Addis Ababa University

College of Natural and Computational Sciences

Development of a Transformer-Based Model for Amharic-to-English Neural Machine Translation

Surafiel Habib Asefa

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science Specialized in Software Engineering

Addis Ababa University

College of Natural and Computational Sciences

Surafiel Habib Asefa

Advisor: Yaregal Assabie (PhD)

This is to certify that the thesis prepared by Surafiel Habib Asefa titled: *Development of a Transformer-Based Model for Amharic-to-English Neural Machine Translation* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science specialized in Software Engineering complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

	Name	**Signature**	**Date**
Advisor:	Yaregal Assabie (PhD)	_____	_____
Examiner:	Ayalew Belay (PhD)	_____	_____
Examiner:	Solomon Gizaw (PhD)	_____	_____

Abstract

Amharic is the working language of Ethiopia and, owing to its Semitic characteristics, the language is known for its complex morphology. It is also an under-resourced language, presenting significant challenges for natural language processing tasks like machine translation. The primary challenges include the scarcity of parallel data, which increases the risk of overfitting and limits the model's ability to generalize effectively, and the complex morphology of Amharic, which further complicates learning patterns in translation tasks. This study proposes a Transformer-based Amharic-to-English neural machine translation model that leverages character-level embeddings and integrates advanced regularization techniques, including dropout, L1, L2, and Elastic Net. By focusing on character-level embeddings, the model captures the intricate morphological patterns of Amharic and effectively handles out-of-vocabulary words. Our model significantly improves upon the previous state-of-the-art result in the Amharic-to-English neural machine translation benchmark, achieving a BLEU score of 40.59, which is 7% higher than the previous state-of-the-art result. Among the regularization techniques tested, the integration of L2 regularization with dropout applied to the point-wise feed-forward network yielded the best translation performance. Additionally, the proposed model significantly reduces the parameter count from 75 million to just 5.4 million, demonstrating substantial computational efficiency while maintaining high accuracy. Extensive experiments demonstrated improvements in test accuracy, loss reduction, and translation fidelity compared to word-level embedding models. This research provides valuable insights into addressing the challenges of low-resource and morphologically complex languages, while also offering promising directions for future work, including the exploration of multilingual models, attention mechanism optimization, and the broader application of hybrid regularization techniques in the Transformer model architecture.

Keywords: Neural Machine Translation, Transformer. Morphologically Complex Languages, Low Resource Languages, Regularization Techniques, Amharic-English Translation

Dedication

To my beloved daughter, *Yemariam Surafiel* (*ሳሱ'ዬ*)

and

To all the innocent Ethiopians who have lost their lives and have been displaced solely because of the language they speak and/or their ethnic identity, in different regions of the country

"የዓለም ሁሉ ቋንቋ አንድ፤ ንግግሩም አንድ ነበረ . . . እግዚአብሔርም ቋንቋቸውን ለያየ . . . ስለዚህም ስምዋ ባቢሎን ተባለ፤ እግዚአብሔር በዚያ የምድርን ቋንቋ ሁሉ ደባልቋልና፤ ከዚያም እግዚአብሔር በምድር ሁሉ ፊት ላይ እነርሱን በትኗቸዋል።"

ኦሪት ዘፍጥረት 11 ፥ 1 - 9

"The whole earth used the same language, the same words . . . For this reason, it is called Bavel [confusion] — because there Adonai confused the language of the whole earth, and from there Adonai scattered them all over the earth."

Genesis 11:1 - 9

Acknowledgments

First and foremost, I would like to extend my deepest gratitude to the Almighty Holy Trinity, God the Father, God the Son, and God the Holy Spirit, the One God, for granting me wisdom, patience, and perseverance throughout this journey. It is through His grace that I have found the endurance and determination to complete this work. I am also immensely grateful to St. Virgin Mary for her intercession, guiding me and standing by me during challenging times.

I would like to express my heartfelt thanks to my advisor, Dr. Yaregal Assabie, for his invaluable support, guidance, and continuous encouragement. From the beginning, he provided me with the foundational concept to build a Transformer-based model for Amharic to English neural machine translation. His consistent follow-up and insightful suggestions pushed me to explore further at every phase of the study. His keen eye for detail and his ability to see perspectives I often missed greatly contributed to my growth throughout this process. His academic guidance, as well as his personal encouragement and appreciation of my efforts, pushed me to go beyond my limits, try new approaches, and conduct additional experiments that enriched the quality of this research. Dr. Yaregal's mentorship has been and will continue to be invaluable to my current and future academic endeavors.

I would also like to extend my sincere appreciation to Mr. Brook Tarekegn and Mr. Yoseph Berhanu, for their crucial assistance in accessing research papers that were otherwise unavailable to me due to subscription-based access restrictions. Their generosity in helping me obtain necessary materials from IEEE Xplore, ACM Digital Library, and ScienceDirect played a key role in shaping this work. In particular, I am deeply thankful to Mr. Brook Tarekegn for his kind support in providing me with access to a subscription-based cloud computing environment, which was vital for the execution of my experiments.

To everyone who supported me along this journey, both academically and personally, I am truly grateful. Your contributions, encouragement, and assistance have made this achievement possible.

Table of Contents

List of Tables iv

List of Figures v

List of Algorithms i

1 Introduction 1

1.1 Background 1

1.2 Motivation 2

1.3 Statement of the Problem 3

1.4 Objectives 6

1.5 Methods 6

1.6 Scope and Limitations 7

1.7 Application of Results 8

1.8 Thesis Organization 8

2 Literature Review 10

2.1 Introduction 10

2.2 Overview of Amharic Language 10

2.2.1 Amharic Morphology 11

2.2.2 Amharic Phrase 19

2.2.3 Amharic Sentences 22

2.2.3.1 Simple Sentences 23

2.2.3.2 Complex Sentences 25

2.2.4 Amharic Punctuation 25

2.3 Overview of English Language 26

2.3.1 English Morphology 26

2.3.2 English Phrase 29

2.3.3 English Sentences 31
2.4 Evolution of Machine Translation 34
2.4.1 Rule Based Machine Translation 35
2.4.2 Example Based Machine Translation 37
2.4.3 Statistical Machine Translation 38
2.4.4 Neural Machine Translation 40
2.4.4.1 Attention Mechanism 43
2.4.4.2 Transformer 45
2.5 Hybrid Machine Translation 48
2.6. Multilingual Machine Translation 49
2.7 Machine Translation Evaluation Metrics 50
2.7.1. Human Machine Translation Evaluation Metrics 51
2.7.2 Automatic Machine Translation Evaluation Metrics 51
2.8 Regularization Techniques in Neural Networks 54
2.9 Available Parallel Corpora for Amharic-to-English Machine Translation 55
3 Related work 57
3.1 Introduction 57
3.2 Character-Level Transformer Model for Neural Machine Translation 57
3.3 Subword Modeling for Morphologically Rich Languages 58
3.4 Regularization Techniques in Low-Resource NMT and other DL Domains 61
3.5 Summary 64
4 Design of the Proposed Model 65
4.1 Introduction 65
4.2 Proposed Model Architecture 65
4.2.1 Character Vectorization 67

4.2.2 Positional Encoding 69
4.2.3 Encoder Stack 71
4.2.4 Decoder Stack 73
4.2.5 Linear Vocabulary Projection and Softmax 81
4.2.6 Output Sequence 81
5 Experimentations and Evaluation 84
5.1 Introduction 84
5.2 Dataset Collection 84
5.3 Experimental System Environment 84
5.4 Dataset Preparation 85
5.5 Experiments and Results 90
5.5.1 Experiment on the Baseline Transformer Model 90
5.6 Experiment on Proposed Model (Character Embedding and Combined regularization Techniques) 93
5.6.1 Character Embedding and Dropout with Elastic net Regularization 93
5.6.2 Character Embedding and Dropout with L1 Regularization 96
5.6.3 Character Embedding and Dropout with L2 Regularization 98
5.7 Discussion on the Experimental Results of the Study 100
6 Conclusion and Future Work 105
6.1 Introduction 105
6.2 Conclusions 105
6.3 Future Work 106
References 107
Annexes 122
Annex I - Sample Source Code 122
Annex III – Sample Translated Sentences from Each Experiment 134

List of Tables

Table 2.1: Plural noun formation in Amharic 12

Table 2.2: Amharic Noun Definiteness Marking Patterns 12

Table 2.3: Gender Marking in Amharic Nouns 13

Table 2.4: Personal pronouns in Amharic 17

Table 2.5: Amharic demonstrative pronouns 17

Table 2.6: Amharic Punctuations 25

Table 2.7: The Eight English Inflectional Morphemes [57] 27

Table 2.8: English Sentence Patterns 33

Table 2.9: Sample Amharic-English Bilingual Corpus for EBMT Illustration 38

Table 2. 10: Summary of the Baseline Model Experimental Results 93

Table 2.11: Summary of the Proposed model Experimental Results 99

Table 2.12: Overfitting size comparison between the Baseline and Proposed model, highlighting reductions in Accuracy and Loss overfitting 101

Table 2.13: BLEU score comparison between the proposed model and other benchmark models for Amharic-to-English NMT 104

List of Figures

Figure 2.1: Sample complex word structure in English 28
Figure 2.2: Generic English sentence structure 32
Figure 2.3: Detailed English sentence structure 32
Figure 2.4: The General Architecture of RBMT 36
Figure 2.5: A Word-to-Word Statistical Machine Translation System 39
Figure 2.6: The Transformer Model Architecture 45
Figure 4.1: Architecture of the Proposed Model 66
Figure 5.1: Amharic Cumulative Token frequency distribution 87
Figure 5.2: Amharic Token frequency distribution 88
Figure 5.3: English Cumulative Token frequency distribution 88
Figure 5.4: English Token frequency distribution 89
Figure 5.5: Baseline Transformer Model Summary 91
Figure 5.6: The Baseline Model Training Results 91
Figure 5.7: Training and Validation Loss and Accuracy Curves for the Baseline Model 92
Figure 5.8: Baseline Model Evaluation Results on a Test Dataset 92
Figure 5.9: Maximum and Average BLEU Score Results of the Baseline Model 92
Figure 5.10: Proposed Model Summary 93
Figure 5.11: Proposed Model Character Embedding and Dropout with Elastic Net Results 94
Figure 5.12: Training and Validation Loss and Accuracy Curves for the proposed model with Character Embedding and Dropout with Elastic Net 95
Figure 5.13: Proposed Model Character Embedding and Dropout with Elastic Net Combined Regularization Evaluation Results on a Test Dataset 95
Figure 5.14: Maximum and Average BLEU Score Results of the Proposed Model with Character Embedding and Dropout with Elastic Net Combined Regularization 96
Figure 5.15: Proposed Model Character Embedding and Dropout with L1 Combined Regularization Training Results 96
Figure 5.16: Training and Validation Loss and Accuracy Curves for the proposed model with Character Embedding and Dropout with L1 97

Figure 5.17: Proposed Model Character Embedding and Dropout with L1 Combined Regularization Evaluation Results on a Test Dataset 97
Figure 5.18: Maximum and Average BLEU Score Results of the Proposed Model with Character Embedding and Dropout with L1 Combined Regularization 97
Figure 5.19: Proposed Model Character Embedding and Dropout with L2 Combined Regularization Training Results 98
Figure 5.20: Training and Validation Loss and Accuracy Curves for the proposed model with Character Embedding and Dropout with L2 98
Figure 5.21:Proposed Model Character Embedding and Dropout with L2 Combined Regularization Evaluation Results on a Test Dataset 99
Figure 5.22: Maximum and Average BLEU Score Results of the Proposed Model with Character Embedding and Dropout with L2 Combined Regularization 99
Figure 5.23: Comparison of Test Loss, Test Accuracy, and BLEU Score Across the Baseline and Proposed Model with Character Embedding and Different Combined Regularization Approaches 103

List of Algorithms

Algorithm 4.1: Character Vectorization Process Pseudocode 69

Algorithm 4.2: Positional Encoding Pseudocode 70

Algorithm 4.3: Encoder Stack Processing Pseudocode 73

Algorithm 4.4: Decoder Stack Processing Pseudocode 76

Algorithm 4.5: L1 Regularization Combined with Dropout in the Point-wise FFN Pseudocode 78

Algorithm 4.6: L2 Regularization Combined with Dropout in the Point-wise FFN Pseudocode 79

Algorithm 4.7: Elastic Net Regularization Combined with Dropout in the Point-wise FFN Pseudocode 80

Algorithm 4.8: Linear Vocabulary Projection and Softmax Pseudocode 82

Algorithm 4.9: Output Sequence Generation Pseudocode 83

Acronyms and Abbreviations

AI	Artificial Intelligence
AMTEM	Automatic Machine Translation Evaluation Metric
ANN	Artificial Neural Network
BiRNN	Bidirectional Recurrent Neural Network
BLEU	Bilingual Evaluation Understudy
CGM	Convolutional Gating Module
CLIR	Cross Language Information Retrieval
CNN	Convolutional Neural Network
CBMT	Corpus Based Machine Translation
CSM	Convolutional Sequence Model
DD	Data Dropout
DL	Deep Learning
DNN	Deep Neural Network
EBMT	Example Based Machine Translation
FD	Feature Dropout
FFN	Feed Forward Network
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
GTM	Global Translation Model
HMT	Hybrid Machine Translation
IBM	International Business Machines

LASSO	Least Absolute Shrinkage and Selection
LSTM	Least Short-Term Memory
ML	Machine Learning
METOR	Metric for Evaluation of Translation with Explicit ORdering
MMT	Multilingual Machine Translation
MT	Machine Translation
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
NMT	Neural Machine Translation
NP	Noun Phrase
OOV	Out of Vocabulary
PP	Prepositional Phrase
RAM	Random Access Memory
RBMT	Rule Based Machine Translation
RCTM	Recurrent Convolutional Translation Model
RNN	Recurrent Neural Network
SA	Self-Attention
SD	Structure Dropout
SL	Source Language
SMT	Statistical Machine Translation
SOV	Subject Object Verb
SVA	Subject Verb Adjunct
SVC	Subject Verb Complement

SVO	Subject Verb Object
SVOA	Subject Verb Object Adjunct
SVOC	Subject Verb Object Complement
SVOO	Subject Verb Object Object
TER	Translation Edit Rate
TL	Target Language
VP	Verb Phrase
WER	Word Error Rate

Chapter 1

Introduction

1.1 Background

Natural Language Processing (NLP) is the field that focuses on how computers can recognize and manipulate natural language text or speech to perform useful tasks [1]. NLP draws from various disciplines, including computer and information sciences, linguistics, mathematics, electrical and electronic engineering, artificial intelligence (AI), robotics, and psychology [1]. It has a wide range of applications across different fields, such as machine translation (MT), natural language text processing and summarization, user interfaces, multilingual and cross-language information retrieval (CLIR), speech recognition, AI, and expert systems [1, 2].

MT is a specialized area within NLP that focuses on the automatic translation of text from one human language to another using computing systems [3, 4, 5]. Over the years, MT has evolved through various methods, from Rule-based Machine Translation (RBMT) [6] to Statistical Machine Translation (SMT), which dominated MT research for decades [3, 4, 5, 7], and more recently to Neural Machine Translation (NMT) [3, 4]. NMT has been shown to outperform SMT across 27 natural languages, as evidenced by comparative experimental results across 30 languages [8]. Consequently, NMT has become the core technology for commercial online translation systems [9].

An end-to-end NMT system implements automatic translation between natural languages through neural networks. NMT systems typically employ an encoder-decoder framework or an embed-encode-attend-decode paradigm, using Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and transformer-based self-attention/feed-forward network (SA/FFN) architectures for sequence-to-sequence conversion.

Traditional NMT approaches based on RNNs have limitations in processing speed and accuracy, particularly with long sentences [10, 11]. To address these challenges, the Transformer model was introduced by Vaswani *et al*. [10] as a new neural network architecture that relies solely on attention mechanisms, eliminating the need for recurrent or convolutional layers. The model connects the encoder and decoder using attention, enabling it to focus on different parts of the

input sequence when generating an output [10]. Attention mechanisms allow the model to capture dependencies across various positions in a sequence, regardless of their distance, and facilitate focused processing of relevant segments during output generation [10, 12].

The Transformer model employs SA mechanisms and FFNs to capture dependencies between different positions in input and output sequences. SA enables each position to attend to all other positions, allowing the model to effectively capture global context and dependencies [9]. FFNs, functioning independently at each position, are used to capture complex patterns and relationships [10].

This research aims to develop a Transformer-based model tailored for Amharic-to-English MT, incorporating SA and FFNs as foundational elements. Despite advancements in NMT, developing a robust Amharic-to-English system remains a considerable challenge due to the distinctive linguistic features of Amharic [13]. The Transformer model, recognized as the state-of-the-art in NMT since 2022, is noted for its superior quality, increased parallelizability, and reduced training time compared to other models [11]. It surpasses RNNs in translation quality and the ability to infer unseen directions [12]. Therefore, this study will construct a Transformer-based model for Amharic-to-English MT, adopting the comprehensive universal Transformer architecture outlined in [10].

1.2 Motivation

There are compelling reasons to emphasize the urgent need for translating Amharic, particularly into English. According to a report by Tomedes[1] dated February 9, 2024, Amharic is the most spoken language in Ethiopia, with 31.8 million native speakers and an additional 25 million who speak it as a second language, representing a substantial portion of the country's population of 115 million. In a nation characterized by linguistic diversity, Amharic plays a crucial role in communication, serving as a bridge between various communities.

The primary motivation for developing a transformer-based model for Amharic-to-English NMT arises from the critical need to accurately translate essential documents from Amharic into English. These documents encompass a wide range of types, including legal texts, academic papers,

[1] https://www.tomedes.com/translator-hub/languages-ethiopia

government documents, and cultural artifacts. Translating such content is not merely about making it linguistically accessible; it directly impacts our ability to share knowledge, foster international collaborations, and ensure effective global communication. Furthermore, a significant barrier exists, as most Amharic speakers have limited proficiency in English. This linguistic gap poses a considerable obstacle, restricting access to information, opportunities, and resources that are predominantly available in English. For businesses and organizations aiming to expand their influence across Ethiopia, Amharic translation holds profound significance, extending beyond personal and community-level considerations to impact broader socio-economic development.

1.3 Statement of the Problem

MT has undergone significant evolution, progressing from rule-based systems to example-based methods and eventually to SMT in the late 1980s. The advent of deep neural networks in 2012 marked a paradigm shift, leading to the rise of NMT systems. These systems have outperformed traditional methods by learning directly from data without heavily relying on linguistic rules [3].

However, the success of NMT systems depends heavily on the availability of large parallel datasets, a requirement that presents substantial challenges for low-resource languages like Amharic [14, 15, 16]. The scarcity of parallel data intensifies issues such as overfitting and poor generalization, particularly in complex models like the Transformer, which are characterized by a high number of parameters [14, 15, 16, 17]. Overfitting occurs when a model learns patterns too specific to the limited training data, capturing noise rather than general trends, which reduces accuracy on new, unseen data and limits model effectiveness [17]. Underfitting, by contrast, arises when the model is too simple or inadequately trained, failing to capture the essential structure of the data. Both overfitting, due to insufficient data, and underfitting, due to a lack of model complexity or training, present significant obstacles to achieving effective and accurate translations.

To address overfitting challenges, various techniques have been explored, with regularization being a key focus. Regularization methods such as Dropout, L1 regularization (Least Absolute Shrinkage and Selection Operator, or Lasso), L2 regularization (Ridge regression), and Elastic Net regularization, which combines both L1 and L2 penalties, have been implemented to reduce overfitting and enhance the generalization of deep neural network (DNN) models [17, 18, 19, 20].

Overfitting in Transformer-based NMT models is a critical concern, particularly in low-resource contexts, as it leads to poor generalization and diminished translation quality. While the Transformer architecture [10] is revolutionary and recognized as the current state of the art in NMT, standing out for its superior translation quality, parallelizability, and reduced training time compared to traditional NMT models, it still faces significant challenges when applied to languages with limited resources. Despite its advantages, including the ability to outperform RNNs in translation quality and infer unseen directions [12], overfitting remains a substantial limitation in low-resource settings.

Amharic, a Semitic language spoken primarily in Ethiopia and the second-most spoken Semitic language in the world after Arabic [16], is highly inflectional with a rich morphology [21, 22, 23]. The limited availability of parallel Amharic-English corpus further complicates the development of effective NMT systems for this language. Existing research on Amharic-to-English translation has predominantly utilized word and subword-based input or embedding approaches. Although some efforts have been made to apply Transformer-based methods [24, 25, 26, 27], these have not fully exploited the potential of optimal input methods within the Transformer model. Moreover, they have not adequately addressed the issue of overfitting, a prevalent problem in low-resource scenarios.

In the baseline Transformer model [10], the conventional dropout regularization technique, as proposed by Srivastava *et al*. [28], is applied to manage overfitting in various layers, including token embeddings, positional encodings in both the encoder and decoder stacks, the FFN, and attention mechanisms. Additionally, several alternative dropout techniques have been introduced for the Transformer. Fan *et al*. [29] propose LayerDrop, a random structured dropout method that drops certain layers of the Transformer during training. Zhou *et al*. [30] introduce DropHead, a structured dropout method for regularizing the multi-head attention mechanism. Furthermore, Wu *et al*. [31] present UniDrop, which integrates three different levels of dropout: feature dropout, structure dropout, and data dropout, into the Transformer model. While dropout is widely used, there is an urgent need to explore and implement more robust regularization methods, such as L1, L2, and Elastic Net regularization, particularly when dealing with morphologically complex and low-resource languages like Amharic. Given the large size of the Transformer architecture, it is more prone to overfitting in such contexts.

Furthermore, the complexities of Amharic morphology and grammar necessitate the use of character-level embeddings, which have shown promise in addressing out-of-vocabulary (OOV) words and reducing model size in low-resource settings [32, 33]. These embeddings have proven particularly effective for morphologically rich languages and low-resource scenarios, outperforming standard word embeddings on small corpora and rare words [34]. Additionally, approaches such as subword-based methods incorporating character n-grams and morphological features [35], character-aware decoders that handle both word-level and subword-level sequences in attention-based encoder-decoder models with CNNs [36], and the integration of character embeddings with coverage models in NMT architectures [37] have all shown promising results in enhancing NMT effectiveness.

However, to the best of the researchers' knowledge, fully character-level embeddings have not yet been integrated into Amharic-to-English Transformer-based NMT models. Furthermore, the exploration of additional regularization techniques, beyond the conventional dropout method, within the Transformer architecture has not been investigated so far, particularly for low-resource languages like Amharic. Given the unique challenges posed by the scarcity of parallel corpora and the complex morphology of such languages, a more comprehensive investigation into how various regularization approaches, including L1, L2, and Elastic Net, can mitigate overfitting and improve model generalization is crucial.

This study aims to address these gaps by developing a fully Transformer-based model for Amharic-to-English NMT that incorporates character-level embeddings and combines dropout with L1, L2, and Elastic Net regularization techniques. This approach seeks to enhance the model's generalization capabilities and translation quality, addressing the unique morphological challenges of Amharic and the limitations posed by the scarcity of parallel corpora. The study will explore the effective integration of character embeddings into the Transformer model to improve Amharic-to-English translation and will examine the impact of combining multiple regularization techniques on the performance of NMT systems for low-resource languages.

1.4 Objectives

General Objective

The general objective of this research is to design a Transformer-based model for Amharic-to-English NMT.

Specific Objectives

In line with the general objective, this thesis work is specifically aimed to at meeting the following specific objectives:

- Review existing MT approaches and methods.
- Collect a bilingual Amharic-to-English corpus.
- Design and propose a fully Transformer-based model for Amharic-to-English NMT.
- Evaluate the performance of the proposed model.

1.5 Methods

The methods that will be employed in this thesis work are as follows:

Literature Review

This research will conduct an extensive literature review, providing an overview of the Amharic and English languages and exploring their linguistic features relevant to translation. It will also include a comprehensive review of the evolution of MT, from rule-based approaches to the current state-of-the-art Transformer-based NMT models. The review will cover MT evaluation metrics such as Bilingual Evaluation Understudy (BLEU), Metric for Evaluation of Translation with Explicit ORdering (METEOR), and Translation Edit Rate (TER), which are essential for assessing translation quality. Additionally, it will examine regularization techniques such as dropout, L1, L2, and Elastic Net to address overfitting, identifying gaps in existing research to guide the development of a more effective Transformer-based Amharic-to-English NMT model.

Data Collection and Preprocessing

We will gather a substantial parallel corpus of Amharic and English texts, despite the challenges associated with Amharic being a low-resource language. Collecting a comprehensive Amharic-English corpus presents a significant challenge, but we will prioritize the identification and collection of high-quality data essential for the NMT system. The corpus will be carefully

preprocessed, cleaned, tokenized, and vectorized to address the unique linguistic characteristics of Amharic, ensuring compatibility with the NMT model's requirements. Character embedding will be applied to account for Amharic's morphological complexity and effectively OOV words

Model Development

In this study, we will design and develop a Transformer-based NMT model specifically for translating from Amharic to English. The model will be built to overcome the challenges posed by the limited availability of parallel data and the complex nature of the Amharic language. The training will be conducted using a parallel corpus, with a focus on using character-level embeddings to handle OOV words and better capture Amharic's rich morphology. We will fine-tune the model's parameters to improve performance and apply various regularization techniques, such as dropout, L1, L2, and Elastic Net, to mitigate overfitting and enhance generalization. By combining these methods, we aim to address overfitting challenges in low-resource settings and improve the quality of Amharic-to-English translations.

Model Evaluation

A comprehensive evaluation of the proposed model will be conducted, focusing on training accuracy and loss, validation accuracy and loss, and test accuracy and loss. Additionally, the BLEU metric will be employed to assess the overall translation quality.

1.6 Scope and Limitations

This thesis focuses on developing a Transformer-based model specifically designed for translating standard Amharic to English. The model is tailored to handle the formal and widely recognized language structures found in standard Amharic texts, ensuring optimal translation of commonly used words, phrases, and sentence structures typical of official documents.

However, this research deliberately excludes certain linguistic elements that present unique challenges for MT. Idiomatic expressions, which often convey meanings that cannot be directly inferred from individual words, are not within the scope of this study. Similarly, the model does not account for sarcasm, where the intended meaning is often the opposite of the literal interpretation. These language forms require a deep understanding of cultural context and nuance, which is beyond the scope of the current model.

Furthermore, the translation of dialects or local variations of Amharic is not addressed in this thesis. Amharic has several dialects with significant differences in vocabulary, pronunciation, and usage. Since the model is dedicated to standard Amharic and does not incorporate dialects in its training, its ability to accurately translate text from various dialects is limited.

1.7 Application of Results

The Amharic-to-English Transformer-based NMT system developed in this thesis addresses the urgent need for accurate and efficient translation of important documents. It helps Amharic speakers overcome language barriers and engage more actively in global discussions. By making communication between Amharic and English speakers easier, this model opens up opportunities for individuals to access information, resources, and participate in international collaboration.

The system also has great potential for businesses and organizations aiming to expand into Ethiopia. By improving communication and collaboration with local communities, it can support economic growth and development, bridging the gap between local enterprises and international markets. Additionally, this model can benefit the education sector by assisting in the translation of educational materials, making them more accessible to Amharic-speaking students and educators, which can lead to better educational outcomes.

Looking to the future, the methods and insights gained from this research could also help in developing NMT systems for other low-resource languages, contributing to the wider field of MT. As MT research advances, the foundation laid by this work can be used to explore more complex translation tasks, such as handling idiomatic expressions, dialects, and culturally specific content, ultimately enhancing the ability of NMT systems to manage different languages and contexts

1.8 Thesis Organization

This thesis is organized into six chapters. Chapter 2 presents a comprehensive literature review, focusing on the linguistic characteristics of the Amharic and English languages, the evolution and evaluation of MT, and an overview of regularization techniques as the foundation of this research. Chapter 3 discusses related work, providing an in-depth analysis of existing research on character embedding in the Transformer model, transformer-based Amharic-to-English NMT, and regularization techniques in DNNs. Chapter 4 details the design of the proposed transformer-based model for Amharic-to-English NMT, outlining the methodologies and techniques employed.

Chapter 5 describes the experiments conducted and presents the evaluation results. Finally, Chapter 6 concludes the thesis by summarizing the key contributions of this work and offering recommendations for future research directions.

Chapter 2

Literature Review

2.1 Introduction

This chapter aims to provide an overview of the linguistic characteristics of the Amharic and English languages, along with the evolution and different types of MT systems. It will cover traditional approaches like rule-based and statistical methods and explore more recent developments, such as NMT, which has revolutionized the field with its ability to generate fluent and accurate translations. Specifically, the attention mechanism, a core component of NMT, allows the model to focus on relevant parts of the source sentence during translation, leading to better context handling. The introduction of the Transformer model has further advanced the field by improving scalability and translation quality through self-attention layers, replacing recurrent architectures. This review will also include hybrid and multilingual MT systems, which combine different approaches to enhance translation quality across multiple languages. Additionally, this chapter will discuss important MT evaluation metrics used to assess translation quality.

The chapter will explore various regularization techniques in neural networks, such as dropout, L1, L2, and Elastic Net, which are essential for preventing overfitting and improving the generalization of NMT models, particularly in low-resource settings like Amharic-to-English translation. Furthermore, the available parallel corpora for Amharic-to-English translation will also be discussed, as they serve as crucial resources for training and evaluating NMT models in this language pair.

2.2 Overview of Amharic Language

Amharic, a Semitic language spoken by the majority of Ethiopians as their mother tongue and as a second language, is also the second most widely spoken Semitic language globally, after Arabic [13]. It serves as the official working language of the Federal Democratic Republic of Ethiopia and has its own unique alphabet, known as Fidel (ፊደል) or Hohey (ሆሄ), which uses a distinct writing system [13, 38]. The Amharic alphabet (Fidel) is derived from Ge'ez, an ancient Ethiopian Semitic language that is recognized as the precursor to other Ethiopian Semitic languages such as Tigrigna [39].

In the Amharic script, there are 34 root alphabets (**ፊደላት/ሆህያት**), each with seven forms that are created by combining a consonant with a vowel, leading to a total of 238 distinct characters (7 * 34) [23]. Additionally, some non-standard alphabets include special features, typically representing labialization. Each character represents a consonant along with its corresponding vowel, and the vowels are incorporated into the consonant form through diacritic markings—strokes attached to the root characters that alter their pronunciation [39].

