and pictures drawn and written in Ge'ez [59]. In Ge'ez language numerous books have been written, compositions have been authored, and ቅኔዎች (Qinnies) have been scripted. Hence, the wisdom and history of Ethiopia have been transferred from generation to generation because there were vellum books written and stored in Ge'ez in different places mainly in Ethiopian Orthodox Church monasteries and caves. The literature includes religious texts (such as the Bible, Apocrypha, Pseudepigrapha, liturgical literature, homilies, theological, acts of martyrs and saints, religious poetry, hymns in honor of Christ and his mother virgin Saint Mary, and Angels), as well as secular writings (such as histories and romances, legal, mathematical, and medical texts) [16] attested in inscription since the early 4th century.

Later the Amharic language starts substituting the Ge'ez language in the 12th century and Ge'ez has completely died out as a spoken language close to 13thC but remained the primary writing language of Ethiopia up to the 21stC and remains only as of the literature and liturgical language of the Ethiopian and Eritrean Orthodox Tewahido Churches, the Ethiopian Catholic Church, and the Bete-Israel Jewish community of Ethiopia [17, 59].

3.1.1 Ge'ez Script Arrangements (ኑባሬ ፊደል)

Ge'ez language has two script arrangements, the former script arrangement, ቀዳማዊ ኑባሬ ፊደል (*k'edamawī nubarē fīdeli*) and the later (current) script arrangement, ደሐራዊ ኑባሬ ፊደል (*dehārawī nubarē fīdeli*) [2, 60].

The former script arrangement of Geez has this 26 basic characters: አ (ā), ቤ (be), ገ (ge), ደ (de), ሀ (hā), ወ (we), ዘ (ze), ሐ (hā), ጎ (ḡā), ጠ (t'e), ሃ (ye), ከ (ke), ለ (le), መ (me), ነ (ne), ሠ (še), ዐ ('ā), ፈ (fe), ጸ (ts'e), ፀ (ts'e), ቀ (k'e), ረ (re), ሰ (se), ተ (te), ጸ (p'e), and ፐ (pe). And the current script arrangement also has this 26 basic characters with different order: ሀ (hā), ለ (le), ሐ (hā), መ (me), ሠ

(še), ረ (re), ሰ (se), ቀ (k'e), ቦ (be), ተ (te), ኀ (hā), ነ (ne), አ (ā), ከ (ke), ወ (we), ዐ ('ā) ዘ (ze), የ (ye), ደ (de), ገ (ge), ጠ (t'e), ጰ (p'e), ጸ (ts'e), ፀ (ts'e), ፈ (fe) ፐ (pe). The complete script arrangement of both the former and the current basic forms and their family (the derived forms) are depicted in [Annex C, Table C.1](#). As it is depicted in this Annex, in table B.1 (a) and B.1 (b), Ge'ez language has 182 letters (7*26) with two arrangements (Previous and Current), as well B.1 (c) and B.1 (d) shows derived letters of Ge'ez language from the basic letters [2, 60]. Which means 182+22+16=220 unique characters. Table (d) are special derived *letters*, which are created by leaving the second and the seventh order appearance and by changing their shape and sound.

3.1.2 Ge'ez Numerals (አኃዝ)

Ge'ez has its own non-positional numerals. The Amharic language also takes these numbers as they are. These numbers are used in the Ethiopian yearly calendar, birr notes, and some other national profiles. Some of the non-positional Ge'ez numbers are ፩, ፪, ፫, ፬, ፭, ፮, ፯, ፰, ፱, ፲ (1 - 10), ፳, ፴, ፵, ፶, ፷, ፸, ፹, ፺, ፻, and ፺፱ (20 – 1000). A table in [Annex C ii, Table C.2](#), shows more the Ge'ez numerals [61]. We have prepared 3078 Ge'ez-Latin numeral datasets to handle number translation.

3.1.3 Similar Letters (ተመኩሳይያን)

They are letters that have similar sounds. Even though they are having similar sounds, the letters are different in shape orthographically.

Table 3.1 similar letters in Ge'ez and Amharic

Sound	Letter
ha	ሀ፣ሐ፣ኀ
se	ሠ፣ሰ
a'	አ፣ዐ
ts'e	ጸ፣ፀ

These letters have unique importance in Ge'ez script. That means they make words have different meanings. For example, ሠረቀ (šerek'e) = ወጣ (wet'a) and ሰረቀ (serek'e) = ሰረቀ (serek'e) to mean (He went out) and (He stole), አመት (āmeti) = አገልጋይ (āgeligayi) and ጻመት (ameti) = ዘመን (zemeni) to mean (server) and (Year), ሰዐለ (se'āle) = ስዕለ ሳለ (si'ili sale) and ሰአለ (se'āle)= ለመነ (lemene) to mean (He painted) and (He begged), መሀረ (mehāre)= አስተማረ (āsitemare) and መከረ (mehāre)= ይቅር አለ (yik'iri āle) to mean (He Taught) and (He Forgave), ኀለየ (hāleye)= አመሰገነ (āmesegene) and ከለየ (hāleye)= አሰበ (āsebe,) to mean (He Thank) and (He Thought), as well, ፈጸመ (fets'eme)= ጨረሰ (ch'eres) and ፈፀመ (fets'eme)= ነጩ (nech'e) to mean (He Finished) and (He scraped) respectively.

However, these similar characters do not have any difference in the Amharic writing system, except writing trends adopted from Ge'ez, such as ኅመተ ምህረት ('*amete mihireti*) and አሜሪካ (*āmērīka*). Though, they have a great influence on machine translation quality. We apply normalization to handle these problems. Ge'ez also has words like መካን (*mekan*) that have different meanings when they are spoken accentuated (means *Barren*) and smoothly (*place*) [62] and these are also other challenges for Machine Translation tasks.

3.2 The Amharic Language

Amharic is the second most widely spoken Semitic language in the world, next to Arabic and the second largest language in Ethiopia (after Afan Oromo) [17]. It is the official working language of the Federal government of Ethiopia, where it has over 100 million native and nonnative speakers where the overall population of Ethiopia was estimated over 120 million, and that makes the native speakers more than 83.3% of the population [14]. Amharic uses a Ge'ez script called hohieyat (ሆሄያት) which is written in a tabular format of seven columns. Both Ge'ez and Amharic languages share the same scripts and writing system as depicted in [Annex C, Table C.1](#). The first column represents the basic form and the other orders are derived from it by slight or considerable modifications indicating the different vowels. Amharic has 34 bases 8 derived characters leads to having $246 = (34 * 7 + 8)$ characters where the 26 characters are derived from Ge'ez. The remaining 8 of them were by modifying Ge'ez characters; namely, ሰ to ሰ̄ ፡ ተ to ተ̄ ፡ ነ to ነ̄ ፡ ከ to ከ̄ ፡ ዘ to ዘ̄ ፡ ደ to ደ̄ ፡ ጠ to ጠ̄ and ቢ to ቢ̄ [2]. As well, the second supplementary 8 derived characters are formed from the first 8 derived ones. This is depicted in the following table.

Table 3.2 Amharic Script (a) added script, (b) Derived script

	ግዕዝ	ካዕብ	ሣልስ	ራብዕ	ሐምስ	ሳድስ	ሳብዕ
፩	ሰ	ሰ̄	ሰ̈	ሰ̇	ሰ̆	ሰ̅	ሰ̄
፪	ቸ	ቸ̄	ቸ̈	ቸ̇	ቸ̆	ቸ̅	ቸ̄
፫	ኘ	ኘ̄	ኘ̈	ኘ̇	ኘ̆	ኘ̅	ኘ̄
፬	ኸ	ኸ̄	ኸ̈	ኸ̇	ኸ̆	ኸ̅	ኸ̄
፭	ዠ	ዠ̄	ዠ̈	ዠ̇	ዠ̆	ዠ̅	ዠ̄
፮	ደ	ደ̄	ደ̈	ደ̇	ደ̆	ደ̅	ደ̄
፯	ጠ	ጠ̄	ጠ̈	ጠ̇	ጠ̆	ጠ̅	ጠ̄
፰	ቢ	ቢ̄	ቢ̈	ቢ̇	ቢ̆	ቢ̅	ቢ̄

(a)

ሰ̄	ቸ̄	ኘ̄	ኸ̄	ዠ̄	ደ̄	ጠ̄	ቢ̄
----	----	----	----	----	----	----	----

(b)

3.3 Linguistic Relationships of Ge'ez and Amharic

3.3.1 Writing System

The writing system of Ge'ez and Amharic is similar. Both languages use the Abugida or alpha syllabaries writing system from six different types of writing systems namely, Alphabets (English, Russian, Greek), Abjads (Arabic, Hebrew), Abugidas or alpha syllabaries (Devanagari, Thai, Ge'ez, Amharic), Featural alphabets (Hangul), Syllabaries (Japanese, Cherokee), and Logographic systems (E.g., Chinese characters). Before Aba Selama (Friemnatos), the first Patriarch of Ethiopia, Ge'ez was written from right to left but now it is written from left to right like Amharic [12, 60]. The two types of Ge'ez alphabet arrangement are አበገደ (the previous) and ሀሁ (the current) [2]. Amharic uses the current Ge'ez script arrangement (ደሐራዊ ኑባሬ ፊደል) with its new derived characters.

3.3.2 Syntactic Language Structure (Word Order)

The general syntactic structure of Amharic is Subject-Object-Verb (SOV) word order e.g. አበበ መኪና ገዛ (*ābebe mekina geza*) meaning, “Abebe bought a Car”. However, if the object is tropicalized it may precede the subject (OSV) E.g. መኪናውን ተመስገን ያመጣዋል (*mekīnawini temesigeni yamet’awali*) meaning, “Temesgen will bring the Car”. Whereas Ge'ez follows somewhat free word order structure such as Subject-Verb-Object (SVO), Verb-Subject-Object (VSO), and Object-Verb-Subject (OVS) word orders.

- A. *Subject-verb-object (SVO)*: አልማዝ ገብረት ጽብሳ (*ālimaz gebiret ts'ibih*) → አልማዝ ወጥ ሰራች (*ālimaz wet'i serachi*). ኤሌያስ መሀረ ትምህርተ (*ēlēyas mehāre timihirite*) → ኤሌያስ ትምህርትን አስተማረ (*ēliyas timihiritin āsitemare*).
- B. *Verb-subject-object (VSO)*: ርዕዩ መላእክት በሰማይ (*ri'eye mela'ikit besemayi*) → መላእክትን በሰማይ አየ (*mela'iketin besemayi āye*). Meaning (*He saw angels in heaven*)
- C. *Object-Verb-Subject (OVS)*: ለሙሴ ነበረ እግዚአብሔር (*lemusē nebebo igizī'ābihēr*) → እግዚአብሔር ሙሴን ተናገረው (*igizī'ābihēri musēni tenagerewi*) and
- D. *Subject-Object-Verb (SOV)*: እግዚአብሔር ለሙሴ ነበረ (*igizī'ābihēri lemusē nebebo*) → እግዚአብሔር ሙሴን ተናገረው (*igizī'ābihēr musēn tenagerew*) meaning “God spoke to Moses”.

From the above word orders, the Ge'ez language mostly follows the Subject-verb-object (SVO). For instance, the sentence “ውእቱ መጻኢ ጎበ ቤቱ (*we'etu mets'a habe bietu*)” in Ge'ez is similar to “እሱ ወደቤቱ መጣ (*esu wedebietu met'a*)” in Amharic, which means “He came home” where “እሱ (*esu*)”, “ወደቤቱ (*wedebietu*)” and “መጣ (*met'a*)” are the *subject*, *object*, and *verve* that are equivalent

to “ውእቱ (weetu)”, “ኅበ ቤቱ (*habe bietu*)”, and “መጽኢ (*metsti 'a*)” in Ge'ez respectively. But usually, pronouns are omitted in both languages and become part of the verb when they are used as a subject “መጽኢ ኅበ ቤቱ (*metsti 'a habe bietu*)” equivalent to “ወደቤቱ መጣ (*wedebietu met 'a*)” [17].

3.3.3 Grammar structure of Ge'ez and Amharic (ሰዋሰው)

In linguistics, grammar or ሰዋሰው is a set of structural constraints controlling the composition of words, phrases, and clauses in any given natural language such as Ge'ez and Amharic. It is the study of such rules, which includes phonology, morphology, and syntax, often complemented by phonetics, semantics, and pragmatics [2].

Words or lexicons are the basic units of many languages. Even though most speakers know and use only a relatively small number of words, languages have tens of thousands of words. Each word has its own part of speech (word class) [2, 17].

Based on parts of speech grammarians classified words into eight major parts of speeches in both Ge'ez and Amharic. These are Nouns (ሰም), Adjectives (ቅጽል), Verbs (ግስ), Adverbs (ተውሳክ ግስ), Pronoun (ተውላጠ ሰም), Preposition (መስተዋድድ), Conjunction (መስተጻምር) and Interjection (ቃለ ኢጋኖ).

However, many references divide parts of speech in terms of *form (Major)* and *structure (Minor)* classes. The form (major) parts of speeches are Nouns/ሰም, Adjectives/ቅጽል, Verbs/ግስ, and Adverbs/ተውሳክ ግስ, which are words that carry the content or meaning of a sentence. The structure (minor) parts of speech are Pronoun/ተውላጠ ሰም, Preposition/መስተዋድድ, Conjunction/መስተጻምር, and Interjection/ቃለ ኢጋኖ, which are words that serve primarily to indicate grammatical relationships and are frequently referred to as structure words [2, 16, 17]. Here the major ones are discussed a little below.

3.4 Major Parts of Speech

3.4.1 Noun (ሰም)

A noun is a name that refers to a person, animal, place, thing, feeling, or abstract idea. It can tell who or what. In Ge'ez and Amharic, there are different types of nouns such as concrete and abstract, proper nouns, collective nouns, countable and uncountable nouns. E.g, name of a person ሰላማዊት (*Selamawit*), places ደብረ-ብርሃን (*DebreBerhan*) or feeling ደስታ (*desta*) means Happiness [2]. Ge'ez language has two ways of forming plural forms of nouns [2]. These are:

- ❖ Pattern replacement (*broken or internal*): ደብር (*debr*) to አድባር (*adbar*).

- ❖ Addition of an ending (*external*): አመት (*Amet*) to አመታት (*Ametat*), መምህር (*memhir*) መምህራን (*memhiran*), ገዳም ገዳማት... etc

The two endings used to form external plurals are -ān (አን) and -āt (አት). -ān is to denoting masculine and -āt (አት) is for feminine. As Amharic by itself is derived from Ge'ez, it shares both the *internal* and *external* ways of forming plural nouns in addition to its own. E.g. ደብር (*debr*) to አድብር (*adbar*) or ደብሮች (*debroch*) by adding -ኛች(-och), only in Amharic and መምህር (*memhir*) to መምህራን (*memhiran*). However many people say አድብራቶች (*adbaratoch*), መምህራኖች (*memhira-noch*) or ገዳማቶች (*gedamatoch*) by adding both the Ge'ez and Amharic plurality forms which makes the words neither Ge'ez nor Amharic and makes any MT model to train in a wrong way. Because the Ge'ez-Amharic corpus does not have such unknown words and therefore, knowing the exact ways of forming plurality is vital during corpus preparation in order to get a quality translation.

3.4.2 Adjective/ቅጽል

Adjectives are words or constructions used to identify, qualify, describe, or further define nouns or pronouns. Adjectives express things behavior or characteristics, such as shape, size, color, type, and property. For example, ጸሐዳ ርግብ (*ts'e āda rigibi*) equivalent with ነጭ ርግብ (*nech'i rigibi*) meaning “white dove”.

The use of adjectives in Ge'ez and Amharic sentences is not the same. In Ge'ez, language adjectives are used before and after nouns whereas in Amharic adjectives are mostly used before nouns [2, 16, 17]. For example, ፍንዋን እደው ይነግሩ መልእክተ። (*finiwani idewi yinegiru meli'ikite*) and እደው ፍንዋን ይነግሩ መልእክተ (*idewi finiwani yinegiru meli'ikite*) have the same arrangement in Amharic that is የተላኩ ወንዶች መልእክት ይናገራሉ። (*yetelaku wenidochi meli'ikiti yinageralu*) meaning “The messenger boys tell the message”.

Adjectives in Amharic are either primary adjectives, derived from nouns or verbs or other combinations. Examples: Primary adjective: ጥቁር (*t'ikur*) meaning black, and Derived from the noun ኃይል (*hayl*) meaning force, ኃይለኛ (*hayilenya*) meaning forceful.

3.4.3 Verb (ግስ)

A verb is a word derived from roots that refers to an action, occurrence, or state of being or condition, and forming the main part of the sentence. E.g. ሐየለ (*hayele*) → አየለ (*ayele*), መልሐ (*melha*) → መዘዘ (*mezeze*). Based on an affix Ge'ez and Amharic verbs are divided into regular and irregular. Regular verbs are main verbs that have four types; ቀዳማዬ ወይም ጎላፊ (past or perfect)

tense, ካልአይ ወይም የአሁንና እና የመጻኢ. (present and future imperfect), ትዕዛዝ (command) or ዘንድ (to be verbs) [2, 16, 17].

Root Verbs (አርእስተ ግስ)

Root verbs are regular verbs used as base words for other verbs to use and follow their morphology style. Root Verbs in Ge’ez are eight with having their characteristics⁴⁰ [61].