2.2.1 Amharic Morphology

Similar to all other Semitic languages, Amharic is one of the most morphologically complex languages, exhibiting a root-pattern morphological phenomenon [21, 40]. A root is a set of consonants (radicals) that carries a basic lexical meaning [21, 40]. A pattern consists of a set of vowels that are inserted among the consonants of a root to form a stem [41]. The complexity of Amharic morphology can be illustrated by its word formation processes, which include inflection, derivation, and affixation [21]. Word classifications in the Amharic language are divided into categories such as noun (**ስም**), verb (**ግስ**), adjective (**ቅፅል**), preposition (**መስተዋድድ**), and adverb (**ተውሳከ ግስ**) [42]. However, Belata Mersea Hazen Woldekirkos (**ብላታ መርስዔ ኀዘን ወልደቂርቆስ**) adds three additional word classes: pronoun (**ተውላጠ ስም**), article (**መስተዓምር**), and exclamation (**ቃለ አጋኖ**) [43].

A. Noun

A noun is a name that represents a person, place, animal, feeling, or idea. It is a class of words used to identify things [39, 44]. Nouns are used to indicate gender, and in Amharic, there are two genders: masculine and feminine. Masculine gender is used for males, and feminine gender is used for females, whether they are persons or animals [45]. For example:

ተማሪ ነው - He (masculine) is a student

ተማሪ ናት - She (feminine) is a student

ተማሪው - The (masculine) student

ተማሪዋ - The (feminine) student

ያ ውሻ - That (masculine) dog

ያች ውሻ - That (feminine) dog

In Amharic, nouns can be made plural by adding suffixes such as "**አች፣ ዎች፣ ን፣ ት**", a prefix "**እነ**", or by repeating the noun using the infix '**ኣ**'. These modifications involve vowel changes or morphemes, as presented in Table 2.1.

Table 2.1: *Plural noun formation in Amharic*

Singular noun form	Morpheme	Plural noun form
ንብ	**-ኦች**	**ንቦች**
ሠራተኛ	**-ዎች**	**ሠራተኞች**
ህያው	**-ን/-ኦች**	**ህያዋን/ህያዋኖች**
ደብር	**-ት/ኦች**	**ደብራት/ደብሮች**
ሱራፌል	**እነ-**	**እነሱራፌል**
ትል	**-ኣ-**	**ትል-ኣ-ትል (ትላትል)**

As shown in Table 2.2, Amharic nouns are designated for a word category of definiteness, which can be achieved by appending morphemes or vowels according to the noun's number, gender, and/or ending.

Table 2.2: *Amharic Noun Definiteness Marking Patterns*

Indefinite Amharic noun	Number	Gender	Definite Amharic noun
ደብተር	Singular	Feminine	**ደብተር-ዋ [ደብተሯ]**
		Masculine	**ደብተር-ኡ [ደብተሩ]**
	Plural		**ደብተር-ኦቹ [ደብተሮቹ]**
ፍየል	Singular	Feminine	**ፍየል-ዋ [ፍየሏ] ፍየል-ኢቱ [ፍየሊቱ]**
		Masculine	**ፍየል-ኡ [ፍየሉ]**
	Plural	neutral	**ፍየል-ኦቹ [ፍየሎቹ]**

In Amharic, there exists a special colloquial form of personal nouns. For instance, nouns like "**ሰው**" when referring to "man", not "person", and "**ሴት**" when referring to "woman", not "female", often add the suffixes '**ዬ**' for masculine and '**ዮ**' for feminine, but only when used to refer to one person, not collectively or in plurals. Additionally, the definite articles '**ው**' in the case of masculine and '**ዋ**' in the case of feminine are further added, though not necessarily in the case of the two nouns already mentioned [46].

Examples:

ሰውዬ - man **ሴትዮ** - woman **ሰውዬው** - the man **ሴትዮዋ** - the woman

አባትዬው - the father **እናትዮዋ** - the mother **ባልዬው** - the husband **ሚስትዮዋ** - the wife In Amharic, nouns are distinguished by gender through the affixation of the morpheme "**-ኢት**" [46, 47]. For instance, **መለኛ-ኢት** [**መለኚት**], **ፍየል-ኢት** [**ፍየሊት**], and **አሮጌ-ኢት** [**አሮጊት**]. Moreover, nouns in Amharic are categorized by case, which can be either objective or possessive. The objective case is indicated by the morpheme '**-ን**', as seen in **በግ-ን** [**በግን**]. Meanwhile, possessive case is denoted by affixes or vowels depending on person, number, gender, and the ending of the noun. For personal pronouns, the possessive form is constructed by adding '**የ-**' as in **የ-ሱራፌል** [**የሱራፌል**].

Table 2.3: *Gender Marking in Amharic Nouns*

Subjective Case	Person	Number	Gender	Possessive case
ጎጆ	First	Singular	Neutral	**ጎጆ-ኤ** => **ጎጆዬ**
		Plural		**ጎጆ-ኣችን** => **ጎጇችን**
	Second	Singular	Feminine	**ጎጆ-ሽ** => **ጎጆሽ**
			Masculine	**ጎጆ-ህ** => **ጎጆህ**
		Plural	Neutral	**ጎጆ-ኣችሁ** => **ጎጇችሁ**
	Third	Singular	Feminine	**ጎጆ-ዋ** => **ጎጆዋ**
			Masculine	**ጎጆ-ው** => **ጎጆው**
		Plural	Neutral	**ጎጆ-ኣቸው** => **ጎጇቸው**
ላም	First	Singular		**ላም-ኤ** => **ላሜ**
		Plural		**ላም-ኣችን** => **ላማችን**
	Second	Singular	Feminine	**ላም-ሽ** => **ላምሽ**
			Masculine	**ላም-ህ** => **ላምህ**
		Plural	Neutral	**ላም-ኣችሁ** => **ላማችሁ**
	Third	Singular	Feminine	**ላም-ዋ** => **ላምዋ**
			Masculine	**ላም- ኡ** => **ላሙ**
		Plural	Neutral	**ላም-ኣቸው** => **ላማቸው**

B. Verb

In Amharic, verbs derive from roots and employ a combination of prefixes and suffixes to convey various grammatical features such as person, number, voice (active/passive), tense, and gender [44, 47, 48]. These roots, typically composed of three radicals, serve as the foundation of the verb, with the radicals themselves remaining constant while their forms may change to indicate different grammatical aspects. Moreover, prefixes and suffixes further modify the verb, but the core radicals persist, allowing us to identify the verb [45]. For example, let's consider the verb "**ነገረ**" (tell), which comprises three radicals: n, g, and r. If we examine one of its conjugated forms, "**ትነግሪአለሽ**" (you (feminine) will tell), we observe that the 2nd radical has taken the 6th form '**ግ**', and the 3rd radical has taken the 3rd form '**ሪ**'. Additionally, we note the prefix '**ት**' and the suffix "**አለሽ**". Despite these modifications and additions, the original three radicals, n, g, and r, persist, providing us with the identity of the verb.

A distinguishing characteristic of Amharic verbs lies in their placement within sentences, where they consistently occupy the sentence's end [42]. Additionally, these verbs exhibit another unique trait in that they bear a suffix indicating the subject of the sentence [42]. For instance, in the sentence "**ተማሪው መፅሐፉን ገለጠ**" (The student (masculine) opened the book), the final word "**ገለጠ**" functions as the verb, with the suffix '**ኧ**' (**ገለጠ-ኧ**) indicating that the subject corresponds to the third person singular masculine pronoun, "**እሱ**" (he).Furthermore, we can examine the Amharic sentence "**ልጅቷ ውሀውን ጠጣች**" (The girl drank the water), where the final word serves as the verb, and the suffix '**-ች**' signifies that the subject aligns with the third person singular feminine pronoun, "**እሷ**" (she).

Amharic verbs can be derived from verbal roots by affixing the vowel -**ኧ**- to produce the form C**ኧ**C1C1**ኧ**C-, e.g., **ክ-ብ-ር ➔ ክ**ኧ**ብብ**ኧ**ር**, which yields "**ከበር**" involving the repetition of the penultimate consonants and the affixing of the vowels -**ኧ**-. Verbal stems can be formed by affixing morphemes such as '**ተ-**', '**አ-**', "**አስ**", e.g., **ሰረቅ** (verbal stem) + '**ተ**-' (morpheme) **➔ ተሰረቅ.**

C. Adjective

An adjective is a word that describes, identifies, or adds more meaning to a noun or pronoun. While nouns convey the essence of an object, adjectives depict its qualities or attributes, such as shape, size, color, type, and property. It typically precedes the noun it modifies, as illustrated in the phrase

"**ረጅም ዛፍ ነው** - It is a tall tree," where the adjective "**ረጅም**" (tall) comes before the noun "**ዛፍ**" (tree) to indicate the tree's height [45]. If there are multiple adjectives, the first one will take the definite article, as seen in the phrase "**አዲሱ ፈጣን ባቡር** - the new fast train". Here, the adjectives "**አዲሱ**" (new) and "**ፈጣን**" (fast) both modify the noun "**ባቡር**" (train), but only the first adjective, "**አዲሱ**" is preceded by the definite article (the new) [45]. In the Amharic language, adjectives indicate gender and number through affixes, resulting in inflected words [42, 44]. In Amharic, adjectives sometimes take the plural suffix '**ች**' to agree with their nouns, particularly when accompanied by a definite article. For instance, "**አደገኛ መንገዶች** (**አደገኞች መንገዶች**) - "Dangerous roads". The correct expression is "**አደገኞቹ መንገዶች** - "the dangerous roads" [45]. Some common adjectives in Amharic form a special "reduplicated form" by doubling one of their letters. This reduplicated form is optionally used to indicate plural or "collective singular" force [28]. For example, **ትልቅ** (big) becomes **ትልልቅ**, **ትንሽ** (little) becomes **ትንንሽ**, **ታላቅ** (great/elder) becomes **ታላላቅ**, **አዲስ** (new) becomes **አዳዲስ**, **ቀይ** (red) becomes **ቀያይ**, and **ጥቁር** (black) becomes **ጥቋቁር**. Similarly, **አጭር** (short) becomes **አጫጭር**, **ነጭ** (white) becomes **ነጫጭ**, and **ረጅም** (long) becomes **ረዣጅም**. These are just a few examples of this reduplicated form in Amharic adjectives.

In Amharic, a distributive or selective sense is conveyed by reduplicating complete adjectives. For example:

- **ረጅም ረጅም እንጨት ልፈልግ** - Shall I look for big woods (i.e., big ones distributed among the others)?
- **ቆንጆ ቆንጆ ልጃገረዶች አሉ** - There are beautiful girls (i.e., there are beautiful ones from among the girls).

Adjectives can also function as nouns by themselves. In this case, the word "one", which is generally added in English, does not have to be translated: **ትልቁን ስጠን** - Give us the big one.

When an adjective used as a noun is in the plural, it must take the plural suffix "**ኦች**":

- **ደግ-ኦች => ደጎች** (Kinds)
- **ነጭ-ኦች => ነጮች** (whites)

The standard method of rendering a noun adjectival is to prefix the preposition '**የ**' (of), for instance: **የከተማ እድገት** - Urban development/Development of urban.

According to [45], there are derived adjectives described under nouns and verbs with the following suffixes:

- "**-ኣዊ**", which conveys the meaning of "appertaining to" and is used to indicate identity.
  Example: The word "**ወንጌላዊ**" uses the suffix "**-ኣዊ**" in the singular form and "**-ኣዊኣን**" in the plural form.
- "**-ኣም**", which is comparable to the English suffix "-ous".
  Example: The word "**መልካም**" uses the suffix "**-ኣም**" in the singular form and "**-ኣምኦች**" in the plural form.
- "**-ኣማ**", which approximates the English suffixes "-ful" or "-y."
  Example: Words like "**ፍሬያማ**", "**ወርቃማ**" and "**ጤናማ**" use the suffix "**-ኣማ**" in the singular form and "**ኣማኣች**" in the plural form.

Additionally, we can consider "**ሰካራም**" and "**ወፍራም** as adjectives derived from verbs **ሰከረ** and **ወፈረ** with the suffix "**ኣም**". Adjectives of Amharic language are marked for gender and number and results in an inflected word with affixes [42, 44].

In English, when we want to describe something using an adjective and the verb "to be" Amharic often accomplishes the same idea with just a verb. These verbs, called Adjectival Verbs, inherently carry the meaning of an adjective. They are often treated as verbs that signify a change or a state of becoming. So, while we might use the verb "to be" in English to translate them, their true meaning is more like an adjective followed by "to become". For example, the Amharic verb "**ከበደ**" (it became heavy) translates to "it is heavy" in English. However, there are instances where Adjectival Verbs are used more like the verb "to be" itself rather than indicating a change. In such cases, "it is heavy" could be literally translated in Amharic as "**ይከብዳል**" using the Present Imperfect form [45].

D. Pronoun

Pronouns are words that are used instead of nouns or noun phrases. They are used to avoid repetition and make sentences easier to understand. In Amharic, personal pronouns function as the

subject of a verb and are inherently integrated into the verb form, thus they are discussed within the sections dedicated to verbs [45].

Table 2.4: *Personal pronouns in Amharic*

Person	Number		Gender
	Singular	Plural	
First person	**እኔ** (I)	**እኛ** (We)	Neutral
Second person	**አንቺ** (You)	**እናንተ** (You)	Feminine
Second person	**አንተ** (You)	**እናንተ** (You)	Masculine
Second person (Polite form)	**እርስዎ** (You) respectful	-	Neutral
Third Person	**እሷ** (She/it)	**እነርሱ** (They)	Feminine
Third Person	**እሱ** (He/It)	**እነርሱ** (They)	Masculine
Third Person	**እሳቸው** (She/He)	-	Neutral

Unlike English, in Amharic, the nominative or subjective pronouns are identical to the objective pronouns. There are instances in Amharic where personal pronouns need to be expressed independently as separate words. There are the polite forms of personal pronouns in Amharic, as well as forms of the verb, used when addressing someone (i.e., a second person) or referring to someone (i.e., a third person) whose age or status merits respectful treatment. Though mainly derived from the third person plural, they exclusively convey singular meaning, without any distinct "polite forms" for plural use [45].

The Amharic demonstrative pronouns from [47] are shown in Table 2.5.

Table 2.5: *Amharic demonstrative pronouns*

Demonstrative pronoun			
	Feminine	**Masculine**	**Polite**
Nominative	**ይህች** (This)	**ይህ** (This)	**እኚህ** (This)
Accusative	**ይህችን** (This)	**ይህን** (This)	**እኚህን** (This)
Nominative	**ያች** (That)	**ያ** (That)	**እኚያ** (That)
Accusative	ያችን (That)	**ያን/ያንን**(That)	**እኚያን** (That)

Nominative	**እነዚህ/እነኚህ** (These)
Accusative	**እነዚህን/እነኚህን** (These)
Nominative	**እነዚያ/እነኚያ** (Those)
Accusative	**እነዚያን/እነኚያን** (Those)

E. Preposition

In Amharic, prepositions and articles can be viewed as a single word category within the preposition group, as they share similar characteristics [42]. In Amharic, as in English, prepositions communicate the relationship between two or more nouns or pronouns and typically appear before a noun or pronoun in a sentence or phrase. Prepositions possess two distinctive characteristics that set them apart from other word classes: they do not generate other words through derivation or any other methods, and they do not incorporate any suffixes [47]. Prepositions in Amharic indicate various relationships such as "place where," "time when," instrument, means or way, dative case (indicating the indirect object of the verb), ablative case (from, of), genitive (possessive case), direction, cause, subject matter, similarity or accord, inclusiveness, etc. [45]. The following words are examples of Amharic prepositions:

ለአባቱ/ላባቱ **ከ**እናቱ/ከናቱ **ከ**ኋላ/**ከ**ፊት/**በ**ላይ/ **በ**ውስጥ/ **በ**ታች/**በ**ስር/ **በ**ጎን **ከ**ቤት የለም

በባቡር ሐዲ ጥፋቷን **ለ**እናቷ ነገረች **ከ**ሀገሬ ወጣሁ ሰፊው **ከ**ጠባቡ ይሻላል

Prepositions with more than one letter, such as "**እንደ**" and "**ወደ**", are often written as separate words, as demonstrated in the following phrases: "**እንደ አባቱ/እንዳባቱ**" – as his father and "**ወደ እናቱ**" - to his mother. The following are Sentence level examples:

ወደ ቤቱ ሐዲ **እንደ** ነገሩ አደረገችው ጮክ ብሎ **ስለ** ተናገረ ሁሉም ሰምቷል

F. Adverb

Adverbs are words that elaborate or modify verbs or adjectives, answering questions of how, where, when, and what. The purpose of adverbs is to modify various aspects of the verb, such as its place, time, degree, and so on. In Amharic, adverbs typically come before the verb they modify [46]. The following are some common adverbs in Amharic: **ቶሎ** (quickly), **ገና** (still/yet), **ዛሬ**

(today), **ትናንትና** (yesterday), **ነገ** (tomorrow), **የት** (where), **መቼ** (when), **ብቻ** (only), **አሁን** (now), **በጣም** (very), **በፍጥነት** (rapidly), etc. [45]. In a sentence, for instance, in the Amharic sentence "**ሱራፌል ሀገሩን በጣም ይወዳል**" (Surafiel loves his country very much), the adverb is "**በጣም**/very", which elaborates the verb "**ይወዳል**".

In Amharic, adverbs may take compound forms, which are created by combining a preposition with other words. Consequently, short adverbial phrases exist. For example, **እንደገና** (again), **ሁልጊዜ** (always), **አንዳንድ ጊዜ** (sometimes), **በአንድ ቦታ/የሆነ ስፍራ** (somewhere), **በሁሉም ስፍራ** (everywhere), **እንደምን/እንዴት** (how), **ለምን/ስለምን** (why), **ስለዚህ** (therefore), **ለብቻው** (by himself), **ወዴት** (where), etc. [45]. Furthermore, in Amharic, there are words that serve as both nouns and adverbs to denote time and place. Examples include **ውስጥ** (inside), **ውጭ** (outside), **ላይ** (above/on top), **ታች** (below), **ፊት** (before), **ኋላ** (after), **አጠገብ** (beside), **ዙሪያ** (around), **መካከል/መኻከል** (middle), etc. [45].

2.2.2 Amharic Phrase

Phrases are syntactic structures that consist of one or more words but lack the subject-predicate organization of a clause. These phrases are composed of either only a head word or other words or phrases combined with the head. The other words or phrases combined with the head in phrase construction can function as specifiers, modifiers, and complements [48, 49]. In Amharic, phrases are categorized into five types, namely noun phrases (NP), Verb phrases (VP), adjectival phrases (AdjP), Adverbial phrases (AdvP), and Prepositional phrases (PP) [44, 46].

A. Noun Phrase

Noun Phrases are integral components of sentence construction, characterized by their structure around a noun as the head of the phrase, typically positioned at the phrase's end. NPs can be either simple or complex in their construction [48, 49].

Simple Noun Phrases: These comprise a single noun or pronoun without any subordinate clauses. For example, words like "**ፍየል**" (goat), "**ባቡር**" (train), "**እሱ**" (he), and "**እሷ**" (she) exemplify simple noun phrases.

Complex Noun Phrases: They consist of a noun accompanied by other constituents such as specifiers, modifiers, and complements. At least one of these constituents must be a sentence. An

illustration of a complex NP is: "**ኢትዮጵያ የምትገኝበት የቀይባሕር ቀጠና**" (The Red Sea region where Ethiopia is located), where "**ኢትዮጵያ የምትገኝበት**" (where Ethiopia is located) acts as a modifier, while "**የቀይባሕር ቀጠና**" (The Red Sea region) functions as the complement.

NPs can include various elements such as specifiers, modifiers, and complements. These elements precede the head noun in the construction. Specifiers, which can be NPs, adjectival phrases, or even sentences, serve to specify or identify the noun. For example, in "**የክብር እንግዳ**" (Guest of honor), "**የክብር**" (honor) acts as a specifier for the noun "**እንግዳ**" (guest). Modifiers in Amharic NPs can be genitive, deictic, or quantifier, providing additional information about the noun. For example, in the phrase "**አስቱር አወቀ ያወጣችው የመጀመሪያው አልበም**" (The album that Aster Awoke released for the first time), "**የመጀመርያው**" (the first) serves as a modifier.

B. Verb Phrase

A verb phrase comprises a verb as its head, located at the end of the phrase, along with other elements such as complements, modifiers, and specifiers. However, not all verbs take the same type of complement [49]. Based on this, verbs can be divided into two categories: transitive and intransitive [49]. Transitive verbs require transitive noun phrases as complements, while intransitive verbs do not [49]. For instance, in "**አየለ <u>ቤቱን</u> ሸጠ**" (Ayele sold his house) and "**አልማዝ <u>ልብስ</u> አጠበች**" (Almaz washed clothes) "**አበበ ዛሬ <u>ስልክ</u> ገዛ**" (Abebe bought a phone today) the underlined words serve as transitive noun phrases, functioning as complements of the respective head verbs. These verbs take one complement, but there are also verbs that take two complements, as illustrated in sentences such as "**አልማዝ <u>ለአበበ</u> <u>ቤት</u> ገዛችለት**" (Almaz bought a house for Abebe") and "**አበበ <u>ለእናቱ</u> <u>ገንዘብ</u> ሰጣት**" (Abebe gave money to his mother). In these examples, two complements are underlined.

In contrast, intransitive verbs do not take noun phrases as complements; instead, they take prepositional phrases, as seen in "**አበበ <u>ወደ ጎንደር</u> ሔደ**" (Abebe went to Gondar) and "**አልማዝ <u>ስለ ሥራ</u> ተጨንቃለች**" (Almaz is worried about work). Here, the underlined phrases **ወደ ጎንደር** and **ስለ ሥራ** represent prepositional phrases. Modifiers in verb phrases can manifest as PPs, NPs, adverbial phrases, or sentences. Like other constituents in VP construction, these elements may co-occur [49]. For example, "**በተደጋጋሚ ለአበበ በፖ᎓ስታ ደብዳቤ ልኬለታለው**" (I have sent

letters to Abebe many times through the post) includes a specifier "**በተደጋጋሚ**" (many times), a modifier **በፖ᎓ስታ** (through the post), and a complement **ለአበበ ደብዳቤ** (letters to Abebe).

In Amharic, VPs can be either simple or complex. The aforementioned examples are instances of simple VPs in Amharic. Amharic VPs become complex when they contain more than one verb or an embedded sentence [49]. For instance, "**<u>አበበ ሥራ እንደ ጀመረ</u> ሰማን**" (We heard that Abebe has started work) is a complex VP because it contains an embedded sentence **አበበ ሥራ እንደ ጀመረ** (that Abebe has started work) and two verbs **እንደ ጀመረ** (that he has started) and **ሰማን** (we heard) within the phrase construction.

C. Prepositional Phrase

In Amharic, a prepositional phrase consists of a preposition as the head and other elements such as nouns, NPs, or other PPs. Unlike some other phrase constructions, prepositions cannot stand alone as a phrase; instead, they must combine with other elements [49]. These elements may precede or follow the preposition, which acts as the head of the phrase. Generally, when the complements are nouns or NPs, the preposition precedes them, whereas if the complements are PPs themselves, the preposition tends to come at the end of the phrase [49].

For example:

- Head (Preposition) + Complement (nouns or NPs) – "**እንደ ጥሩ ጓደኛ**" (As/ like a good friend)
- Complement (PP) + Head (Preposition) – "**ከቤቱ በላይ**" (Above the house)

Note that, prepositional phrases may not only consist of complements but also prepositional modifiers.

D. Adjectival Phrase

Many adjective words in Amharic are derived from nouns or verbs. An adjectival phrase typically consists of an adjective as the head, positioned at the end, along with other elements such as complements, modifiers, and specifiers [49]. For example: "**አበበ ሁልጊዜም እንደ እናቱ ሰው አክባሪ ነው**" (Abebe is always respectful like his mother). Here, the underlined phrase is an AdjP. This phrase construction includes a modifier "**ሁልጊዜም**" (always) and a specifier "**እንደ እናቱ**" (like his mother). In the construction of adjectival phrases, complements always appear after

modifiers or specifiers, while modifiers come after specifiers. In the above example, the specifier "**ሁልጊዜም**" comes before the modifier "**እንደ እናቱ**" which is a PP, and the complement "**ሰው**" comes after all the constituents that are modifier and specifier. Similar to VPs and NPs, AdjP can be either simple or complex. Complex AdjP may contain embedded sentences. For instance, in the phrase "**<u>አልማዝ ከአበበ የተሻለ</u> ውጤት አስመዝግባለች**" (Almaz scored better than Abebe), the underlined phrase (**<u>አልማዝ ከአበበ የተሻለ</u>**) is an embedded sentence, making the overall phrase complex.

E. Adverbial Phrase

Adverbial phrases in Amharic consist of one adverb as the head word and one or more other lexical categories, including adverbs themselves, as modifiers. The head of the adverbial phrase is also found at the end. Unlike other phrases, adverbial phrases do not take complements. Most often, the modifiers in adverbial phrases are prepositional phrases that always precede the adverb [49]. For example, in the sentence "**ኢትዮጵያ <u>እንደ ገዛ ታሪኳ የከፋ</u> ጠላት የላትም**" (Ethiopia has no worse enemy than its own history), the underlined phrase is an adverbial phrase, and the head word is "**የከፋ**" (worse). The modifier within the adverbial phrase is "**እንደ ገዛ ታሪኳ**" (its own history), which is a PP.

2.2.3 Amharic Sentences

The Amharic language follows a Subject-Object-Verb (SOV) grammatical pattern, in contrast to English, which employs a Subject-Verb-Object (SVO) sequence of words [40, 41]. For example, consider the Amharic sentence "**አልማዝ በሩን ዘጋችው**" where "**አልማዝ**" functions as the noun, "**በሩን**" serves as the object, and "**ዘጋችው**" acts as the verb. In English, this sentence is translated as "Almaz closed the door" with "Almaz" as the subject, "closed" as the verb, and "the door" as the object.

Amharic sentences can be constructed using either simple or complex NPs and simple or complex VPs [41]. In terms of their order within a sentence, an Amharic sentence typically begins with the NP followed by the VP. For instance, "**ሁሉም የስብሰባው ተሳታፊዎች**" (All participants of the meeting) "**በአጀንዳው ላይ ተስማምተዋል**" (agreed on the agenda). "**ሁሉም የስብሰባው ተሳታፊዎች በአጀንዳው ላይ ተስማምተዋል**" translates to "All participants of the meeting agreed on the agenda" in English. This sentence presents a complete idea, unlike the preceding phrases.

It provides comprehensive information about the meeting agenda and the agreements of the participants. This final construction answers questions regarding the status of the agenda in relation to participant agreements. Within this final construction, there are NPs and VPs forming the sentence. The NP is "**ሁሉም የስብሰባው ተሳታፊዎች**" (All participants of the meeting), and the VP is "**በአጀንዳው ላይ ተስማምተዋል**" (agreed on the agenda). Other phrases within the sentence are constructed within NPs or VPs. From this structure, sentences can be categorized as either simple or complex.

2.2.3.1 Simple Sentences

Simple sentences are formed by using a simple NP followed by a VP that contains only one verb [41, 44, 46]. For instance, in the sentence "**አበበ መኪናውን አጠበው**" (Abebe washed the car), there is only one verb, "**አጠበው**" (he washed), which is a transitive verb that takes only one object, "**መኪናውን**" (the car). Another example is "**አበበ ለከበደ ገንዘብ ሰጠው**" (Abebe gave money to Kebede) where again, there is only one verb, and "**ሰጠው**" (gave), making it a simple sentence. However, in this example, the verb "**ሰጠው**" (gave) is transitive and has two objects, "**ለከበደ**" (Kebede) and "**ገንዘብ**" (money). Generally, all the examples mentioned above are simple sentences that contain various types of verbs. Simple sentences may include intransitive verbs, transitive verbs with one object, and transitive verbs with two objects. As described in [44, 46, 50], simple sentences are categorized into four types: declarative sentences, interrogative sentences, negative sentences, and imperative sentences.

A. Declarative Sentences

The Declarative sentence states a fact in either affirmative or negative form [51]. It is used to express the speaker's ideas and feelings about various things, events, etc., whether physical, mental, real, or imaginary. In Amharic, declarative sentences always conclude with the Amharic punctuation mark "**።**" which is equivalent to the period (.) in English [48].

Example: "**ሰራተኞቹ ትናንት ሐዲዋል**" (The workers went yesterday)

This sentence is a declarative sentence as it describes the action of the workers going somewhere. The head verb indicating movement from one place to another is "**ሐዲዋል**" (went), and its subject is the noun phrase "**ሰራተኞቹ**" (the workers). Additionally, the complement of the verb is the

Adverb Phrase “**ትናንት**” (yesterday). Furthermore, the VP “**ሰራተኞቹ ትናንት ሔደዋል**” (the workers went yesterday) is a projection of the head verb.

B. Interrogative Sentences (Questions)

The interrogative sentence, identified by its indirect word order and the use of function words, serves the communicative purpose of seeking information [51], questioning the subject, the complement, or the action specified by the verb [48]. For instance, in the sentence, “**አንተ መኪና መንዳት ትችላለህ?** (Can you drive a car?)” The pronoun “**አንተ**” (you) serves as the subject of the sentence. The verb “**መንዳት**” (drive) functions as the head of the overall sentence structure. Its complement is the noun “**መኪና**” (car). The auxiliary verb “**ትችላለህ**” (can) adds grammatical content to the information expressed by “**መንዳት**” (drive), which is considered the main verb.

C. Negative Statements

Negative sentences are used to express the opposite or negation of a declarative statement about something. For example, “**አስተማሪው በሰዓቱ አልመጣም** (The teacher did not come on time)”. In this instance, the sentence is negative because the prefix “**አል**” (did not) of the verb “**መጣ**” (come) indicates that the action of coming, expressed by the verb “come” did not happen.

D. Imperative Sentences

Imperative sentences convey instructions or commands, often using a second-person pronoun as the subject implied by the verb suffix. However, commands directed towards third persons can also use third-person pronouns or nouns as subjects [48]. These sentences, also known as jussives or directives, can be short or long, simple or complex, depending on context. They may express a wide range of intentions, such as giving orders, making requests, issuing warnings, or persuading others, and may end with either an exclamation mark or a period [51]. For example, in the sentence “**መስኮቱን ክፈት**” (open the window), the subject is implied in the verb, indicating second person, singular, either feminine or masculine in gender. The main structure of the sentence revolves around the verb “**ክፈት**” (open), with the complement “**መስኮቱን**” (the window). In another example, “**አንቺ ለአንድ ወር ያህል መጠበቅ አለብሽ**” (You have to wait a month), the subject is the noun “**አንቺ**” (you). The main structure revolves around the verb “**መጠበቅ**” (wait), with the complement “**ለአንድ ወር**” (a month), where the preposition ‘**ለ**’ merges with the determiner

"**አንድ**" to form a determiner phrase. The auxiliary verb "**አለብሽ**" (you have to) helps the main verb "***መጠበቅ***" (wait).

In [51] the exclamatory sentence is separately mentioned as one of the simple sentences in Amharic as described as it articulates thoughts and emotions and often begins with the pronoun what or the adverb how. It always has direct word order. The sentence has a falling tone in speaking and an exclamation mark in writing. For example, "**ምንኛ ቀርፋፋ መኪና ነው** (What a slow car it is!)" and "**ጨዋታው ድንቅ ነበር**! (It was a brilliant game)".

2.2.3.2 Complex Sentences

Complex sentences are those that have at least one complex NP), complex VP, or both a complex NP and VP [44, 46, 50]. Complex NPs are phrases that include at least one sentence embedded within the phrase structure. This embedded sentence can function as a complement. For example, consider this sentence "**በጃፋር ማተሚያ ቤት የታተመው የቅርብ ጊዜ ልብ ወለድ አዎንታዊ ግምገማዎችን እያገኘ ነው** (The recent novel published by Jafar publishing house is receiving positive reviews)". In this sentence, the head of the noun phrase is "**በጃፋር የታተመው የቅርብ ጊዜ ልብ ወለድ** (The recent novel published by Jafar publishing house)", with the complement "**የቅርብ ጊዜ** (the recent)" forming a simple noun phrase "**የቅርብ ጊዜ ልብ ወለድ** (The recent novel)". This noun phrase has been combined with the embedded sentence or clause "**በጃፋር ማተሚያ ቤት የታተመው** (published by Jafar publishing house)" to form a complex noun phrase.

2.2.4 Amharic Punctuation

The Amharic language in writing incorporates both indigenous punctuations and those derived from English. The common punctuations used in Amharic are presented in Table 2.6 [45].

Table 2.6: *Amharic Punctuations*

Amharic Punctuation	Equivalent English punctuation
: (ሁለት ነጥብ)	Space
:: (አራት ነጥብ)	. (Full stop)
? (ጥያቄ ምልክት)	? (Question mark)
! (ቃለ አጋኖ)	! (Exclamation mark)
፣ (ነጠላ ሰረዝ)	, (Comma)
፤ (ድርብ ሰረዝ)	; (Semi-colon)
" " (ትእምርተ ጥቅስ)	" " (Quotation mark)
… (ነጠብጣብ)	… (Ellipsis)

2.3 Overview of English Language

The English language has become the dominant global common language, used as a medium of communication in business, education, science, and technology, and it serves as the medium for international communication, commerce, and diplomacy, with over 1.5 billion speakers worldwide [52]. In essence, the English language is a Germanic language brought over by invading tribes from the European continent into what would eventually be recognized as England [53]. The English alphabet follows a Latin script structure comprising 26 letters, each with both upper and lower case forms. Among these, there are 5 vowels (a, e, i, o, and u), while the remaining 19 letters serve as consonants.

2.3.1 English Morphology

The term “morphology”, derived from the Greek word “Morph” meaning “shape or form”, refers to the study of forms and, in linguistics, specifically to the study of words, including their formation and internal structure [54, 55]. A word is defined as a series of letters separated by spaces or, sometimes, by punctuation, and it carries meaning, serving as a building block of sentences and forming the essential elements of coherent language expression [56]. Words can vary in complexity, being composed of smaller units known as morphemes, which represent the smallest components of a word that carry grammatical function or meaning [57]. A word that can stand alone without depending on other morphemes for meaning is called a free morpheme, whereas morphemes that must be attached to other word parts to convey meaning are referred to as bound morphemes [57].

Word classes, traditionally known as parts of speech in English, fall into two categories: open and closed classes. Open classes, which include nouns, verbs, adjectives, and adverbs, readily accept new words and make up the majority of the vocabulary. Closed classes, on the other hand, rarely incorporate new words and consist of a finite list, such as pronouns and auxiliaries. For example, while it's feasible to list all pronouns, listing all nouns is impractical due to their vast number and the continual creation of new ones [53, 55]. The form of a word is influenced by its grammatical function within each word class, which may be associated with factors such as tense for verbs, plurality and possession for nouns, and degree of comparison for adjectives [55].

Morphemes can be further categorized as root, derivational, or inflectional [55, 57]. A root morpheme serves as the fundamental form to which other morphemes are attached, providing the core meaning of the word. For instance, the morpheme {saw} functions as the root in the word "sawers" [55]. The processes of inflectional and derivational morphology involve the use of affixes, which can be prefixes and/or suffixes [57].

Inflectional morphology refers to the process of creating new word forms by adding specific affixes, or sometimes making changes without affixes at all [55]. These forms do not create distinct words but modify the word to indicate grammatical properties such as plurality, as in the {-s} of "books," or past tense, as in the {ed} of "corrected" [57]. For example, the variations of the word "help" such as "helps" and "helped" represent different forms of "help" that reflect its grammatical function within a sentence, influenced by number and tense. This process is known as inflectional morphology.

Table 2.7: *The Eight English Inflectional Morphemes [57]*

Word class/ phrase	Affixation	Example
Noun	{-s} plural	The lions
Noun Phrases	{-s} genitive/possessive	The lion's family
Adjectives/Adverbs	{-er} comparative	Slower
	{-est} superlative	Slowest
Verbs	{-s} Third person singular present tense	Proves
	{-ed} past tense	Proved
	{-ing} progressive/present participle	is proving
	{-en} past participle	has proven/was proven

The primary purpose of derivational morphology is to facilitate the creation of new lexemes [58]. Lexemes are members of lexical categories such as nouns, verbs, and adjectives, and the derived lexemes may belong to a different category than their bases [55, 58]. Lexemes can be thought of as families of words that differ only in their grammatical endings or forms; for example, the singular and plural forms of a noun like "class" and "classes", the present, past, and participle forms of verbs such as "walk", "walks", "walked", and "walking", or the different forms of a pronoun such as "I", "me", "my", and "mine", all represent a single lexeme [59]. Simply put,

derivational morphemes are added to word forms to create distinct words. For instance, the suffix {-er} is a derivational morpheme that, when added to a verb, typically transforms it into a noun indicating the person or thing performing the action denoted by the verb. For example, combining "teach" with {-er} produces "teacher", meaning "someone who teaches" [57].

The distinction between inflectional and derivational morphemes is that inflection creates various word forms from existing lexemes, whereas derivation generates new lexemes from other lexemes. Inflectional morphemes can be attached to derivational ones, but derivational morphemes cannot be attached to inflectional ones.

Complex words, which are made up of more than one morpheme, are not simply random sequences of morphemes. For example, in the word "butterflies", the plural suffix {-es} must be attached to the whole compound "butterfly" rather than just to "fly" and then adding "butter". This is because the plural suffix is only applicable to nouns, not to verbs or particles. The words "butter" and "fly" together form a noun only when combined as a compound [57]. Another example is the word "unsustainability", which can be broken down as [noun[Adjective[Verb sustain]abil]ity]. This structure can also be represented using a tree diagram as follows:

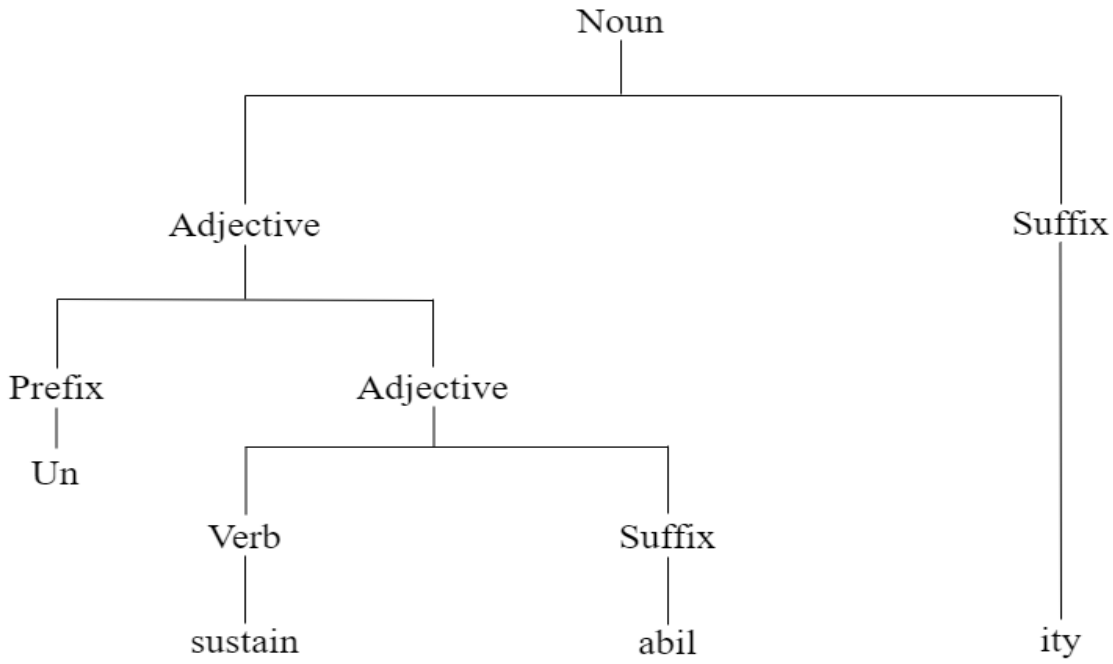

Figure 2.1: *Sample complex word structure in English*

2.3.2 English Phrase

A phrase is a word or group of words that acts as a single unit within a sentence and always contains a head, or headword. The five types of phrases in English are the noun phrase, verb phrase, adjective phrase, adverb phrase, and prepositional phrase [53, 60].

Noun phrase: A noun phrase has as its head a noun, a pronoun, a nominal adjective, or a numeral [53]. Most noun phrases also include a noun signaler or marker known as a determiner [60]. Noun phrases with a noun as their head are commonly introduced by the definite article “the’ or the indefinite articles ‘a’ or “an”. “The” and ‘a’ are the most commonly used determiners in this class, which also comprises words like “some”, “both” and "this" [53]. Noun phrases may have modifiers, which provide additional information to specify more precisely the referent of the head noun. Modifiers depend on the head and can generally be omitted without disturbing the sentence structure. However, they often convey crucial information, so leaving them out can impair communication [53].

Consider the following example, “During the thrilling 2000 Sydney Olympics, the 10,000-meter race showcased exceptional athleticism, with Haile Gebrselassie narrowly defeating Paul Tergat to win a gold medal”. In this sentence, “the 10,000-meter race” is a noun phrase introduced by the definite article “the” and modified by the number “10,000-meter”, which acts as a premodifier for the head noun “race”. The phrase “a gold medal” is another NP, introduced by the indefinite article ‘a’, providing specific detail about the prize for which the athletes were competing. Both “Haile Gebrselassie” and “Paul Tergat” function as proper NPs, referring to the prominent athletes in this competition.

Verb Phrase: A verb phrase uses a main verb as its head, which may be accompanied by auxiliary verbs. These auxiliary verbs fall into two main categories: primary auxiliaries and modals, or secondary auxiliaries [53]. The primary auxiliaries include “be”, “have”, and “do”. The auxiliary “be” serves two purposes: first, it combines with an “-ing” participle to form the progressive tense, e.g., “is playing”, and second, it combines with an “-ed” participle to form the passive voice, e.g., “is played”. The auxiliary "have" forms the perfect aspect when combined with an “-ed” participle, e.g., “has played”. The auxiliary "do" functions as a placeholder operator, used to form interrogative and negative sentences when no other operator is present, e.g., “Did they play?” or “They didn't play”.

Modal auxiliaries, such as “can”, “could”, “may” and “might” convey ideas of certainty or permission and are followed by an infinitive [53]. The order of appearance for these auxiliaries in a verb phrase is as follows: Modal - perfect “have” - progressive “be” - passive “be” - main verb. It's not common for all of these auxiliaries to be present in one verb phrase, but it's certainly possible. For instance, in a sentence “Should have been being played” each auxiliary is followed by the appropriate verb form: “should” (modal), “have” (infinitive), “been” (-ing participle), “being” (-ing participle), and “played” (-ed participle).

Adjective Phrase: An adjective phrase uses an adjective as its head word, which may be preceded by premodifiers and followed by postmodifiers. Adjectives can appear in a hierarchical sequence, where each adjective modifies the next [53]. For example, in the sentence “We had some nice crisp white wine to go with it", “wine” is first described by “white”, then further specified by “crisp” and “nice”. Adjective phrases serve two main purposes: as premodifiers of nouns and as subject predicatives. In the sentence “Well, it's a much less popular route”, the adjective phrase “much less popular” modifies the noun “route”. In “No, I mean John was extraordinarily ugly”, the phrase “extraordinarily ugly” functions as the subject predicative.

Adverb Phrase: An adverb phrase is headed by an adverb, which can be preceded by premodifiers and, less frequently, followed by postmodifiers. These phrases can modify adjectives, adverbs, or verbs by acting as adverbials or complements [53]. For example, in “We are far too close to it”, the adverb phrase “far too close” modifies the adjective “close”. In “You need to have your teeth extremely thoroughly cleaned”, the phrase “extremely thoroughly” modifies the adverb “cleaned”. Additionally, in “Refunds of fees are not normally available”, the adverb phrase “not normally” acts as an adverbial.

Intensifying adverbs, such as “very”, typically premodify other adverbs, as in “wear this occasionally but very rarely now”. They can also postmodify adverbs, with examples like “It was quoted often enough in the recent debate”. Comparative clauses or phrases can postmodify adverbs when linked to a comparative inflection, as in “I think he's feeling the time going more slowly than I am since he's the one left behind” [53].

Prepositional Phrase: A prepositional phrase consists of a preposition followed by an object, usually a noun phrase, and may appear in various parts of a sentence, either as part of a noun phrase or modifying a verb. When functioning within a noun phrase, a prepositional phrase typically follows the noun, adding descriptive detail or clarification [60]. For instance, in “Our new neighbors across the hall became our best friends”, the phrase “across the hall” modifies “neighbors”. In a different context, such as "Our good friends live across the hall," the same phrase serves as an adverbial, specifying “where” the action takes place.

Modifiers of nouns are referred to as "adjectivals," while modifiers of verbs are called "adverbials," regardless of the form they take. Prepositional phrases can function as postmodifiers of nouns, e.g., "Everybody questions the significance of the results," or as postmodifiers of adjectives, e.g., "He was ignorant of the crucial lack of an extradition treaty." They may also serve as subject predicatives, e.g., “The sun was just as it is in Ethiopia”, object predicatives, e.g., “She never slept from the time I brought her out of the hospital”, or as adverbials, e.g., “Every Saturday, I stood there waiting by the door”.

2.3.3 English Sentences

A sentence conveys a complete thought, formed by a group of words working together to express a full idea. Each sentence arranges words meaningfully, representing a collection of words conveying a complete idea [53]. Understanding sentence structure begins with recognizing its two main parts: the subject and the predicate [45]. The subject is the main focus, representing the central theme, while the predicate provides information about the subject. Both components describe specific functions within the sentence. For example, in the sentence “The teacher is coming,” “the teacher” is the subject, functioning as a noun phrase made up of the definite article “the” and the noun “teacher.” “is coming” serves as the predicate, consisting of the auxiliary verb “is” and the present participle “coming,” forming a verb phrase that indicates the action or state of the subject [53].

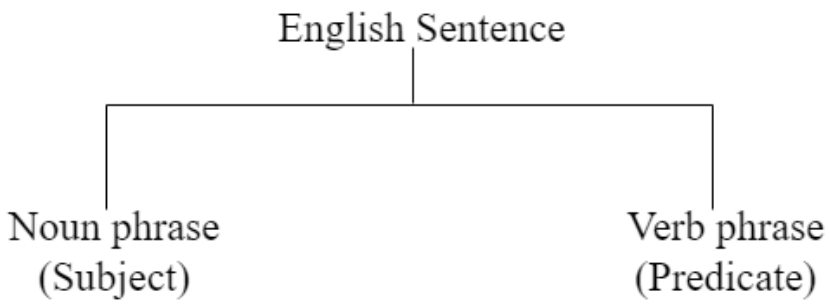

Figure 2.2: *Generic English sentence structure*

A sentence contains one or more clauses, with a clause being a group of words that includes a subject and a verb that are related [38]. Both the subject and predicate can be further divided into smaller parts, as illustrated in Figure 2.3. The subject may be simple or compound, and the verb may be transitive or intransitive [60].

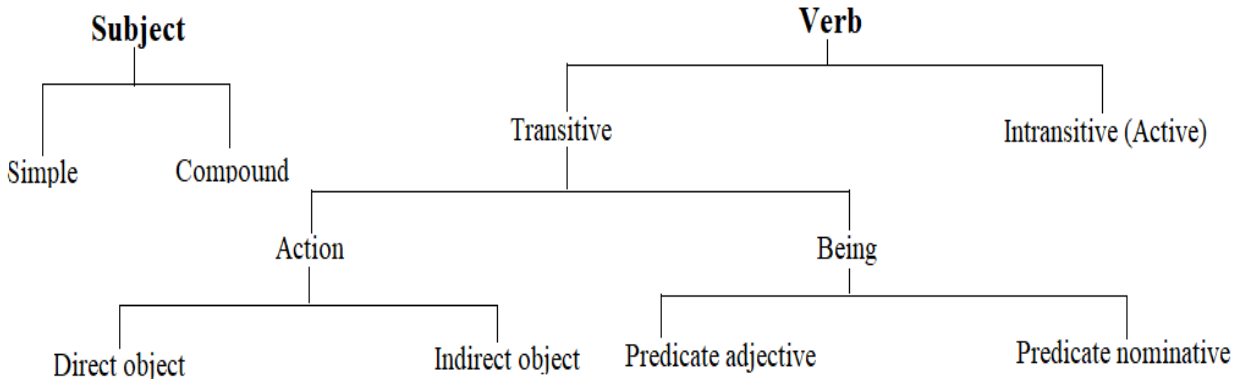

Figure 2.3: *Detailed English sentence structure*

English sentences can have multiple elements making up the subject, linked by conjunctions such as “and” or “or”. For instance, in the simple subject sentence “The cat is sleeping on the mat”, the subject is “cat”. In a compound subject sentence, such as “The cat and the dog are sleeping on the mat”, both “cat” and “dog” are the subjects, joined by “and”. Typically, in the subject + predicate structure, English sentences use an intransitive verb, which does not require an object, rather than linking verbs like “am,” “is”, “are”, “was”, “were”, or stative verbs [53, 60]. The subject can be a single noun, pronoun, or noun phrase [53, 60]. For example, in the sentence “Children play”, the subject is the plural noun “children.” In “They laugh,” the subject is the pronoun “they”. In the sentence “The happy children play,” the subject is the noun phrase “The happy children”.

In addition to the subject + predicate pattern, English sentences follow other structures such as SVA (Subject + Verb + Adjunct), SVC (Subject + Verb + Complement), SVO (Subject + Verb + Object), SVOA (Subject + Verb + Object + Adjunct), SVOC (Subject + Verb + Object + Complement), and SVOO (Subject + Verb + Object + Object) [53, 60]. Table 2.8 provides example sentences for each pattern.

Table 2.8*: English Sentence Patterns*

Pattern	Example	Subject	Verb	Object(s)	Adjunct	Complement
SV	We sleep.	We	sleep	-	-	-
SVA	The bird sings joyfully.	The bird	sings	-	joyfully	
SVC	She became a professor.	she	became	-	-	a professor
SVO	I love Ethiopia	I	love	Ethiopia	-	-
SVOA	I ate dinner early.	I	ate	dinner	early	-
SVOC	She considers him a friend.	She	considers	him	-	a friend
SVOO	He gave her a book.	She	gave	her *and* a book	-	-

There are two types of clauses: independent and dependent clauses. A sentence contains at least one independent clause and may also include one or more dependent clauses [60]. An independent clause, or main clause, is a complete thought that can stand alone. A dependent clause, or subordinate clause, is an incomplete thought that cannot stand alone, and recognizing it requires more than spotting the subordinating conjunction, as subordinating conjunctions like “after”, “although”, “because”, “if”, “since”, and “while” introduce dependent clauses that rely on the rest of the sentence for meaning [53]. For example, in “Although Ethiopia is known for its rich natural resources, many of its people still face economic challenges”, “Although Ethiopia is known for its rich natural resources” is a dependent clause, while “many of its people still face economic challenges” is an independent clause that can stand alone.

English sentences are traditionally classified by structure and complexity into four types: simple, compound, complex, and compound-complex [53]. Simple sentences consist of one main clause without subordination. For example, “Merry walked to the park,” “What time does the match start?” and “He plays football every weekend” are simple sentences. Compound sentences consist of two or more main clauses, often linked by coordinating conjunctions such as “for”, “and”, “but”,

“or”, “yet”, and “so”, usually preceded by a comma. For example, “I enjoy reading books, but my wife prefers watching movies”. Complex sentences consist of a main clause with one or more subordinate clauses. For instance, “Although it was raining, she went for a walk” contains one independent and one dependent clause, connected by a subordinating conjunction. A compound-complex sentence combines both forms, containing one or more independent clauses and one or more dependent clauses. For example, “She went for a walk, although it was raining, and she took an umbrella” illustrates a compound-complex sentence.

Additionally, English sentences can be categorized based on their purpose as declarative, interrogative, imperative, or exclamatory [53, 60].

Declarative sentences are used to make statements or convey information, providing facts, opinions, or descriptions [53, 60]. For instance, “The sun rises in the east”, “I enjoy reading poems”, and “Mount Ras Dashen is the tallest mountain in Ethiopia” are all examples of declarative sentences.

Interrogative sentences, on the other hand, are designed to ask questions, seeking information or clarification [53, 60]. Examples include “What time is the meeting?”, "Have you finished your thesis work?" and “Where is the nearest hotel?”.

Imperative sentences are used to give commands or orders, make requests, or offer suggestions [53, 60]. Examples such as “Please pass the book”, “Close the door behind you” and “Study for your exams” demonstrate the imperative form. In imperative sentences, the subject is often not stated. The recipient of the sentence is understood to be the subject [53]. For example, in “clean the room” the subject “You” is understood as “you clean the room”.

Exclamatory sentences express strong emotions such as surprise, excitement, or emphasis [53, 60]. For instance, “What a beautiful game!”, “I can't believe we won the game!” and "How delicious this cake is!” are the exclamatory types.

2.4 Evolution of Machine Translation

In its early stages, machine translation (MT) systems were primarily developed for military applications [61]. Petr Petrovich Troyanskii is considered a pioneer in MT. In 1939, he submitted proposals for mechanical translation to the Soviet Union Academy of Sciences. Despite initial

discussions, these proposals were never implemented, and Troyanskii's groundbreaking efforts to mechanize translation during the 1930s and 1940s remained largely unknown [3, 62].

In 1949, American scientist Warren Weaver, in collaboration with British engineers, proposed the idea of using computers to tackle the task of MT [3, 6, 63, 64]. This marked a turning point in the field. In 1954, Georgetown University, in partnership with International Business Machines (IBM) Corporation, conducted a successful Russian-English machine translation experiment using the IBM-701 computer. This significant achievement not only demonstrated the feasibility of MT but also initiated further research in the field, bringing the long-held dream of MT closer to reality [7, 61, 64].

MT can be broadly categorized into two main types based on methodology: RBMT and corpus-based or data-driven MT (CBMT). RBMT relies on linguistic rules defined by human experts to describe the translation process, requiring substantial human input [6, 61, 62]. In contrast, CBMT methods derive knowledge automatically by analyzing translation examples from parallel corpora, which are manually constructed by human experts [65]. Data-driven MT techniques, which include example-based MT (EBMT), SMT, and NMT, aim to learn translation patterns from large amounts of parallel data. Generally, an increase in the volume of training data leads to improvements in translation quality [61]. RBMT aligns with rationalism, while CBMT follows an empirical approach [66].

2.4.1 Rule Based Machine Translation

RBMT, one of the earliest MT approaches, employs bilingual dictionaries and grammar rules to translate source language (SL) texts into target language (TL) texts [6, 61]. These systems rely on linguistic information from dictionaries and grammars, covering semantic, morphological, and syntactic aspects of each language. This helps generate output sentences through analysis of both the source and target languages [65]. Despite its solid linguistic foundation, RBMT faces challenges such as labor-intensive rule creation, difficulty in rule maintenance, and limited scalability for open-domain and multilingual translation [61].

Also known as knowledge-based or classical MT, RBMT utilizes morphological, syntactic, semantic, and contextual knowledge of both SL and TL to perform translation tasks. This is facilitated by computer-accessible dictionaries and grammar rules derived from theoretical

linguistic research [64]. The core approach involves linking the structure of the input sentence with that of the output sentence while preserving their meanings [65].

RBMT systems are classified into three primary methods: literal (direct) translation, interlingua-based translation, and transfer-based translation [6]. These methods operate under different linguistic principles, leading to varying types and depths of analysis.

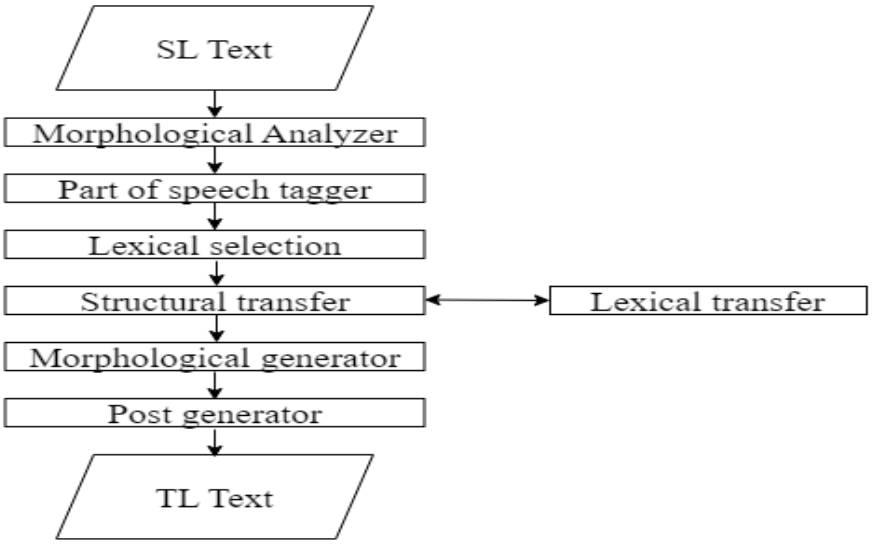

Figure 2.4: *The General Architecture of RBMT*

Literal (Direct) RBMT: Also known as word-based or dictionary-based translation, literal RBMT translates words similarly to consulting a dictionary without accounting for nuanced meaning correlations [6, 67]. It works at the word level, directly mapping SL words to TL equivalents without intermediary representation, resulting in a one-way translation [65]. This approach involves minimal syntactic and semantic analysis, often following a word-for-word translation with minor grammatical adjustments [65]. While effective for limited sentence structures, the literal approach lacks deeper semantic understanding. An example is the Washington MT system from the 1950s, which translated Russian to English but produced unsatisfactory results [64]. Another notable system, SYSTRAN, launched in 1978, initially translated Russian to English and later expanded to other languages [6, 61].

Transfer-Based RBMT: This method translates SL texts by analyzing sentence structures and applying linguistic rules across languages, using dictionaries such as SL, SL-TL bilingual, and TL dictionaries [6]. It creates translations based on an intermediate representation, transferring

syntactic structures between languages. The process involves three stages: SL analysis, transfer of representations to TL equivalents, and generation of the final TL texts [61]. Transfer-based systems operate on sentence structures rather than linear word strings, focusing on syntactic relationships between languages [64].

Interlingual RBMT: This approach offers a more abstract alternative by dividing translation into two phases: converting the SL into an intermediate, language-neutral form (interlingua), and then into the TL [6]. The main advantage is its ability to handle multilingual translation with fewer translation pairs, requiring only 2N pairs for N languages, as opposed to N*(N-1) pairs in other systems [68]. However, defining an effective interlingua is challenging, limiting its use to specific domains [6]. The interlingual method involves transforming the SL into an abstract representation that captures its full semantic, morphological, and syntactic content before generating the TL [64].

The three RBMT approaches can be summarized as follows: Direct systems map the input sentence directly to the output sentence, Transfer systems employ morphological and syntactic analysis to translate sentences, and Interlingual RBMT systems transform the input sentence into an abstract representation before mapping it to the final output [3].

2.4.2 Example Based Machine Translation

EBMT, also known as memory-based translation, operates on the principle of translating by analogy, a concept introduced by Makoto Nagao in 1984 [69, 70]. This approach leverages bilingual corpora containing parallel texts as its primary knowledge source, retrieving analogous language pairs to generate translations [3, 65, 71]. EBMT systems are trained on sets of SL sentences and their corresponding TL translations, establishing point-to-point mappings [4, 65]. When a new input sentence is encountered, EBMT systems identify similar examples from the corpus and use them to generate translations [69, 71]. The core idea is that if a sentence has been translated before, a similar translation should be applicable to a similar sentence, allowing EBMT to generate accurate results based on past examples [70].

Despite its reliance on bilingual corpora and analogy-based translation, EBMT has shown promise in various translation tasks [65]. The process of EBMT involves four main tasks: example acquisition, example base management, example application, and synthesis, all grounded in the principle of translation by analogy [69]. However, EBMT's coverage is limited by the inability of bilingual corpora to represent all linguistic phenomena in language pairs [61]. As a result, EBMT

is most commonly used in computer-aided translation systems [61]. Additionally, the process of analogy-based machine translation can be time-consuming [69].

Table 2.9: *Sample Amharic-English Bilingual Corpus for EBMT Illustration*

English Sentence	Amharic Sentence
I need a **new phone**.	**አዲስ ስልክ እፈልጋለሁ።**
I need a **different job**.	**የተለየ ሥራ እፈልጋለሁ።**
He likes to **drink coffee**.	**ቡና መጠጣት ይወዳል።**
He likes **watching a movie**.	**ፊልም ማየት ይወዳል።**
Is this the **correct answer**?	**ትክክለኛው መልስ ይህ ነው?**
Is this the **right direction**?	**ትክክለኛው አቅጣጫ ይህ ነው?**
Are you going to the **home today**?	**ዛሬ ወደ ቤት ትሄዳለህ?**
Are you going to the **church tomorrow**?	**ነገ ወደ ቤተክርስቲያን ትሄዳለህ?**

A bilingual parallel corpus in EBMT, as illustrated in Table 2.9, consists of sentence pairs in both the source (Amharic) and target (English) languages, following their respective syntactic structures of SOV and SVO. The table displays a minimal pair, where sentences differ by only one element, making it easier to understand translations of smaller linguistic units.