Table 3.3 Root or main Verbs (አርእስተ ግስ) of Ge'ez

ተ.ቁ	Verbs- ግእዝ አርእስተ ግስ (Pronunciation -አነባበብ)	Meaning-ትርጉም (Pronunciation -አነባበብ)
1	ቀተለ (<i>k'etele</i>)	ገደለ (<i>gedele</i>)
2	ቀደሰ (<i>k'edese</i>)	አመሰገነ (<i>āmesegene</i>)
3	ተንበለ (<i>tenibele</i>), ደንገጸ (<i>denigets'e</i>)	ለመነ (<i>lemene</i>) ደነገጠ (<i>deneget'e</i>)
4	ባረከ (<i>bareke</i>)	ባረከ (<i>bareke</i>)
5	ማሕረከ (<i>maḥireke</i>)	ማረከ (<i>mareke</i>)
6	ሴሰየ (<i>siēseye</i>)	መገበ (<i>megebe</i>)
7	ክህለ/ብእለ (<i>kihile/bi'ile</i>)	ቻለ (<i>chale</i>)
8	ጦመረ (<i>t'omere</i>)	ጻፈ. (<i>ts'afe</i>)

Verbs in Amharic mostly are placed at the end of the sentence whereas in most Ge’ez sentences the verbs are placed in the middle [17]. For example ውኑት ቀተለ አንበሳ (*wu'etu k'tele anbesa*) → እሱ አንበሳ ገደለ (*esu anbesa gedele*) meaning “He killed a lion”.

3.4.4 Adverb (ተውሳክ ግስ)

Adverbs (ተውሳክ ግስ) are words that give additional meaning to verbs. The job of adverbs is to tell the verb’s place, time, degree, and so on by giving information how, why, where... etc. about the verb. Mostly Ge’ez adverbs come after the verb they modify and the Amharic adverbs precede the verb they modify. For instance in the sentence, ሮፀ ኃይሌ ፍጡነ (*rots'e hayilē fit'une*) → ኃይሌ በፍጥነት ሮጠ (*hayilē befit'ineti rot'e*) meaning “Haile ran faster”, the Adverb ፍጡነ (*fit'une*) follow the verb ሮፀ (*rots'e*) in the Ge’ez sentence. However, the adverb በፍጥነት (*befinet*) precede the verb ሮጠ (*rot'e*) in Amharic sentence [17].

⁴⁰ Elam Aba, አርእስተ ግስ-የግስ አለቆች (2) <https://www.youtube.com/watch?v=82pcq0RVdHw>, Jun 24, 2020, Accessed March 26, 2021

3.5 Minor Parts of Speech

As discussed in subsection 3.3.3 the structure or minor parts of speeches are Pronoun (ተውላጠ ስም), Preposition (መስተዋድድ), Conjunction (መስተጻምር), Interjection (ቃለ ኢጋኖ), Demonstrative (አመላካች), possessive (አገናዛቢ), and punctuation marks (ትምህርተ ጥቅስ), which are used to indicate grammatical relationships between other words [2, 16, 17]. However, we do not do syntactic translation and we do not use those parts of speech (except punctuations) as a feature in our deep learning-based machine translation methods. Hence, we only discuss the punctuation marks used in both languages.

Punctuation Marks

Both Amharic and Ge'ez use similar punctuation marks for different purposes. A few decades back, the individual word-separator in the sentence “ሁለት ነጥብ (*Hulet netib*)”, two dots arranged like colon (:) were used in both Ge'ez and Amharic. However, today the use of *Hulet Neteb* is paused and replaced by space in modern typing systems. The basic punctuation marks include the sentence-separator, “አራት ነጥብ (*arat netib*)” that is four dots arranged in a square pattern (::), lists separator “ነጠላ ሰረዝ (*netela serez*)” that is two dots like colon with upper bar (፣) which is equivalent with comma and “ድርብ ሰረዝ (*derib serez*)” that is two dots like colon with upper and lower bars (፤) equivalent with a semicolon. The symbol ‘?’ is used to represent questions in Amharic but no in Ge'ez. The interrogative word or character in Ge'ez is placed at the end of a word or sentence. It is pronounced at a low level and the style of pronunciation by itself shows an interrogation. For example; ሁ (hu), ኑ (nu), ሁ (u), ኢ (ī), ት (ti), አ (a), አይቴ (āyitē) used for words such as ሰባኑ (*sobenu*) meaning “When?” ተአምሩኑ (*te'āmirunu*) meaning (*Do you know?*), አንትሙሁ (*ānitimuhu*) meaning “are you?”, ተአምረኒኢ (*te'āmireni'ī*) or ተአምረኒኑ (*te'āmireni'ī*) meaning “do you know me?” [2, 17]. However, we removed punctuation marks during preprocessing while preparing the dataset for Machine Translation.

CHAPTER FOUR: RESEARCH METHODOLOGY

4.1 Overview

This chapter discusses the selected type of research method used for this study, the proposed architecture of a bi-directional Ge'ez-Amharic MT system, corpus preparation, and how the model train and translate sentences between the two languages are annotated. In general, this chapter briefly explains how this study was exhaustively done.

4.2 Research Design

For this study, To develop a Bi-Directional Ge'ez Amharic Machine Translation, a Design Science Research Methodology (DSRM) is selected as a general approach that is one of the approaches used in the field of information systems. Creating an applicable solution to a problem is an accepted research paradigm in different disciplines, such as Computer science, and engineering. Design science research (DSR) is a process of creating and evaluating information technology artifacts aimed to solve known organizational or community problems. This problem-solving approach tries to improve human knowledge through the creation of innovative artifacts and the creation of design knowledge by innovative solutions to real-world problems [63]. Figure 4.1 below shows a DSR framework for understanding, executing, and evaluating a Bi-Directional Ge'ez Amharic MT.

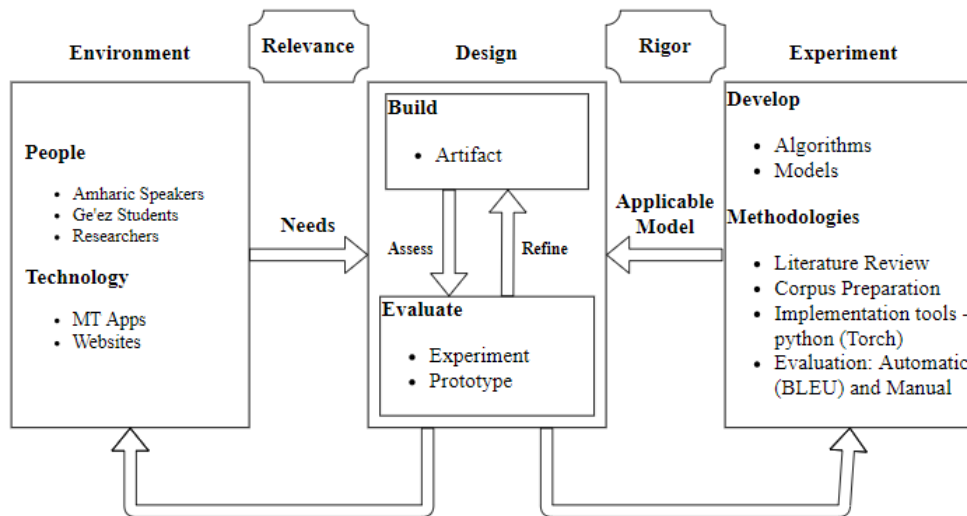


Figure 4.1 Design Science Research Framework Adopted From [63]

The environment expresses the problem space where the phenomena of interest exist in. It includes people, existing or planned technologies, and organizations. As well, goals, tasks, problems, and opportunities are defined in the environment. Needs are assessed and evaluated within the context of the requirement of language speakers. The Design phase consists of building the artifacts and evaluating them with different experiments and showing the demonstration or prototype. The experiment holds development and methodologies where different algorithms and models are developed. The methodology provides steps and guidelines used in this research. The rigor is achieved by properly applying existing development and methodologies.

The DSR approach has been selected because of the following reasons.

- ❖ DSR is a popular new research methodology and paradigm in the department of Information Systems, for which numerous research approaches have been developed.
- ❖ DSR supports practical problem solving that is solution-oriented, such as developing MT models that can be used by Ge'ez and Amharic language users [64].
- ❖ The DSR can decrease the gap between theory and practice by producing practical knowledge, which can serve as a reference for the coming researchers. Figure 4.1 depicts the general DSR framework and the relationship between two essential factors for the achievement of the investigation: relevance and rigor.

Design Science Research Methodology

For this study, the design science research methodology (DSRM) process model was chosen. According to [65], DSR is a methodical problem-solving method for producing relevant, new, and innovative information systems solutions within a specific domain. Iteratively, the alternatives and revised designs are evaluated until the best solution to the problem is identified.

Peppers, et al. [65], suggest a DSRM, which is reliable with prior literature and provides a design science research methodology process model to present the DSR. The authors claim that this process model provides support for researchers, and this is a good way for researching the design science paradigm. In this section, we justify that the DSRM process model of [65]. Brocke et al. [64], indicate that the research method can be applied differently according to the type of problem and the research objective and its starting point can be modified according to the targets of the researcher. Based on Peppers, et al. [65] Considering these starting points, the entry point of this research is the problem-centered initiation, the objective-centered solution, and the development-

centered solution. The DSRM process model consists of six activities, which cover the complete study from the start (motivation) to the end (communication). These phases are problem identification, objective definition, design and development, demonstration, evaluation, and finally communication as shown in Figure 4.2 below.

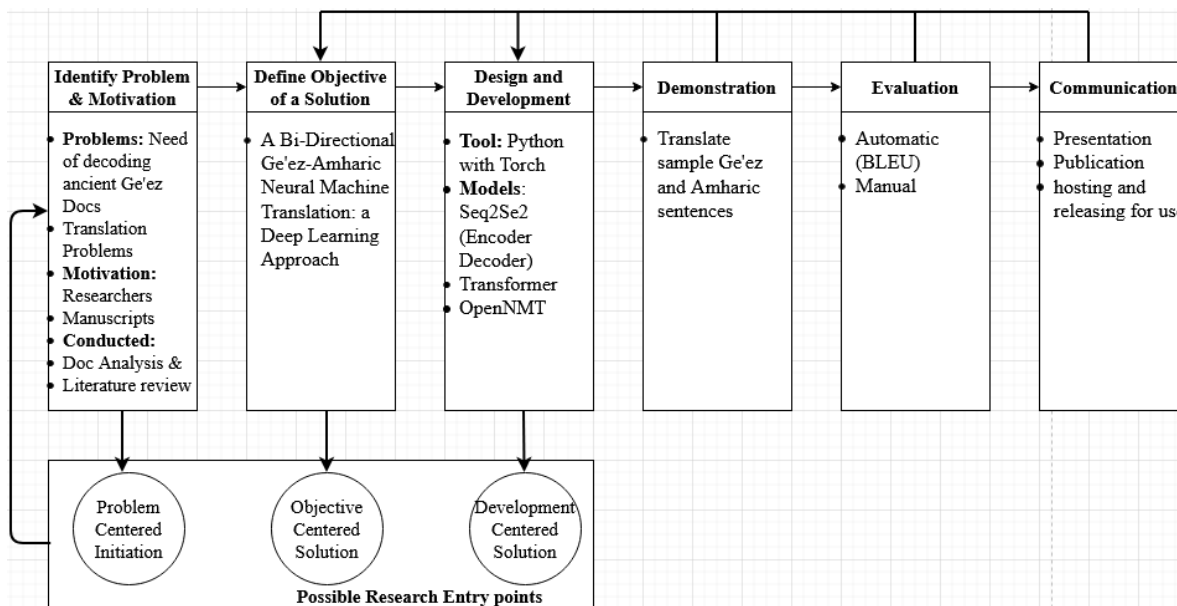


Figure 4.2 DSR Methodology Process Model Adopted from [63]

A brief explanation of each DSR step, shown in Figure 4.2 is discussed as follows:

i. Problem Identification and Motivation

This step describes the specific research problem and justifies the value of a solution. As discussed in section 1.2, we are motivated to conduct this research for different reasons, such as there are many ancient scripts encoded in the Ge'ez language and there is a high need to manipulate those documents using technologies such as NMT for mining old knowledge. As well, there are new youth researchers that are interested to study the Ge'ez language. Filling the gaps from prior researches is also another motivation. To achieve these all, vast document analysis and review of literature are accessed.

ii. Define the objectives for a solution

After the problem definition, the next step is the objectives of a solution and identifying the feasible solution. E.g., a question like “how the proposed Ge'ez Amharic NMT is useful for society?” can bring the objective of a solution.

iii. Design and development

The third step is creating an artifact and determining the intended functionality of the artifact, and its proposed architecture. According to [63], such artifacts are perhaps concepts, models, methods, and instantiations. In this study, the proposed system architecture shown in section 4.3 and the implementation tools listed in section 5.3 explain the design and development process of this research in detail.

iv. Demonstration

This step proves the use of artifacts or systems to solve the identified problem. This might include experiments, simulation, proof, or another appropriate step in different fields of study. In this work, we conducted four experiments to prove the use of the new system as discussed in section 5.4.

v. Evaluation

This step measures how well the artifact is done and how it supports a solution to the problem. After evaluation, the researchers can decide whether to iterate back to step three to try to improve the efficiency of the model or to continue to step four and leave further improvement to the following tasks. In this study, we used two types of evaluation methods namely the automatic (BLEU score) and the manual (human) evaluations metrics.

vi. Communication

Finally, all parts of the problem and the designed artifact are presented to the relevant stakeholders, organizations, or departments. Journal publication is also accomplished under this step.

4.3 The proposed system

The architecture of a Bi-Directional Ge'ez-Amharic Machine Translation system depicted in Figure 4.3 shows the overall workflow of the translation model.

This architecture works through six major different stages, namely the *preprocessing*, *data splitting*, *embedding*, *encoder*, *decoder*, and *evaluation* phases. First, the bi-lingual corpus goes to preprocessing and has cleaning, normalization, padding, tokenization, and other preprocessing tasks. Then the preprocessed corpus is divided into two core sets called to train and test set. Afterward apply embedding to make the corpus readable to the model, which is transforming words into vocabulary then converts them to vectors of continuous real value numbers. Next, Positional encoding is applied to know the relationship between tokens from the sequence by considering word orders in the case of Transformers. This intern enhances the representation of the transformer to give attention to any required token from any position in the sequence.

Moreover, the encoder encodes the input sequence to an internal representation called ‘context vector’, which is used by the decoder to generate the output sequence, as well, the decoder decodes the encoded sequence as per the input language to be translated to a sentence in the output language. The output embedding and the output positional encoding are also applied at the end just like the stages in the input process. Finally, Evaluating the model then go back to data splitting for another experiment if the translation quality is poor. These concepts are discussed in detail below and the general structure of system architecture is depicted as follows in Figure 4.3.

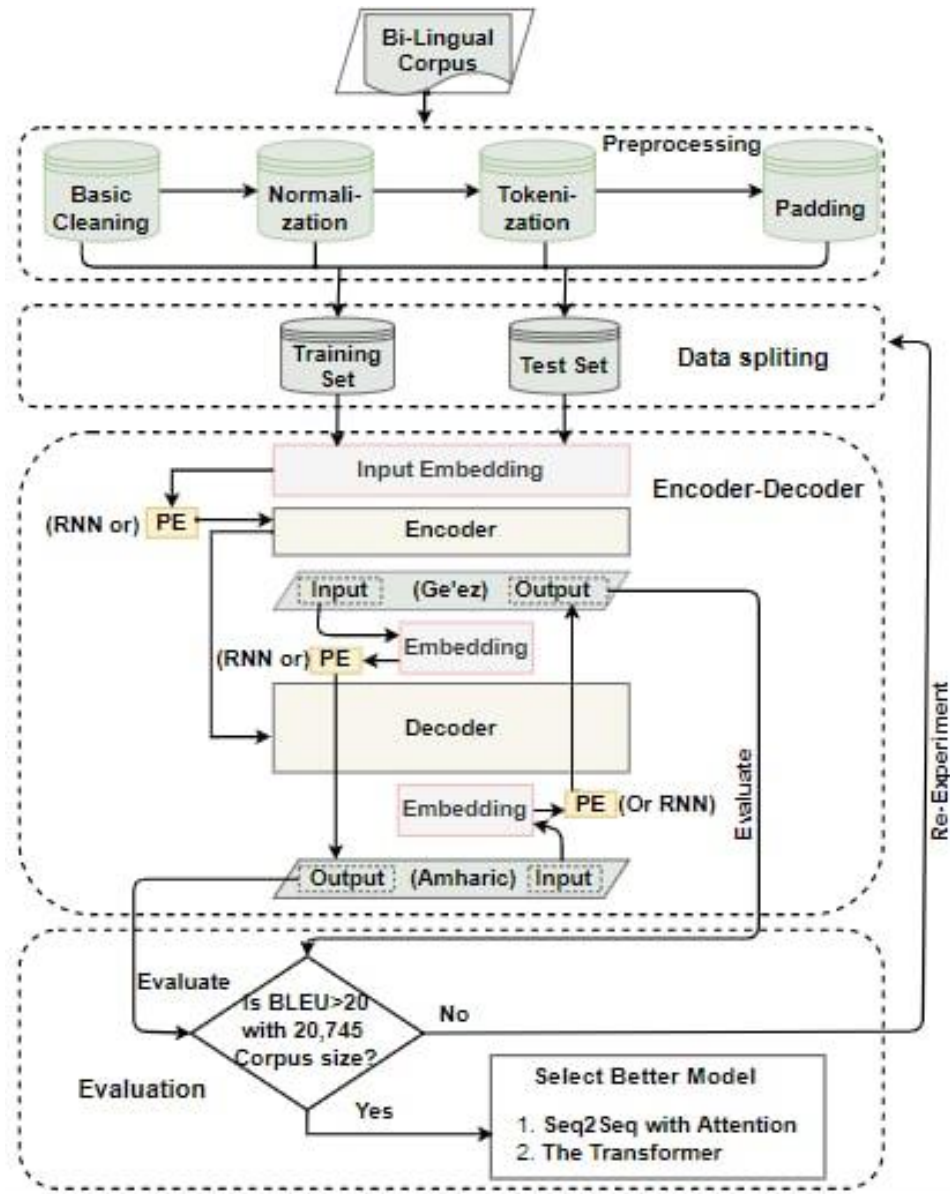


Figure 4.3 Overall architecture of the proposed system

4.4 Preprocessing

We conduct preprocessing, the first and most important task during machine translation. It includes cleaning text, normalization, padding, and tokenization.