2.4.3 Statistical Machine Translation

Statistical Machine Translation (SMT) is a data-driven approach that employs probabilistic models to predict translations [4, 5]. It uses machine learning (ML) techniques to automatically derive translation rules from large bilingual corpora, eliminating the need for manual rule creation by human experts [61]. SMT assumes that any sentence in one language can be translated into another with varying probabilities [6]. The goal of SMT is to find the sentence with the highest probability of being the correct translation.

SMT is composed of three main problems: model problems, which involve defining a probability model for translation; training problems, which use parallel corpora to determine the parameters of the model; and decoding problems, which focus on finding the most probable translation of an input sentence [6].

The foundational SMT model, introduced by IBM researchers such as Brown *et al*. in 1990, is based on a statistical framework that primarily handles lexical (word-level) translations [6, 72]. The process of SMT relies on the probability distribution P(e | a), where P(a | e) represents the translation model and P(e) represents the language model. By applying Bayes' theorem, SMT models find the most probable translation for a given source sentence (SL) in a target language (TL), such as translating from Amharic to English [65, 72]. For example, if a decoder is given an English sentence (e), it finds the Amharic sentence (â) that maximizes the probability P(a | e) to identify the most accurate translation.

SMT relies on three key components: the SL model, the translation model, and a decoding algorithm. The translation model produces potential translations, and the SL model ensures that these translations are grammatically correct [65, 72].

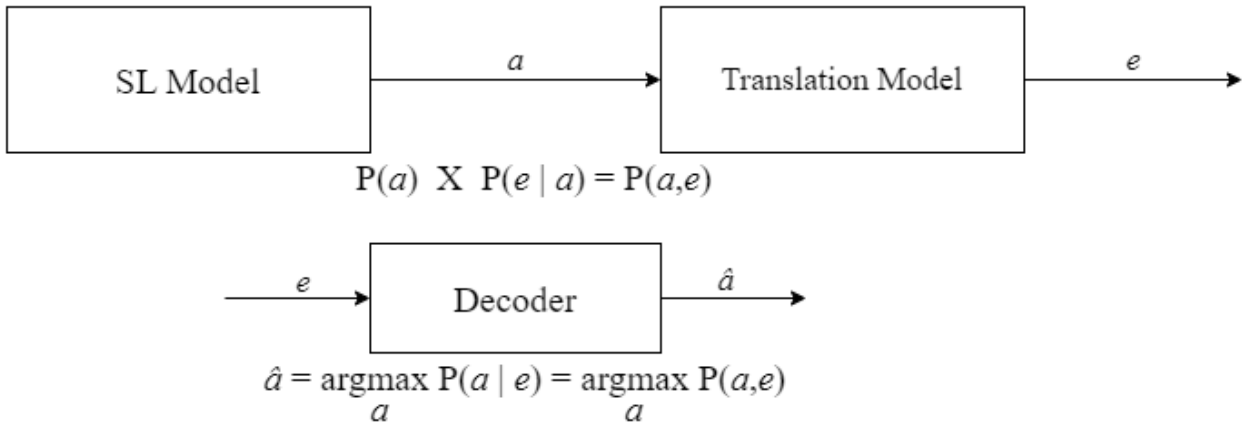

Figure 2.5: *A Word-to-Word Statistical Machine Translation System [72]*

Despite the initial breakthrough with the word-based SMT model by Brown *et al*. [72], it has a notable limitation: it primarily focuses on modeling lexical dependencies between individual words. This approach overlooks larger linguistic units such as phrases, leading to challenges in translating sentences accurately. To overcome this limitation, phrase-based models were introduced by Vogel *et al*. [73], Marcu and Wong [74], and Och and Ney [75]. These models treat phrases rather than single words as the basic units of translation, enabling more flexible and contextually appropriate translations.

Several enhancements to SMT have been explored to address its limitations further. Factored SMT models [62] incorporate additional linguistic information, such as morphology, to improve

translation quality. Hierarchical SMT models [63] handle complex syntactic structures by translating sentences based on hierarchical phrase rules, and syntax-based SMT models utilize parsing trees on the source or target language side to provide better structural understanding of the input sentences [76, 77, 78, 79, 80].

Phrase-based SMT has transformed machine translation by breaking down sentences into phrases, translating each phrase independently, and reordering them in the target language (TL) [71, 81]. Unlike word-to-word SMT systems, phrase-based translation does not require aligning individual words directly, which increases its flexibility [4, 74]. In phrase-based models, each lexical unit is a sequence of words rather than a single word, and each unit is associated with a score or weight that reflects its appropriateness as a translation [3]. For example, in an Amharic-to-English phrase-based SMT model, the lexicon entries could be represented as tuples (e, a, k), where e is a sequence of English words, a is a corresponding sequence of Amharic words, and k is the associated score [3].

The translation process in phrase-based SMT involves three main steps: segmenting sentences into phrases, translating each phrase, and arranging the translated phrases in the proper order. This method significantly improves translation quality compared to word-based models [5, 82]. Phrase-based models rely on probability tables, learned from bilingual corpora, to translate phrases from the SL to the TL [3]. However, they face challenges with long-distance reordering, which can sometimes lead to disjointed translations, a phenomenon known as "phrase salad" [5]. Though efforts have been made to incorporate syntactic information into phrase-based models, their compatibility with hierarchical models remains limited [83, 84, 85, 86, 87].

Various alternatives to traditional phrase-based models have been explored, such as the alignment template model and the discontiguous phrase-based translation model [75, 88]. These variants aim to enhance translation by allowing greater flexibility in phrase alignment and reordering. Popular toolkits for phrase-based translation include the Pharaoh Toolkit by Koehn and its open-source successor, Moses [5, 89, 90].

2.4.4 Neural Machine Translation

NMT, an end-to-end model, employs an encoder-decoder framework, typically consisting of a single large neural network with separate sub-networks for the encoder and decoder, respectively

[9, 91, 92, 93]. Essentially, NMT combines an encoder, trained to compute a representation of each word in an input sentence (referred to as word embedding), with a decoder that processes one word at a time from the sentence representation and generates the target translation [94, 95, 96]. This model does not require separate components like word aligners, translation rule extractors, or other feature extractors. Instead, it utilizes a single sequence model to predict one word at a time [9]. An artificial neural network (ANN), also known as neural networks, predicts the probability of a sequence of words, simulating the human brain by establishing connections between processing elements, functioning similarly to neurons [97]. In NMT, the encoder converts SL words into semantic vector representations, which are then used by the decoder to generate the target sentence word by word [9, 92, 93, 94]. It directly models the conditional probability p(t|s) of translating an SL sentence to a TL sentence. RNN, CNN, and transformer SA/FFN are commonly used approaches based on the NMT paradigm [10, 98, 99, 100]. These approaches, often referred to as deep neural networks (DNNs), aim to learn syntactic and semantic representations for discrete words, phrases, structures, and sentences in real-valued continuous space, positioning similar words, phrases, or structures close to each other [7, 91].

The encoder reads each symbol in input sequences one word at a time until it reaches the end-of-sequence symbol, effectively compressing the input sequence into hidden states [3, 95]. This encoding process is simple, using RNN architectures widely used for computing the source sentence representation [94]. The encoder RNN processes the source sentence $\boldsymbol{S} = (s_1, s_2, s_3, \ldots\ldots s_{T_s})$ where T_s is the length of $\boldsymbol{S}$, into hidden states **h** = (h_1, h_2, h_3 …. h_{Tn}) [46, 89]. Typically, the encoder takes the last hidden state, h_{Tn}, as the representation of the source sentence [61]. On the other hand, the decoder, which also operates as an RNN, generates translations based on the encoded representation. It updates its hidden state h_t at each time step **t** based on the summary C of the entire input sequence, along with the previous hidden state h_{t-1} and the preceding target word t_{t-1} [3]. The conditional distribution of the next symbol, i.e., the next word in the target language sentence given the source language sentence, is calculated using $p(s_t|s_1 \ldots,s_{t-1}) = g(h_t,y_{t-1},C)$ [3]. The RNN encoder-decoder model is trained jointly to maximize the conditional log-likelihood [3]. Generally, the encoder RNN processes an input sequence S = (s_1, s_2, s_3..., s_n) of n elements and returns state representations h = (h_1, h_2, h_3..., hn), while the decoder RNN takes h and generates the output sequence T = (t_1, t_2, t_3..., tn) one element at a time [92].

The vanishing gradient problem occurs in basic RNNs when learning signals weaken during backpropagation through many layers, hindering the updating of network weights [96]. This issue becomes more pronounced in RNNs with long sequences, where gradients diminish significantly due to activation functions like sigmoid, leading to vanishing gradients for earlier layers [96]. To resolve this, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures were proposed as alternatives to basic RNNs [4, 96, 101, 102, 103]. LSTMs address the problem by introducing memory cells with gates controlling information flow, including the forget gate, input gate, and output gate, enabling them to maintain gradients over longer sequences and effectively learn long-term dependencies [101]. Similarly, GRUs address the vanishing gradient problem by introducing a gating mechanism that regulates information flow within the network, managing both the forgetting of past information and the integration of new information into the hidden state [102, 103].

In addition to RNNs, other model architectures like CNNs, commonly used in computer vision for image and video processing and analysis, have been suggested for NMT. Early work integrating CNNs was proposed by Kalchbrenner *et al*. [100], who introduced a recurrent continuous translation model featuring a CNN encoder and RNN decoder. CNNs create hierarchical representations across sequences, allowing for the capture of long-distance dependencies through shorter paths and enabling fully parallelized computations during training [61]. The CNN architecture consists of convolution and pooling layers, providing a standard framework for transforming variable-length sentences into fixed-size distributed vectors [7]. While CNNs offer advantages over RNNs, such as their ability to perform parallel computations (which enhances training speed and addresses the vanishing gradient problem), they also have limitations. Specifically, their fixed filter width limits their capacity to capture word dependencies within sentences, making them less effective in translating long sentences due to the fixed-size vector representation [91].

The convolutional sequence-to-sequence learning introduced by Kalchbrenner *et al*. [100] incorporates different components such as the Recurrent Convolutional Translation Model (RCTM), Convolutional Sequence Model (CSM), and Convolutional Gating Module (CGM). The RCTM merges features from recurrent and convolutional architectures, enabling effective capture of short- and long-range dependencies in input sequences. In contrast, the CSM, based entirely on

CNNs, processes input sequences using convolutional layers, facilitating faster training via parallelized computations compared to recurrent models. Additionally, the CGM integrates gated linear units and attention mechanisms, enhancing gradient propagation and decoding accuracy, thereby improving the model's ability to focus on relevant parts of the input sequence during decoding. The gated recursive CNN, proposed by Cho *et al*. [104], which replaced the RNN in the encoder, demonstrated strong performance in learning the grammatical structure of source sentences.

2.4.4.1 Attention Mechanism

The performance of the basic encoder-decoder NMT can vary depending on the length of the input sentence and the occurrence of unknown words. According to [104], the performance tends to degrade as the length of the source sentence increases [3, 4, 105]. Additionally, the size of the vocabulary in both the SL and TL can significantly impact translation performance. Attention mechanisms have revolutionized MT by addressing the limitation of fixed-sized vectors in traditional or basic encoder-decoder RNNs, which struggle with longer sentences, resulting in poor translations. It enables the modeling of dependencies regardless of their distance in the input or output sequences. However, except for a few cases [106], such attention mechanisms are typically used alongside a recurrent network [10].

Bahdanau *et al*. [99] introduced the Embed-Encode-Attend-Decode architecture, applying the attention mechanism to NMT for the first time [3, 4, 91, 92, 96, 107]. This architecture enables the encoder to map the source sentence into a sequence of vectors, with the decoder dynamically selecting relevant vectors during prediction. They also presented RNNsearch, where a bidirectional RNN (BiRNN) encoder captures both past and future information, thereby improving translation quality.

By utilizing BiRNNs for encoding and unidirectional RNNs for decoding, the model dynamically focuses on different parts of the input sequence. The BiRNN encoder captures contextual information from both left-to-right and right-to-left directions, represented by $\overrightarrow{h} = \{\overrightarrow{h_1}, \overrightarrow{h_2}, \overrightarrow{h_3}, \ldots, \overrightarrow{h_{T_X}}\}$ and $\overleftarrow{h} = \{\overleftarrow{h_1}, \overleftarrow{h_2}, \overleftarrow{h_3}, \ldots, \overleftarrow{h_{T_X}}\}$. These hidden states are then concatenated, allowing the model to dynamically focus on different parts of the input sequence during the decoding process. This improves the representation of the input sequence and provides a richer context vector for the decoder in generating the output sequence [61, 96, 99].

The attention mechanism assigns weights to parts of the input sequence during decoding, facilitating the alignment of source and target words for accurate translations. Unlike basic encoder-decoder architectures, attention computes a weighted sum of hidden states, preserving the relevance of each source word in context. This approach contrasts with hard alignments in SMT, offering soft alignments with varying weights, significantly enhancing translation quality and representing a substantial advancement in MT history [61, 100]. Furthermore, during decoding, the attention mechanism dynamically focuses on different parts of the input sequence, assigning weights based on relevance to the current step. This enables selective attention to crucial information, aiding the effective alignment of source and target words for accurate translations [99]. In general, the implementation process of the attention mechanism involves two steps: first, computing the attention distribution, and second, computing the context vector based on the attention distribution [96].

The attention method proposed by [99] falls under soft or deterministic attention, where it computes a weighted average of all keys to generate the context vector [96]. In soft attention, the attention module is differentiable with respect to the inputs, allowing the entire system to be trained using standard back-propagation methods [96]. Conversely, hard or stochastic attention was introduced by Xu *et al*. [108], where the context vector is derived from randomly sampled keys.

The hard attention model uses less computing resources compared to the soft attention model because it doesn't calculate attention weights for every element at each step. However, making hard decisions at each position of the input feature makes the module non-differentiable and difficult to optimize [109]. Therefore, the entire system can be trained by maximizing an approximate lower bound or using reinforcement learning [110].

Luong *et al*. [94] introduced two distinct attention mechanisms for NMT. The first mechanism, called global attention, is very similar to the soft attention proposed by Bahdanau *et al*. [99], where all source words are considered simultaneously. However, Luong *et al*.'s [94] global attention approach has a simpler structure and entails some extended calculations when determining the value of the attention matrix compared to Bahdanau *et al*.'s [99] model [3, 107]. On the other hand, their second mechanism, known as local attention, combines concepts from Xu *et al*.'s [108] hard and soft attention approaches. Local attention focuses on a subset of source words at any given time, reducing computational complexity. This design not only enhances efficiency but also

maintains differentiability throughout, distinguishing it from hard attention models. This characteristic makes local attention more suitable for implementation and training processes [94, 99, 96, 108]. Basically, global and local attention differs in terms of whether the “attention” is placed on all source positions or only a few [6].

2.4.4.2 Transformer

The primary drawback of RNN-based NMT is its inability to parallelize effectively, as it relies on past words to compute the current word. In contrast, CNNs create hierarchical representations over sequences, allowing them to capture long-distance dependencies through shorter paths and perform fully parallelized computations during training. Building on this concept, Vaswani *et al*. [10] introduced the Transformer model, which relies solely on an attention-based architecture. Unlike RNNs, the Transformer uses an attention mechanism to establish global dependencies between inputs and outputs, eliminating the need for sequence-aligned RNNs or CNNs. It employs self-attention, also known as intra-attention, which relates different positions within a sequence to compute its representation. The Transformer follows the general encoder-decoder architecture, with both encoder and decoder utilizing stacked self-attention and fully connected FFNs.

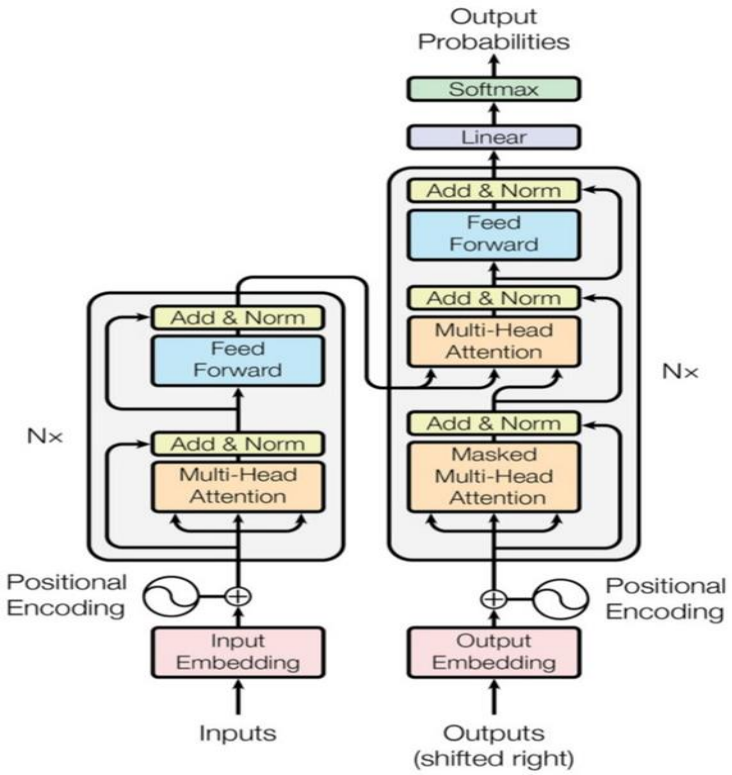

Figure 2.6: *The Transformer Model Architecture [10]*

The Transformer encoder processes the input sequence to produce continuous representations. It consists of six identical layers, each with two sub-layers: a multi-head self-attention mechanism that weighs the importance of different input positions and a fully connected FFN that processes the attention mechanism's output. The decoder generates the output sequence based on these encoded representations, also consisting of six layers. Each decoder layer has three sub-layers: two multi-head self-attention layers and a fully connected FFN.

In the decoder, the first sub-layer, masked multi-head attention, uses queries from the decoder's current state and compares them with keys representing the input sequence to determine relevance during decoding. It combines the queries, keys, and values from the encoder output with attention scores to form a context vector, which the decoder uses to generate the next output. The second sub-layer, encoder-decoder attention, focuses on attending to the encoder's output, allowing the decoder to leverage input information for accurate predictions. The FFN refines this information before generating the output.

The decoder's masked multi-head attention ensures the model's auto-regressive property by restricting predictions to depend solely on previously generated symbols. This is achieved through masking, which prevents the model from attending to future positions during decoding. The model also shifts the output embeddings by one position to maintain positional dependency, allowing it to generate sequences in a step-by-step manner.

Each sub-layer in the encoder and decoder includes residual connections and layer normalization to stabilize training and improve gradient flow. Residual connections help the model learn complex representations by bypassing sub-layers such as self-attention and FFN, while layer normalization ensures stable outputs from each sub-layer.

Vaswani *et al*. [10] introduced a specific attention mechanism called scaled dot-product attention. This mechanism maps a query and a set of key-value pairs to an output, where each component (query, key, value, and output) is represented as a vector. The output is a weighted sum of the values, with each weight determined by a compatibility function between the query and its corresponding key. Queries and keys have a dimension of d_k, while values have a dimension of d_v. The dot products of the query and keys are computed, scaled by $\sqrt{d_k}$, and passed through a softmax function to derive the attention weights.

Multi-head attention involves applying several attention mechanisms in parallel, allowing the model to focus on different representation subspaces and positions simultaneously. The Transformer uses eight parallel attention heads, with each head having a reduced dimensionality of 64. This design maintains computational efficiency while enhancing the model's ability to capture diverse relationships in the data. Multi-head attention is used in three key areas: encoder-decoder attention, where queries come from the decoder and keys and values from the encoder; self-attention within the encoder, where queries, keys, and values all come from the encoder; and self-attention within the decoder, where the decoder attends to its own previous states while maintaining masking to preserve auto-regressive properties.

Position-wise FFNs consist of two linear transformations with a ReLU activation in between, allowing the model to capture non-linear relationships in the sequence. Although the linear transformations are shared across positions, their parameters differ from layer to layer, contributing to the model's flexibility in learning.

The Transformer uses learned embeddings to convert input and output tokens into vectors of dimension d_{model}. These embeddings are crucial for representing tokens in high-dimensional space, helping the model process and manipulate representations. A shared weight matrix is used for both input/output embedding layers and the softmax linear transformation that generates the predicted probabilities for the next token. To scale the embeddings, their weights are multiplied by $\sqrt{d_{model}}$, ensuring proper dimensional alignment and facilitating effective learning.

Positional encodings are added to the embeddings at the bottom of the encoder and decoder stacks to provide information about the relative or absolute position of tokens within the sequence. For a token at position *pos* and embedding dimension *i*, the positional encoding $PE(pos, i)$ is calculated as follows:

$$PE(pos, 2i) = sin\left(\frac{pos}{1000^{2i/d}model}\right) \quad (1)$$

$$PE(pos, 2i + 1) = cos\left(\frac{pos}{1000^{2i/d}model}\right) \quad (2)$$

where pos is the position of the character in the sequence, i is the dimension index and d_{model} is the dimension of the model.

Self-attention layers in the Transformer model offer several advantages over traditional recurrent and convolutional layers. They reduce computational complexity, enhance parallelization, and improve the model's ability to learn long-range dependencies. Unlike recurrent layers, which rely on sequential processing, self-attention allows for parallelized computation, making the Transformer more efficient. Additionally, self-attention minimizes the number of operations needed to relate signals from distant positions, allowing the model to better capture long-range dependencies.

In summary, the Transformer model improves the efficiency of sequence-to-sequence tasks by replacing recurrence with attention mechanisms, allowing for parallel processing and faster training. This design has led to state-of-the-art performance in MT and other NLP tasks, such as pretraining models like BERT [111] and ERNIE [112]. However, it also has some drawbacks, such as focusing on irrelevant information and a quadratic increase in computational costs with sentence length. The attention mechanism assigns weights to all values, even those with low relevance, leading to inefficiencies. Furthermore, the quadratic growth of computational cost arises from the dot product calculation between every pair of words in the source and target sequences [92, 113, 114].

2.5 Hybrid Machine Translation

Hybrid Machine Translation (HMT) systems emerged as a fusion of rule-based and statistical paradigms, aiming to enhance translation quality [71, 115, 116, 117]. These systems integrate statistical knowledge into rule-based architectures and incorporate rules into corpus-based approaches [118]. By combining linguistic and statistical features, hybrid MT methods seek to improve both translation accuracy and speed [71, 119]. This integration occurs at various levels, including sentence, sub-sentential, and graph levels, to generate more accurate translations [120]. Experimental studies using evaluation metrics such as NIST and BLEU have demonstrated the superiority of hybrid MT systems over individual approaches [119].

Additionally, HMT combines the strengths of SMT and NMT approaches to improve translation quality [96, 105, 121]. This integration takes various forms, such as combining NMT scores with traditional SMT features in a log-linear model or incorporating SMT-style lexical translation tables into the NMT decoder [96, 105]. HMT methods also include rescoring and reranking, where a fully trained SMT system is fused with an independently trained NMT system [96]. Another

strategy involves cascading SMT and NMT systems, with one system feeding input to the other, thus leveraging the strengths of both approaches [121]. Moreover, unsupervised NMT models can benefit from SMT acting as a regularizer, enhancing neural network performance [96, 105].

The rationale for combining SMT and NMT in HMT is to strike a balance between the fluency of NMT and the accuracy of SMT [121]. While NMT produces fluent translations, it may lack accuracy in certain cases, whereas SMT systems excel in accuracy but may lack fluency. By integrating these approaches, SMT-NMT systems can balance accuracy and fluency, leading to improved translation quality [105, 121]. Additionally, SMT-NMT systems can utilize multilingual language models to embed source sentences and translations into the same vector space, further enhancing the overall translation process [122].

2.6. Multilingual Machine Translation

MT systems can be categorized as bilingual or multilingual, operating in unidirectional or bidirectional modes [123, 124]. Most systems function bidirectionally between a SL and TL, like Amharic-to-English and vice versa. However, some are unidirectional, translating solely from SL to TL, such as Amharic-to-English [121, 124]. Both systems rely on parallel corpora for training and evaluation [124]. In contrast, Multilingual Machine Translation (MMT) aims to translate among multiple languages without requiring parallel data for all language pairs [125].

Multilingual NMT seamlessly integrates multiple languages within the NMT framework, enabling translation between more than two languages and efficiently using linguistic resources to handle scenarios like multiway, low-resource, and multi-source translations [61, 9]. In multiway translation, a single NMT system accommodates various scenarios, including one-to-many, many-to-one, or many-to-many translations. Low-resource translation addresses situations with limited parallel corpora by using auxiliary languages to improve translation between low-resource language pairs. Multi-source translation involves leveraging existing multilingual content on the source side to enhance translation quality [9].

Both SMT and NMT methods face challenges in MMT due to differences in language structure and morphology [61]. Multilingual NMT models aim to overcome these challenges by leveraging large amounts of parallel data to learn translation knowledge [46]. Various techniques, such as back-translation and unsupervised translation, address the scarcity of training data for low-

resource languages [61]. These models eliminate the need for separate translation models for each language pair and task, providing a universal solution capable of translating multiple languages across various tasks, such as multiway translation or training approaches [61]. Multilingual training, especially for low-resource languages, exploits syntactic or semantic similarities between languages, enabling the transfer of processing abilities across language boundaries [126]. Different training settings, such as one-to-many and many-to-one, are explored, each with varying impacts on performance [126]. Shared parameters across languages improve performance, particularly in low-resource language settings, and methods like parameter sharing based on related language sets further enhance results without requiring extensive linguistic knowledge, making them valuable for less-studied languages [126].

2.7 Machine Translation Evaluation Metrics

Assessing machine translation (MT) systems is essential to ensure output reliability and guide system improvements [127]. Koby *et al*. [128] define a quality translation as one that accurately and fluently conveys the intended message while meeting specific requirements set by both the requester and provider. The translation's purpose, as defined by the requester, directs the process, making functional equivalence between the source and target texts crucial [127]. Key criteria for evaluating MT quality include fluency in the TL, adequacy in semantic and pragmatic equivalence, and adherence to requester specifications [127]. The goal is to achieve a translation quality comparable to human-level outputs, with recent advances aiming for human parity, which is statistically defined as no significant difference between human and machine-generated quality scores [127].

MT evaluation typically falls into two categories: glass-box and black-box evaluation. Glass-box evaluation focuses on assessing system quality through internal mechanisms, while black-box evaluation focuses solely on output without examining the internal workings [129]. In black-box evaluation, source and target sentence pairs are used to objectively assess system performance, and careful selection of evaluation sets is critical to ensure coverage of significant data features and system robustness [130, 131]. Black-box evaluation techniques include both human evaluation and automatic metrics, with semi-automated metrics sitting between these two categories [116].

2.7.1. Human Machine Translation Evaluation Metrics

Human evaluation, or manual evaluation, is the primary method for assessing MT quality, typically performed by expert evaluators. They assess translations from two main perspectives: fluency and accuracy [127, 129, 131]. Fluency evaluation focuses on how well the translation adheres to the norms of the TL, including its grammaticality and clarity, without reference to the source text [129]. Accuracy evaluation, in contrast, measures the translation's fidelity to the source text, ensuring the target text accurately conveys the original content. Bilingual evaluators assess this fidelity [129].

Human evaluation methods are further divided into directly expressed judgment (DEJ-based) and non-DEJ-based metrics. DEJ-based methods involve subjective assessments of translation quality, while non-DEJ-based methods focus on task-oriented evaluation, such as error classification or gap filling [127, 132].

The earliest human MT evaluation method was introduced by the Automatic Language Processing Advisory Committee (ALPAC) in 1966, focusing on intelligibility and fidelity [133]. Intelligibility measures how naturally a translation reads, ensuring it is as clear as a well-edited native sentence. Fidelity assesses how faithfully the translation conveys the source text's meaning, minimizing distortion or contradiction [131, 133].

In the 1990s, the Advanced Research Projects Agency (ARPA) developed a methodology that evaluates adequacy, fluency, and informativeness (also known as comprehension) [131, 133]. Evaluators rate each translation fragment on a 1-to-5 scale based on syntactic constituents and information content [133]. This method measures conveyed information, fluency, and the system's overall translation quality, including its adherence to the source text.

However, human evaluation has drawbacks, as it is time-consuming, costly, and inherently subjective. To mitigate bias, multiple experts often assess the same translations, and their judgments are statistically validated [129].

2.7.2 Automatic Machine Translation Evaluation Metrics

Automatic machine translation evaluation metrics (AMTEM) play a critical role in assessing MT outputs without human intervention, offering a cost-effective alternative to manual evaluation, especially for large-scale data tasks like collection, annotation, or reference translation generation

[127]. While automatic metrics are more efficient, they are generally considered less accurate than human evaluations [134]. Nevertheless, their ability to facilitate system comparison and tuning during development has made them widely used [133].

AMTEMs were developed to address the challenges of human evaluation, such as high costs, subjectivity, and lack of reproducibility, by providing objective measures that can be automated for system optimization [81]. These metrics compare MT outputs against reference translations, and their correlation with human judgments often determines their effectiveness [130].

In their survey, Lee *et al*. [134] categorize AMTEMs into several groups. Lexical-based metrics measure the similarity between the generated text and reference translations using mathematical or heuristic methods, without relying on ML algorithms. These metrics are further divided into:

- Word-based metrics, which include N-gram measures such as BLEU, METEOR, NIST, and GTM.
- Edit distance measures, including Translation Edit Rate (TER) and Word Error Rate (WER).
- Character-based metrics, like chrF, which evaluate similarity at the character level.

Embedding-based metrics use ML or deep learning (DL) algorithms, representing texts as dense vectors to measure similarity by considering the words' deeper meanings. These metrics focus on capturing both the lexical and contextual nuances of translation. Metrics can also be classified into matching, regression, ranking, and generation, depending on the training task involved.

Chauhan and Daniel [135] provide a comprehensive survey of fully automated AMTEMs, categorizing them into lexical, character-based, syntax-based, semantics-based, and combined syntactic and semantic-based evaluation metrics. Lexical metrics assess word similarity, character-based metrics focus on character n-grams, and syntax-based metrics evaluate sentence structures using part-of-speech tags or phrase types. Semantics-based metrics incorporate features like named entities and paraphrasing.

WER, introduced by Su *et al*. [136], is a metric that considers word order and editing operations such as insertion, deletion, and substitution, determining the minimum number of steps required to align two sequences [133]. While simple and reproducible, WER struggles with issues like

penalizing valid translations that differ in word order and failing to capture semantic or syntactic nuances, especially in morphologically complex languages [124].

Extensions of WER, such as Position-Independent Word Error Rate (PER) and TER, address some of these shortcomings by considering the reordering of words and phrases. For example, TER incorporates block movements or shifts within the output sentence, which allows for more flexible editing processes [135].