4.4.1 Major Cleaning

Cleaning text includes removal of numerals, special characters, punctuation marks, unwanted spaces, extremely long sentences, mistranslated sentences, and other non-Ge'ez characters. Punctuation marks do not have major relevance for translation and are excluded from the corpus.

Algorithm 4.1 Algorithm for general cleaning (Preprocessing) of a corpus

```
##Defining a Regular-expression R, Containing punctuations and non-Ge'ez characters
# Then preprocessing the corpus by matching each character in a dataset with regular expression R
DEFINE REGULAR EXPRESSION (R) #R Contains punctuation marks and non-Ge'ez characters
OPEN AND READ TEXT FILE #Input
WHILE READING LINE DO
  FOR C IN LINE #Read each character from each line
    IF C IS IN R #if C is punctuation or is not in Ge'ez characters (not in  $\mathcal{V}$ )
      THEN
        REPLACE C WITH SPACE
      END IF
    END WHILE
  CLOSE FILE
```

4.4.2 Normalization

In NLP, normalization is a process of mapping different variants of the same word type to a single string. In the Amharic language, some characters have the same sound and meaning but different shape and annotations for instance ['ሀ', 'ሐ', 'ሐ' and 'ሐ'], ['ፀ' and 'ፀ'], ['አ' and 'ዐ'], ['ሰ' and 'ሠ'] are the same, that represents ['hä'], ['ts'e'], ['a'], ['se'] respectively. This character normalization is done for Amharic but not in Ge'ez. For instance, the word “ሰአሊ/ፍላጊ” and “ሰዓሊ/ፍላጊ” have the same meaning, which means to “Draw” in Amharic⁴¹. However, in Ge'ez, the two words have no similar meaning, which has the meaning of “Beg” [feminine] and “Painter” respectively. Mainly the normalization function in Amharic selects homophone Ge'ez characters and replaces

⁴¹ However, the right way of writing the word 'Draw' is ሰዓሊ, which is

characters with the same pronunciation into a single character. Algorithm 4.2 in the following box shows the algorithm of normalization and basic cleaning tasks.

Algorithm 4.2 Algorithm for Amharic text normalization

```

DEFINE NORMALIZATION LIST (N) WITH VALUE (V) # E.g. ሃኅሐሐኻ=>ሀ, ኅሐኸ=>ሁ...ሠ=>ሰ..
OPEN AND READ AMHARIC CORPORA # This is input
WHILE READING LINE DO
FOR C IN LINE #Read each character from each line
 IF C IS IN N # if C is in a normalization list
 THEN
 REPLACE C WITH NORMALIZED VALUE (V) # E.g. if C is ኃ, ኅ or ኈ then replace C with ሀ V
 END IF
END WHILE
CLOSE FILE

```

4.4.3 Tokenization

Tokenization is the process of segmenting a given text into a piece of sentence, word, and sub-words. It includes separating words from running text. E.g. ዓ/ም or ዓም. to ዓመተ ምህረት, and segmenting a word into its sub-word elements.

We use *Sentencepiece* tokenizer for sub-word segmentation for the Transformer model. *SentencePiece* is an unsupervised text tokenizer and detokenizer designed primarily for Neural Network text generation systems and the vocabulary size is known before training the neural model⁴². *SentencePiece* implements sub-word units (e.g. byte-pair-encoding (BPE) and unigram language model) with the extension of direct training from raw sentences. It helps to make a purely end-to-end system that does not depend on language-specific pre or post-processing.

The sentence-piece model accepts a set of sentences, then chunks to word as a starting vocabulary to find out sub-words. The vocabulary is built starting from the characters. Then each possible sub-word created by concatenating characters is selected based on the frequency then appended to the vocabulary [66]. Later the most frequent sub-words are concatenated based on probability. Algorithm 4-2 Shows the segmentation process in the below box.

⁴² <https://github.com/google/sentencepiece> accessed on Jan 2021

Algorithm 4.3 Amharic sub-word segmentation Algorithm

```

#Amharic sub-word segmentation Algorithm
DEFINE TOKENIZATION (STRING_LIST: LIST[STR], INT K) -> VOCAB: LIST[STR]:
VOCAB = <LIST OF UNIQUE CHARACTERS IN STRING_LIST>
OPEN AND READ AMHARIC CORPORA #The Input
WHILE READING LINE DO
FOR I IN RANGE(0, K+1):
    C_LEFT,C_RIGHT=MOST FREQUENT PAIR OF ADJACENT TOKENS IN STRING_LIST
    C_NEW = C_LEFT + C_RIGHT # Create A new bigram
    VOCAB = VOCAB + C_NEW # Add the bigram to teh vocabulary
    REPLACE EACH OCCURENCE OF C_LEFT, C_RIGHT WITH C_NEW #Update the corpus
    RETURN(VOCAB)
END WHILE
CLOSE FILE

```

First, the segmentation model accepts the pre-processed sentence and then chunks it into words. Then we have a tokenized article that is ready for further processing. For example, in the sentences “ወጸዐኑ እከሎሙ ዲበ አእዱጊሆሙ ወኅለፉ እምህየ።”⁴³ [*wets'e ānu ikilomu dibe ā'idugīthomu wehālefu imihiye.*]” the possible sub-word segments are shown as follows⁴⁴.

Word: ['_ወጸ', 'ዐ', 'ኑ', '_እ', 'ከ', 'ሎሙ', '_ዲ', 'በ', '_አእ', 'ዱ', 'ጊ', 'ሆሙ', '_ወ', 'ኅለፉ', '_እምህየ', '::']

ID: [448, 51, 85, 59, 49, 260, 14, 673, 135, 430, 118, 5, 1957, 817, 4]

Here “_” marker is used to show the white space which is used as a delimiter for words. In addition, all possible sub-words are identified efficiently.

Two tokens, the start of the sentence (<S>) and end of the sentence (</S>)⁴⁵ are added to every pair of sentences during training which allows the model to know where to start and end translating and, we set the maximum sentence length to 32.

4.4.4 Padding

It is inserting zeros to the end of shorter sentences to make them equal to the longest one.

⁴³ እህላቸውን በአህዮቻቸው ላይ ጫኑ ከዚያም ተነሥተው ሄዱ። [*ihilachewini be'āhiyochachewi layi ch'ānu kezīyami tenešitewi hēdu*]

⁴⁴ Generated with pytorch Sentencepiece

⁴⁵ We used <SOS> and </EOS> as *Start of the sentence* and *End of the sentence* in Seq2Seq

4.5 Input Embedding

Embedding is an initial representation generated for each word of the input language to their corresponding numeric values for further process. In other words, Word Embedding is turning text into numbers. The embedding step happens in the most-bottom of the encoder layer. After embedding the words in the input sequence, each of them flows through each of the two layers of the encoder. Then, the encoder uses the embedding to generate the key, query, and value vectors for each of the words in Transformers. Then the encoder receives a list of vectors (the default size is 512) based on [47]. Though the size of this list is a hyper-parameter we can set, it would be the length of the longest sentence in the training dataset.

4.6 The Encoder

In the typical Seq2Seq encoder-decoder, the encoder converts the input sequence to an internal representation known as a *context or thought vector*, which the decoder uses to create the output sequence. Because there is no clear one-to-one relationship between the input and output sequences, the lengths of the input and output sequences can differ. Here the encoder and decoder are both made up of a stack of RNN, LSTM, or GRU units. It works in two steps. First, the LSTM in the encoder processes the entire input sentence and encodes it into a context vector, which is the final LSTM or RNN's hidden state. This should be an accurate summary of the input sentence. All the other encoder's intermediate stages are ignored, and the last state is taken as the decoder's initial hidden state. The primary flaw with this strategy is an event, that is the translation will be bad if the encoder makes a faulty context vector. When the encoder tries to grasp longer sentences, it provides a terrible summary. It's known as the RNN or LSTM *long-range dependency* problem.

In the Transformer Model, on the other hand, an encoder consists of a stack of $N = 6$ equal layers in which each layer has two major components: a self-attention mechanism and a feed-forward neural network. It receives a list of vectors as input and processes this list bypassing these vectors into a self-attention layer, then into a feed-forward neural network, then sends out the output upwards to the next encoder. The self-attention mechanism accepts input encodings from the previous encoder and weighs their relevance to each other to generate output encodings. The feed-forward neural network further processes each output encoding individually. These output encodings are then passed to the next encoder as its input, as well as to the decoders. The first encoder takes positional information and embedding of the input sequence as its input, rather than

encodings. The PoE (Positional Encoding) information is necessary for the transformer to make use of the order of the sequence, and to analyze the relationship between words.

By using the tensors embedding, the encoder processes the input to produce a context vector. The attention values of the input are passed to the feed-forward network and produce the encoded representation of the input sequence. After that, the resulted vector is inputted to the multi-headed self-attention layer. Next, the multi-headed self-attention is computed to calculate the attention score input sequence. Multi-headed self-attention is expressed by stacking of self-attention N times and calculated in parallel then concatenate. Thus, self-attention is calculated in terms of Scaled Dot-Product Attention and Multi-Head attention

The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. A residual connection around each of the two sub-layers, followed by layer normalization is employed. That is, the output of each sub-layer is $\text{LayerNorm}(x + \text{Sublayer}(x))$, where $\text{Sublayer}(x)$ is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $\text{model} = 512$. This is finally then sent to the decoder as input.

4.7 The Decoder

Similarly, as in the encoder, the decoder of the usual Seq2Seq architecture is made up of LSTM (or sometimes GRU) models. And this decoder's first state is initialized to the *last hidden states* of the Encoder. Using these initial states, the decoder starts generating the output sequence in the way that the input to the decoder at each time step is the output from the previous time step. In this way, it is passing the encoded meaning of the input sentence to the Decoder to be translated to a sentence in the output language. However, unlike the Encoder, the Decoder shall output a translated sentence of variable length. Therefore, the Decoder will output a prediction word at each time step until it has outputted a complete sentence. First, input a <SOS> tag as the input at the first time step in the Decoder. Just like in the Encoder, the Decoder will use the <SOS> input at time-step $t=1$ to update its hidden state. However, Instead of just going on to the next time step, the Decoder will use an additional *weight matrix* to create a probability over all of the words in the *output* vocabulary. In this way, the word with the highest probability in the output vocabulary will become the first word in the becoming predicted output sentence and continues until the <EOS> is predicted.

Whereas In the Transformer model, the decoder functions similarly to the encoder, but an additional attention mechanism and the masked multi-head attention are added in the decoder, which instead draws relevant information from the encodings generated by the encoders.

Like the first encoder, the first decoder takes positional information and embedding of the output sequence as its input, rather than encodings. The transformer must not use the current or future output to predict an output, so the output sequence must be partially masked to prevent this reverse information flow. The last decoder is followed by a final linear transformation and Softmax layer, to produce the output probabilities over the vocabulary. The General Structure of the encoder and decoder is depicted below in Figure 4.4 with two stacked encoders and one decoder.

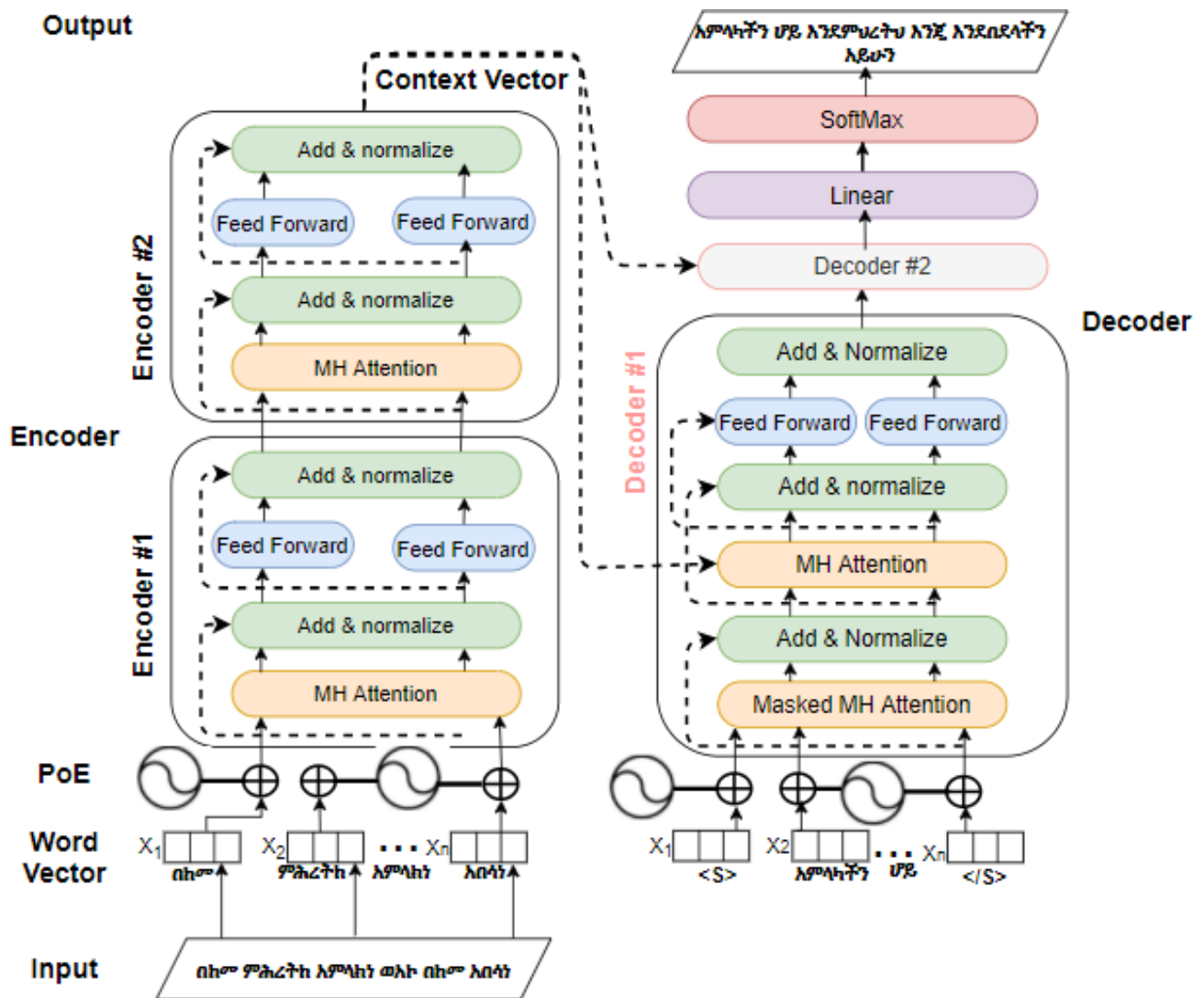


Figure 4.4 The detailed structure of Transformer's Encoder and Decoder architecture

4.8 The Evaluation

After the candidate models (The Seq2Seq, the Transformer, and the OpenNMT) have been trained with the available bi-lingual corpora, and they will be evaluated based on the BLEU score metrics. However, the type of experiment may differ from one to another by the hyperparameters. The corpus size and the ratio of dataset split may also be other distinct. The main experiment used for comparison and evaluation of all models is conducted with the 20,745 corpus size. Hence, we set a threshold value of a BLEU score of 20. If the result is less than the BLEU score of 20 it has to be re-experimented iteratively with a different percentage split and hyperparameter until it attained equal or more than 20 BLEU score. The hyperparameter includes the number of hidden layers, the dropout rate, and the optimizer. After re-experiment if the result is equal or greater than the threshold value it will be compared to the corresponding model and the better model will be selected, otherwise, the experiment will be conducted again. The ratio percentage split is changed between the minimum of 70% with 30% and the maximum of 90% with 10% training and testing sets respectively. If the dataset size is different from the main one (20,745) and the result is less than the threshold value, the experiment will be conducted again and again by changing the percentage split such as into 90% with 10%, 80% with 20%, and 70% with 30% training and testing sets respectively until no BLEU score increment of change is seen with the respective hyperparameters. Finally, the better model with a higher BLEU score will be selected. The linguist evaluation is applied after the better model has been selected at the end to ensure whether the BLEU score is reliable or not. We are not using the manual evaluation for model comparison and re-experiment.