The BLEU metric, developed by Papineni *et al*. [137], is one of the most widely used language-independent evaluation tools for MT. It measures the overlap between N-grams in the MT output and reference translations, with a penalty for brevity to avoid favoring short translations [129, 135]. However, BLEU has limitations, such as its strict focus on N-gram precision without considering recall, and its challenges in handling morphologically rich languages or syntactic variation [134].

Metric for Evaluation of Translation with Explicit Ordering (METEOR), proposed by Banerjee and Lavie [138], is another widely used automatic metric. METEOR improves upon BLEU by considering recall alongside precision and incorporating stemming, synonym matching, and reordering flexibility to better capture linguistic variation. This makes METEOR more sensitive to meaning preservation and word choice, often resulting in higher correlation with human judgments than BLEU [138, 139]. Despite its advantages, METEOR can be computationally intensive and struggles with handling highly inflected languages or evaluating diverse sentence structures [138].

National Institute of Standards and Technology (NIST), a variant of BLEU, improves upon it by assigning weights to less common, more informative N-grams, providing a more nuanced approach to sentence similarity [138]. However, NIST also struggles with the evaluation of languages with complex morphology and fails to fully account for syntactic elements [135].

Global Translation Model (GTM) introduced by Turian *et al*. [139], evaluates the quality of text by calculating precision, recall, and the harmonic mean (F-score) based on matching word sequences between the hypothesis and reference. GTM prioritizes longer matching sequences and correlates more strongly with human judgment than BLEU and NIST [130].

2.8 Regularization Techniques in Neural Networks

Regularization in neural networks refers to a set of techniques designed to improve model generalization by preventing overfitting. Overfitting occurs when a model performs exceedingly well on training data but poorly on unseen data due to its excessive complexity, capturing noise rather than underlying patterns [142]. This issue is common in neural networks because of their high capacity to model complex relationships. As the model becomes more complex, the risk of overfitting increases as it may memorize the training data rather than generalize from it [143, 144].

Historically, regularization methods such as L1 and L2 regularization were first introduced in linear models and statistical learning [145]. These techniques have since been extended to neural networks to manage complexity and enhance generalization. The evolution of regularization methods in DL includes Dropout and Batch Normalization, which specifically address the challenges posed by deep architectures [28, 146].

Regularization plays a crucial role in DNNs for several key reasons. Firstly, it helps prevent overfitting, which is a common issue in large neural network models with extensive numbers of parameters. By introducing constraints, regularization limits the model's capacity, reducing the likelihood of memorizing training data and enhancing performance on unseen data [147, 148]. Secondly, regularization improves the model's generalization ability by penalizing overly complex models, ensuring that the network captures essential patterns rather than fitting to noise, which promotes better generalization [10, 149]. Furthermore, as DL models increase in complexity with numerous layers and parameters, the risk of overfitting grows. Regularization helps manage this complexity by controlling the growth of model parameters through the introduction of penalties [101, 150]. Finally, in situations where data is scarce, such as in low-resource language tasks, regularization is particularly beneficial. It reduces the risk of overfitting in these small datasets, enabling the model to handle limited data more effectively [151, 152].

Various regularization techniques employ distinct strategies to mitigate overfitting in neural networks. L1 regularization adds a penalty equal to the absolute value of the magnitude of the coefficients, which often results in sparse models by driving some weights to zero, effectively performing feature selection [153, 154, 155]. In contrast, L2 regularization introduces a penalty proportional to the square of the magnitude of the coefficients. This method encourages smaller, non-zero weights, leading to a smoother model that avoids extreme parameter values [156, 157].

Elastic Net combines the benefits of both L1 and L2 regularization, balancing sparsity from L1 with smoothness from L2, making it particularly useful when correlations exist between input features [158, 159, 160]. Another widely used technique is dropout, proposed by [28], which involves randomly setting a fraction of neurons to zero during each training iteration. This prevents neurons from becoming overly reliant on specific patterns, thus reducing overfitting. The dropout rate, which controls the fraction of neurons set to zero, is especially effective in deep networks [161, 162]. Lastly, early stopping monitors the model's performance on a validation set and halts training when the validation error begins to rise, indicating potential overfitting. While not a formal regularization technique, early stopping helps prevent over-training and enhances the model's generalization capabilities [163, 164].

The Transformer model architecture [10], known for its high capacity and flexibility, is prone to overfitting due to its complexity and the large number of parameters it involves. Regularization plays a crucial role in addressing this issue in Transformer-based models for several reasons. Firstly, the high dimensionality of the self-attention (SA) mechanism results in a large number of parameters, as it computes attention weights between every pair of input tokens, whether they are words, subwords, or characters, depending on the type of input embeddings used. Regularization techniques, such as dropout, help to mitigate overfitting by randomly dropping some of these attention connections during training, thereby improving the model's generalization capabilities [165, 166]. Secondly, the large parameter space in the Transformer, which often involves millions of parameters, particularly in the multi-head attention mechanism, requires regularization to prevent over-reliance on specific parameters. This helps the model to generalize better to unseen data by encouraging it to distribute learning more evenly across the parameters [167, 168]. Finally, for low-resource languages like Amharic, where training data is limited, the risk of overfitting is particularly high. In these scenarios, regularization techniques are critical to improving the model’s generalization and preventing it from overfitting to the sparse training data [169, 170].

2.9 Available Parallel Corpora for Amharic-to-English Machine Translation

In the rapidly evolving field of NMT, the availability and quality of parallel corpora are crucial for developing effective translation models. The limited availability of high-quality datasets intensifies the challenge of developing NMT for language pairs such as Amharic-to-English. Being

a low-resource language, Amharic faces additional challenges in accessing sufficient linguistic data.

There are a few options to find a parallel corpus for Amharic-English. One source is religious documents like the Bible, which are available on platforms such as YouVersion[2], Jehovah's Witnesses (JW)[3], the Ethiopic Bible[4], and the Ebible[5], all of which offer parallel Amharic and English versions of religious documents.

Another option is legal documentation. The Ethiopian Legal Information Portal[6], Chilot blog[7], and the FDRE Ministry of Justice's Negarit Gazeta[8] all provide parallel Amharic and English versions of proclamations, legislation, and directives. Additionally, there is a GitHub repository[9] that offers Amharic-English parallel corpora, including the Bible, historical texts, and legal documents.

[2] https://www.bible.com/

[3] https://www.jw.org/

[4] https://www.ethiopicbible.com/

[5] https://ebible.com/

[6] https://lawethiopia.com/

[7] https://chilot.blog/

[8] https://www.eag.gov.et/en-us/Laws/Law/Negarit-Gazeta

[9] https://github.com/MarsPanther/Amharic-English-Machine-Translation-Corpus

Chapter 3

Related work

3.1 Introduction

This chapter presents a comprehensive review of research efforts in Transformer-based NMT, focusing specifically on low-resource languages such as Amharic, which is both low-resource and morphologically complex. Although NMT has made substantial progress with the advent of Transformer architectures, accurately translating morphologically rich and underrepresented languages remains a considerable challenge. Factors such as limited availability of parallel corpora, linguistic diversity, and the inherent morphological complexity of these languages contribute to the difficulty in achieving high-quality translations.

Previous studies have explored several strategies to address these challenges, including character-level embeddings and subword modeling techniques, which are designed to capture the rich morphological structure of languages and mitigate issues like OOV words. These approaches improve translation accuracy by handling word variations more effectively, especially when dealing with languages with complex word formations and sparse data resources. Additionally, regularization techniques such as dropout, L1, L2, and Elastic Net, which are widely used across DL tasks to prevent overfitting, have been adapted in NMT models to enhance performance on small datasets.

This review highlights how these character-level and subword techniques, along with regularization strategies, have been employed to address the specific challenges posed by low-resource and morphologically complex languages like Amharic. It also discusses the limitations of current approaches, setting the stage for the proposed model to address the existing gaps.

3.2 Character-Level Transformer Model for Neural Machine Translation

Character-level models have emerged as an effective solution to address the challenges posed by low-resource languages. For instance, Banar *et al.* [171] introduced the CharTransformer, a model that incorporates character embeddings into the Transformer architecture. This model successfully managed linguistic phenomena such as transliteration and proper noun translation, outperforming existing character-level models like CharRNN across various language pairs. Notably, the

CharTransformer achieved a competitive BLEU score of 28.63 in the German-English translation task, compared to 29.72 for subword-level Transformers, while also being 34% faster. This efficiency highlights the potential of character-level models in low-resource settings. However, the model faced challenges when dealing with languages with highly complex morphological structures, underscoring the need for further exploration in this area.

3.3 Subword Modeling for Morphologically Rich Languages

Morphologically rich languages pose additional challenges in MT. Baniata *et al*. [172] addressed these challenges in their Transformer-based NMT model, which translated Arabic vernaculars into Modern Standard Arabic (MSA). By using a Word-Piece model to create subword units, their approach effectively managed OOV words, resulting in a BLEU score of 63.71 for Levantine-to-MSA translation. While this method proved strong in handling complex morphologies, its application to other low-resource languages has yet to be fully generalized. This study emphasizes the importance of subword models in improving NMT for languages with complex morphological features.

In the context of Amharic-to-English translation, Andargachew Gezmu *et al*. [24] worked on improving translation performance by applying subword-based models and transliteration techniques. They used a transformer-based NMT architecture, testing three model sizes: transformer-tiny, transformer-small, and transformer-large, each optimized for low-data settings. The training data was the Amharic-English parallel corpus, consisting of 140,000 sentence pairs, which they used to evaluate their models. The results showed that the transformer-tiny model achieved BLEU scores of 24.0 for Amharic-to-English and 17.8 for English-to-Amharic, while the transformer-small model scored 25.4 and 18.9, respectively. The transformer-large model performed the best, with BLEU scores of 32.2 for Amharic-to-English and 26.7 for English-to-Amharic translations. The study also compared various NMT models and found that subword-based models consistently outperformed word-based models by about 3 to 4 BLEU points, with the best subword-based models exceeding the baseline by 6 to 7 BLEU scores. However, one limitation of the paper was the lack of a fully implemented character embedding approach, which could potentially improve translation quality, especially given the complex morphology of the Amharic language.

Expanding on this, Tadesse Destaw Belay et al. [25] combined the multilingual pre-trained model Facebook M2M100 with homophone normalization to improve English-Amharic translation performance. They used a dataset of 8,603 parallel sentences. Their method significantly enhanced translation quality, achieving a BLEU score of 34.12 for the standard model, and an even higher score of 37.79 after applying homophone normalization for Amharic-to-English translation, setting a new state-of-the-art result. This study demonstrated the value of using pre-trained models alongside language-specific techniques, although relying on pre-trained models may limit generalization in very low-resource environments. The key lessons from their work are that subword modeling, along with linguistic methods like transliteration and homophone normalization, is essential for improving Amharic-to-English translation. However, they noted some limitations, particularly the relatively small dataset, which may affect the generalizability of the findings. Future research with larger, more diverse datasets and improved embedding methods is suggested to further advance Amharic-to-English MT.

Andargachew Gezmu and Nürnberger [26] conducted an in-depth exploration of NMT for low-resource languages, particularly Amharic-English and Turkish-English translation pairs. Their work utilized the Transformer-based architectures, proposing three variations: TransformerShallow1, TransformerShallow2, and TransformerDeep, which were trained on benchmark datasets, including the Amharic-English parallel corpus from the Data and Engineering group at the University of Magdeburg[10] and Turkish-English datasets from the 18th Workshop on Machine Translation (WMT18)[11]. The TransformerDeep model significantly outperformed the baselines, achieving BLEU scores of 32.2 and 26.7 for Amharic-to-English and English-to-Amharic translations, respectively, while the best baseline (BaselineMERT) scored 25.8 and 20.2. Similar improvements were observed in Turkish-English translations, where TransformerDeep achieved 16.3 for Turkish-to-English and 12.6 for English-to-Turkish, compared to BaselineMERT's 10.7 and 7.8. They also highlighted the benefits of synthetic data generated from the Contemporary Amharic Corpus[12] (CACO) corpus, boosting BLEU scores further to 27.8. Additionally, the authors introduced a morpheme-based model using MorphoSeg to address the

[10] http://dx.doi.org/10.24352/ub.ovgu-2018-145

[11] http://data.statmt.org/wmt18/translation-task/preprocessed/

[12] http://dx.doi.org/10.24352/ub.ovgu-2018-144

high OOV rate in Amharic, which demonstrated promising improvements with BLEU scores of 25.3 for Amharic-to-English and 30.1 for English-to-Amharic. However, they acknowledged challenges such as extensive hyperparameter tuning in low-data conditions and morphological complexity, suggesting room for further refinement.

In their subsequent work, Andargachew Gezmu and Nürnberger [27] focus on addressing the challenges of NMT for low-resource fusion languages, particularly the Amharic-English translation pair. They propose a word segmentation method called MorphoSeg, which is designed to restore morpheme boundaries that have been altered by phonological and orthographic processes in fusion languages. This method relies on a language's morphological analyzer or treebank, allowing more accurate segmentation of words into morphemes. The study compared MorphoSeg with conventional subword segmentation methods such as Byte Pair Encoding (BPE), Word-Piece, and Sentence Piece with Unigram Language Modeling (SPULM) using a baseline Transformer-based architecture for their NMT system. The evaluation was conducted using an Amharic-English parallel corpus sourced from CACO, containing approximately 22 million tokens.

Models were trained with varying vocabulary sizes, specifically 1K (1,000 unique tokens), 4K (4,000 unique tokens), 8K (8,000 unique tokens), 16K (16,000 unique tokens), and 32K (32,000 unique tokens), referring to the number of unique subword units or tokens in the model's vocabulary. The morpheme-based models, especially those using MorphoSeg, significantly outperformed conventional subword models, with BLEU scores showing improvements such as 32.8 for the Word-Piece-4K model in Amharic-to-English translation compared to 32.2 for Word-Piece-1K and 33.0 for Word-Piece-8K. In the English-to-Amharic direction, the Word-Piece-4K model achieved a BLEU score of 26.1, while the Word-Piece-16K model scored 26.7. Gezmu and Nürnberger acknowledged limitations, including the lack of part-of-speech disambiguation in morphological analysis, which could impact segmentation accuracy. Additionally, they faced resource constraints that limited them to a single run per configuration, suggesting that multiple replications could lead to more robust results. While they noted that character embedding is an extreme approach due to the longer sequences it generates, which creates challenges in both modeling and computation, they also emphasized that character embeddings remain effective

despite these challenges. However, the reliance on a morphological analyzer may limit the method's applicability to languages that lack such resources.

3.4 Regularization Techniques in Low-Resource NMT and other DL Domains

Regularization techniques play a vital role in improving the performance of NMT models, especially in low-resource settings where overfitting is a common challenge. Various methods have been developed to address these issues, and recent studies highlight their importance in enhancing model robustness and generalizability.

Variš and Bojar [174] introduced Elastic Weight Consolidation (EWC) to prevent overfitting during fine-tuning with limited parallel data. Their approach achieved a BLEU score of 27.5 for Basque-to-English translation. The strength of this technique lies in its ability to mitigate overfitting effectively. However, its complexity of implementation may be a limitation. The key takeaway from this study is that robust regularization techniques like EWC are critical in low-resource settings.

Gao *et al*. [175] proposed Cross-lingual Consistency Regularization (CrossConST), a technique aimed at enhancing multilingual NMT by enforcing consistency in semantic representations across languages. This method significantly improved zero-shot translation performance, reaching a BLEU score of 28.5 on the IWSLT17[13] benchmark. A notable strength of CrossConST is its capacity to boost multilingual translation without the need for additional training data. However, its application may be limited to multilingual systems. This study emphasizes the importance of enforcing semantic consistency to improve generalization in NMT models, particularly in zero-shot scenarios.

Fan *et al*. [29] introduced LayerDrop, a structured dropout method for Transformer architectures that dynamically prunes layers during inference. Tested on the WMT English-German translation benchmark, LayerDrop achieved a BLEU score of 30.2, surpassing the baseline Transformer score of 29.3. The method's adaptability and efficiency are key strengths, though the complexity involved in hyperparameter tuning may be challenging. The lesson learned from this work is the effectiveness of structured pruning techniques in enhancing model efficiency and resilience.

[13] https://paperswithcode.com/dataset/iwslt-2017

Zhou *et al*. [30] developed DropHead, a regularization method that selectively drops entire attention heads in Transformer models. This technique was evaluated on multiple NLP tasks, including WMT14 English-to-German dataset[14], where it achieved a BLEU score of 29.9, and on IWSLT14 German-to-English[15], reaching 34.4. A key advantage of DropHead is its ability to reduce overfitting while improving model interpretability, although it introduces additional computational overhead during training. The main takeaway is that attention-head-level regularization significantly boosts performance and generalization.

Wu *et al*. [31] proposed UniDrop, a regularization method aimed at preventing overfitting in Transformer models by combining three key dropout techniques: feature dropout (FD), structure dropout (SD), and data dropout (DD), without significantly increasing computational costs. These techniques target different levels of the model. FD removes specific features from input representations to prevent overfitting at various stages, including four distinct types of feature dropouts. SD eliminates entire components, such as layers, to regularize the learning process, while DD omits words or tokens during training to enhance model robustness. The method was evaluated on the IWSLT14 dataset, which includes language pairs such as English and German, Romanian, Dutch, and Portuguese-Brazil, with approximately 170,000 to 190,000 translation pairs. BPE was applied to handle the vocabulary.

The results showed that UniDrop enabled the Transformer model to achieve state-of-the-art performance, particularly in the IWSLT14 Dutch -to-English translation task, with a BLEU score of 36.88. An ablation study was performed, where each of the proposed dropout techniques was removed in isolation to evaluate its specific contribution to the overall performance. The study revealed that removing FD resulted in a BLEU score of 35.68, SD resulted in a score of 36.70, and DD led to a score of 36.38. These results demonstrate that the removal of any proposed dropout technique resulted in lower BLEU scores, justifying the necessity of each component in the UniDrop method. However, the study acknowledged certain limitations, such as the possibility that the proposed dropout techniques might not generalize well to all datasets or tasks.

[14] https://paperswithcode.com/sota/machine-translation-on-wmt2014-english-german
[15] https://paperswithcode.com/sota/machine-translation-on-wmt2014-german-english

Moreover, insights from other DL domains can be applied to NMT regularization. For example, Yang *et al*. [176] explored the application of L1 and L2 regularization on high-dimensional datasets, while Farhadi *et al*. [17] integrated multiple regularization techniques to enhance CNN performance. These principles of regularization, particularly the L1 and L2 methods, can be adapted for NMT models to improve robustness and generalizability in low-resource environments.

In their study, Farhadi *et al*. [17] address the overfitting problem in DL models, particularly deep CNNs, by integrating L1, L2, Elastic Net regularization, and dropout methods. The CNN architecture they designed consists of five convolutional layers and three fully connected layers. Using a dataset of gold prices per ounce, the authors transformed the data into a regression format and applied various regularization techniques. Among the different configurations, the Elastic Net regularization yielded the best performance, with a mean squared error (MSE) of 0.0129773, root mean squared error (RMSE) of 0.1139183, and mean absolute error (MAE) of 0.0748202. These findings highlight the effectiveness of combining regularization techniques in improving the accuracy and robustness of DL models.

On the other hand, Yang *et al*. [176] present significant findings regarding the impact of L1 and L2 regularization on the performance of DNNs in predicting corporate credit risk. They highlight a major limitation of traditional DNNs, where L1 regularization fails to function effectively when batch normalization layers are used. To overcome this, they propose a high-dimensional deep neural network (HDNN) that combines L1 regularization for feature selection and L2 regularization to mitigate overfitting. Their results demonstrate that the HDNN outperforms standard DNNs and other ML models, such as Support Vector Machines and Logistic Regression. In testing, the HDNN achieved an AUC of 80.12%, compared to Support Vector Machines with an AUC of 73.8% and Logistic Regression with an AUC of 71.7%. Even with varying proportions of training sets at 60%, 70%, and 80%, the HDNN consistently outperformed these models, highlighting its robustness and predictive accuracy. This approach effectively balances feature selection and generalization, making it particularly useful for handling high-dimensional data. These findings underscore the critical role of regularization in improving the predictive accuracy of DNNs in various domains, including NMT, where text datasets are often high-dimensional due to large vocabularies, variable sequence lengths, and complex contextual dependencies.

Regularization techniques can therefore help NMT models handle this complexity, leading to more robust and generalizable performance.

3.5 Summary

The reviewed works highlight significant advancements in NMT, particularly for low-resource languages like Amharic. Subword modeling and character-level representations have shown promise in handling OOV words and morphological complexity, while advanced regularization techniques such as Elastic Weight Consolidation and Cross-lingual Consistency Regularization have been instrumental in preventing overfitting in small datasets. However, challenges remain, particularly in fully addressing the complexities of highly morphologically rich languages like Amharic and the generalization of techniques developed for more well-resourced languages.

This thesis aims to bridge these gaps by developing an optimized Transformer-based model for Amharic-to-English NMT. Specifically, it explores the use of character embeddings to better handle the Amharic complex morphology and integrate regularization techniques such as L1, L2, and Elastic Net to enhance the model's robustness in low-resource settings. By doing so, this research contributes to the development of more effective NMT systems for underrepresented languages, providing a potential solution to the persistent challenges identified in the related work.

Chapter 4

Design of the Proposed Model

4.1 Introduction

This chapter focuses on the design of a Transformer-based model for Amharic to English NMT. The model is built on the foundation of the Transformer architecture introduced by Vaswani *et al*. [10], with adjustments made to handle the translation from Amharic input sequences to accurate English outputs. To better capture the complex linguistic structures of Amharic, character embeddings are used, allowing the model to represent detailed linguistic patterns more effectively. The model also includes an advanced regularization technique in the Point-wise FFN to help improve its generalization and prevent overfitting. This is important, especially when dealing with the challenges of translating between two languages with significant structural differences. The use of SA mechanisms and other DL techniques helps the model better understand and translate long-range dependencies and the subtle meanings that can vary between Amharic and English. In this chapter, the role of each component within the architecture is explained, along with the reasons for their inclusion, providing a clear understanding of how the model works to achieve accurate and contextually appropriate translations.

4.2 Proposed Model Architecture

The proposed Amharic to English Transformer-based NMT model architecture comprises essential elements such as character vectorization, positional encoding, encoder stack, decoder stack, linear vocabulary projection, softmax, and the output sequence. Character vectorization begins with tokenization, where the Amharic input sequence is divided into characters, enabling the model to capture the language's morphological complexity. These tokenized characters are then converted into numerical representations, allowing the model to process fine-grained linguistic details at the character level. After vectorization, positional encoding is applied to embed the positional information of each character in the sequence, ensuring that the model recognizes the order of the characters. The encoder stack processes these positionally encoded vectors through layers of multi-head self-attention and point-wise FFNs, extracting relevant contextual information from the Amharic input. The decoder stack uses these encoded representations, along with previously

generated translation outputs, to predict the English sequence. This is followed by linear vocabulary projection, transforming the decoder's outputs into a space corresponding to the target or English vocabulary. Finally, the softmax layer generates a probability distribution over the English vocabulary, producing the output sequence that completes the translation from Amharic to English.

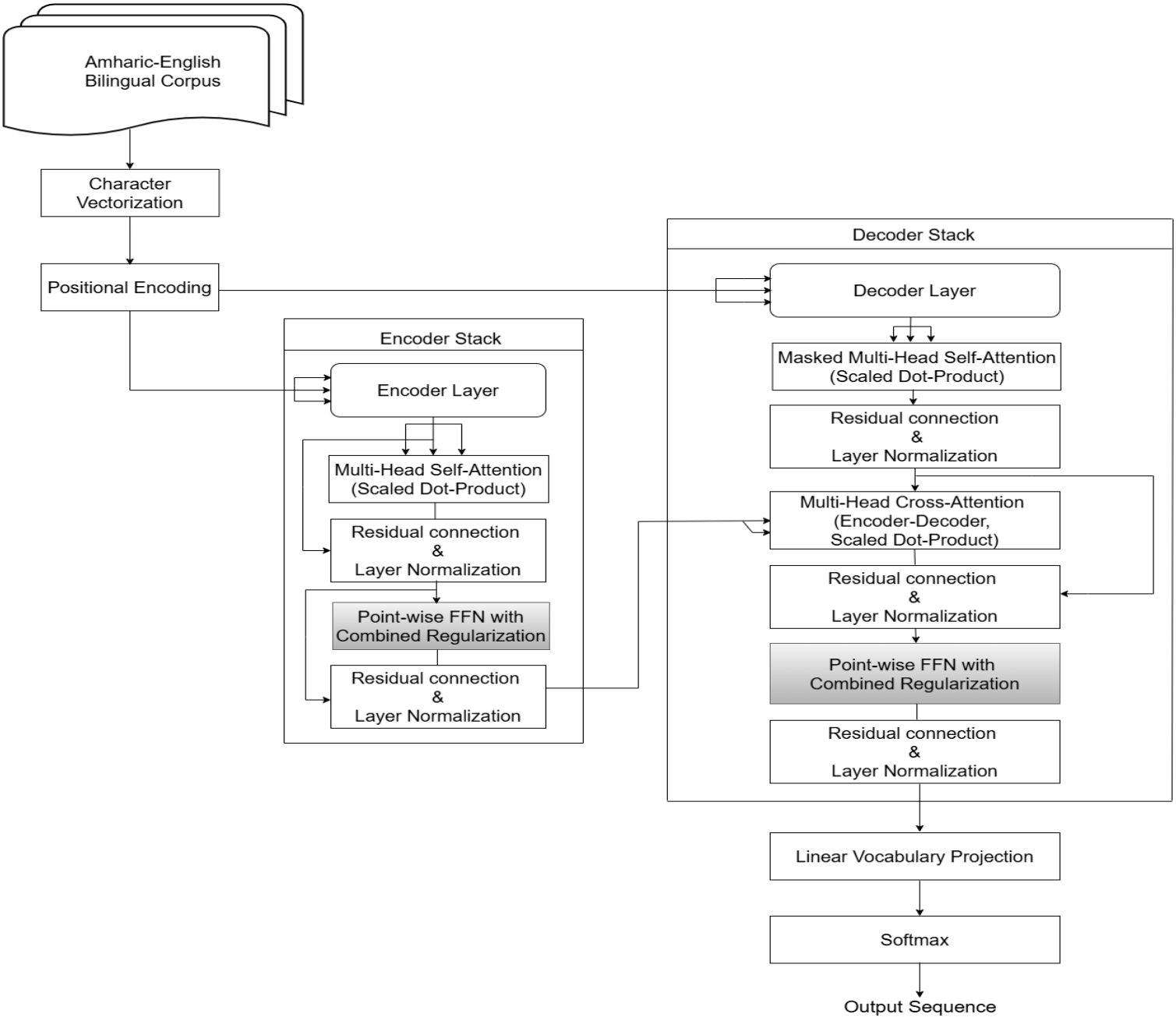

Figure 4.1: *Architecture of the Proposed Transformer-Based Amharic-to-English Neural Machine Translation Model*

4.2.1 Character Vectorization

In our approach, we propose using character-level embeddings, which utilize character-level tokenization instead of the standard word embeddings. This method captures more fine-grained linguistic nuances and handles OOV words more effectively. This strategy is particularly beneficial for morphologically rich languages like Amharic, where words vary greatly. By focusing on characters, our model more effectively captures the internal structure of words, improving translation accuracy for both common and rare words.

Character embedding transforms each character into a dense vector of fixed dimensions. This process is mathematically represented by transforming each character into a corresponding embedding vector through multiplication with an embedding matrix. For instance, the Amharic character 'υ' is converted into its embedding vector, where the one-hot vector representing 'υ' is multiplied by the embedding matrix.

The preprocessing pipeline begins by reading the source language (Amharic) and target language (English) sentences from text files. The data is then cleaned and normalized to ensure consistency and address potential Unicode issues. In English, contractions like "don't" are expanded to "do not" to simplify the text and reduce variability, enhancing the model's learning efficiency. In Amharic, Unicode ranges are employed to retain only relevant characters, including Amharic script characters, punctuation marks, and numbers, while removing irrelevant ones.

For instance, for the following Amharic-English paired sentences "**ከሥነ ሥርዓት ነጻ ስለመሆን**" and "Exemption from Procedures", after preprocessing, the Amharic sentence becomes a sequence of Amharic characters: ['**ከ**', '**ሥ**', '**ነ**', ' ', '**ሥ**', '**ር**', '**ዓ**', '**ት**', ' ', '**ነ**', '**ጻ**', ' ', '**ስ**', '**ለ**', '**መ**', '**ሆ**', '**ን**']. Similarly, the English sentence is tokenized into characters: ['E', 'x', 'e', 'm', 'p', 't', 'i', 'o', 'n', ' ', 'f', 'r', 'o', 'm', ' ', 'P', 'r', 'o', 'c', 'e', 'd', 'u', 'r', 'e', 's'].

Each character from the Amharic and English sequences is then converted into a dense vector using the embedding matrix. For example, let us denote the character embeddings for Amharic characters as C_1, C_2, …., C_n where *n* is the index of the last character. For example, for the first character '**ከ**', its embedding vector can be represented as:

$$C_i = E * I_i \qquad (3)$$

where $i = 1 \ldots \ldots \ldots n$. This process is repeated for each character in both the Amharic and English sentences, producing sequences of embedding vectors that represent the entire sentence.

For the given sentence pair, the first character '**ከ**' in Amharic would be mapped to its corresponding embedding vector, followed by vectors for '**ሥ**', '**ነ**', and so on. In English, the character 'E' would be mapped to its embedding vector, followed by vectors for 'x', 'e', etc. This character-level tokenization method is particularly advantageous for Amharic, where the rich morphology leads to many word variations. By focusing on characters, the model captures the internal structure of words more effectively, which improves translation accuracy across both common and rare words.

After tokenization, the data is filtered to remove zero-length sentences and tagged with special tokens [start] and [end] to mark sentence boundaries. The data is then split into training, validation, and test sets in a stratified manner to ensure representativeness across these sets. To convert the text into sequences of integers, we utilize the TextVectorization layer from TensorFlow, which tokenizes the text at the character level and maps each character to a unique integer, thereby creating a vocabulary of characters. This vocabulary is built by adapting the TextVectorization layer to the training data, ensuring that the model can recognize and process all characters in the training set.

For instance, consider the previously tokenized Amharic sentence "**ከሥነ ሥርዓት ነጻ ስለመሆን**" and its English counterpart "Exemption from Procedures". In the text vectorization process, each character from these sentences is mapped to a unique integer based on the vocabulary created by the TextVectorization layer, which can be represented as:

$$V = Vectorize(C) \qquad (4)$$

where V is the vectorized representation of the text and C is the sequence of character embeddings.

For example, in the Amharic sentence, the character '**ከ**' might be mapped to an integer such as 5, while '**ሥ**' might be mapped to 12, and so on. Similarly, in the English sentence, the character 'E' might be mapped to an integer such as 3, and 'x' to 15. This conversion transforms the character sequences into integer sequences, which can be processed by the model.

The mapping for the Amharic phrase "ከሥነ ሥርዓት ነጻ ስለመሆን" might look something like this: [5, 12, 7, 22, 12, 9, 18, 11, 1, 7, 21, 1, 12, 10, 14, 8, 6] and for the English sentence "Exemption from Procedures": [3, 15, 9, 13, 16, 20, 11, 14, 7, 1, 6, 17, 14, 13, 1, 18, 17, 14, 19, 9, 16, 21, 5].

This vectorized representation is crucial as it converts the character sequences into a numerical format that the neural network can process, thereby allowing the model to learn from and generate text based on these sequences.

ALGORITHM 4.1: CHARACTER VECTORIZATION PROCESS

Input: Raw Amharic (source) text and English (target) text
Output: Sequence of character embedding vectors for both Amharic and English

Start
Preprocess the input text
  Clean and normalize both Amharic and English texts
  Remove irrelevant characters, retaining only Amharic and English scripts, punctuation, and numbers
Tokenize the text into individual characters
  Tokenize the Amharic text into individual Amharic characters
  Tokenize the English text into individual English characters.
For each character in both tokenized sequence:
  Convert each Amharic character to its one-hot vector representation
  Convert each English character to its one-hot vector representation
Multiply the one-hot vectors by the corresponding embedding matrix to obtain the character embeddings for both Amharic and English
Return: The sequence of character embedding vectors for both Amharic and English
End

4.2.2 Positional Encoding

To enable the proposed model to capture the order of the sequence, positional encoding is added to the input embeddings. This encoding allows the model to distinguish between different positions in the sequence, which is crucial for handling sequential data, such as characters.

Positional encodings are derived using sinusoidal functions, ensuring that each position in the sequence has a unique encoding, while nearby positions have similar encodings. This method helps the model understand the relative positioning of tokens in the sequence, which is vital for tasks like translation.

Considering the previously vectorized Amharic phrase "**ከሥነ ሥርዓት ነጻ ስለመሆን**" and its English counterpart "Exemption from Procedures", let's say we have an embedding dimension d_{model} = 512. For the first character '**ከ**' at position 1, the positional encoding for the first dimension (i.e., i = 0) would be calculated using (1) as:

$$PE(1,0) = \sin\left(\frac{1}{1000^{\frac{0}{512}}}\right) \quad (5)$$

for the next dimension (i.e., i = 1), the positional encoding would be calculated using (2) as:

$$PE(1,0) = \cos\left(\frac{1}{1000^{\frac{2}{512}}}\right) \quad (6)$$

This process is repeated for each character in both the Amharic and English sequences, generating positional encodings that are added to the corresponding character embeddings. By combining these positional encodings with the character embeddings, the model gains information about the order of characters in the sequence, which enhances its ability to understand and generate text in the correct order.

ALGORITHM 4.2: POSITIONAL ENCODING

Input: Sequence of character embeddings, maximum sequence length (max_len), embedding dimension (d_model)

Output: Sequence of character embeddings with positional encoding (PE) added

```
Start
For each position pos in the sequence (0 to max_len-1)
    For each dimension i in the embedding (0 to d_model-1)
        If i is even:
            PE(pos, i) = sin(pos / (10000^(2i/d_model)))
        Else
            PE(pos, i) = cos(pos / (10000^(2i/d_model)))
Add the calculated positional encodings to the input embeddings
Return: The combined embeddings and positional encodings
End
```

4.2.3 Encoder Stack

Our proposed model architecture comprises an Encoder stack and a Decoder stack, designed specifically for Amharic-to-English translation. The foundation of our model is the Scaled Dot-Product Attention mechanism, which allows the model to focus on the most relevant parts of the input sequence by calculating attention scores.

The Encoder stack consists of multiple layers, with each layer containing Multi-Head Attention and FFNs. The role of the Encoder is to process the source language (Amharic) input sequence and generate a contextually rich representation of it. To ensure the model captures the sequential nature of the input, positional encoding is added to the input embeddings.

The Multi-Head Self-Attention mechanism within the Encoder enables each token in the input sequence (Amharic characters) to attend to all other tokens in the sequence. This is computed using scaled dot-product attention, where the query, key, and value vectors are derived from the same input sequence. This mechanism allows the Encoder to capture complex relationships between tokens, improving the model's ability to represent the input sequence in a meaningful way for translation.

For the input sequence "**ከሥነ ሥርዓት ነጻ ስለመሆን**" the characters '**ከ**', '**ሥ**', '**ነ**', ' ', '**ሥ**', '**ር**', '**ዓ**', '**ት**', ' ', '**ነ**', '**ጻ**', ' ', '**ስ**', '**ለ**', '**መ**', '**ሆ**', and '**ን**' are passed into the Encoder as individual embeddings. Let the sequence of characters be represented as: $[C_1, C_2, C_3. \ . \ ., C_n]$.

Each character C_i is first embedded into a vector X_i, and then these embeddings are transformed into query (Q), key (K), and value (V) vectors through learned weight matrices W_Q, W_K, and W_V, respectively. The transformations for each character in the sequence are given by:

$$Q_i = X_i W_Q,\ K_i = X_i W_k,\ V_i = X_i W_V \qquad \forall i \in \{i,\ i+1,\ \ldots\ i+n\} \qquad (7)$$

For instance, the transformations for the first character C_1= '**ከ**' are:

$$Q_1 = X_1 W_Q,\ K_1 = X_1 W_k,\ V_1 = X_1 W_V \qquad (8)$$

where X_1 represents the embedding of the character '**ከ**', and W_Q, W_k, and W_V are the weight matrices for query, key, and value transformations, respectively.

This transformation applies to all characters in the sequence, with X_i representing the embedding of each character at position *i*, and the weight matrices W_Q, W_K, and W_V used to produce the query, key, and value vectors for each respective character.

The attention scores are then computed for each pair of characters using the following formula:

$$attention\_scores\_i = \frac{Q_{c_i} K_{C_{i+1}}^T}{\sqrt{d_k}} \qquad (9)$$

where d_K is the dimension of the key vectors. This scaling factor of $\sqrt{d_K}$ ensures stable gradients during training. Q_{c_i} is the query vector for character *i*, derived from its embedding, and $K_{C_{i+1}}$ is the key vector for *i*+1, also derived from its embedding. The dot product $Q_{c_i} K_{C_{i+1}}^T$ measures the attention that C_i should pay to C_{i+1} scaled by $\sqrt{d_K}$ (the square root of the key vector dimensionality) for gradient stability. This process continues for all consecutive pairs of characters, calculating the attention scores between C_1 and C_2, C_2 and C_3, and so forth, up to C_{n-1} and C_n or C_i and C_{i+1} where $i = 1 \ldots .. n$.

For instance, the attention score between '**ह**' represented by C_1 and '**ऱ्**' represented by C_2, in the given sequence of characters. using (9) is computed as:

$$attention\_scores\_1 = \frac{Q_{c_1} K_{C_2}^T}{\sqrt{d_k}} \qquad (10)$$

The computed attention scores are then applied to the value vectors, allowing the proposed model to focus on the most relevant parts of the input when processing each character. The final attention output for each character is a weighted sum of the value vectors, enabling the proposed model to generate a contextually rich representation of the input sequence by capturing specific relationships between characters. Each Encoder layer includes this Multi-Head Attention mechanism, as well as a Point-wise FFN.

The FFN consists of two fully connected layers separated by a ReLU activation, enabling complex data transformations. To stabilize learning and facilitate gradient propagation, residual connections and layer normalization are applied around both the Multi-Head Attention and FFN components.

ALGORITHM 4.3: ENCODER STACK PROCESSING

Input: Sequence of character embeddings with positional encoding, denoted as X

Output: Contextualized representation of the input sequence

```
Start
For each Encoder layer
    Apply Multi-Head Attention
        Compute the queries (Q), keys (K), and values (V)
                Q = X * W_Q    K = X * W_K    V = X * W_V
        Calculate the scaled dot product of Q and K
                            QK^T = Q * K^T
        /* where K^T is the transpose of the matrix K,
        switching its rows and columns */
        Apply softmax to the scaled dot product
          Attention(Q, K, V) = softmax(QK^T / sqrt(d_k)) * V
        /* where d_K is the dimensionality of the keys, and
        the division by sqrt d_k is used to scale the dot
        product to avoid large values that could lead to small
        gradients */
    Residual Connection and Layer Normalization
                 LayerNorm(X + MultiHeadAttention(X))
    Apply the Point-wise FFN with combined regularization
        FFN(X) #(see Algorithms 4.5, 4.6 and 4.7)
    Residual Connection and Layer Normalization
                        LayerNorm(X + FFN(X))
Return: The final layer output
                #Output generated by the last Encoder layer
End
```

4.2.4 Decoder Stack

The Decoder stack mirrors the Encoder but includes additional components for generating the English or target language sequence. It processes the Encoder's output along with previously generated tokens in the target sequence. Each Decoder layer consists of Masked Multi-Head Self-Attention, which functions similarly to the multi-head attention used in the Encoder but with a crucial difference: it is masked to prevent the Decoder from attending to future tokens during training. The mask ensures that the prediction for a given token can only depend on previously generated tokens, preserving the autoregressive nature of translation generation. In this case, the

attention mechanism is considered self-attention because the model is attending to the target sequence it is generating. The Masked Multi-Head Self-Attention is defined as:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_K}} + M\right) * V \qquad (11)$$

Where Q (Query) is a matrix derived from the current token's hidden state, K (Key) is a matrix representing the hidden states of previous tokens, and V (Value) is a matrix representing either token embeddings or hidden states used for generating output. The term d_K refers to the dimensionality of the keys and queries, used to scale the dot product. The matrix K^T represents the transpose of the Key matrix, where rows and columns of *K* are swapped, allowing for the dot product QK^T, which computes the similarity between queries and keys. The scaled dot product ensures that the resulting values are normalized by dividing by $\sqrt{d_K}$ preventing large values from dominating the softmax function. The mask matrix M is then applied to ensure that the model does not attend to future tokens during training, enforcing an autoregressive pattern.

For example, during training, when generating the target sequence "Exemption from Procedures" from the source "**ከሥነ ሥርዓት፡ ነጻ ስለመሆን**", at any point in time, the proposed model can only attend to previously generated tokens in the target sequence. Masked Multi-Head Self-Attention ensures that, when predicting the token 'm' in the word "Exemption", the model can only attend to the earlier tokens 'E', 'x', and 'e'. It cannot access subsequent tokens like 'p', 't', and 'i', preserving the left-to-right decoding order. This masking mechanism enforces a strict autoregressive pattern, where the model learns to predict each token based only on the tokens it has generated so far, without seeing future tokens.

After processing the target sequence with Masked Multi-Head Self-Attention, the Decoder applies another multi-head attention mechanism: the Multi-Head Encoder-Decoder Attention, which is an example of cross-attention. Unlike self-attention, cross-attention allows the Decoder to attend to the output from the Encoder. In this case, the query vectors come from the Decoder's current state, which is the partial English target sequence generated so far, while the key and value vectors come from the Encoder's output, which represents the encoded Amharic source sequence. The cross-attention mechanism allows the Decoder to focus on relevant parts of the Amharic source sequence when generating the English target sequence, using information from the entire input sequence rather than just attending to the target sequence, as it does with self-attention.

For instance, when the Decoder is generating the English translation for "**ከሥነ ሥርዓት**" ("from procedures"), the Multi-Head Encoder-Decoder Attention allows it to attend to the relevant source tokens such as '**ከ**', '**ሥ**', '**ነ**', ' ', '**ሥ**', '**ር**', '**ዓ**', and '**ት**'. The query vectors from the Decoder focus on the target tokens generated so far, such as 'E', 'x', 'e', 'm', 'p', 't', 'i', 'o', and 'n', while the key and value vectors from the Encoder highlight the source tokens corresponding to "**ከሥነ ሥርዓት**". This attention mechanism ensures that the model generates the translation correctly by aligning the source and target tokens effectively. In both attention mechanisms, the scaled dot product is critical to ensure that the similarity scores between query and key vectors are normalized, thus preventing any single value from disproportionately influencing the attention weights after applying the softmax function. By utilizing both self-attention within the target sequence, English, and cross-attention from the source sequence, Amharic, the Decoder generates the English translation, informed by the preceding context and aligned with the original Amharic input.

The point-wise FFN within both the encoder and decoder of the transformer model architecture is integral to transforming high-dimensional representations generated by the attention mechanisms into more refined and task-specific features. Despite its importance, the FFN's dense connections and high dimensionality make it particularly prone to overfitting, a challenge worsened by the limited training data often available for low-resource languages like Amharic. Overfitting in this context can lead to the model memorizing specific patterns in the training data, rather than learning generalizable linguistic structures, which diminishes its performance on unseen data.

To address overfitting in our proposed model, we applied advanced regularization techniques specifically within the FFN, integrating standard dropout with L1, L2, and Elastic Net regularization to encourage smaller, more meaningful weights. Focusing on the FFN leverages its crucial role in transforming attention-derived features into outputs for translation tasks, directly impacting the model's ability to capture complex dependencies. While our current emphasis is on the FFN, this combined regularization approach could also be applied effectively to other layers within the Transformer, establishing a foundation for future model enhancements.

ALGORITHM 4.4: DECODER STACK PROCESSING

Input: Encoder output, previously generated target (English)sequence
Output: Probability distribution (PD) over next token

Start
For each Decoder layer
    Apply Masked Multi-Head Self-Attention
        Apply attention to the target sequence with a mask to prevent attending to future tokens
        Compute Query (Q), Key (K), and Value (V) matrices from the target sequence Y:
            $Q = YW_Q \quad K = YW_K \quad V = YW_V$
        #where W_Q, W_K, and W_V are the projection weight matrices
        Compute Masked attention as:
            Attention(Q, K, V) = softmax((QK^T / sqrt(d_k)) + M) * V
        /* where, M represents the mask matrix that prevents attention to future tokens, and d_k is the dimensionality of the key vectors */
    Apply Residual Connection and Layer Normalization:
        Output = LayerNorm(Y + Masked Multi-Head Self-Attention(Y))
    Apply multi-head attention between the target (English) sequence and encoder output
        The Query (Q) comes from the Decoder's current state, while the Key (K) and Value (V) come from the Encoder's output E:
            $Q = YW_Q, \quad K = EW_K, \quad V = EW_V$
        Compute the attention
            Attention(Q, K, V) = softmax(QK^T / sqrt(dk)) * V
    Apply Residual Connection and Layer Normalization
        Output = LayerNorm (Y + Multi-Head Self Attention(Y, E))
    Apply the Point-Wise FFN with combined regularization
        FFN(Y) #(see Algorithms 4.5, 4.6 and 4.7)
    Apply Residual Connection and Layer Normalization:
        Output = LayerNorm(Y + FFN(Y))
Apply linear vocabulary projection and softmax
    The final Decoder output is projected to the vocabulary size and passed through a softmax to compute probabilities:
        PD = softmax(Linear Vocabulary Projection(Y))
Return: PD over the vocabulary for the next token
End

Standard dropout is employed to randomly deactivate a subset of neurons in the proposed model, preventing the FFN from becoming overly reliant on particular pathways and encouraging more robust learning. By randomly setting a fraction of the input units to zero during each update, dropout reduces interdependent learning among neurons.

$$y = dropout\ (x, p) \qquad (12)$$

where x denotes the input vector, p is the dropout rate (the probability of setting a unit to zero), and y is the resulting output vector after dropout has been applied. This technique mitigates overfitting by ensuring that no single neuron is only responsible for predictions, thereby promoting generalization.

L1 regularization encourages sparsity in the proposed model by penalizing the absolute values of the weights. This can lead to a more interpretable model and reduce the reliance on specific features. The penalty applied by L1 regularization is given by:

$$\text{L1\_Penalty} = \lambda \sum_i |w_i| \qquad (13)$$

where λ is the regularization strength parameter controlling the impact of the L1 regularization on the overall loss function, i denotes the index of each weight in the model, and w_i represents the individual weights.

L1 regularization promotes sparsity in the network by pushing many weights towards zero, aiding in feature selection. When combined with dropout, this creates a powerful regularization effect. Dropout randomly deactivates a portion of neurons during training, preventing complex co-adaptations and forcing the model to learn more robust features. The interaction between L1 and dropout is complementary: L1 encourages sparse representation, while dropout ensures this sparsity is distributed across different feature sets in each training iteration. This combination is particularly effective in preventing overfitting, especially with highly correlated input features or limited training data.

ALGORITHM 4.5: L1 REGULARIZATION COMBINED WITH DROPOUT IN THE POINT-WISE FFN

Input: Input tensor for the Point-wise FFN layer
Output: Regularized and normalized output tensor

Start
Initialize the L1 regularizer with λ as the regularization strength parameter controlling the impact of L1 regularization
Define the point-wise FFN
    Create a Sequential model
    Add the first Dense layer with the hidden layer dimension units, ReLU activation, and L1 regularization
    Add the second Dense layer with the model dimension units and L1 regularization
    Add a Dropout layer to reduce over-reliance on specific neurons, complementing the sparsity induced by L1 regularization
    Add a Layer Normalization layer to stabilize and accelerate training
Implement the Point-wise FFN in the model
Process the input tensor through the Point-wise FFN
Return: The processed output tensor
End

L2 regularization helps prevent individual weights from becoming too large, promoting a more balanced and stable learning process in the proposed model. This regularization method encourages the model to distribute the weight values more evenly, reducing the risk of overfitting. The penalty applied by L2 regularization is given by:

$$\text{L2\_Penalty} = \lambda \sum_i w_i^2 \qquad (14)$$

where λ is the regularization strength parameter that controls the impact of L2 regularization on the overall loss function, i denotes the index of each weight in the model, and w_i represents the individual weights in the model.

L2 regularization, also known as weight decay, encourages the model to use all inputs moderately rather than relying heavily on a few. It adds a penalty term proportional to the sum of squared weights, leading to smaller weights across the board. The combination with dropout creates a balanced regularization effect. While L2 pushes all weights to be small but non-zero, dropout

effectively creates many different 'thinned' networks during training. This synergy provides consistent pressure to keep weights small while introducing randomness that prevents over-dependence on specific neurons. The result is a model that spreads learning across all neurons, enhancing robustness and reducing the risk of overfitting. This approach is especially beneficial in transformer-based NMT models for low-resource languages, where overfitting is a significant concern.

ALGORITHM 4.6: L2 REGULARIZATION COMBINED WITH DROPOUT IN THE POINT-WISE FFN

Input: Input tensor for the Point-wise FFN layer
Output: Regularized and normalized output tensor

Start
Initialize the L2 regularizer with λ as the regularization strength parameter controlling the impact of L2 regularization
Define the point-wise FFN
    Create a Sequential model
    Add the first Dense layer with the hidden layer dimension units, ReLU activation, and L2 regularization
    Add the second Dense layer with the model dimension units and L2 regularization
    Add a Dropout layer to reduce the risk of co-adaptation among neurons, complementing the balance induced by L2 regularization
    Add a Layer Normalization layer to stabilize and accelerate training
Implement the Point-wise FFN in the model
Process the input tensor through the Point-wise FFN
Return: The processed output tensor
End

Elastic net regularization, which combines both L1 and L2 penalties, offers a balanced approach that utilize the benefits of both methods. By blending the strengths of L1 and L2, this approach provides a robust regularization framework that can adapt to various types of model configurations, making it particularly effective in the proposed model. The penalty applied by Elastic Net regularization is given by:

$$Elastic\_Net\_Penalty = \lambda\left(\frac{1-p}{2}\sum_i w_i^2 + p\sum_i |w_i|\right) \quad (15)$$

where λ is the overall regularization strength, p is the mixing parameter between L1 and L2 penalties and where w_i represents the weights of the model.

ALGORITHM 4.7: ELASTIC NET REGULARIZATION COMBINED WITH DROPOUT IN THE POINT-WISE FFN

Input: Input tensor for the Point-wise FFN layer
Output: Regularized and normalized output tensor

Start
Initialize the Elastic Net regularizer with λ as the overall regularization strength and p as the mixing parameter between L1 and L2 penalties
Define the point-wise FFN
    Create a Sequential model
    Add the first Dense layer with the hidden layer dimension units, ReLU activation, and Elastic Net regularization
    Add the second Dense layer with the model dimension units and Elastic Net regularization
    Add a Dropout layer to complement the mixed regularization effects of L1 and L2 (with mixing parameter p) by preventing the model from overfitting to specific neurons
    Add a Layer Normalization layer to stabilize and accelerate training
Implement the Point-wise FFN in the model
Process the input tensor through the Point-wise FFN
Return: The processed output tensor
End

Elastic Net regularization combines L1 and L2 penalties, offering a balanced approach that leverages the benefits of both methods. This combination aims to maintain L1's feature selection properties while benefiting from L2's stability and generalization improvements. When used with dropout, it creates a powerful three-way regularization effect. The L1 component promotes sparsity and aids in feature selection, the L2 component encourages smaller weights across the board, and dropout ensures the network doesn't overly rely on specific features or neurons. This combination results in a highly regularized model capable of capturing complex data patterns while maintaining good generalization. The interaction between Elastic Net and dropout allows for fine-tuned regularization by adjusting the mixing parameter and dropout rate, making it adaptable to various data characteristics and problem types.

4.2.5 Linear Vocabulary Projection and Softmax

After the Decoder stack, the output undergoes two final transformations: a Linear Vocabulary Projection layer followed by a Softmax function. The Linear layer transforms the high-dimensional output into the vocabulary space or dimension of the target language (English), preparing the data for the final classification step. This linear transformation can be expressed as:

$$Z = W * X + b \qquad (16)$$

where Z is the output of the linear layer, W is the weight matrix, X is the input to the linear layer, and b is the bias term.

Following this linear transformation, the Softmax function is applied. This function converts the output scores into probabilities, indicating the likelihood of each token being the correct one in the sequence. The Softmax function is defined as:

$$Softmax\ (Z)_i = \frac{e^{z_j}}{\sum_{K=1}^{K} ez_K} \qquad (17)$$

where Z_j is the j-th element of the input vector z and K the number of possible tokens.

Together, these two operations play a critical role in the final stage of translation. The Linear Vocabulary Projection layer maps the Decoder's output to the target or English vocabulary space, while the Softmax function normalizes these scores into a probability distribution over possible tokens, enabling the model to make its final prediction for each position in the output sequence. The linear vocabulary projection and softmax algorithm is presented in Algorithm 4.8.

4.2.6 Output Sequence

The final component of our proposed model involves generating the output sequence by iteratively selecting tokens based on the probabilities computed by the Softmax layer. For each time step, the token with the highest probability is chosen as the next word in the sequence. This process continues until the entire output sequence is generated.

The token selection at each time step t can be formally expressed as:

$$y_t = argmax\ (softmax\ ((z_t))) \qquad (18)$$

where y_t is the token selected at time step *t* and z_t is the linear layer output at time step *t*.

The argmax function identifies the index of the maximum value from the Softmax output probabilities, effectively selecting the token with the highest predicted probability as the next word in the generated sequence. This iterative process continues until either an end-of-sequence token is generated or a predetermined maximum sequence length is reached, resulting in the complete translated output. The output sequence generation algorithm is presented in Algorithm 4.9.

ALGORITHM 4.8: LINEAR VOCABULARY PROJECTION AND SOFTMAX

Input: Decoder output X (from the final decoder layer)
Output: Probability distribution (PD) over the target (English) vocabulary

1 **Start**
2 Apply Linear Projection:

$$Z = W * X + b$$

/* $W \in R^d$ is the weight matrix, projecting the decoder output into the vocabulary space of size k, $X \in R^d$ is the decoder output from the final decoder layer, with d being the hidden dimension of the model, $b \in R^k$ is the bias vector and the output $Z \in R^k$ is a vector of logits, representing the unnormalized scores for each token in the vocabulary */
3 Apply Softmax
4 | For each token j in the vocabulary
5 | | $P(j) = \exp(Z_i) / \text{sum}(\exp(Z_k) \text{ for all } k)$
| | /* Z_j is the j-th element of the input vector Z, representing the logit (unnormalized score) for token j, k is the total number of possible tokens (vocabulary size) and the denominator sums over all k tokens, ensuring that the output is a valid probability distribution */
6 **Return:**
PD = [$\text{Softmax}(Z)_1$ (P1), $\text{Softmax}(Z)_2$ (P2) …, Softmax(Z)k (Pk)]
/* Return PD, which contains the probabilities for each token in the target (English) vocabulary */
7 **End**

ALGORITHM 4.9: OUTPUT SEQUENCE GENERATION

Input: Proposed trained Transformer Model for Amharic-to-English Neural Machine Translation and an Amharic phrase or sentence

Output: Translated English phrase or sentence

Start
Encode the Amharic sentence or phrase using the Encoder stack
Initialize English sequence with start token
While the end token has not been generated and the length of the sequence is less than the maximum length
    Pass the current English sequence and the Encoder stack output through the Decoder stack
    Get probability distribution for the next token
    Select the token with the highest probability
    Append the selected token to the English sequence
Return: Generated English sequence
End

Chapter 5

Experimentations and Evaluation

5.1 Introduction

To validate the effectiveness of our proposed Transformer-based Amharic-to-English NMT model, a series of experiments were conducted. This chapter outlines the data collection process, the experimental system environment, and the experiments performed on both the baseline Transformer model and our proposed model, which incorporates character embedding and various combined regularization techniques applied in the FFN. Additionally, the implications and interpretations of the experimental results are discussed by comparing the models to each other and to previous benchmarks.

5.2 Dataset Collection

For our experiments, we utilized a dataset consisting of 100,481 Amharic-English sentence pairs. These data were sourced from various parallel corpora, as discussed in Chapter 2, Section 2.8, including religious and legal documents. Of the total paired dataset, 77.97% (78,345 sentence pairs) are from religious texts, primarily from the Bible, while the remaining 22.03% (22,136 sentence pairs) are derived from legal documents such as proclamations, directives, and legislations.

5.3 Experimental System Environment

In order to experiment with the baseline and our proposed model, the system environment was carefully selected to ensure optimal performance for the computationally intensive tasks involved in training the Transformer-based NMT model. The experiments were conducted on a subscription-based cloud computing environment, Google Colab Pro+, which provided access to NVIDIA A100 GPUs with 40GB of GPU memory and 100GB of disk space. These resources were critical for fully supporting character-level embeddings and managing the large computational load required for the multi-head attention mechanisms in our proposed model.

Python 3.9 was selected as the programming language for its rich ecosystem of libraries suited to our experiments. The model implementation was built using TensorFlow 2.15.1, a DL framework

known for its efficient GPU acceleration, which ensured optimal performance during training and testing. TensorFlow's extensive support for neural networks made it an ideal choice for building the Transformer architecture, including components such as the self-attention mechanism, encoder-decoder stacks, and FFNs.

The following packages were employed to support various aspects of the baseline and our proposed models:

- **NumPy**: This package was used for handling numerical computations in the model, especially for matrix operations, which are crucial when processing large input data for character embeddings and attention mechanisms.
- **Pandas**: In the context of our proposed model, pandas was used for managing and manipulating the Amharic-English sentence pairs. It efficiently handled data loading, structuring, and preprocessing, allowing us to organize and analyze the parallel corpora before feeding the data into the model for training.
- **Matplotlib**: This visualization library was utilized to plot the cumulative token frequency distribution and token frequency distribution for both Amharic and English. It was also utilized to plot the learning curves of the baseline and our proposed models, showing key metrics such as training and validation loss, as well as accuracy, helping us monitor the model's performance during the training phase.
- **Adam Optimizer**: The Adam optimizer was employed for its adaptive learning rate capabilities, which allowed the model to converge faster. This was critical given the complexity of training a Transformer-based model with character-level embeddings. To further improve the optimization process, a PolynomialDecay schedule was applied, gradually reducing the learning rate over time.
- **BLEU Score (NLTK)**: The BLEU score module from NLTK was used to evaluate the translation performance of the model, providing a quantitative assessment of how well the model translated Amharic sentences into English.

5.4 Dataset Preparation

The dataset preparation phase began with the upload of two text files, "Amharic.txt" and "English.txt", to Google Drive. These files contained parallel sentences in Amharic and English,

respectively. The files were then loaded into Google Colab with careful attention to encoding to ensure proper handling of Amharic characters.

The initial step in preprocessing involved cleaning the raw text data. For Amharic text, a specialized function was developed to remove any characters that were not part of the Amharic Unicode range (U+1200 to U+137F), Amharic punctuation marks (U+1361 to U+1368), Amharic numerals (U+1369 to U+1371), or Arabic digits (U+0030 to U+0039). This ensured that only valid Amharic text remained. Additionally, homoglyphs normalization was performed for Amharic characters such as ኧ and ዐ, ሀ, ሐ, and ኀ, as well as ሰ and ሠ, and ጸ and ፀ. For English text, the preprocessing included expanding contractions, for example "don't" to "do not", "I'm" to "I am" and so on to standardize the text. Unicode normalization was also applied to ensure consistent representation of characters, especially for handling any non-ASCII characters that might appear in the English text.

After the initial cleaning, the dataset consisted of 100,481 sentence pairs. To ensure a balanced dataset suitable for our baseline and proposed model training, sentences were filtered based on length. The minimum sentence length was 1 word, while the maximum was 229 words for both languages. This filtering process resulted in a slightly reduced dataset of 100,450 pairs, which were then ready for further processing and splitting.

The dataset was divided into training, validation, and test sets using a stratified sampling approach. This method ensures that the distribution of sentence lengths is similar across all sets, which is crucial for maintaining the dataset's representativeness. Stratified sampling was chosen because it helps to preserve the proportion of different sentence lengths in each subset, which is important for training a model that can handle various sentence structures effectively. The split resulted in 64,288 pairs (64%) for training, 16,072 pairs (16%) for validation, and 20,090 pairs (20%) for testing.

The training dataset is used to train both the baseline and proposed models on translation patterns. During each training epoch, the model adjusts its parameters based on the errors it makes on this dataset. The validation set plays a crucial role in the training process by providing an unbiased evaluation of the model's performance after each epoch. It helps in hyperparameter tuning by allowing us to adjust parameters such as learning rate, batch size, or model architecture to improve generalization. Additionally, the validation dataset is useful for early stopping. If the model's

performance on the validation set starts to degrade while still improving on the training set, it indicates overfitting. This signals that training should be stopped to prevent the model from memorizing the training data. The test set, which is completely held out during training and validation, provides a final, unbiased evaluation of the model's performance on unseen data.