CHAPTER FIVE: EXPERIMENT AND RESULT ANALYSIS

5.1 Overview

This chapter presents in detail how the experiments are conducted. It explains how the corpus used for this study was prepared and used, as well, the challenges during corpus preparation. The chapter also explains the hardware, software, and tools used for the experiment, the variety of conducted experiments, and the analysis of the result.

5.2 Corpus Preparation

Deep Learning needs a huge amount of data for training. Neural Machine Translation also uses deep learning techniques to teach itself to translate texts from one natural language to another. Most TensorFlow datasets such as “*wmt13_translate/fr-en*”⁴⁶ use at least 250,000 parallel sentences (44.65MiB) and equal or more than 40.8 million (40,827,433 parallel sentences, which is about 14.64GiB of data size). Hence, we need a very large amount of bilingual corpus to train the proposed model and for better translation qualities, as well, to be comparable with those international workshop translation tasks. However, this is unthought-of and infeasible to collect this much of bilingual corpus manually in the time available for this work, especially for low-resource languages such as Ge’ez and Amharic.

Even so, for this research, we tried to collect manually about 6,958 parallel sentences in addition to the available domain-specific dataset. The dataset includes domain-generic conversation sentences prepared from Ge’ez learning books such as “የግዕዝ ቋንቋ ንግግር ማስተማሪያ እና መግባቢያ (yegi’izi k’wanik’wa nigigiri masitemarīya ina megibabīya)” [70] and linguists, as well the remaining domain-specific parallel sentences were collected from a bible and battel of Saints. From the collected 6,958 corpora, the 3,021 paired sentences of the corpus were conversations and the remaining 3,937 were from the religious domain particularly from the ገድለ ቅዱስ ገብረ ክርስቶስ (*gedile k’idusi gebire kirisitos*). Moreover, an additional 3,078 parallel Ge’ez Latin numeric corpus was prepared to handle numeral translations such as “፫ቱ እደው ሐነዱ ቤተ። (3tu idewi hanets ’u bēte።)” which means “*The three boys built a house*”. This is discussed in experiment 4. in section 5.4. Hence, a total of 10,036 Ge’ez Amharic parallel corpora was newly added for this study.

⁴⁶ https://www.tensorflow.org/datasets/catalog/wmt14_translate accessed on May 26, 2021

The previous corpus, with 13,787⁴⁷ parallel sentences, was prepared by Tadesse Kassa [2] collected from different online Ethiopian Orthodox Church websites and Mahiber Kidusan repositories⁴⁸. The data set includes texts from the Old Testament of the Holy Bible.

We also added some extra domain-specific corpus from the bible and the battle of saints that were not included by the former researcher.⁴⁹ This includes ገድለ ቅዱስ ገብረክርስቶስ [*gedile k'idusi gebirekirisitosi*], the book of Baruch (Barok), and some parts of Enoch.

During corpus preparation, we followed a bottom-up approach. That is aligning first each books' verse level, merge the aligned books and finally merge all the books to the respective languages. After we collect the whole corpus, we used different python codes as stipulated in the system architecture and methodology parts for data cleaning purposes. The fragment code is depicted in Annex D ii, Figure D.2.

The corpus we have used for this experiment has three arrangements. In the first one, the Ge'ez and the Amharic sentences have been saved in different text files separately, named *ge.txt* and *am.txt*. We use this dataset for the Seq2Seq encoder-decoder (LSTM) Model. The second one is the Ge'ez and Amharic sentences were divided into six distinct files named *ge_train.txt*, *ge_test.txt*, *ge_valid.txt*, *am_train.txt*, *am_test.txt*, *am_valid.txt* (For Training, Testing, and validation sets respectively) with different percentage splits. We used this corpus to train the OpenNMT tool to compare findings with previous works. The last one is the corpus that contains the Ge'ez and the Amharic parallel sentences in one text file named *Geez_Amharic.txt* by separating the source and the target language sentences with the tab (\t) delimiter. Whereas, on the first two types of file arrangements, the sentence in the first line of say *ge_train.txt* will have its translated sentence in the first line of the *am_train.txt* and so forth. One of these corpus types is shown in Figure 5.1 below.

⁴⁷ Reported as 13,833 parallel sentences in their work, but we got 13787 instead, and we added 46 parallel sentences from new religious domain corpus to be comparable with their work.

⁴⁸ <https://www.ethiopicbible.com>, <http://ethiopianorthodox.org>, <http://eotcmk.org>, Accessed on May 26, 2021

⁴⁹ <https://www.stepbible.org/?q=version=Geez>, <https://www.tau.ac.il/~hacohen/Biblia.html> Accessed May 2021

a)	b)
500 እስማኤል ውለቱ	500 እስማኤል ነው
501 ስምዖ እስማኤል ውለቱ	501 ስሜ እስማኤል ነው
502 ስሙ ዚአዮ እስማኤል ይሰመደ	502 ዩኔ ስም እስማኤል ይባላል
503 አይቴ ውለቱ ዘመዳእከ ዚአክ	503 አንተስ ከየት ነው የመጣኸው
504 አንሰ ዘመዳእከ እምነ ኤርትራ ውለቱ	504 እኔስ የመጣሁት ከ ኤርትራ ነው
505 እምነ ኤርትራ ውለቱ ዘመዳእከ	505 ከኤርትራ ነው የመጣሁጥ
506 ውስተ ኤርትራ እኔሉ	506 ኤርትራ ውስጥ እየራሉሁ
507 ተአምሮ ዘብእሲ ውብእሲት	507 የወንድና የሴት ሰላምታ
508 በሐክ እንዮ	508 እንዴት አይርክ ወንድሜ
509 እግዚአብሔር ይሰባሕ በሐኪ እንትዮ	509 እግዚአብሔር ይመስገን እንዴት አይርክ እንጥ
510 መኑ ስመከ እሁዮ	510 ዩኔ ወንድም ስምህ ማን ይባላል
511 ኤልያስ እሰመደ	511 ኤልያስ እባላሉሁ
512 ስመዚአዮ ኤልያስ ውለቱ	512 ዩኔ ስም ኤልያስ ይባላል
513 ኤልያስ ይሰመደ ስምዮ	513 ስሜ ኤልያስ ይባላል
514 ወመኑ ስሙ ዚአኪ	514 ያንቺስ ስም ማን ነው
515 ኮከብ ውለቱ	515 ኮከብ ይባላል
516 አስቴር እሰመደ	516 አስቴር እባላሉሁ
517 ስመዚአዮ አስቴር ውለቱ	517 ዩኔ ስም አስቴር ነው
518 ኮከብ ይሰመደ ስምዮ	518 ስሜ ኮከብ ይባላል
519 እስፍንት ውለቱ እድሜኪ	519 እድሜሽ ስንት ነው
520 እስፍንት ውለቱ መዋዕልኪ	520 እድሜሽ ስንት ነው
521 እስፍንት ውለቱ እድሜኪ ኮከብ	521 ኮከብ እድሜሽ ስንት ነው
522 እስፍንት ውለቱ መዋዕልኪ አስቴር	522 እድሜሽ ስንት ነው አስቴር
523 እድሜዮ እስራ ወሀመስቱ ውለቱ	523 እድሜዮ ሃዖ አምስት ነው
524 እድሜዮ ፳፮ ውለቱ	524 እድሜዮ 25 ነው
525 ዘተወለድኩ በዐሥርቱ ወተስዓቱ ምዕት ተስዓ ወሰመንቱ ዓም	525 የተወለድኩት በአስራ ዘጠኝ መቶ ዘጠና ስምንት ዓም ነው
526 ዘተወለድኩ በ፲፱፻፲፰ ዓም	526 የተወለድኩት በ1989 ዓም ነው

Figure 5.1 Sample dataset from Ge'ez Amharic corpus in distinct files

Challenges during corpus preparation

During the preparation of the bilingual corpus, we encountered numerous difficulties such as Misaligned parallel sentence verses even from the previously available corpus, prepared for the earlier research work [2]. For example from *ge.txt* file line 2781 the Ge'ez verse says “ወተከዘ ኣካኣብ ወሖረ ወሰከበ ውስተ ምስካቢሁ ወተከድነ ገጾ ወኢበልዕ እክለ። [wetekeze āka’ābi weḥore wesekebe wisite misikabīthu wetekedine gets’o we’ībeli ‘ā ikile]” which means, “[Saddened, Ahab went to bed. He covered his face, He did not eat bread]” and the Amharic verse says “**ኢይዝራኤላዊውም ናቡቴ የአባቶቹን ርስት አልሰጥም ብሎ ስለ ተናገረው** አከዓብ ተቈጥቶና ተናድዶ ወደ ቤቱ ገባ በአልጋውም ላይ ተጋድሞ ፊቱን ተሸፋፈነ እንጀራም አልበላም። [īyizira’ēlawīwimi nabutē ye’ābatochēni risiti āliset’ihimi bilo sile tenagerewi āki’abi tek’ot’itona tenadido wede bētu geba be’āligawimi layi tegadimo fītuni teshefafene inijerami ālibelami]” instead of saying “አከዓብ ተከዘ ወደ አልጋውም ሄዶ ተኛ፤ ፊቱን ተሸፋፈነ፤ እንጀራም አልበላም። [āki’abi tekizo wede āligawimi hēdo tenya። fītuni teshefafene፤ inijerami ālibelami]”. Here the bolded phrase “**ኢይዝራኤላዊውም ናቡቴ የአባቶቹን ርስት አልሰጥም ብሎ ስለ ተናገረው**” is irrelevant. We found just such many mistranslated parallel sentences. We have found out that, this type of mistranslation is caused because the new King James Version of the Amharic Bible with the very old version of Ge'ez translation was used. Even though the idea is similar, the way of expression and writing verses is different, as shown in the above sentence. This type of mistranslation leads

the translation model to create undesirable translation patterns. As a result, to solve such a problem, we try to manually check each sentence in the text file line by line as much as possible with the help of Ge'ez linguist. Manual preparation of corpus is too tedious and time-consuming. However, we did not have any option other than validating the paired sentences manually, in order to add conversation sentences to make the corpus a little bit domain generic.

5.3 Experimental setup

SW and HW Tools Used for Experiment

During the experiment, we used the following hardware and software tools.

Table 5.1 Hardware and Software Requirements

No	Hardware	Software	Purpose
1.	HP laptop with Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz, x64-based processor. 8.00 GB RAM	Windows 10 Pro, 64-bit operating system Jupyter Notebook with Python 3.7	For corpus preparation and preprocessing, for writing the thesis report, and accessing Colab.
2.	GPU enabled 25 GB RAM (Colab ⁵⁰ , a cloud resource provided by Google)	Colab Jupyter Notebook with Python 3.7	For coding and experiment

We used the HP Laptop for corpus preparation and preprocessing, for writing the thesis report. The Colab, a Google cloud GPU, is used as the main laboratory environment for training the proposed model.

Moreover, different neural network modules and libraries such as Pytorch, and Tensor flow 2.0 are used to implement the proposed model and to support tensor-based data structure. Other libraries like Numpy, Scipy, Pandas, and Transformers are used to vectorize the data at the tensor level. Sample codes for our experiments are included under [Annex D \(i, ii, iii, iv\), Figure D.1, D.2, D.3](#).

⁵⁰ Google Co-laboratory

5.4 Experiments

We conducted two dual (four single) and two quadruple (eight single) experiments with different corpus sizes and algorithms. For this experiment, we used three distinct corpus types and formats.

- ❖ The old corpus collected from the previous researcher [2] has a size of 13,787 parallel Ge'ez-Amharic sentences and was a domain-specific dataset, which is completely from the bible. However, the researcher reported as 13,833 parallel sentences were used in their work, but we got only 13,787 instead, and we added 46 religious parallel sentences from the new prepared one to be comparable with their work.
- ❖ The old *preprocessed* 13,787 corpus plus the new conversation corpus, has a size of 6,958 and makes the new one 20,745 parallel sentences
- ❖ The 20,745 parallel sentences with the addition of a new numeric corpus having 3,078 lines, that contains Ge'ez numerals in a different writing system and their corresponding equivalent values in Latin number and Amharic text, which makes the new corpus 23,823. This numeral corpus was added to handle number translations.

The experiments are grouped based on the corpus type or size and the type of model. This is depicted in Figure 5.2 below.

Table 5.2 The different experiments, models, and corpus size used

No	Experiment	Model	Corpus size	Percentage Split
1	Experiment 1:	OpenNMT	13787(Old)	84%:8%:8% (T:T:V)
2	Experiment 2:	Transformer, OpenNMT	20,745(new)	80%:10%:10% (T:T:V)
3	Experiment 3:	Seq2Seq (LSTM with Attention)	20,745(new)	90%:10% (T:T)
4	Experiment 4:	Transformer, OpenNMT	23,823(new with numeric)	80%:10%:10% (T:T:V)

***NB:** *T:T:V* stands for Training, Testing, and Validation sets respectively

The Amharic corpora have 20,745 sentences having 248, 114 tokens (words), with an average of 12 words per sentence which has 1,217,793 characters without line ending, 39437 Types, with the longest sentence having 142 words and the shortest sentence with 1 word. Similarly, the Ge'ez corpora have 20,745 sentences having 242,946 words, with an average of 12 words per sentence and 1,166,242 characters without line ending, with the longest sentence with 152 words and the

shortest sentence with 1 word. The other corpus types and their experimental detail shown below in table 5.3.

Table 5.3 The length of sentence, token and type used for each experiment

	Total corpus					Percentage Split	Ge'ez			Amharic		
	Sentence	Token	Type	Token	Type		Sentence	Token	Type	Sentence	Token	Type
Experiment-1	13,833	163,321	34,229	166,131	37,672	Training	11,067	129,828	29,601	11,067	132,468	32,584
		788,652 characters		819,908 characters		Testing	1,383	16,657	4,860	1,383	16,574	5,052
						Validation	1,383	16,836	6,159	1,383	17,089	6,605
Experiment-2	20,745	242,946	36,050	248,114	39,437	Training	16,596	201,936	34,983	16,596	205,235	38,345
		1,166,242 characters		1,217,793 characters		Testing	2,075	26,630	9,594	2,075	27,936	10,115
						Validation	2,075	14,380	5,248	2,075	14,943	5,772
Experiment-4	23,823	248,551	38,047	253,733	40,489	Training	19,059	199,885	35,974	19,059	203,184	38,184
		1,190,746		1,236,735		Testing	2,382	30,300	10,177	2,382	31,344	10,740
						Validation	2,382	18,366	6,345	2,382	19,205	7,064

Evaluation Metrics: We used both automatic and manual evaluation metrics for this experiment. Mainly a standard BLEU score is used to evaluate our training results and to be comparable with the previous researches. However, we also used linguist manual evaluators after the models generate the final translated texts.

Experiment 1: In this experiment, we used a total of 13,833 parallel sentences (the old corpus with 13,787 with 46 added parallel sentences) to test how far the NMTs are from the typical SMT model reported by [2]. To conduct this experiment we used the OpenNMT model, with a standard *Sequential LSTM* neural network architecture. Moreover, the dataset was split into three different files⁵¹ as explained in section 5.2, with 84% training, 8% test, and 8% validation sets for each Ge'ez and Amharic language after trying different percentage splits. That means the corpus was split into 11619, 1107, and 1107 parallel sentences for training, testing, and validation sets respectively. This is why because the dataset we have used in the experiment is smaller than what the deep neural networks need. We have tried different dataset splitting ratios such as the common MT dataset split method with 80%, 10%, and 10% for training, testing, and validation respectively. However, the result was not that much better (a BLEU score less than 17) as the corpus size is

⁵¹ Namely *am-train.txt*, *am-test.txt*, and *am-test.txt* for Amharic as well, *ge-train.txt*, *ge-test.txt*, and *ge-test.txt* for Ge'ez corpus

small and the training set should have a sufficient amount of parallel sentences for the model. We observed that when the training and testing sets are too large (>15%) and too small (<5%) the model decreases its learning capability and results in poor translation quality.

In the beginning, before preprocessing and Amharic normalization have been applied, we have conducted the first experiment with the 13, 833 parallel sentences. After this uncleaned corpus has been feed to OpenNMT, we got BLEU scores of 1.2 and 1.9 from Ge'ez to Amharic and vice versa. On the other hand, after preprocessing and the Amharic normalization are applied, **15.79** and **16.94** are achieved for Ge'ez to Amharic and Amharic to Ge'ez with 2 hidden layers, 500 neurons, and 10000 training steps. During this experiment, we observed that the Neural Networks have an advisable training rate and translation quality than SMT models achieved by Tadesse [2] which were the BLEU scores of **15.14** and **16.15**. Even though it is difficult to compare both models with this small amount of corpus [23], the new sequential OpenNMT showed an amendment result over the previous SMT results with a BLEU score of **0.65** and **0.79** from Ge'ez to Amharic and Amharic to Ge'ez that is **2.46%** and **4.66%** improvement respectively.

The following snapshots, (a), (b), and (c) in Figure 5.2 shows the training and result of the experiment. More than 30 thousand vocabulary size, 2 hidden layers with 500 hidden neurons, a dropout of 0.3, and a SoftMax activation function were used as shown in Figure (a).

```

! onmt_train --data /NMT_Code/MyDrive/NMT_Code/OpenNMT-py/OldCorpus/model2am2ge/am-ge --save_model /NMT_Code/MyDrive/NMT_Code/
[2021-06-05 17:54:57,144 INFO] * src vocab size = 33920
[2021-06-05 17:54:57,144 INFO] * tgt vocab size = 30905
[2021-06-05 17:54:57,144 INFO] Building model...
[2021-06-05 17:55:08,598 INFO] NMTModel(
  (encoder): RNNEncoder(
    (embeddings): Embeddings(
      (make_embedding): Sequential(
        (emb_luts): Elementwise(
          (0): Embedding(33920, 500, padding_idx=1)
        )
      )
    )
    (rnn): LSTM(500, 500, num_layers=2, dropout=0.3)
  )
  (decoder): InputFeedRNNDecoder(
    (embeddings): Embeddings(
      (make_embedding): Sequential(
        (emb_luts): Elementwise(
          (0): Embedding(30905, 500, padding_idx=1)
        )
      )
    )
    (dropout): Dropout(p=0.3, inplace=False)
    (rnn): StackedLSTM(
      (dropout): Dropout(p=0.3, inplace=False)
      (layers): ModuleList(
        (0): LSTMCell(1000, 500)
        (1): LSTMCell(500, 500)
      )
    )
    (attn): GlobalAttention(
      (linear_in): Linear(in_features=500, out_features=500, bias=False)
    )
  )
)

```

(a)

Again, as shown in Figure (b), the training epoch is iterating every 50 steps, more than 10 thousand tokens are processed per second, and the accuracy of each iteration is calculated at each time step.

```

[2021-06-05 18:06:06,919 INFO] Step 9250/10000; acc: 77.26; ppl: 2.43; xent: 0.89; lr: 1.00000; 11009/11446 tok/s; 658 se
[2021-06-05 18:06:06,919 INFO] Loading dataset from /NMT_Code/MyDrive/NMT_Code/OpenNMT-py/OldCorpus/model2am2ge/am-ge.train.0.
[2021-06-05 18:06:07,236 INFO] number of examples: 11785
[2021-06-05 18:06:10,623 INFO] Step 9300/10000; acc: 79.16; ppl: 2.29; xent: 0.83; lr: 1.00000; 10313/10897 tok/s; 662 se
[2021-06-05 18:06:14,076 INFO] Step 9350/10000; acc: 78.92; ppl: 2.29; xent: 0.83; lr: 1.00000; 11289/12223 tok/s; 665 se
[2021-06-05 18:06:17,659 INFO] Step 9400/10000; acc: 78.56; ppl: 2.33; xent: 0.85; lr: 1.00000; 10651/11450 tok/s; 669 se
[2021-06-05 18:06:20,235 INFO] Loading dataset from /NMT_Code/MyDrive/NMT_Code/OpenNMT-py/OldCorpus/model2am2ge/am-ge.train.0.
[2021-06-05 18:06:20,548 INFO] number of examples: 11785
[2021-06-05 18:06:21,576 INFO] Step 9450/10000; acc: 78.78; ppl: 2.28; xent: 0.83; lr: 1.00000; 10183/10673 tok/s; 673 se
[2021-06-05 18:06:25,066 INFO] Step 9500/10000; acc: 78.99; ppl: 2.29; xent: 0.83; lr: 1.00000; 11190/11858 tok/s; 676 se
[2021-06-05 18:06:28,465 INFO] Step 9550/10000; acc: 81.27; ppl: 2.08; xent: 0.73; lr: 1.00000; 11010/11944 tok/s; 680 se
[2021-06-05 18:06:32,075 INFO] Step 9600/10000; acc: 79.31; ppl: 2.25; xent: 0.81; lr: 1.00000; 10896/11360 tok/s; 683 se
[2021-06-05 18:06:33,568 INFO] Loading dataset from /NMT_Code/MyDrive/NMT_Code/OpenNMT-py/OldCorpus/model2am2ge/am-ge.train.0.
[2021-06-05 18:06:33,887 INFO] number of examples: 11785
[2021-06-05 18:06:35,918 INFO] Step 9650/10000; acc: 79.59; ppl: 2.20; xent: 0.79; lr: 1.00000; 10155/10688 tok/s; 687 se
[2021-06-05 18:06:39,430 INFO] Step 9700/10000; acc: 79.54; ppl: 2.23; xent: 0.80; lr: 1.00000; 11140/12059 tok/s; 691 se
[2021-06-05 18:06:42,745 INFO] Step 9750/10000; acc: 82.37; ppl: 1.97; xent: 0.68; lr: 1.00000; 11114/11892 tok/s; 694 se
[2021-06-05 18:06:46,509 INFO] Step 9800/10000; acc: 78.79; ppl: 2.29; xent: 0.83; lr: 1.00000; 10911/11427 tok/s; 698 se
[2021-06-05 18:06:46,923 INFO] Loading dataset from /NMT_Code/MyDrive/NMT_Code/OpenNMT-py/OldCorpus/model2am2ge/am-ge.train.0.
[2021-06-05 18:06:47,230 INFO] number of examples: 11785
[2021-06-05 18:06:50,262 INFO] Step 9850/10000; acc: 81.24; ppl: 2.10; xent: 0.74; lr: 1.00000; 10260/10871 tok/s; 702 se
[2021-06-05 18:06:53,724 INFO] Step 9900/10000; acc: 80.21; ppl: 2.16; xent: 0.77; lr: 1.00000; 11220/12112 tok/s; 705 se
[2021-06-05 18:06:57,236 INFO] Step 9950/10000; acc: 80.33; ppl: 2.14; xent: 0.76; lr: 1.00000; 10804/11683 tok/s; 709 se
[2021-06-05 18:07:00,273 INFO] Loading dataset from /NMT_Code/MyDrive/NMT_Code/OpenNMT-py/OldCorpus/model2am2ge/am-ge.train.0.
[2021-06-05 18:07:00,581 INFO] number of examples: 11785
[2021-06-05 18:07:01,251 INFO] Step 10000/10000; acc: 80.28; ppl: 2.13; xent: 0.76; lr: 1.00000; 10021/10374 tok/s; 713 se
[2021-06-05 18:07:01,251 INFO] Loading dataset from /NMT_Code/MyDrive/NMT_Code/OpenNMT-py/OldCorpus/model2am2ge/am-ge.valid.0.
[2021-06-05 18:07:01,268 INFO] number of examples: 1001
[2021-06-05 18:07:01,767 INFO] Validation perplexity: 150172
[2021-06-05 18:07:01,767 INFO] Validation accuracy: 14.276
[2021-06-05 18:07:01,958 INFO] Saving checkpoint /NMT_Code/MyDrive/NMT_Code/OpenNMT-py/OldCorpus/model2am2ge/ethio-ge-am-model

```

(b)

Finally, after the training is completed, the model starts to predict translation patterns based on what it has learned before. Figure (c) shows some of the predicted Amharic output sentences of the model given a Ge'ez corpus as a source language.

```
[ ] [2021-05-18 20:23:50,286 INFO]
SENT 4: ['ወነሥሉ', 'ሎቱ', 'አንስቲያ', 'አምነ', 'አቀልደ', 'ከናአን', 'ሐዳሰ', 'ወለተ', 'ኤሎን', 'ቤተያዌ', 'ወኤሌማ', 'ወለተ', 'ሐና', 'ወልደ', 'ሴሴስ', 'ኤው]
PRED 4: ዓላው ከከነዓን ልጆች ሚስቶችን አገባ የኬጢያዊውን የዓሎንን ልጅ ዓዳን የአዊያዊው የፀብያን ልጅ ዓና የወለዳትን አሀሊባማን
PRED SCORE: -6.1937

[2021-05-18 20:23:50,286 INFO]
SENT 5: ['ወቤሌምት', 'ወለተ', 'ይስማኤል', 'አጎቱ', 'ለናቡኦት']
PRED 5: የእስማኤልን ልጅ የነባቶትን አጎት ቤሌምትን
PRED SCORE: -0.0904

[2021-05-18 20:23:50,286 INFO]
SENT 6: ['ወወለደቶ', 'ሐዳሰ', 'ለኤልፋዝ', 'ወቤሌምት', 'ወለደቶ', 'ለራጉኤል']
PRED 6: ዓዳ ለዓላው ኤልፋዝን ወለደች ቤሌምትም ራጉኤልን ወለደች
PRED SCORE: -1.1154

[2021-05-18 20:23:50,286 INFO]
SENT 7: ['ወኤሌማ', 'ወለደቶ', 'ለዮሐል', 'ወለይጉሜል', 'ወለቆሬ', 'አሉ', 'ደቄቆ', 'ለዓሳው', 'አለ', 'ተወልዱ', 'ሎቱ', 'በምድረ', 'ከናአን']
PRED 7: አሀሊባማም የዑስን የፅላምን ቆሬን ወለደች በከነዓን ምድር የተወለዱለት የዓላው ልጆች አነዚህ ናቸው
PRED SCORE: -0.7795

[2021-05-18 20:23:50,286 INFO]
SENT 8: ['ወነሥሉ', 'ዓላው', 'ደቄቆ', 'ወአንስቲያሁ', 'ወጠሎ', 'ነባሬ', 'ቤቱ', 'ወጠሎ', 'አንስሳሁ', 'ወጠሎ', 'ንቀዮ', 'ዘአጥረየ', 'በምድረ', 'ከናአን', 'ወተረ',
PRED 8: ዓላውም ሚስቱን ለውዝ ኤርምን ከሚበሉ እንጭቶች አርጥብ በትርን ወስዶ በበትሮቹ ውስጥ ያለው ነጭ አንዲታይ ነጭ ሽመልመሌ አድርጎ ላጣቸው
PRED SCORE: -5.6153

[2021-05-18 20:23:50,287 INFO]
SENT 9: ['እስመ', 'ብዙጎ', 'ንቀዮሙ', 'ወኢክሀሉ', 'ጎቡረ', 'ነቢረ', 'በአንተ', 'ብዝሃ', 'ንቀዮሙ']
PRED 9: በኃጢአተኛ ለሲደናውያንም አምላክ ለአስታሮት የያዕቆብም አምላክ ናቀ
PRED SCORE: -8.2754
```

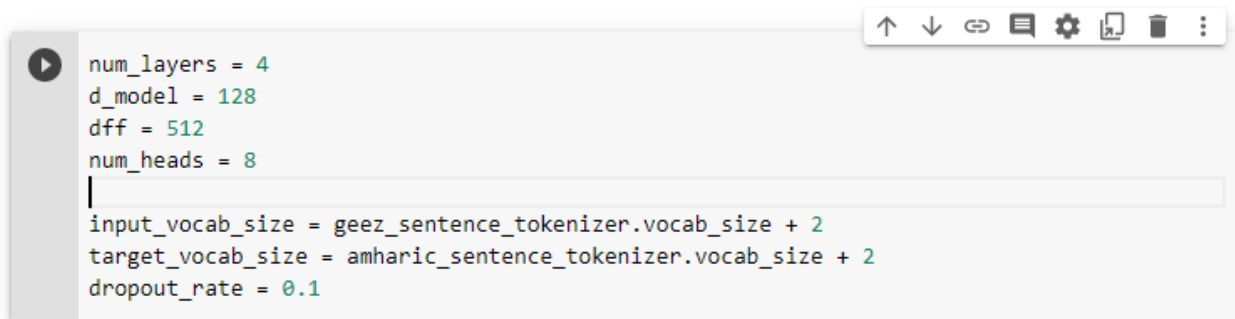
(c)

Figure 5.2 Snapshot of OpenNMT (a) training, (b) loss, and (c) output

Experiment 2: In this trial, we conducted two sub experiments with the written Transformer model and OpenNMT. The first experiment was conducted using the available optimum corpus (20,745)⁵² parallel sentences with a split ratio of 80% training, 10% of testing, and 10% of validation sets. That is an old preprocessed corpus plus new 6,958 added parallel sentences. The corpus is prepared under distinct single text files, labeled “Geez-Amharic.txt” and “Amharic-Ge’ez.txt”, having a parallel sentence with a “\t” delimiter in between both languages. We feed this datasets⁵³ for the state-of-the-art architecture, the *Transformer*, with the following hyper-parameters.

⁵² We put the corpus we have used for this study at this Git Address: [Geez-Amharic-DS/README.md at main · Amdework21/Geez-Amharic-DS \(github.com\)](https://github.com/Amdework21/Geez-Amharic-DS)

⁵³ In this thesis, dataset indicates a bilingual parallel Corpus.

A screenshot of a code editor window with a light gray background. The code is written in a dark font. The parameters are: num_layers = 4, d_model = 128, d_ff = 512, num_heads = 8, input_vocab_size = geez_sentence_tokenizer.vocab_size + 2, target_vocab_size = amharic_sentence_tokenizer.vocab_size + 2, dropout_rate = 0.1. The editor has a play button on the left and a toolbar with icons for back, forward, search, settings, print, and delete on the right.

```
num_layers = 4
d_model = 128
d_ff = 512
num_heads = 8
input_vocab_size = geez_sentence_tokenizer.vocab_size + 2
target_vocab_size = amharic_sentence_tokenizer.vocab_size + 2
dropout_rate = 0.1
```

Figure 5.3 Hyper-parameters of the Transformer model

The values of the parameters from the image are initial and we adjusted them with different values in different experiments. We overserved that once the number of layers (num_layers) is increased the training time and the result are increased as well.

We experimented with the Transformer model with the default parameters having only 2 hidden layers but shown smaller BLEU scores of **19.8** and **24.8** from Ge'ez to Amharic and Amharic to Ge'ez. Once the number of the hidden layers has increased the result is also increased. As a result, with 4 hidden layers, a BLEU score of **22.9** and **29.7** was achieved from Ge'ez to Amharic and vise versa. The state-of-the-art optimizer, Adaptive Moment Estimation (Adam) was used because it is the best among the adaptive optimizers and its simplicity of implementation. Besides, Rectified Linear Unit (ReLU) activation function is used as a threshold value in hidden layers and a Softmax activation function is used in the Decoders output. Because the ReLU function is faster to compute and is a general activation function that is used in most cases these days. As well, SoftMax is helpful in the output probabilities range.

The second experiment was conducted with OpenNMT using the same amount of corpus size. Why we used OpenNMT is because, it is a product-ready model, verified by many companies, and in order to ensure our model is acceptable or reliable based on its result.

As the OpenNMT requires separated text files for each training, testing, and validation sets, we split the corpus into three distinct files with 80%, 10%, and 10% for training, testing, and validation sets respectively as of in the first sub-experiment. Here the corpus size is a bit larger than the size of the corpus used in experiment 1 and we had to try a different percentage split ratio. Even so, we had tried to use the percentage split ratio of experiment 1 for this experiment. However, the result decreases and we have tried any other dataset split ratios up and down. The 80%, 10%, and 10% split for training, testing, and validation respectively attained better results.

clear reason for that; it could be the unbalanced percentage split of the dataset, the difference of unknown hyperparameters, and or the hardware facilities they run on.

Experiment 4: The last experiment was conducted with the addition of more numeric corpora on the available optimum corpus (The 20,745 parallel sentences). The numeric dataset was prepared manually from Ge'ez books and has 3,078 parallel Ge'ez-Latin numbers with their description. The numeric dataset looks like as shown in Figure 5.6. The aim of preparing these numeric corpora was to handle the numeric translations. Because Ge'ez language mostly uses numbers and numeral descriptions for days, names, and other quantitative things. For example, “እንዘ ፩ ፫ቱ ወእንዘ ፫ቱ ፩ ይህለሱ በአካላት ወይትወሐዱ በመለኮት።⁵⁴ (*enze 1 (āḥādu) šelesitu (3) we'inize 3tu (šelesitu) 1 (ahadu) yišēlesu be'ākālati weyitiweḥādu bemelekot*)” meaning (*When they are one, three, when they are three, one, When they are three in body, they become one in divinity*), and “፪ኤ⁵⁵ አንስት መጽአ እም ቤተክርስቲያን። (*2'ē ānisiti mets'i'a I'm bētekirisitiyan*)”, meaning (*two girls come from church*) are some instances of Ge'ez sentences with numeral description.