To better understand the token distribution in our dataset, we analyzed the frequency of tokens in both Amharic and English. We generated cumulative token frequency and token frequency distribution plots for both languages. These figures illustrate the distribution of token frequencies in our dataset. The cumulative frequency plots show how many tokens are required to cover a certain percentage of the text. The token frequency distributions demonstrate the long-tail nature of both Amharic and English, where a small number of tokens occur very frequently, while a large number of tokens appear rarely.

Figure 5.1 and Figure 5.2 show the cumulative token frequency and token frequency distribution for Amharic, respectively.

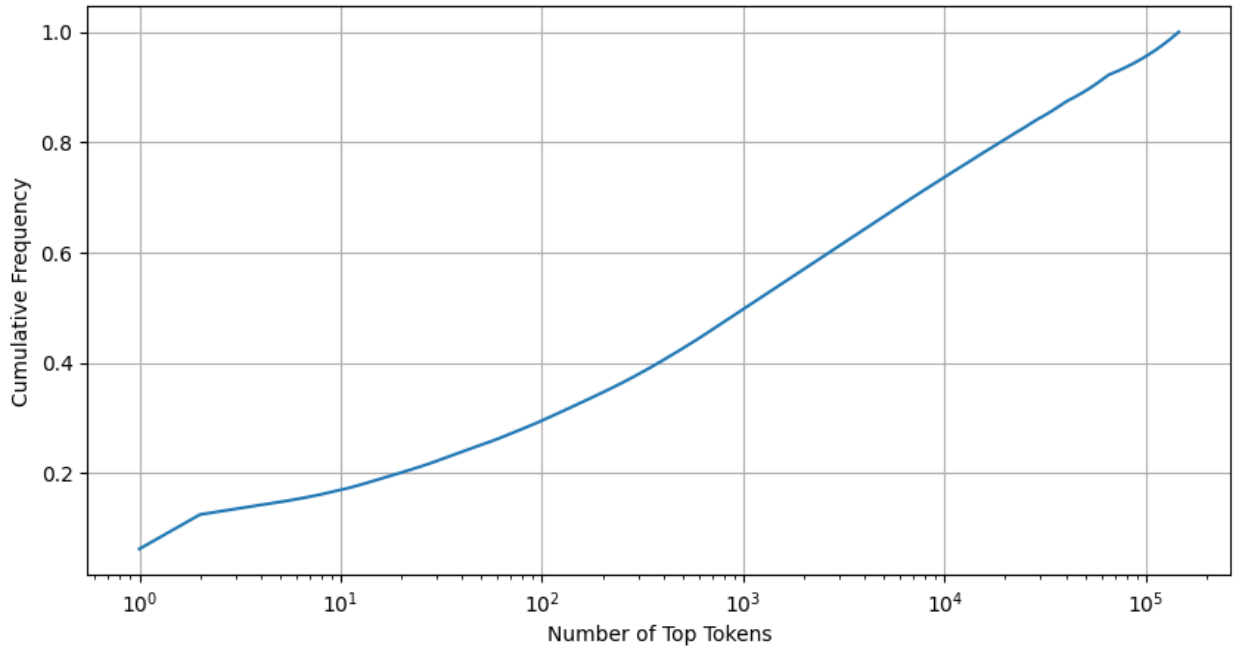

Figure 5.1: *Amharic Cumulative Token frequency distribution*

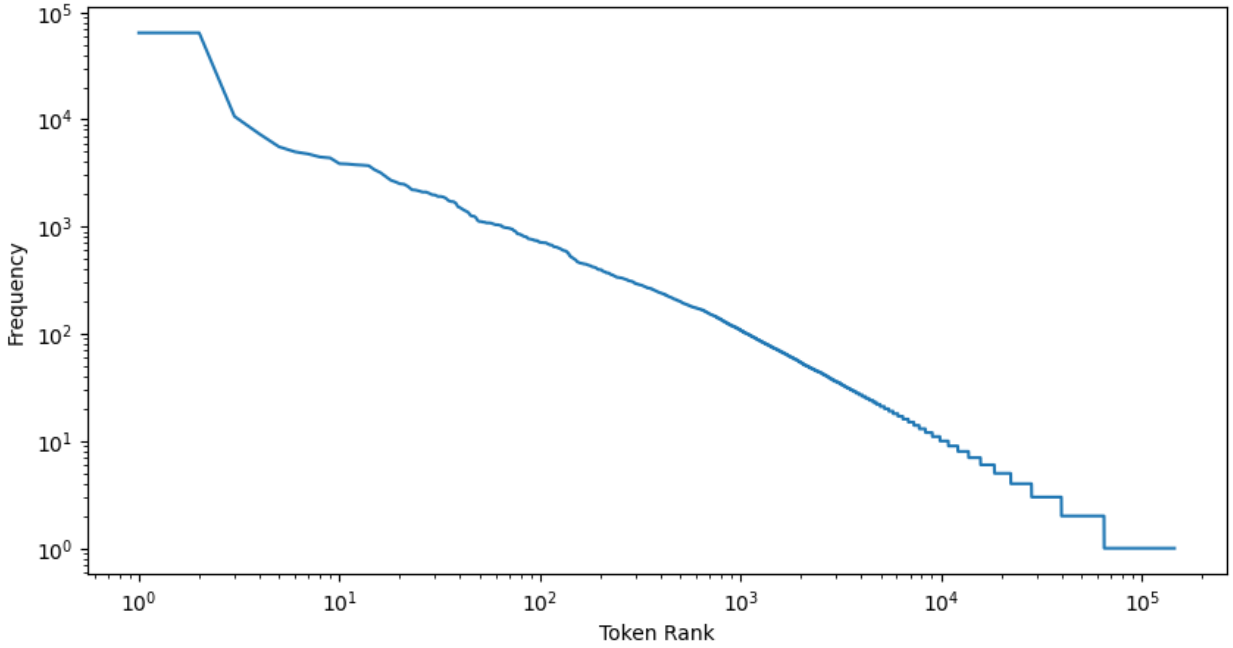

Figure 5.2: *Amharic Token frequency distribution*

Similarly, Figure 3 and Figure 4 present the cumulative token frequency and token frequency distribution for English.

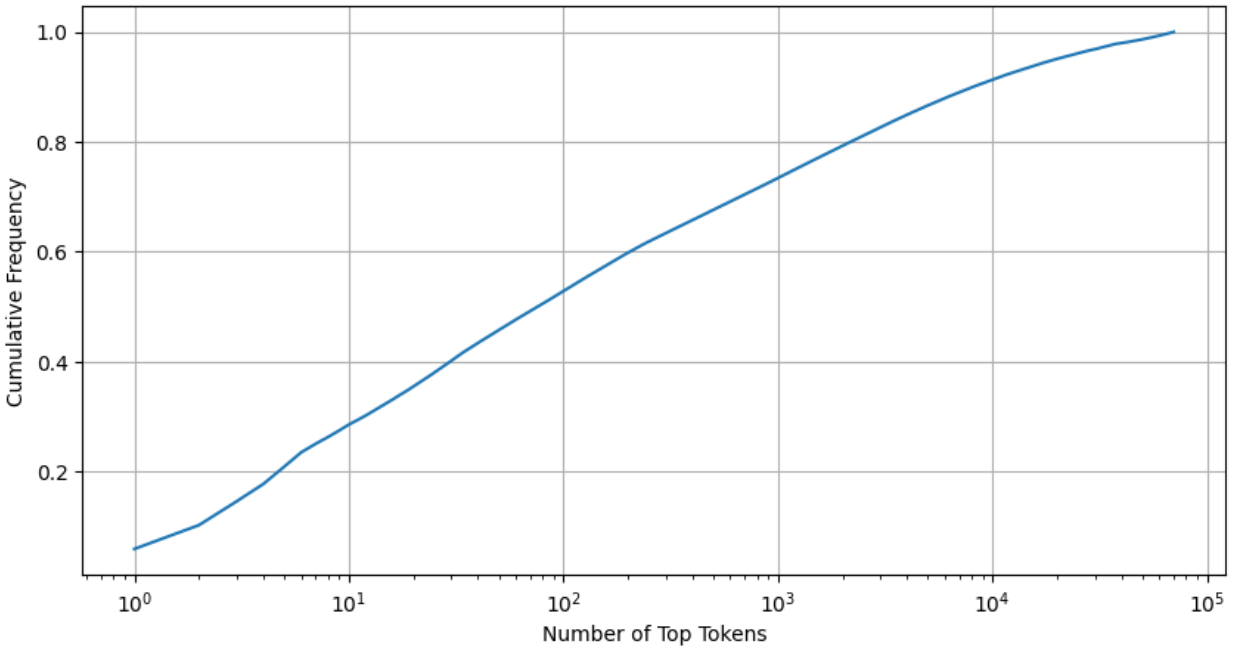

Figure 5.3: *English Cumulative Token frequency distribution*

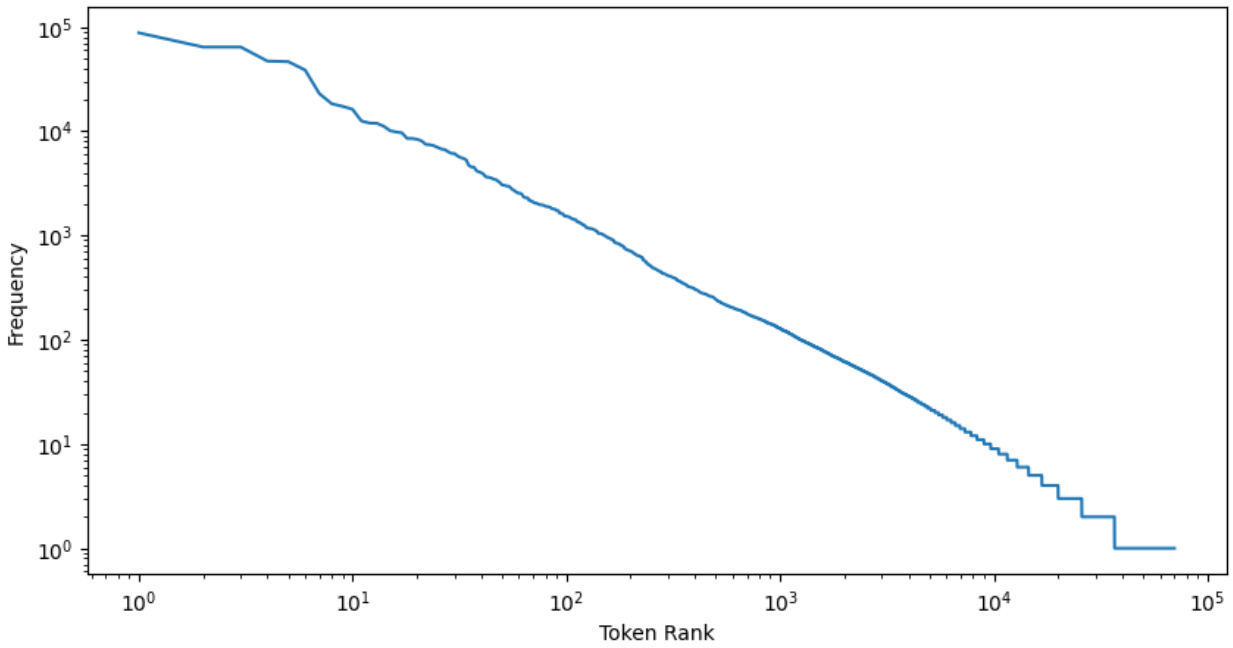

Figure 5.4: *English Token frequency distribution*

For our baseline transformer model which utilizes word embedding, tokenization was performed by splitting the text on whitespace. The maximum number of tokens observed in the Amharic sentences was 179, while for English it was 162. To determine a balanced sequence length for padding or truncation, the 95th percentile of token counts was calculated, resulting in 31 tokens for Amharic and 46 for English. These values were chosen to cover the majority of sentences without excessive padding, which could lead to inefficient processing.

Text vectorization was then applied to convert the tokenized text into integer sequences. The TextVectorization layer from TensorFlow's Keras API was utilized to convert the preprocessed or tokenized Amharic and English texts in both word and character embeddings. This layer handles the process of converting text strings to sequences of integer indices, making it easier to feed the data into the neural network. For word-level embedding, the vocabulary size for Amharic was found to be 146,221 unique words, while English had 62,960. This significant difference in vocabulary size reflects the morphological richness of the Amharic language compared to English.

For our proposed model which utilizes character-level embedding, a similar process was followed, but with characters as the basic units instead of words. The TextVectorization layer was configured to split the text into individual characters rather than words. The maximum sequence length for Amharic characters was 1,053, and for English, it was 1,733. The character-level vocabulary sizes

were much smaller: 318 for Amharic and 55 for English, reflecting the limited set of characters used in each language. This approach allows the model to learn sub-word patterns and may be particularly beneficial for morphologically rich languages like Amharic.

This comprehensive analysis of frequency distribution for tokens and characters helped in determining appropriate sequence lengths and vocabulary sizes for both word and character-level models, striking a balance between coverage and computational efficiency.

5.5 Experiments and Results

We conducted experiments on both the baseline Transformer model and our proposed Transformer model, which incorporates character embedding and combined regularization techniques in the Point-wise FFN. The details of each experiment, along with the results for both models, are presented in the following subsections.

5.5.1 Experiment on the Baseline Transformer Model

We first conducted experiments on the baseline Transformer model proposed by Vaswani *et al*. [10] using standard word embedding. The model was configured with a sequence length of 64, determined based on the 95th percentile token counts in both languages: 31 tokens for Amharic and 46 tokens for English, as indicated in section 5.4. A sequence length of 64 was chosen to ensure sufficient coverage of most sentences in both languages, allowing the model to capture the majority of linguistic patterns while maintaining computational efficiency.

Other model configurations included a batch size of 64, an embedding dimension of 256, a latent dimension of 512, eight attention heads, five encoder and decoder layers, and a dropout rate of 0.3. The dropout was applied similarly to the residual dropout in the original Transformer model [10]. The model was compiled using the Adam optimizer with a custom learning rate schedule, and the loss function used was SparseCategoricalCrossentropy, which is implemented in the TensorFlow tf.keras.losses module. This loss function is commonly used for multi-class classification problems, particularly when dealing with integer labels, and works by comparing the predicted class probabilities to the actual class labels, using an efficient sparse representation.

The model, comprising 75,012,848 parameters (286.15 MB), was trained on the training dataset for 20 epochs, with early stopping implemented to prevent overfitting. Early stopping monitored the validation dataset performance, halting training if the validation loss stopped improving for 5 consecutive epochs. This approach ensured that the model did not overfit to the training data while still aiming for optimal generalization.

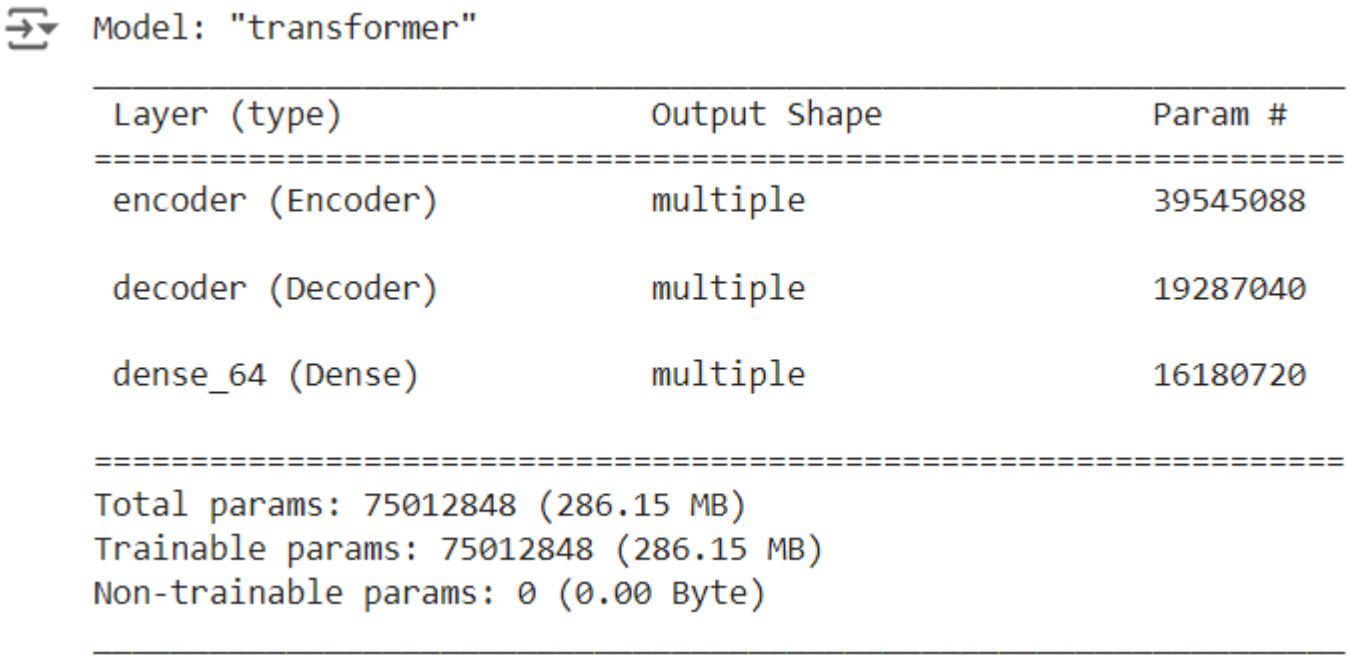

Figure 5.5: *Screenshot of the Baseline Transformer Model Summary*

The training process yielded an average training accuracy of 0.6716, an average training loss of 2.1488, an average validation accuracy of 0.6558, and an average validation loss of 2.3977.

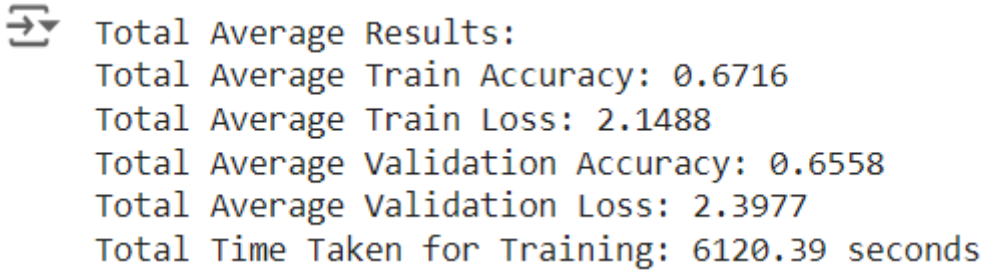

Figure 5.6: *Screenshot of the Baseline Model Training Results*

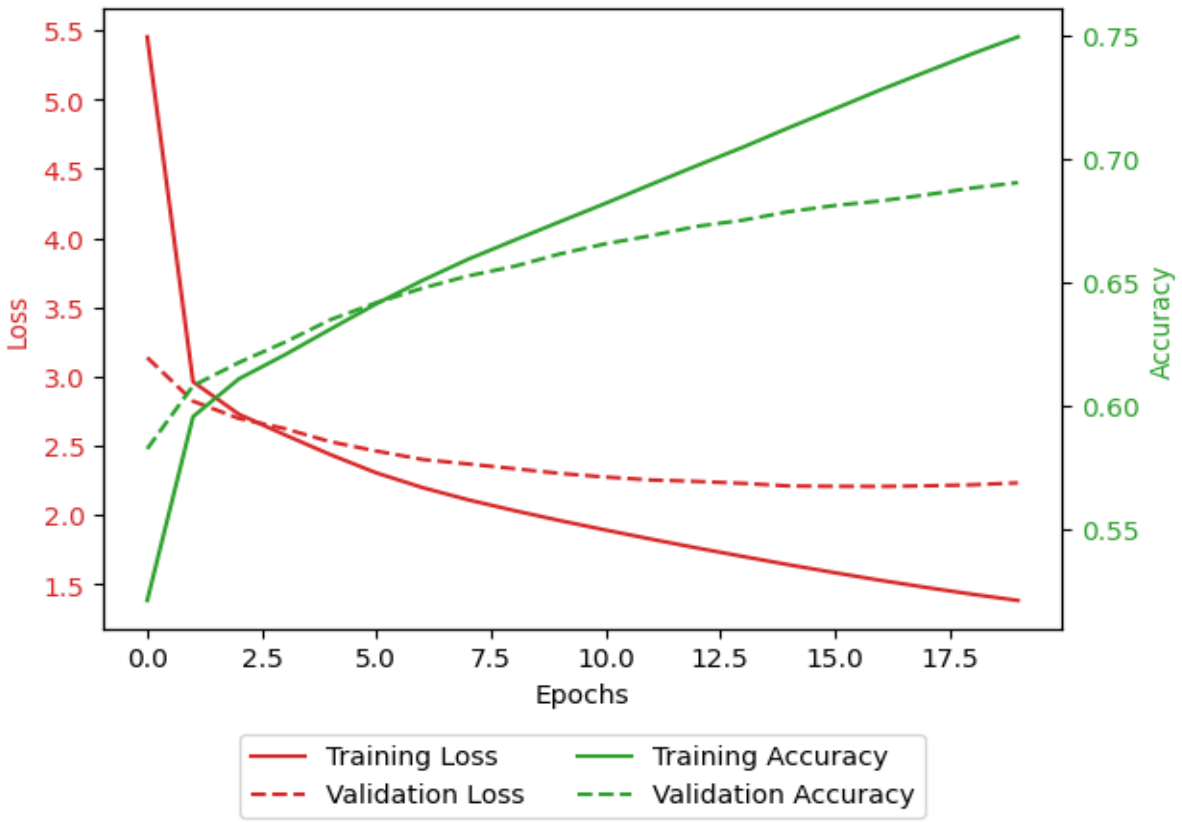

Figure 5.7: *Training and Validation Loss and Accuracy Curves for the baseline Transformer model with Word Embedding and Dropout only*

When evaluated on a test dataset of 20,090 pairs, the baseline model achieved a test loss of 2.2087 and a test accuracy of 0.6832.

```
Test loss: 2.2087487965632397
Test accuracy: 0.6832802547770697
```

Figure 5.8: *Screenshot of the Baseline Model Evaluation Results on a Test Dataset*

Additionally, the average BLEU score, calculated on a subset of 3,000 pairs from the test dataset, was 31.04.

```
Maximum BLEU score: 0.8932
Average BLEU score: 0.3104
```

Figure 5.9: *Screenshot of the Maximum and Average BLEU Score Results of the Baseline Model*

Table 2. 10: *Summary of the Baseline Model Experimental Results*

Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss	BLEU
0.6716	2.1488	0.6558	2.3977	0.6833	2.2087	31.04

5.6 Experiment on Proposed Model (Character Embedding and Combined regularization Techniques)

We conducted experiments on our proposed Transformer model for Amharic-to-English NMT using character embedding, applying various combined regularization techniques in the Point-wise FFN component. The use of character embedding significantly reduced the model size to 5,440,663 parameters (20.75 MB), resulting in a substantial improvement in efficiency.

```
Model: "transformer"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 encoder (Encoder)           multiple                  2218544

 decoder (Decoder)           multiple                  3207984

 dense_64 (Dense)            multiple                  14135

=================================================================
Total params: 5440663 (20.75 MB)
Trainable params: 5440663 (20.75 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 5.10: *Screenshot of the Proposed Model Summary*

5.6.1 Character Embedding and Dropout with Elastic net Regularization

The first experiment we conducted on our proposed model involved applying character embedding as the input method. In this experiment, the model was designed with a sequence length set to the maximum for each language: 318 for Amharic and 55 for English, along with all the data preparation steps applied to our proposed model, as discussed in the data preparation section (5.4).

The configuration included a batch size of 64, an embedding dimension of 256, a latent dimension of 524, eight attention heads, and 5 encoder and decoder layers. A dropout rate of 0.3 was applied, similar to the residual dropout used in the baseline Transformer model. However, in this experiment, the dropout was combined with Elastic Net regularization in the Point-wise FFN, incorporating both L1 and L2 penalties with equal regularization strengths of 0.01

The model was trained for 20 epochs, resulting in an average training accuracy of 0.7901 and a training loss of 1.1782. The validation performance showed an average accuracy of 0.7788 and a validation loss of 1.4191.

```
Total Average Results:
Total Average Train Accuracy: 0.7901
Total Average Train Loss: 1.1782
Total Average Validation Accuracy: 0.7788
Total Average Validation Loss: 1.4191
Total Time Taken for Training: 6852.64 seconds
```

Figure 5.11: *Screenshot of the Proposed Model Character Embedding and Dropout with Elastic Net Combined Regularization Training Results*

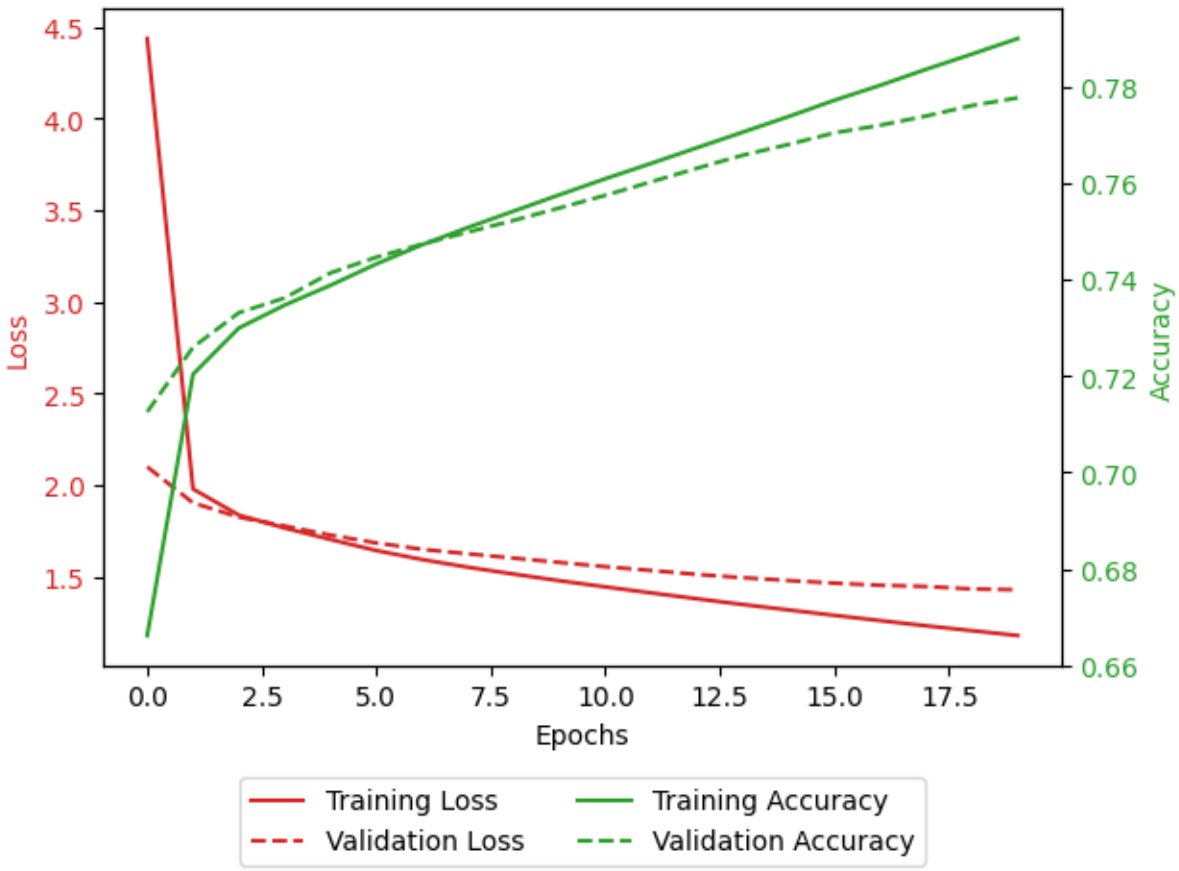

Figure 5.12: *Training and Validation Loss and Accuracy Curves for the proposed model with Character Embedding and Dropout with Elastic Net*

When evaluated on the test dataset, the model achieved a test loss of 1.4121 and a test accuracy of 0.7792.