1	0	0	358	፻፲፩	111	358	፻፲፩	111	
2	አልቦ	ዛሮ	359	፻፲፩	111	359	፻፲፩	111	
3	፩	1	360	ምዕት	ዐወርቱ	ወአሐዱ	መቶ	አስራ	አንድ
4	አሐዱ	አንድ	361	፻፲፪	112	361	፻፲፪	112	
5	አሐደ	አንድ	362	፻፲፪	112	362	፻፲፪	112	
6	ዋሕድ	አንድ	363	ምዕት	ዐወርቱ	ወክልኢቱ	መቶ	አስራ	ሁለት
7	አሚሩ	መጀመሪያ	364	፻፲፫	113	364	፻፲፫	113	
8	አሚሩ	አንደኛ	365	፻፲፫	113	365	፻፲፫	113	
9	፩ይ	፩ኛ	366	ምዕት	ዐወርቱ	ወሠለስቱ	መቶ	አስራ	ሶስት
10	፩ይ	1ኛ	367	፻፲፬	114	367	፻፲፬	114	
11	ቀዳማይ	ቀዳማ	368	፻፲፬	114	368	፻፲፬	114	
12	ቀዳማይ	አንደኛ	369	ምዕት	ዐወርቱ	ወአርባዕቱ	መቶ	አስራ	አራት
13	፪	2	370	፻፲፭	115	370	፻፲፭	115	
14	ክልኤ	ሁለት	371	፻፲፭	115	371	፻፲፭	115	
15	ክልኤቱ	ሁለት	372	ምዕት	ዐወርቱ	ወገምስቱ	መቶ	አስራ	አምስት
16	ክልኤተ	ሁለተኛ	373	፻፲፮	116	373	፻፲፮	116	
17	ክልዳይ	ሁለተኛ	374	፻፲፮	116	374	፻፲፮	116	
18	ሠትዩ	ሁለተኛ	375	ምዕት	ዐወርቱ	ወስድስቱ	መቶ	አስራ	ስድስት
19	፪ይ	፪ኛ	376	፻፲፯	117	376	፻፲፯	117	
20	፪ይ	2ኛ	377	፻፲፯	117	377	፻፲፯	117	
21	፫	3	378	ምዕት	ዐወርቱ	ወሰብዐቱ	መቶ	አስራ	ሰባት
22	ሠለስቱ	ሰባት	379	፻፲፰	118	379	፻፲፰	118	
23	ሠለስተ	ሰባቱ	380	፻፲፰	118	380	፻፲፰	118	
24	ሠሎሱ	ሰባተኛ	381	ምዕት	ዐወርቱ	ወስምንቱ	መቶ	አስራ	ስምንት
25	ሣልሳይ	ሰባተኛ	382	፻፲፱	119	382	፻፲፱	119	
26	፫ይ	፫ኛ	383	፻፲፱	119	383	፻፲፱	119	
			384	ምዕት	ዐወርቱ	ወተስባቱ	መቶ	አስራ	ዘጠኝ

Figure 5.6 The numeric corpus

After adding the numeric dataset to the available corpus, We conducted two separate experiments with the developed Transformer model and the OpenNMT model using similar hyperparameters

⁵⁴ እንዘ አሐዱ ሠለስቱ ወእንዘ ሠለስቱ አሐዱ ይህለሱ በአካላት ወይትወሐዱ በመለኮት። => አንድ ሲሆን ሶስት፣ ሶስት ሲሆኑ አንድ። በአካል ሶስት ሲሆኑ በመለኮት አንድ ይሆናል። ማለት ነው።

⁵⁵ It reads like ክልኤ (*kili'ē*) which means *two*

Table 5.4 The proposed Models and their BLEU score result

Model	Corpus size	Ge'ez to Amharic	Amharic to Ge'ez	Hidden size of encoder and decoder	GPU Time	CPU Time
OpenNMT	13833(Old)	15.79	16.94	2	18m 8s	-
	20,745(new)	21.7	28.2	4	20m 25s	-
	23,823(new with numeric)	11.2	18.4	4	13m 54s	-
Sequential (LSTM with Attention)	20,745(new)	19.3	23.4	4 with 440 hidden neurons	30m 7s	10h 46m 33s
Transformer	20,745(new)	19.8	24.8	2	18m:13s:46ms	-
		22.9	29.7	4	21m:15s:52ms	-
	23,823(new with numeric)	16.5	20.1	4	28m:23s:12ms	-

Again Table 5.4 below depicts the four experiments and their corresponding results arranged based on the type of experiment conducted.

Table 5.5 The four Experiments and their BLEU score result

No	Experiment	Model	Ge'ez to Amharic	Amharic to Ge'ez	Hidden size	Corpus size	Percentage Split (T:T:V)	GPU Time	CPU Time
1	Experiment 1:	OpenNMT	15.79	16.94	2	13833(Old)	80%:10%:10%	18m 8s	-
2	Experiment 2:	Transformer	19.8	24.8	2	20,745(new)	80%:10%:10%	18m:13s:46ms	24.8
			22.9	29.7	4			21m:15s:52ms	
		OpenNMT	21.7	28.2	4	20,745(new)	80%:10%:10%	20m 25s	-
3	Experiment 3:	Seq2Seq (LSTM with Attention)	19.3	23.4	4	20,745(new)	90%:10%(T:T)	0h 30m 7s	10h 46m 33s
4	Experiment 4:	Transformer	16.5	20.1	4	23,823(new with numeric)	80%:10%:10%	28m:23s:12ms	-
		OpenNMT	11.2	18.4	4	23,823(new with numeric)	80%:10%:10%	13m 54s	-

Some of these experimental results are depicted furthermore on the following histogram in Figure 5.8. The histogram shows the BLEU scores of the experiments conducted with 20,745 corpus size.

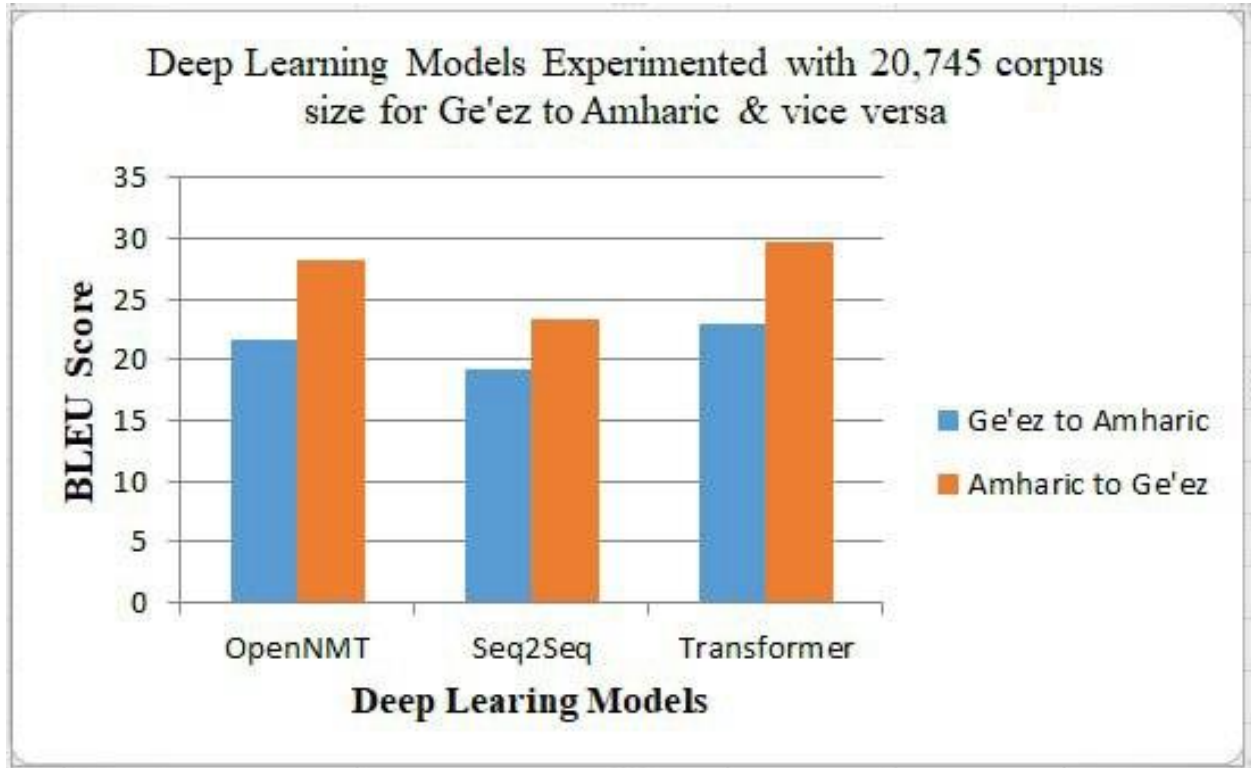


Figure 5.8 Comparison of Deep Learning models

5.5 Linguist Evaluation

As we have already noted in the experiment (Section 5.4), we used a standard BLEU score evaluation metric to evaluate translated results of the proposed model. Besides this automatic evaluation, the linguist manual evaluation is also applied in order to ensure whether the proposed model is acceptable or not. Hence, we asked five Ge'ez language experts (2 Merigetas, 1 priest, and 2 deacons) selected purposively, to evaluate the translated Ge'ez Amharic texts by the OPenNMT, Seq2Seq, and Transformer models. These scholars were selected purposively because they know Ge'ez and Amharic languages very well and sufficient knowledge of Ge'ez was required for preparing Ge'ez and Amharic Corpus, evaluating results, and corrective actions. Table 5.5 below shows how the translated texts are prepared for manual evaluation as well Points of Translation Quality (PTQ) and their corresponding values are listed below.

1. ምንም አልተተረጎመም = 0 (Nothing Translated = 0) Or Bad Translation (abbreviated as *NT*)
2. ብዙው አልተተረጎመም = 1 (Most not Translated = 1) (abbreviated as *MnT*)

3. በከፊል አልተተረጎመም =2 (Partially not Translated = 2) (abbreviated as *PnT*)
4. ተተርጉሟል ማለት ይቻላል =3 (Almost Translated = 3) (abbreviated as *AT*)
5. በትክክል ተተርጉሟል=4 (Correctly Translated = 4) Or Nice Translation (abbreviated as *CT*)

The snapshot of a sample manual evaluation is shown under [Annex E, Figure E.1, and E.2](#).

Table 5.6 Parallel sentences prepared for manual evaluation (Translated by OpenNMT)

በOpenNMT የተተረጎሙ ዓ/ነገሮች (Sentences Translated via OpenNMT)	የትርጉም ጥራት					Average (4pts)
	Points of Translation Quality (PTQ)					
	NT (0)	MnT (1)	PnT (2)	AT (3)	CT (4)	
ግብዓት626: ['ባለ', 'ወንግሮ', 'ለፈርዖን', 'ንጉሠ', 'ግብጽ', 'ከመ', 'ይፈንዎሙ', 'ለደቂቀ', 'አስራኤል', 'እምድሩ'] ውጽዓት626: ግባ የአስራኤልንም ልጆች ከአገሩ ይለቅቅ ዘንድ ለግብፅ ንጉሥ ለፈርዖን ንገር				2 (40%)	3 (60%)	3.6 (90%)
ግብዓት627: ['ወተናገረ', 'ሙሴ', 'ወይቤ', 'ቅድመ', 'እግዚእ', 'ናሁ', 'ደቂቀ', 'አስራኤል', 'ኢስምዑኒ', 'ፈርዖን', 'አፎ', 'ይሰምዐኒ', 'ወአነ', 'በሃም'] ውጽዓት627: ሙሴም በእግዚአብሔር ፊት እነሆ የአስራኤል ልጆች አልሰሙኝም እንዴትስ ፈርዖን ይሰማኛል ይልቁንም እኔ ከንፈረ ቁላፍ ነኝ ብሎ ተናገረው				2 (40%)	3 (60%)	3.6 (90%)
ግብዓት628: ['ወይቤሎሙ', 'እግዚእ', 'ለሙሴ', 'ወለአሮን', 'ወአዘዙሙ', 'ይባልዎ', 'ለፈርዖን', 'ንጉሠ', 'ግብጽ', 'ያውጽአሙ', 'ለደቂቀ', 'አስራኤል', 'እምድረ', 'ግብጽ'] ውጽዓት628: እግዚአብሔርም ሙሴንና አሮንን በኤዶምያስ ምድር በኤዶምያስ ዳርቻ ባለው በሐር ተራራ ናችሁ አላቸው			2 (40%)	3 (60%)		2.6 (65%)
ግብዓት629: ['ወአሉ', 'እሙንቱ', 'መላእክት', 'በበ', 'ቤተ', 'አበዊሆሙ'] ውጽዓት629: የአባታቸውም ቤት አለቆች እነዚህ ናቸው የአስራኤል የበተር ልጅ			1 (20%)	3 (60%)	1 (20%)	3 (75%)
ግብዓት630: ['ደቂቀ', 'ሮቤል', 'በኩሩ', 'ለአስራኤል', 'ሄና', 'ወፍሉሶ', 'ወአስሮን', 'ወከርሚ', 'ዘውአቱ', 'ትውልዲሁ', 'ለሮቤል'] ውጽዓት630: የሮቤል ልጆች ሄናን ፈሉስ አስሮን ከርሚ እነዚህ የሮቤል ልጆች ናቸው			2 (40%)	2 (40%)	1 (20%)	2.8 (70%)
ግብዓት631: ['ደቂቀ', 'ስምዖን', 'የምኤል', 'ወያሚን', 'ወአኦድ', 'ወያክን', 'ወሳኦር', 'ወሰኦል', 'ዘእምነ', 'ፈኒስ', 'ከናናዊት', 'ዘውአቱ', 'ትውልዱ', 'ለስምዖን'] ውጽዓት631: የኤልፋዝም ልጆች እነዚህ ናቸው ስማቸውም የይሁዳ ልጆች እነዚህ ናቸው የዲሶንም ልጆች እነዚህ ናቸው	1 (20%)	3 (60%)	1 (20%)			1 (25%)
Total Average	1 (3.3%)	3 (10%)	6 (20%)	12 (40%)	8 (26.7%)	2.766 (69.15%)

*The average 69.15% is only for the six sentence pairs of this table.

Average is calculated as shown in equation 2.22 below.

$$Average = \frac{\sum_{i=0}^n (PTQ * ValueGivenByEvaluator)}{noOfTotalEvaluators (5 \text{ here})} \quad (2.25)$$

For example, in Table 5.6, the average of the “ግብዓት881” could be calculated as 1*3 +4*4 (1 evaluator gave it 3 or AT and 4 evaluators gave it 4 or CT) divided by the total number of evaluators, which is 5 and 19/5 will be 3.8.

Table 5.7 Parallel sentences prepared for manual evaluation (Translated by Transformer)

በ Transformer የተተረጎሙ ዓ/ገገሮች (Sentences Translated via Transformers)	የትርጉም ጥራት					Average (4pts)
	Points of Translation Quality (PTQ)					
	NT (0)	MnT (1)	PnT (2)	AT (3)	CT (4)	
ግብዓት881: <ሕዝቡንም ለሦስተኛው ቀን ተዘጋጁ ወደ ሴቶቻችሁ አትቅረቡ አለ> ውጽዓት881: ወይቤ ለሕዝቡ ተደለዉ ለሦሎስ ዕለት ወኢትቅረቡ አንስተ				1 (20%)	4 (80%)	3.8 (95%)
ግብዓት882: <ግብፃውያንም በፈርዖንና በሰረገሎቹ በፈረሰኞቹም ላይ ክብር ባገኘሁ ጊዜ እኔ እግዚአብሔር እንደ ሆንሁ ያውቃሉ> ውጽዓት882: ወያእምሩ ከሱ ግብጽ ከሙ አነ እግዚአብሔር ተሰቢሕየ በፈርዖን ወበሰረገላሁ ወበከሱ አፍራሲሁ				2 (40%)	3 (60%)	3.6 (90%)
ግብዓት883: <ሙሴም የአማቱን ቃል ሰማ ያለውንም ሁሉ አደረገ> ውጽዓት883: ወሰምዐ ሙሴ ቃለ ሐሙሁ ወገብረ				2 (40%)	3 (60%)	3.6 (90%)
ግብዓት884: <ዮቶርም ከግብፃውያንና ከፈርዖን እጅ ያዳናችሁ ከግብፃውያንም እጅ ሕዝቡን ያዳነ እግዚአብሔር ይባረክ አለ> ውጽዓት884: ወይቤ ዮቶር ቡሩክ እግዚአብሔር ዘእድነነ ሕዝቡ እምእደ ግብጽ ወእምእደ ፈርዖን				3 (60%)	2 (40%)	3.6 (90%)
ግብዓት885: <አቤቱ ቀኝህ በኃይል ከበረ አቤቱ ቀኝ እጅህ ጠላቶችን በተናቸው> ውጽዓት885: የማንክ እግዚአ ተሰብሐ በኃይል የማነ እዴክ እግዚአ ሠረወቶሙ ለፀር					5 (100%)	4 (100%)
ግብዓት886: <እግዚአብሔር ተዋጊ ነው ስሙም እግዚአብሔር ነው> ውጽዓት886: እግዚአብሔር ይቀጠቅጥ ፀብአ ወእግዚአብሔር ስሙ				1 (20%)	4 (80%)	3.8 (95%)
Total Average				9 (30%)	21 (70%)	3.733 (93.33%)

*NB: These two tables have only six sample sentence pairs, which are selected randomly from the models' output. However, the total number of sentence pairs from each model evaluated by Ge'ez linguist is 20 (In the beginning 100 pairs of sentences were prepared but minimized to 20 based on the evaluators' comment). Accordingly, the OpenNMT, the Transformer, and the Seq2Seq (Encoder-Decoder) models achieved an average of 81.3%, 83.2%, and 63.1% of translation quality from Ge'ez to Amharic and 85.2%, 86.7%, 65.5% from Amharic to Ge'ez respectively. The outputs of the first two models are taken from experiment 2 and that of the Seq2Seq is taken from experiment 3. This is depicted in Table 5.5 below.

Table 5.8 Manual evaluation results compared with the BLEU score

No	Evaluation Type	OpenNMT (Ex2) G-A and A-G	Transformer (Ex2) G-A and A-G	Seq2Seq (Ex3) G-A and A-G
1	BLEU	21.7 and 28.2	22.9 and 29.7	19.3 and 23.4
2	Manual	81.3% and 85.2%	83.2% and 86.7%	63.1% and 65.5%

*G-A and A-G stands for Ge'ez to Amharic and Amharic to Ge'ez respectively

As shown in table 5.7 the Transformer model achieved the higher result on both automatic and Linguist evaluation. It improved the translation quality by 2.28% and 1.73% of the OpenNMT as well, it improved the translation by 24.15% and 24.45% of the Seq2Seq encoder-decoder model.

5.6 Answering Research Questions

At the beginning of this work, we have formulated three research questions to be answered after the experiment, hence, here is their answer with the findings of the study.

Q1: “How effective are Deep Learning NMT models in MT for low resourced languages such as Ge’ez and Amharic?” Based on the experiments, we have seen that deep learning models are effective and efficient for low-resourced languages such as Ge’ez and Amharic, but sensitive to hyper-parameters and quality of corpus. As compared to the previous works using the same dataset, the deep learning models achieved the promising result by improving over 0.65 and 0.79 BLEU scores from Ge’ez to Amharic and Amharic to Ge’ez in OpenNMT. Which was an improvement of 2.46% and 4.66% respectively.

Q2: “Which NMT algorithm is better for low resource languages like Ge’ez and Amharic?” Section 5.4 discussed different experiments with two state-of-the-art models namely: The Seq2Seq or encoder-decoder (LSTM) with the help of attention and the Transformer model. The seq2seq (encoder-decoder) models are two types: the OpenNMT tool and the Seq2Seq, written by hand. From those models, the Transformer model outperforms the other. we have said that the capability of Transformers in different subsets of the IWSLT¹⁴ training data showed that their effectiveness under low-resource languages is highly dependent on the hyper-parameter settings [67]. As well, we have observed in our experiments, the Transformer model is better for such low-resourced languages if there is a quality corpus and more hidden layers for training.

Q3: What are the main challenges of translation between Ge’ez and Amharic? This is answered below in section 5.7.

5.7 Challenges of Ge'ez and Amharic during machine translation

We faced different challenges during an experiment of Ge'ez and Amharic MT. Some of them are: **Metaphors or Proverbs:** Such as “አንበሳ ነህ የኔ ልጅ!” (*ānibesa nehi yenē liji!*) perhaps translated as, “*You are a Lion! My son*” intended to say, “You are brave”. Such translations of metaphors are yet not handled even by Google translate⁵⁶. A large Ge'ez-Amharic and vice versa proverb corpus might be required to handle such translations.

Smooth and accentuated words: There are words, which have a different meaning when they are spoken strongly and smoothly in both Ge'ez, such as መካን (*mekan*), and Amharic such as ገና (*gena*). These words are not comfortable for text translation, as they have a different meaning when they are spoken. E.g. the sentence “ገና ሲመጣ ማውራት ጀመሩ” (*gena sīmet'a mawirati jemeru*) can be translated as “as soon as he came, they began talking” or “*when X-mass came, they began talking*” based on the context and the stress of sound. This type of translation is not handled on text translation; even Google Translate suffered from miss translating these types of sentences⁵⁷.

Improper use of Ge'ez similar letters: section 3.1.3 shows similar (ተመኩሳይያን) Letters of the Ge'ez language, and they make words such as መሀረ → አስተማረ and መሐረ → ይቅር አለ to have a different meaning. We found words with improper use of characters in Ge'ez corpora. We have tried to search and replace with the correct words as much as we can. Yet, there might be such words existing, that lead the model to train in a wrong way.

Translation of Ge'ez numerals: As discussed in section 5.4, experiment 4, we have tried to handle Ge'ez numeral translation by adding 3,078 parallel Ge'ez-Latin numbers. However, the translation result was decried dramatically. We suggested this as future work for the coming researchers.

Corpus quality: Deep learning models are sensitive to hyperparameters and the training data they have fed. However, as discussed in subsection 5.2.1 the corpus collected from the previous researcher lacks quality. Though we have tried to fix this issue manually with linguists line by line, it was inefficient and time-consuming, we have been unable to do the whole corpus. This makes the training challenging for the models.

Handling of Punctuation: Because punctuations of both languages have been removed from the corpus during preprocessing (before training), we are not able to use them on the trained model.

⁵⁶ The sentence “ሚስቱ ንብ ናት።” (*mīsitu nibi nati።*)” is translated as “*His wife is a bee*”, intended to say “Clever”

⁵⁷ The sentence “ገና ሲመጣ ማውራት ጀመሩ” as “*When he arrived, they began talking*”. It didn't understand the word “ገና” (*gena*) is referring to X-mass, unless it is preceding to the word “በዓል” (*be'al*) indicating the festival.

CHAPTER SIX: CONCLUSION AND RECOMMENDATION

6.1 Overview

This chapter finalizes the whole work and gives a general conclusion about the study, contribution, and findings of the work, and suggests some feature works for the upcoming researchers.

6.2 Conclusion

From the first chapter until now, we have seen a vast concept about machine translation and the process of text translation from one language to another (Ge'ez to Amharic). Although there is still work to be done, various researches including this study show that deep learning methods have performed better quality than other machine translation approaches so far. The purpose of this study was to design and develop a bidirectional Ge'ez-Amharic machine translation using deep-learning-based Neural Machine Translation models. From the available NMT models, the Transformer model is the state of the art and has a promising result over the other. Since deep learning algorithms need a huge amount of data, a talked corpus was manually prepared and added to the previously available dataset so it has played a great role in higher translation quality.

Four major experiments were conducted and results were recorded for all translations with BLEU score metrics. In addition, questionnaires were prepared for Ge'ez and Amharic linguists to evaluate the translated texts between the two languages via the proposed models. In the automatic evaluation, the Transformer model achieved a higher result than the other models. Particularly, it improved the translation quality by 1.2 and 1.5 BLEU scores from Ge'ez to Amharic and vice versa over the OpenNMT model. Moreover, with the human manual evaluation, the transformer model outperforms the OpenNMT with 5.4% translation quality.

6.3 Contribution of the Study

Even though the models trained in this study are not deployed and released (hosted) for use, we think this study contributes numerous values for the Ge'ez and Amharic languages, their users, and the coming researchers. We found that the Deep Learning models are good in modeling MT algorithms for low-resourced languages such as Ge'ez and Amharic.

Adding about 6,958 conversations and domain-specific, as well, 3,078 numeric datasets (Total 10,036) to the existing one is also another contribution.

6.4 Future Works and Suggestions

Though our proposed model performs well on the provided corpus, it needs further improvement to reach human-level translation. As we have noted from the above chapters, deep learning requires a huge amount of dataset for training. In contrast, the corpus taken for this study was not enough and cannot represent the languages. Hence, future researches should be conducted using a larger set of corpus for better quality. If the corpus used for this study were large enough, well prepared, and properly reviewed the proposed model would score higher results than the achieved.

The following areas could be explored further as a continuation of this study.

- ❖ Experimenting either by adding much more Ge'ez-Latin numeral corpora to the existing one or by adding a real-world conversation corpus having those numbers in them may enhance the handling of numeral translation.
- ❖ The conversation corpus added for this study was too small as compared to the total. Hence the model is biased to the dominating corpora, adding more talked corpus may improve the translation quality.
- ❖ Using other MTs with deep learning (hybrid) may enhance the translation result.
- ❖ Adding multi-domain corpus for Ge'ez using OCR, Speech to Text, and manual writing from movies, news, and other talked conversions can enhance the quality of translation.
- ❖ Preparing a large Ge'ez-Amharic proverb and metaphor corpus can be used to handle proverbs and metaphors
- ❖ Next Researchers should try to add Geez language in Google Translate.
- ❖ Applying semi-supervised approaches such as adding a monolingual corpus for each language can improve the translation quality.

References

- [1] E. Teshome, "Bidirectional English-Amharic Machine Translation: An Experiment using Constrained Corpus," Masters Thesis submitted to Addis Ababa University, Addis Ababa, 2015.
- [2] T. Kassa, "Morpheme-Based Bi-directional Ge'ez -Amharic Machine Translation, Masters Thesis Submitted to Addis Ababa University," Addis Ababa University, Addis Ababa, 2018.
- [3] S. I. A. aTilde, "Improving SMT for Baltic languages with factored models," *In Human Language Technologies: The Baltic Perspective: Proceedings of the Fourth International Conference, Baltic HLT IOS Press*, vol. 219, p. 125, 2010.
- [4] M. D. Okpor, "Machine Translation Approaches: Issues and Challenges," *International Journal of Computer Science Issues*, vol. 11, no. 5, pp. 159-165, 2014.
- [5] "omniscien," 20 May 2021. [Online]. Available: <https://omniscien.com/blog/migrate-from-smt-to-nmt/>. [Accessed 26 May 2021].
- [6] M. L. Forcada, "Making sense of neural machine translation," *John Benjamins Publishing Company*, p. 291–309, 2017 (also available as: <http://orcid.org/0000-0003-0843-6442>).
- [7] M. HARGRAVE, "Investopedia," investopedia, 30 April 2019. [Online]. Available: <https://www.investopedia.com/terms/d/deep-learning.asp>. [Accessed 17 December 2019].
- [8] C. Nicholson, "Pathmind," Pathmind, 21 Jan 2018. [Online]. Available: <https://pathmind.com/wiki/neural-network>. [Accessed 23 November 2019].
- [9] Missinglink.AI, 12 Feb 2018. [Online]. Available: <https://missinglink.ai/guides/neural-network-concepts/perceptrons-and-multi-layer-perceptrons-the-artificial-neuron-at-the-core-of-deep-learning/>. [Accessed 21 Sep 2019].
- [10] C. K. GN, "Medium," Medium.com, 31 August 2018. [Online]. Available: <https://medium.com/@chethankumargn/artificial-intelligence-definition-types-examples-technologies-962ea75c7b9b>. [Accessed 17 December 2019].
- [11] W. Leslau, "Introductory Grammar of Amharic.," *Wiesbaden: Harrassowitz*, 2000.
- [12] ኪ. ወ. ክፍሌ, መጽሐፈ ሰዋሰው ወግስ ወመዝገብ ቃላት ሐዲስ, ኡዲስ አበባ: ኦርቲስቲክ ማተሚያ ቤት, 1948.
- [13] merriam-webste, "merriam-webste," merriam-webste, 1828. [Online]. Available: <https://www.merriam-webster.com/dictionary/geez>. [Accessed 10 October 2019].
- [14] M. T. Bame, "Transformer Based Amharic Headline Generation Using Sub-word2Vec Representation," A Thesis Submitted to the School of computing in Partial Fulfilment for the Master of Science in Information Technology, Jimma University, Jimma, 2020.
- [15] W. L. Belcher, "wendybelcher," 23 June 2018. [Online]. Available: <https://wendybelcher.com/african-literature/african-language-literature/ethiopic-language-courses/>. [Accessed 26 May 2021].
- [16] B. Abel, "Geez to Amharic Machine Translation," Masters Thesis Submitted to Addis Ababa University, Addis Ababa, 2018.
- [17] D. Mulugeta, "Geez to Amharic Automatic Machine Translation: A Statistical Approach, Masters Thesis Submitted to Addis Ababa University," Addis Ababa, 2015.

- [18] opennmtGroup, "opennmt.net," opennmt, [Online]. Available: <http://opennmt.net>. [Accessed 11 October 2019].
- [19] H. P. C. D. M. Minh-Thang Luong, "Effective Approaches to Attention-based Neural Machine Translation," -, 2015.
- [20] A. Kaufmann, Using Natural Language Processing to Support Interview Analysis. PhD diss., Master Thesis, Friedrich-Alexander University Erlangen-Nürnberg: Department of Computer Science Lanham, ML, 2014.
- [21] T. B. II, Applied Natural Language Processing with Python, San Francisco, California, USA: Apress Media LLC: Welmoed Spahr, 2018.
- [22] M. A. Chéragai, "Theoretical Overview of Machine translation," *Proceedings ICWIT*, vol. 1, pp. 160-169, 2012.
- [23] "Tilde," tilde, 2017. [Online]. Available: <https://www.tilde.com/about/news/316>. [Accessed 22 September 2019].
- [24] S. T. a. J. K. Sarkhel, "Approaches to machine translation," *Annals of Library and Information Studies*, vol. 57, pp. 388-393, 2010.
- [25] P. F. D. P. S. A. D. P. V. J. & M. R. L. Brown, "The mathematics of statistical machine translation: Parameter estimation," *Computational linguistics*, 19(2), vol. 2, pp. 63-311, 1993.
- [26] N. a. B. P. Kalchbrenner, "Recurrent continuous translation models," *Association for Computational Linguistics*, no. Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1700-1709, 2013.
- [27] O. V. Q. V. L. Ilya Sutskever, "Sequence to Sequence Learning with Neural Networks," *In Advances in neural information processing systems*, p. 3104–3112, 2014.
- [28] M. S. Z. C. Q. V. M. N. Yonghui Wu, "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," *arXiv:1609.08144v2 [cs.CL]*, vol. 2, 8 Oct 2016.
- [29] C. J. Dale Janssen, "TechoPedia," techopedia, 15 January 2019. [Online]. Available: <https://www.techopedia.com/definition/5967/artificial-neural-network-ann>. [Accessed 17 December 2019].
- [30] Andovar, 21 July 2018. [Online]. Available: <https://blog.andovar.com/machine-translation>. [Accessed 13 November 2019].
- [31] D. P. Lambert, "Iconic," Iconic, February 2019. [Online]. Available: <https://iconictranslation.com/2019/10/issue-55-word-alignment-from-neural-machine-translation/>. [Accessed 12 May 2020].
- [32] I. S. Q. V. L. O. V. W. Z. Minh-Thang Luong, "Addressing the Rare Word Problem in Neural Machine Translation," *arXiv preprint arXiv 1410.8206*, 2015.
- [33] R. K. Philipp Koehn, "Six Challenges for Neural Machine Translation," *arXiv:1706.03872v1 [cs.CL]*, vol. 1, 12 Jun 2017.
- [34] G. L. L. L. M. M. S. S. Xintong Li, "On the Word Alignment from Neural Machine Translation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics*, Florence, Italy, August 2, 2019.

- [35] G. E. S. O. a. Y.-W. T. Hinton, "A fast learning algorithm for deep belief nets," *Neural computation* 18.7, pp. 1527-1554, 2006.
- [36] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Technical Report IDSIA-03-14 arXiv:1404.7828v4*, vol. 4, p. 85–117, 8 Oct 2014.
- [37] K. S. & A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR arXiv:1409.1556v6 [cs.CV]*, 10 Apr 2015.
- [38] K. H. X. Z. S. R. J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385v1 [cs.CV]*, vol. I, 10 Dec 2015.
- [39] "Encoding Source Language with Convolutional Neural Network for Machine Translation," *Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, Qun Liu*, no. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pp. 20-30, July 26-31, 2015.
- [40] D. Gupta, "Introduction to Recurrent Neural-Networks," *Analytics Vidhya*, December 7, 2017.
- [41] T. M. Y. B. Razvan Pascanu, "On the difficulty of training recurrent neural networks," 2013.
- [42] S. a. J. S. Hochreiter, "Long Short-Term Memory," *Neural computation* 9.8, vol. 9, no. 8, pp. 1735-1780, 15 November 1997.
- [43] C. Olah, "Understanding LSTM Networks," 27 August 2015. [Online]. Available: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. [Accessed 22 August 2019].
- [44] K. B. V. M. C. G. D. B. F. B. H. S. a. Y. B. Cho, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, no. In Proceedings of the Empirical Methods in Natural Language Processing (EMNLP), 2014.
- [45] D. K. C. a. Y. B. Bahdanau, "Neural machine translation by jointly learning to align and translate," *ICLR, arXiv preprint arXiv:1409.0473*, 2014.
- [46] N. a. K. S. Adaloglou, "How attention works in deep learning: understanding the attention mechanism in sequence models," <https://theaisummer.com/>, no. <https://theaisummer.com/attention/>, 2020.
- [47] A. N. S. N. P. J. U. L. J. A. N. G. Ł. K. a. I. P. Vaswani, "Attention is all you need," in *31st Conference on Neural Information Processing Systems (NIPS)*, Long Beach, CA, USA, pp. 5998-6008, 2017.
- [48] T. C. K. C. G. a. D. J. Mikolov, "Efficient estimation of word representations in vector space," *arXiv:1301.3781*, 2013.
- [49] P. G. E. J. A. a. M. Bojanowski, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135-146, 2017.
- [50] M. O. N. G. J. D. M. J. D. C. O. L. M. L. L. Z. V. S. Yinhan Liu, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv:1907.11692v1 [cs.CL]*, 26 Jul 2019.
- [51] S. R. T. W. a. W.-J. Z. Kishore Papineni, "BLEU: a Method for Automatic Evaluation of Machine Translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, July 2002.

- [52] J. Y. C. X. W. P. L. a. W. X. Zhou, "Deep recurrent models with fast-forward connections for neural machine translation," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 371-383, 2016.
- [53] Y. Solomon, "Optimal Alignment for Bi-directional Afaan Oromo-English Statistical Machine Translation," A Thesis Submitted in Partial Fulfillment of the Requirement for the Degree of Masters of Science in Information Science, Addis Ababa, June, 2017.
- [54] A. Birhanu, "Bi-directional English-Afaan Oromo Machine Translation using Convolutional Neural Network," A Thesis Submitted to the School of Graduate studies of Addis Ababa University in Partial Fulfillment of the Requirement for the Degree of Masters of Science in Computer Engineering, Addis Ababa, 14 October 2019.
- [55] I. G. a. H. L. Shashirekha, "Amharic-Arabic Neural Machine Translation," *CS & IT-CSCP, arXiv preprint arXiv:1912.13161*, pp. 55-68, 2019.
- [56] M. M. W. a. M. Meshesha, "Experimenting Statistical Machine Translation for Ethiopic Semitic Languages: The Case of Amharic-Tigrigna," *ICST*, p. 140–149, 2018.
- [57] H. Mekonnen, "Amharic-Awngi Machine Translation: An Experiment Using Statistical Approach," *IJCSE*, vol. Vol.7, no. 8, pp. 2347-2693, Aug 2019.
- [58] G. T. Heyi, "Bidirectional Amharic-Afaan Oromo Machine Translation Using Hybrid Approach," A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science, Addis Ababa, March 2020.
- [59] ማ. ድ. ትንሳኤ, ሰዋሰወ ግእዝ, ኦዲስ አበባ: ትንሳኤ ማሳተሚያ ድርጅት, ፲፱፻፺፪.
- [60] ዘ. አድሐና, መርኖ-ሰዋሰወ ዘልሳኑ-ግዕዝ (በኦዲስ ጥበብ) መጽሐፍ ቀዳማይ, ኦዲስ አበባ: ኢትዮጵያ: ብርሃንና ሰላም , ፲፱፻፺፮ (1996) እ ኢ. ኢ..
- [61] ኢ. ድ. መምህር, መዝገበ አእምሮ ግዕዝ ዐምዳ ወድዳ ለኢትዮጵያ, ጎንደር: ከኮ ማተሚያ ቤት, 2012 ዓ/ም.
- [62] መ. አ. ገ/ዲድቅ, የግእዝ ቋንቋ የሰዋሰወ መጽሐፍ, ደብረ ብርሃን: ፋር ኢስት ትሬዲንግ ኃላየተ.የግ.ማኅበር, 2010.
- [63] A. H. A. M. Jan vom Brocke, "Introduction to Design Science Research," *Researchgate*, pp. 1-19, 2020.
- [64] A. H. A. M. Jan vom Brocke, "Introduction to Design Science Research," *Research Gate*, no. In: vom Brocke J., Hevner A., Maedche A. (eds) Design Science Research. Cases, Cham., 2020.
- [65] T. T. M. A. R. S. C. Ken Peffers, "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45-78, 2014.
- [66] J. R. Taku Kudo, "SentencePiece: A simple and language-independent subword tokenizer and detokenizer for Neural Text Processing," *Conference on Empirical Methods in Natural Language Processing (System Demonstrations)*, pp. 66-71, 2018.
- [67] A. A. C. Monz, "Optimizing Transformer for Low-Resource Neural Machine Translation," *arXiv:2011.02266v1 [cs.CL] 4 Nov 2020*, 2020.
- [68] lios, "Medium.com," Lios Hyper translation echo system , 29 June 2018. [Online]. Available: <https://medium.com/systran-lios/what-is-opennmt-1-smt-vs-nmt-2ad56003cc0e>. [Accessed 11 October 2019].
- [69] D. Demissie, "Amharic Named Entity Recognition Using Neural Word Embedding as a Feature," Addis Ababa University, Addis Ababa, 2017.

- [70] T. Dutoit, "MonkeyLearn," 21 May 2019. [Online]. Available: <https://monkeylearn.com/sentiment-analysis/>. [Accessed 16 May 2019].
- [71] A. S. Y. G. Barbara Plank, "Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss," *arXiv*, 2016.
- [72] L. J. G. D. L. a. H.-Y. S. Zhou, "The design and implementation of xiaoice, an empathetic social chatbot," *Computational Linguistics* 46, vol. 1, pp. 51-93, 2020.
- [73] C. V. J. Million Meshesha, "Optical Character Recognition of Amharic Documents," *AJICT*, vol. III, no. 2, pp. 53-66, 2007.
- [74] "TutorialsPoint," TutorialsPoint, 12 January 2019. [Online]. Available: https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_basic_concepts.htm. [Accessed 21 September 2020].
- [75] J. Brownlee, "Machine Learning Mastery," Machine Learning Mastery Pty. Ltd, 6 August 2019. [Online]. Available: <https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/#:~:text=A%20node%2C%20also%20called%20a,layers%20to%20comprise%20a%20network..> [Accessed 7 August 2020].
- [76] a. Ng, "Coursera," [Online]. Available: <https://www.coursera.org/courses?query=neural%20networks>. [Accessed 5 Jun 2019].
- [77] S. R. T. W. a. W.-J. Z. Kishore Papineni, "BLEU: a Method for Automatic Evaluation of Machine Translation," *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia*, pp. 311-318, July 2002.
- [78] M. A. Chéragai, "Theoretical Overview of Machine translation," *Proceedings ICWIT*, vol. 1, pp. 160-169, 2012.
- [79] መ. ዮ. ዳኘው, የግልጽ ቋንቋ ንግግር ማስተማሪያ እና መግባቢያ (Conversation) ግዕዝ-አማርኛ ቁጥር-1, ጥር ፳፻፯ ዓ.ም 1ኛ እትም.
- [80] Y. M. Omer Osman Ibrahim, "Stemming Tigrinya Words for Information Retrieval," in *Proceedings of COLING 2012*, Mumbai, December 2012.
- [81] M. S. M. a. G. Donaj, Machine Translation and the Evaluation of Its Quality, IntechOpen: Recent Trends in Computational Intelligence, September 7th, 2019.
- [82] M.-W. C. K. L. a. K. T. Jacob Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805v2 [cs.CL]*, 24 May 2019.
- [83] Y. J.-D. R. a. E. J. Xin, "Deep hybrid neural network for named entity recognition," *eBay Inc, U.S. Patent Application No. 15/692,392*, 2019.

Annex

Annex A

Annex-Table A.1 Vocabulary and One-Hot encoding Vector

ነህ	0
አንተ	1
ግን	2
ያው	3

$$\begin{matrix}
 \text{አንተ} = & \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} &
 \text{ግን} = & \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} &
 \text{ያው} = & \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} &
 \text{አንተ} = & \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} &
 \text{ነህ} = & \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}
 \end{matrix}$$

(a) Word Vocabulary with 10 unique words

(b) One –Hot-Encoding Vector

Annex B

i. A 4D dense Word embedding

Annex-Table B.1 Word embedding of a sentence "አንተ ግን ያው አንተ ነህ"

አንተ =>	2.3	0.6	1.1	2.0
ግን =>	1.2	0.3	-0.5	0.4
ያው =>	0.6	0.4	1.8	0.7
ነህ =>	2.1	0.1	4.1	0.3

Annex C

i. The Complete Ge'ez script arrangement. The former (a), current (b) and derived (c&d)

Annex-Table C.1 Ge'ez Script Arrangements

ጎልቁ	ግዕዝ	ክዕብ	ሣልስ	ራብዕ	ሐምስ	ሳድስ	ሳብዕ	ቁጥር	ግዕዝ	ክዕብ	ሣልስ	ራብዕ	ሐምስ	ሳድስ	ሳብዕ
፩	አ	አ	አ	አ	ኤ	አ	አ	ሀ	ሁ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
፪	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ	ለ	ሉ	ሉ	ሊ	ላ	ሌ	ል	ሎ
፫	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ገ	ሐ	ሐ	ሐ	ሐ	ሐ	ሐ	ሐ	ሐ
፬	ደ	ደ	ደ	ደ	ደ	ደ	ደ	መ	ሙ	ሙ	ሚ	ማ	ሜ	ም	ሞ
፭	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ	ሠ	ሡ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
፮	ወ	ወ	ወ	ወ	ወ	ወ	ወ	ረ	ሩ	ሩ	ሪ	ራ	ራ	ራ	ራ
፯	ዘ	ዘ	ዘ	ዘ	ዘ	ዘ	ዘ	ሰ	ሱ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
፰	ሐ	ሐ	ሐ	ሐ	ሐ	ሐ	ሐ	ቀ	ቁ	ቁ	ቁ	ቀ	ቁ	ቀ	ቀ
፱	ኀ	ኀ	ኀ	ኀ	ኀ	ኀ	ኀ	በ	ቡ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
፲	ጠ	ጠ	ጠ	ጠ	ጠ	ጠ	ጠ	ተ	ቱ	ቱ	ቲ	ታ	ቲ	ት	ቶ
፲፩	የ	ዩ	ዩ	ያ	ዩ	ይ	ዮ	ኀ	ኀ	ኀ	ኀ	ኀ	ኀ	ኀ	ኀ
፲፪	ከ	ከ	ከ	ካ	ከ	ከ	ከ	ነ	ኑ	ኑ	ኒ	ና	ኔ	ን	ኖ
፲፫	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ	አ	አ	አ	አ	አ	አ	አ	አ
፲፬	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ	ከ	ከ	ከ	ከ	ካ	ከ	ከ	ከ
፲፭	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ	ወ	ወ	ወ	ወ	ወ	ወ	ወ	ወ
፲፮	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ	ዐ	ዑ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ

iii. Normalization

```
In [17]: #method to normalize character level mismatch such as አሀይ and ፀሐይ
def normalize_char_level_mismatch(self,input_token):
    rep1=re.sub('[ሃጎጋሐሐሻ]', 'ሀ',input_token)
    rep2=re.sub('[ሐጎሽ]', 'ሐ',rep1)
    rep3=re.sub('[ጎሐሺ]', 'ሂ',rep2)
    rep4=re.sub('[ጋሐሽ]', 'ሄ',rep3)
    rep5=re.sub('[ሐጎ]', 'ሀ',rep4)
    rep6=re.sub('[ሩሐሽ]', 'ሆ',rep5)
    rep7=re.sub('[ሠ]', 'ሰ',rep6)
    rep8=re.sub('[ሠ]', 'ሰ',rep7)
    rep9=re.sub('[ሢ]', 'ሲ',rep8)
    rep10=re.sub('[ሣ]', 'ሳ',rep9)
    rep11=re.sub('[ሤ]', 'ሴ',rep10)
    rep12=re.sub('[ሥ]', 'ስ',rep11)
    rep13=re.sub('[ሦ]', 'ሶ',rep12)
    rep14=re.sub('[ገሐዐ]', 'ሐ',rep13)
    rep15=re.sub('[ዐ]', 'ሐ',rep14)
    rep16=re.sub('[ዒ]', 'ሲ',rep15)
    rep17=re.sub('[ዓ]', 'ሴ',rep16)
    rep18=re.sub('[ዕ]', 'ሴ',rep17)
    rep19=re.sub('[ዖ]', 'ሴ',rep18)
    rep20=re.sub('[ጸ]', 'ፀ',rep19)
    rep21=re.sub('[ጹ]', 'ፀ',rep20)
    rep22=re.sub('[ጺ]', 'ቧ',rep21)
    rep23=re.sub('[ጻ]', 'ቧ',rep22)
    rep24=re.sub('[ጼ]', 'ቧ',rep23)
    rep25=re.sub('[ጾ]', 'ፀ',rep24)
    rep26=re.sub('[ጻ]', 'ፀ',rep25)
    #Normalizing words with Labialized Amharic characters such as በሉቶላ or በሉቶላ to በሉቶላ
    rep27=re.sub('(ሉ[ጻሉ])', 'ሉ',rep26)
    rep28=re.sub('(ሙ[ጻሉ])', 'ሙ',rep27)
```

Annex-Figure D.3 Snapshot of fragment code of Normalization

iv. Sample Codes from Experiments

```
#Fragment Code from the Transformer Model
#Importing important libraries
import time
import numpy as np
... //jumped code
from sklearn.model_selection import train_test_split
import unicodedata
import re, os, io

...// Jumped codes
# Code for the encoder Layer of the Transformer
class EncoderLayer(tf.keras.layers.Layer):
    def __init__(self, d_model, num_heads, dff, rate=0.1):
        super(EncoderLayer, self).__init__()
        self.mha = MultiHeadAttention(d_model, num_heads)
        self.ffn = point_wise_feed_forward_network(d_model, dff)
        self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.dropout1 = tf.keras.layers.Dropout(rate)
        self.dropout2 = tf.keras.layers.Dropout(rate)
    def call(self, x, training, mask):
```


ii. Manual Translation Evaluation (For Transformer Model)

ተ.ቁ	በተሰራው አዲስ የማሽን ሞዴል (Transformer) ከአማርኛ ወደ ግእዝ የተተረጎሙ ዓ/ነገሮች	የትርጉም ጥራት				
		(0)ምንም አልተተረጎመም	(1) ብዙው አልተተረጎመም	(2) በከፊል አልተተረጎመም	(3) ተተርጉሟል ማለት ይቻላል	(4) በትክክል ተተርጎሟል
1.	ግብዓት881: <ሕዝቡንም ለሦስተኛው ቀን ተዘጋጁ ወደ ሴቶቻችሁ አትቅረቡ አለ> ውጽዓት881: ወይቤ ለሕዝቡ ተደለው ለሠሉስ ዕለት ወኢትቅረቡ አንስተ					✓
2.	ግብዓት882: <ግብፃውያንም በፈርዖንና በሰረገሎቹ በፈረሰኞቹም ላይ ከብር ባገኘሁ ጊዜ እኔ እግዚአብሔር እንደ ሆኑሁ ያውቃሉ> ውጽዓት882: ወያኔምሩ ኩሉ ግብጽ ከመ እን እግዚአብሔር ተሰቢሐየ በፈርዖን ወበሰረገላሁ ወበኩሉ አፍራሲሁ					✓
3.	ግብዓት883: <ሙሴም የአማቱን ቃል ሰማ ያለውንም ሁሉ አደረገ> ውጽዓት883: ወሰምዐ ሙሴ ቃለ ሐሙሁ ወገብረ					✓
4.	ግብዓት884: <የቶርም ከግብፃውያንና ከፈርዖን እጅ ያዳናችሁ ከግብፃውያንም እጅ ሕዝቡን ያዳኑ እግዚአብሔር ይባረክ አለ> ውጽዓት884: ወይቤ የቶር ቡሩክ እግዚአብሔር ዘእድኅኑ ሕዝቦ እምአደ ግብጽ ወእምአደ ፈርዖን					✓
5.	ግብዓት885: <አቤቱ ቀኝህ በኃይል ከበረ አቤቱ ቀኝ እጅህ ጠላቶችን በተናቸው> ውጽዓት885: የማንከ እግዚአ ተሰብሐ በጎይል የማኑ እዴክ እግዚአ ሠረወቶሙ ለፀር					✓
6.	ግብዓት886: <እግዚአብሔር ተዋጊ ነው ስሙም እግዚአብሔር ነው> ውጽዓት886: እግዚአብሔር ይቀጠቅጥ ፀብአ ወእግዚአብሔር ስሙ					✓
7.	ግብዓት887: <እንዲህም ሆነ አሮን ከእርሱ ጋር ሲነጋገር የአሰራኤል ልጆች ሁሉ በተናገረ ጊዜ ወደ እርሱ በጮኹ ጊዜ> ውጽዓት887: ወሰበ, ይትናገር, አሮን, ለኩሉ, ውሉደ, አሰራኤል, ንቤ, ተጋብኡ, ተመይጡ, ውስተ, ገዳም, ወሙራኔ, እግዚአብሔር, ተርእየ, በደመና			✓		
8.	ግብዓት888: <ሙሴም መሠዊያ ሠራ ስሙንም ይሀዌህ ንሲ ብሎ ጠራው> ውጽዓት888: ወአሕነጸ, ሙሴ, ምሥዋዐ, ለእግዚአብሔር, ወሰመዮ, ሰሞ, ምምሕፃን				✓	
9.	ግብዓት889: <ትዕቢት ባደረጉባቸው ነገር እግዚአብሔር ከአማልክት ሁሉ እንዲበልጥ አሁን አወቅሁ አለ> ውጽዓት889: እምይእዜ, አእመርኩ, ከመ, ዐቢይ, እግዚአብሔር, እምኩሉ, አማልክት, በበይነዝ, ተኩነኑ, ሎሙ				✓	
10.	ግብዓት890: <ሙሴም አማቱን ሰደደው እርሱም ወደ አገሩ ተመለሰ> ውጽዓት891: ወፈነወ, ሙሴ, ሐማሁ, ውስተ, ምድሩ, ወጎሰፈ					✓

Annex-Figure E.2 A sample photo of the Transformer translation manual evaluation

A Bi-Directional Ge'ez-Amharic Neural Machine Translation: a Deep Learning Approach



Amdework Asefa Belay

Declaration

I hereby declare that this thesis represents my own work, which has been done after registration for the degree of Masters at Debre Berhan University and has not been previously submitted for any other degree or professional qualification. I confirm that the work of this thesis is my own, and all sources of materials used for the thesis have been duly acknowledged.


Name: Amdework Asefa

Signature: _____

Date of submission: _____

Place: Debre Berhan University, Debre Berhan, Ethiopia

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

<i>Advisor Name</i>	<i>Signature</i>	<i>Date</i>
Wondwossen Mulugeta (PhD)		_____