```
Test loss: 1.4121106620818848
Test accuracy: 0.779245777370777
```

Figure 5.13: *Screenshot of the Proposed Model Character Embedding and Dropout with Elastic Net Combined Regularization Evaluation Results on a Test Dataset*

Additionally, the average BLEU score on a subset of 3,000 sentence pairs from the test dataset was 37.27.

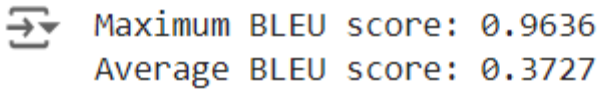

Figure 5.14: *Screenshot of the Maximum and Average BLEU Score Results of the Proposed Model with Character Embedding and Dropout with Elastic Net Combined Regularization*

5.6.2 Character Embedding and Dropout with L1 Regularization

Our second experiment on the proposed model followed the same preprocessing steps, model design, and configuration as in the first experiment conducted on our proposed model. However, in this case, dropout was combined with L1 regularization in the Point-wise FFN, using a regularization strength parameter of 0.01.

The model was trained for 20 epochs, yielding an average training accuracy of 0.7885 and an average training loss of 1.1873. The validation results showed an average accuracy of 0.7771 and a validation loss of 1.4250.

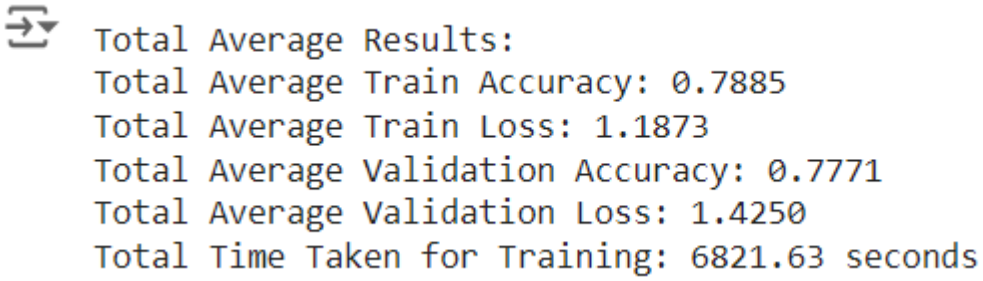

Figure 5.15: *Screenshot of the Proposed Model Character Embedding and Dropout with L1 Combined Regularization Training Results*

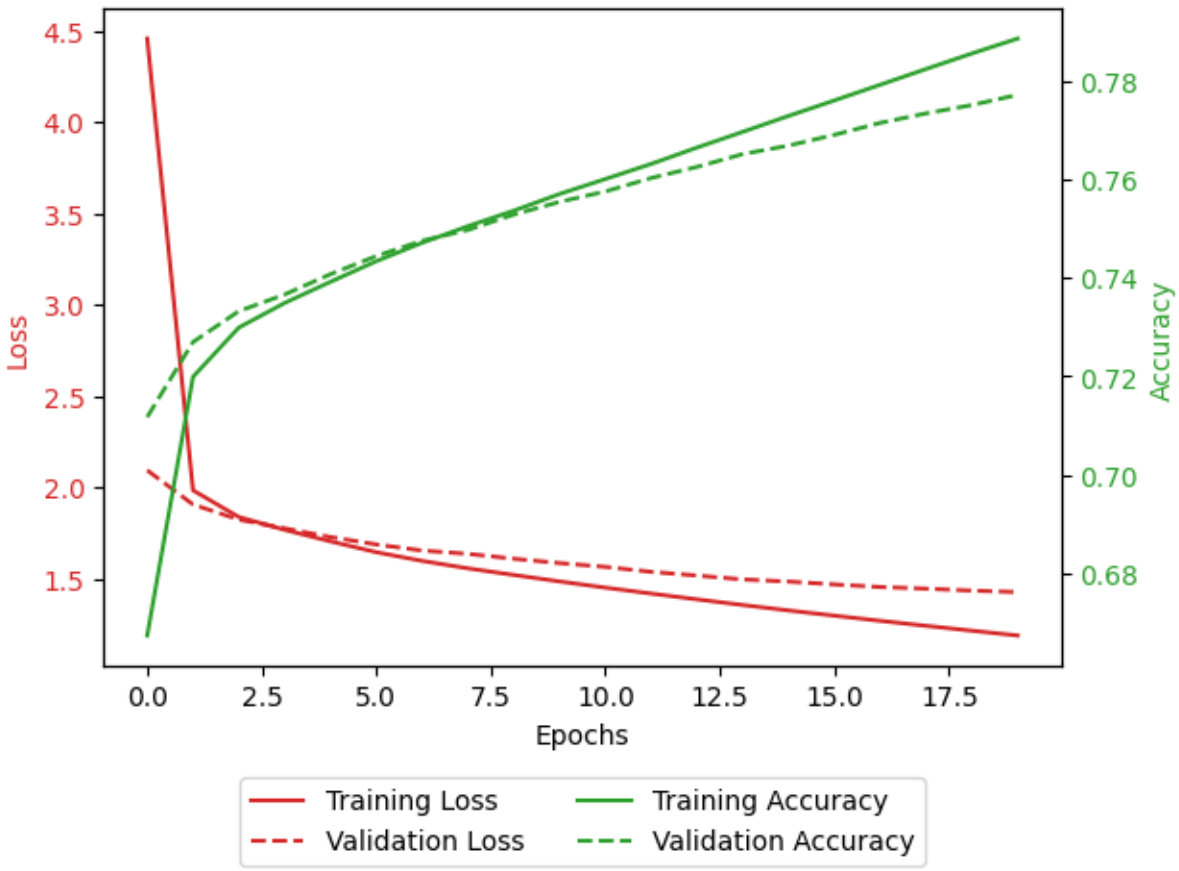

Figure 5.16: *Training and Validation Loss and Accuracy Curves for the proposed model with Character Embedding and Dropout with L1*

On the test dataset, the model achieved a test loss of 1.4110 and a test accuracy of 0.7781.

```
Test loss: 1.4110293036415464
Test accuracy: 0.7781438237688232
```

Figure 5.17: *Screenshot of the Proposed Model Character Embedding and Dropout with L1 Combined Regularization Evaluation Results on a Test Dataset*

The average BLEU score, calculated on a subset of 3,000 sentence pairs from the test dataset, was 36.58.

```
Maximum BLEU score: 0.9391
Average BLEU score: 0.3658
```

Figure 5.18: *Screenshot of the Maximum and Average BLEU Score Results of the Proposed Model with Character Embedding and Dropout with L1 Combined Regularization*

5.6.3 Character Embedding and Dropout with L2 Regularization

In our final experiment on the proposed model, we applied a combination of two regularization techniques: dropout and L2 regularization within the Point-wise FFN, while maintaining character embedding as the input method. The model configuration remained consistent with the previous two experiments.

As in the previous experiments, the model was trained for 20 epochs, achieving an average training accuracy of 0.8001 and an average training loss of 1.0830. The validation results showed an average accuracy of 0.7825 and an average validation loss of 1.3852.

```
Total Average Results:
Total Average Train Accuracy: 0.8001
Total Average Train Loss: 1.0830
Total Average Validation Accuracy: 0.7825
Total Average Validation Loss: 1.3852
Total Time Taken for Training: 6948.48 seconds
```

Figure 5.19: *Screenshot of the Proposed Model Character Embedding and Dropout with L2 Combined Regularization Training Results*

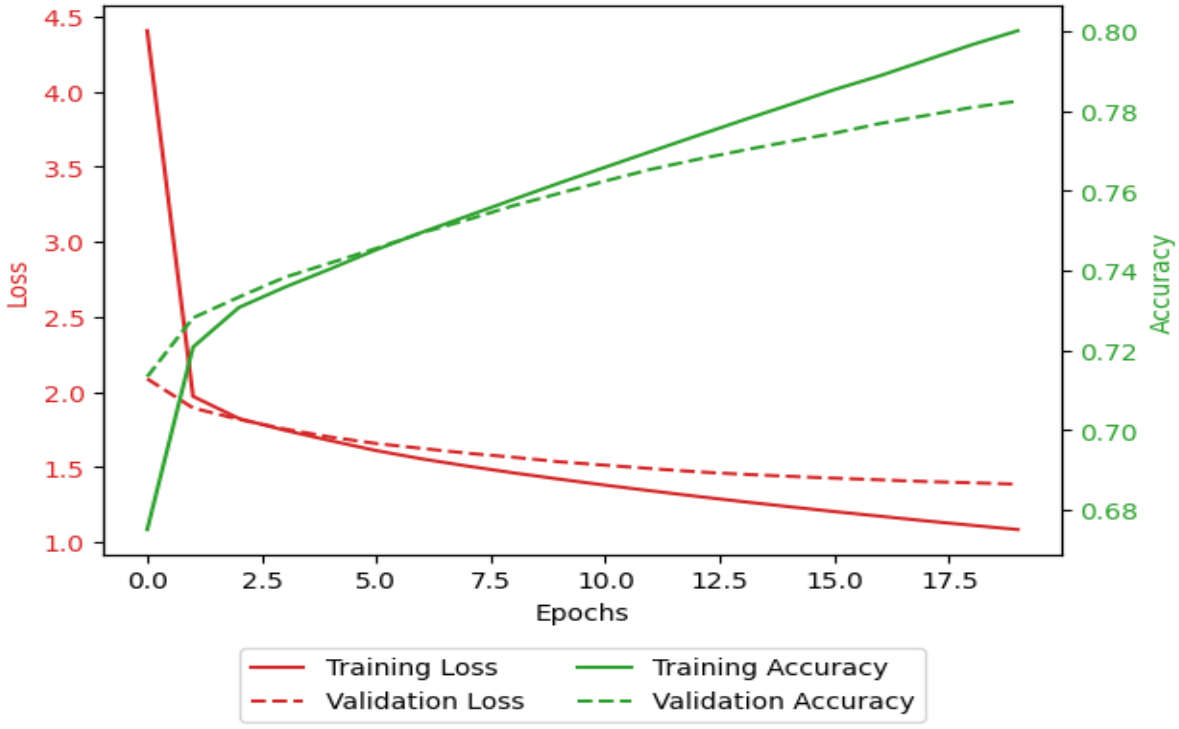

Figure 5.20: *Training and Validation Loss and Accuracy Curves for the proposed model with Character Embedding and Dropout with L2*

On the test dataset, the model achieved a test accuracy of 0.7838 and a test loss of 1.3713.

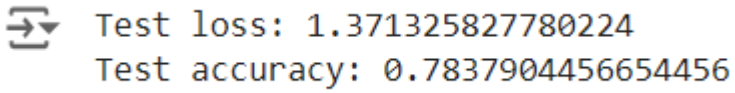

Figure 5.21: *Screenshot of the Proposed Model Character Embedding and Dropout with L2 Combined Regularization Evaluation Results on a Test Dataset*

Additionally, the average BLEU score, calculated from a subset of 3,000 pairs from the test dataset, was 40.59.

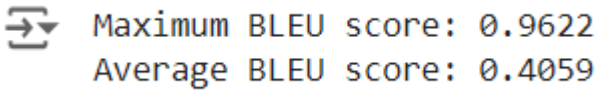

Figure 5.22: *Screenshot of the Maximum and Average BLEU Score Results of the Proposed Model with Character Embedding and Dropout with L2 Combined Regularization*

For the experiments, the baseline model with word embedding used 12 GB of GPU RAM. Preprocessing involved determining the maximum tokens for each language. A sequence length of 64 was chosen for both Amharic and English based on the 95th percentile of their token distributions, as mentioned in subsection 5.6.1, ensuring efficient memory usage while capturing most of the token sequences. This resulted in relatively modest memory consumption. In contrast, the proposed model with character embedding required 32 GB of GPU RAM due to the more complex character-level preprocessing, which was aligned with the respective token counts for each language.

Table 2.11: *Summary of the Proposed model Experimental Results*

Proposed model different approaches	Train Accuracy	Train Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss	BLEU
Character embedding & Dropout with Elastic Net	0.7901	1.1782	0.7788	1.4191	1.4121	0.7792	37.27
Character embedding & Dropout with L1	0.7885	1.1873	0.7771	1.4250	1.4110	0.7781	36.58
Character embedding & Dropout with L2	0.8001	1.0830	0.7825	1.3852	1.3713	0.7838	40.59

To measure overfitting reduction, we calculate the overfitting size for each model by examining both accuracy and loss differences between training and validation sets. Overfitting size is defined as the absolute difference between training and validation metrics, calculated as:

$$Overfittingsize(Accuracy) = AverageTrainingAccuracy - AverageValidationAccuracy \quad (19)$$

$$Overfittingsize(Loss) = AverageValidationLoss - AverageTrainingLoss \quad (20)$$

The baseline model shows an accuracy-based overfitting size of 0.6716 − 0.6558 = 0.0158 and a loss-based overfitting size of 2.3977 − 2.1488 = 0.2489 2. In comparison, the proposed model with character embedding, dropout, and Elastic Net regularization reduces the loss-based overfitting to 1.4191 − 1.1782 = 0.2409, achieving a reduction of 0.2489 − 0.2409 = 0.0080, or approximately 3.21%. The accuracy-based overfitting size for this model is 0.7901 − 0.7788 = 0.0113, representing a reduction of 0.0158 − 0.0113 = 0.0045, or around 28.48%.

The proposed model with character embedding, dropout, and L1 regularization also reduces overfitting, yielding an accuracy-based overfitting size of 0.7885 − 0.7771 = 0.0114, resulting in a reduction of 0.0158 − 0.0114 = 0.0044, or 27.85%. Its loss-based overfitting size is 1.4250 − 1.1873 = 0.2377, leading to a reduction of 0.2489 − 0.2377 = 0.0112, or approximately 4.50%.

In contrast, the proposed model with character embedding, dropout, and L2 regularization slightly increases overfitting compared to the baseline, with an accuracy-based overfitting size of 0.8001 − 0.7825 = 0.0176, representing an increase of 0.0176 − 0.0158 = 0.0018, or around 11.39%, and a loss-based overfitting size of 1.3852 − 1.0830 = 0.3022, leading to an increase of 0.3022 − 0.2489 = 0.0533, or approximately 21.41%.

Table 2.12*: Overfitting size comparison between the Baseline and Proposed model, highlighting reductions in Accuracy and Loss overfitting*

Model	Accuracy Overfitting Size	Reduction Compared to Baseline	Loss Overfitting Size	Reduction Compared to Baseline
Baseline	0.0158	–	0.2489	–
Dropout with Elastic Net	0.0113	0.0045 (28.48%)	0.2409	0.0080 (3.21%)
Dropout with L1	0.0114	0.0044 (27.85%)	0.2377	0.0112 (4.50%)
Dropout with L2	0.0176	+ 0.0018 (11.39%)	0.3022	+ 0.0533 (21.41%)

5.7 Discussion on the Experimental Results of the Study

Our experiments demonstrate that integrating character embeddings and combined regularization techniques within the point-wise FFN component of a Transformer-based model significantly enhances translation performance for low-resource Amharic-to-English NMT. This improvement is particularly noteworthy given the challenges posed by Amharic, a morphologically complex language. Traditional word embeddings often struggle to generalize across the many forms a word can take.

In contrast, character embeddings operate at a more granular level, capturing the internal structure of words, enabling the model to better understand and translate complex morphological patterns. This granular approach is reflected in the improved BLEU scores, increased test accuracy, and decreased test loss observed in our proposed model. Additionally, by capturing intricate inflectional and derivational morphology, character embeddings help mitigate issues related to OOV words, which are particularly problematic in morphologically rich languages.

We tested three combined regularization techniques in our model: dropout with L1, dropout with L2, and dropout with Elastic Net. Among these, the combination of conventional dropout with L2 regularization demonstrated superior performance across several metrics. Specifically, compared to the baseline model, which achieved a test accuracy of 68.33%, a test loss of 2.2087, and a BLEU score of 31.04, the dropout with L2 regularization variant led to a 14.71% increase in test accuracy, a 37.89% reduction in test loss, and a 30.77% improvement in BLEU score, reaching a BLEU score of 40.59.

When comparing dropout with L2 regularization to the other regularization techniques, we observed notable differences. The combination of dropout with Elastic net achieved a BLEU score of 37.27, reflecting a 9.02% improvement over the baseline, while dropout with L1 resulted in a BLEU score of 36.58, a 7.84% improvement over the baseline. Although both approaches also led to improvements in test accuracy and test loss over the baseline model, they did not exceed the performance of the dropout with L2 regularization combination. These results suggest that L2 regularization offers a more optimal balance of capturing complex linguistic patterns and maintaining model generalizability.

This performance can be attributed to L2 regularization's ability to balance fitting the training data well while maintaining generalizability. By discouraging large weight values through a penalty proportional to the square of the weights, L2 promotes a stable learning process and prevents any single feature from dominating the model's predictions. When combined with dropout, which randomly deactivates neurons during training, the model is less prone to overfitting because it does not rely too heavily on any single neuron.

In contrast, the combination of L1 regularization with dropout tends to produce sparse solutions by driving many weights to zero. While this can be useful for feature selection, it may be less effective for capturing the complex linguistic patterns in morphologically rich languages like Amharic, where a nuanced understanding of word structure is necessary.

The Elastic Net approach, which combines L1 and L2 penalties with dropout, achieved the second-best score in our experiment. This method retains some benefits of both L1 and L2 regularization, with dropout continuing to enhance model robustness. However, despite these advantages, Elastic Net with dropout may not offer the optimal trade-off for this specific NMT task compared to the more effective L2 with dropout combination.

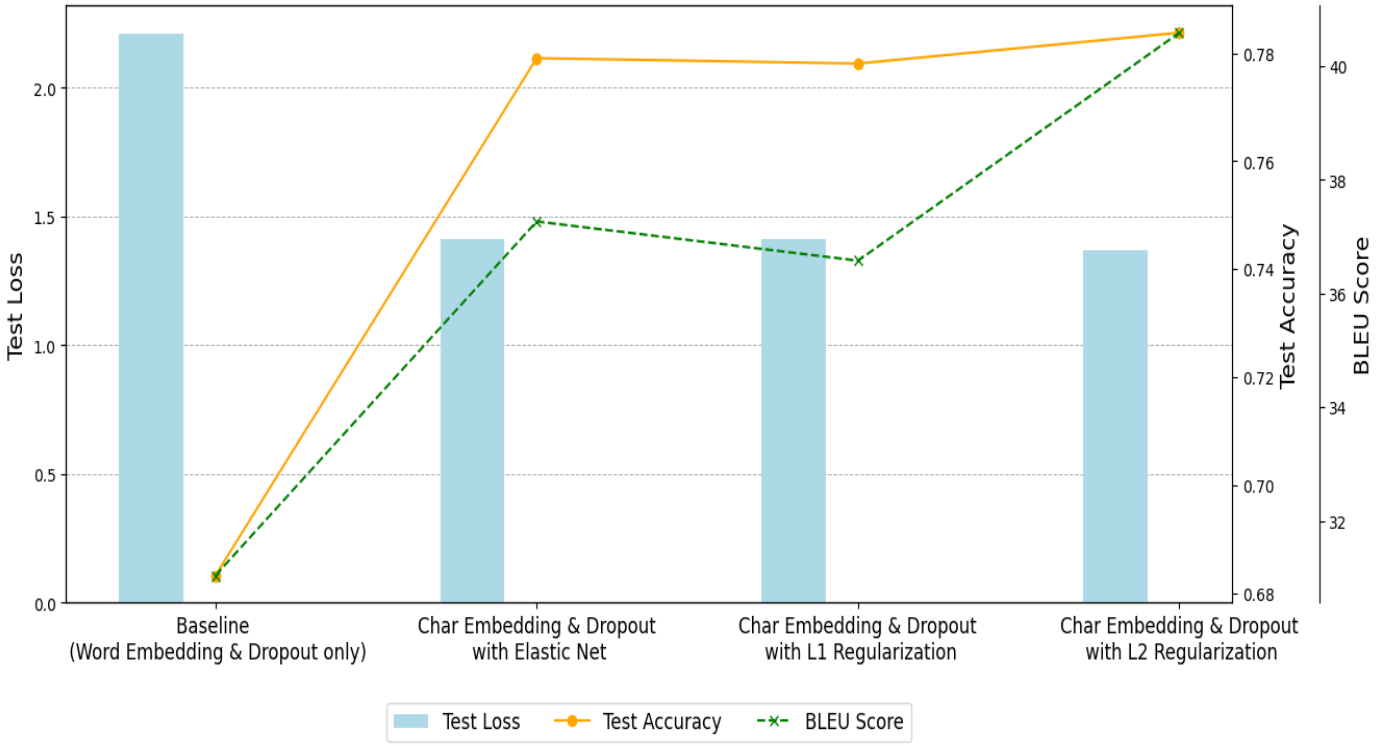

Figure 5.23: *Comparison of Test Loss, Test Accuracy, and BLEU Score Across the Baseline and Proposed Model with Character Embedding and Different Combined Regularization Approaches*

Our proposed model, which combines character embedding along with dropout and L2 regularization in the Point-wise FFN, achieved a BLEU score of 40.59, setting the state-of-the-art for Amharic-to-English NMT. It outperformed the morpheme-based model by Andargachew Gezmu and Nürnberger [26], which recorded a BLEU score of 25.3, by 60.40%. Additionally, it exceeded the transformer-large model by Andargachew Gezmu et al. [24], which achieved a BLEU score of 32.2, by 25.99%. It also outscored the Word-Piece-8K model [27], which had a BLEU score of 33.3, by 21.94%. Furthermore, it surpassed the previous state-of-the-art BLEU score of 37.79, reported by Tadesse Destaw Belay et al. [25], by 7.41%.

Our experiments reveal distinct patterns of overfitting across different regularization techniques. The baseline model shows accuracy-based overfitting of 0.0158 and loss-based overfitting of 0.2489. In comparison, the model with dropout and Elastic Net regularization, along with character embedding, reduces overfitting significantly, with a 3.21% decrease in loss-based overfitting and a 28.48% reduction in accuracy-based overfitting, indicating improved generalization. Similarly, the model with dropout and L1 regularization achieves a 27.85% reduction in accuracy-based overfitting and a 4.50% reduction in loss-based overfitting, enhancing model robustness. In

contrast, the model with dropout and L2 regularization slightly increases overfitting, with accuracy-based and loss-based overfitting values exceeding the baseline by 11.39% and 21.41%, respectively. Nonetheless, dropout with L2 regularization performs best overall, yielding superior training and validation accuracy, lower training and validation loss, as well as improved test accuracy and test loss compared to all other combined regularization methods, not just the baseline. These findings suggest that while dropout with L2 regularization excels in several metrics, Elastic Net and L1 regularization provide more effective control over overfitting in Amharic-to-English NMT, highlighting the importance of choosing the appropriate regularization technique for low-resource settings

Table 2.13: *BLEU score comparison between the proposed model and other benchmark models for Amharic-to-English NMT*

Model	BLEU
Andargachew Gezmu and Nürnberger [26]	25.3
Andargachew Gezmu et al. [24]	32.2
Andargachew Gezmu and Nürnberger [27]	33.3
Tadesse Destaw Belay et al. [25]	37.79
Our Baseline Transformer Model	31.04
Our Proposed Model (Character Embedding + Dropout + Elastic Net)	37.27
Our Proposed Model (Character Embedding + Dropout + L1 Regularization)	36.58
Our Proposed Model (Character Embedding + Dropout + L2 Regularization)	40.59

The increased GPU RAM requirement for the proposed model with character embedding highlights the trade-off between model complexity and computational resource demands. While the character-level approach offers potential improvements in handling morphological complexity and low-resource language translation, it also necessitates more substantial computational resources compared to the word-level approach.

The success of our proposed transformer-based model with character embedding and combined regularization techniques applied in the point-wise FFN for Amharic-to-English NMT suggests that this approach could also benefit other morphologically rich languages underserved by existing NMT solutions. By operating at the character level, our model better captures the complex morphological structures that often complicate NMT in these languages, enhancing generalization capabilities and reducing challenges related to OOV words.

Chapter 6

Conclusion and Future Work

6.1 Introduction

This chapter presents the conclusions of the research, summarizing the work done, addressing the research problems, and achieving the intended objectives. Additionally, the future works that go beyond the scope of this study are discussed, considering different perspectives.

6.2 Conclusions

In this research, we have made substantial progress in NMT for low-resource and morphologically complex languages, specifically focusing on Amharic-to-English translation. We achieved this by employing character-level embeddings and integrating various regularization techniques in a Transformer-based architecture, resulting in a model that surpasses previous benchmarks. The work addresses key challenges, particularly the rich morphological structure of Amharic and the limited availability of parallel data.

By working at the character level, the model effectively handles OOV words and generalizes well across diverse word forms. This granularity has proven crucial in processing the complex word formations typical of Amharic. The use of combined Dropout and L2 regularizations in the FFN has been instrumental in addressing overfitting, which is common in neural networks trained on small datasets. As a result, our model is both more accurate and more robust in translating Amharic into English.

Through extensive experimentation, we demonstrated that our approach sets a new standard for Amharic-to-English translation in low-resource environments. This improvement is not only quantitative but also reflects a qualitative enhancement in translation accuracy. Moreover, we achieved this efficiency by significantly reducing the model size, from over 75 million parameters to just over 5 million.

The major contributions of this thesis work include:

- Implementation of a character-level embedding approach within the Transformer model to address the challenges of morphologically complex languages, with a focus on Amharic.

- Exploration and integration of various regularization techniques, such as dropout, L1, L2, and Elastic Net, to mitigate overfitting in low-resource settings.
- A detailed analysis of how character embeddings, combined with regularization techniques, improve the generalization and translation quality of the Transformer model for morphologically complex languages like Amharic, particularly in low-resource settings such as Amharic-to-English translation

6.3 Future Work

There are several directions for future research that could extend the work presented here. These include:

- Multilingual Translation Systems: Building upon the character-level understanding developed in this study, future research could explore creating multilingual translation models that handle multiple low-resource languages simultaneously. This could lead to transfer learning benefits, where knowledge gained from one language improves the translation quality for others.
- Optimization of Attention Mechanisms: Future research could focus on refining attention mechanisms for character-level inputs, such as by integrating semantic context to better distinguish between Amharic words that are frequently misused, like "ጠበል" (correct) and "ፀበል" (incorrect). Such enhancements would enable the model to capture subtle character distinctions and accurate semantic relationships, potentially leading to more precise translations.
- Extending Combined Regularization Techniques: Combined regularization techniques could be extended to other components of the Transformer model, such as the attention mechanism and embedding layers. Applying these techniques across the entire architecture might improve the model's ability to generalize, reduce overfitting, and enhance translation accuracy.

Our thesis work represents a significant advancement in NMT for both low-resource and morphologically complex languages. The combination of character-level embeddings and advanced regularization techniques has proven effective for Amharic-to-English translation, and the insights gained could be applied to other challenging linguistic scenarios in future research.

References

[1] Chowdhary, K., and K. R. Chowdhary, “Natural language processing,” in Fundamentals of Artificial Intelligence, pp. 603-649, 2020

[2] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: State of the art, current trends and challenges," Multimedia Tools and Applications, vol. 82, no. 3, pp. 3713-3744, 2023

[3] A. Garg and M. Agarwal, “Machine translation: a literature review,” arXiv preprint arXiv:1901.01122, 2018

[4] S. Srivastava, A. Shukla, and R. Tiwari, “Machine translation: from statistical to modern deep-learning practices,” arXiv preprint arXiv:1812.04238, 2018

[5] A. Lopez, “Statistical machine translation,” ACM Computing Surveys (CSUR), vol. 40, no. 3, pp. 1-49, 2008

[6] P. Li, “A survey of machine translation methods,” TELKOMNIKA Indonesian Journal of Electrical Engineering, vol. 11, no. 12, pp. 7125-7130, 2013

[7] J. Zhang and C. Zong, “Deep Neural Networks in Machine Translation: An Overview,” IEEE Intell. Syst., 30(5), 16-25, 2015

[8] Z. Zong and C. Hong, “On application of natural language processing in machine translation,” in 2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE), pp. 506-510, IEEE, Sep. 2018

[9] R. Dabre, C. Chu, and A. Kunchukuttan, “A survey of multilingual neural machine translation,” ACM Computing Surveys (CSUR), vol. 53, no. 5, pp. 1-38, 2020

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, ... & I. Polosukhin, “Attention is all you need,” Advances in neural information processing systems, 30, 2017

[11] T. Tian, C. Song, J. Ting, and H. Huang, “A French-to-English machine translation model using transformer network,” Procedia Computer Science, vol. 199, pp. 1438-1443, 2022

[12] S. M. Lakew, M. Cettolo, and M. Federico, “A comparison of transformer and recurrent neural networks on multilingual neural machine translation,” arXiv preprint arXiv:1806.06957, 2018

[13] A. T. Hadgu, A. Beaudoin, and A. Aregawi, “Evaluating Amharic Machine Translation,” arXiv preprint arXiv: 2003.14386, Mar. 2020

[14] A. Jain, A. Banerjee, and P. Bhattacharyya, “Literature survey: Neural machine translation in low resource setting,” 2021.

[15] A. Al-ani, “A Hybrid Neural Machine Translation Technique for Translating Low Resource Languages,” in Machine Learning and Data Mining in Pattern Recognition, 2019.

[16] H. Khayrallah, B. Thompson, K. Duh, and P. Koehn, “Regularized training objective for continued training for domain adaptation in neural machine translation,” in Proc. 2nd Workshop on Neural Machine Translation and Generation, pp. 36-44, Jul. 2018.

[17] Z. Farhadi, H. Bevrani, and M. R. Feizi-Derakhshi, “Combining regularization and dropout techniques for deep convolutional neural network,” in 2022 Global Energy Conference (GEC), pp. 335-339, Oct. 2022.

[18] À. R. Atrio and A. Popescu-Belis, “On the interaction of regularization factors in low-resource neural machine translation,” in Proc. 23rd Annual Conf. European Association for Machine Translation, 1-3 Jun. 2022.

[19] A. Araabi and C. Monz, “Optimizing transformer for low-resource neural machine translation,” arXiv:2011.02266, 2020.

[20] R. Li et al., “Smart and rapid design of nanophotonic structures by an adaptive and regularized deep neural network,” Nanomaterials, vol. 12, no. 8, p. 1372, 2022.

[21] M. Abate and Y. Assabie, “Development of Amharic morphological analyzer using memory-based learning,” in Advances in Natural Language Processing: 9th International Conference on NLP, PolTAL 2014, Warsaw, Poland, Sep. 17-19, 2014, pp. 1-13. Springer International Publishing.

[22] Nega Alemayehu and Peter Willett, “Stemming of Amharic Words for Information Retrieval,” Literary and Linguistic computing, 17(1): 1-17, 2002.

[23] B. M. Hailu, Y. Assabie, and Y. B. Sinshaw, “Semantic role labeling for Amharic text using multiple embeddings and deep neural network,” IEEE Access, vol. 11, pp. 33274-33295, 2023.

[24] A. M. Gezmu, A. Nürnberger, and T. B. Bati, “Neural Machine Translation for Amharic-English Translation,” in ICAART (1), pp. 526-532, 2021.

[25] T. D. Belay et al., “The effect of normalization for bi-directional Amharic-English neural machine translation,” in 2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA), pp. 84-89, Nov. 2022.

[26] A. M. Gezmu and A. Nürnberger, “Transformers for Low-resource Neural Machine Translation,” in ICAART (1), pp. 459-466, 2022.

[27] A. M. Gezmu and A. Nürnberger, “Morpheme-Based Neural Machine Translation Models for Low-Resource Fusion Languages,” ACM Transactions on Asian and Low-Resource Language Information Processing, vol. 22, no. 9, pp. 1-19, 2023.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] A. Fan, E. Grave, and A. Joulin, “Reducing transformer depth on demand with structured dropout,” arXiv:1909.11556, 2019.

[30] W. Zhou, T. Ge, K. Xu, F. Wei, and M. Zhou, “Scheduled drophead: A regularization method for transformer models,” arXiv:2004.13342, 2020.

[31] Z. Wu, L. Wu, Q. Meng, Y. Xia, S. Xie, T. Qin, and T. Y. Liu, “Unidrop: A simple yet effective technique to improve transformer without extra cost," arXiv:2104.04946, 2021.

[32] S. Papay, S. Padó, and N. T. Vu, “Addressing low-resource scenarios with character-aware embeddings,” in Proc. Second Workshop on Subword/Character Level Models, pp. 32-37, Jun. 2018.

[33] A. Renduchintala, P. Shapiro, K. Duh, and P. Koehn, “Character-aware decoder for translation into morphologically rich languages,” arXiv:1809.02223, 2018.

[34] M. Labeau and A. Allauzen, “Character and subword-based word representation for neural language modeling prediction,” in Proc. First Workshop on Subword and Character Level Models in NLP, pp. 1-13, Sep. 2017.

[35] S. Cao and W. Lu, “Improving word embeddings with convolutional feature learning and subword information,” in Proc. AAAI Conf. on Artificial Intelligence, vol. 31, no. 1, Feb. 2017.

[36] A. Renduchintala, P. Shapiro, K. Duh, and P. Koehn, “Character-aware decoder for translation into morphologically rich languages,” arXiv:1809.02223, 2018.

[37] M. B. Kazimi and M. Ruiz Costa-Jussà, “Coverage for character based neural machine translation,” 2017.

[38] Y. A. Ashengo, R. T. Aga, and S. L. Abebe, “Context based machine translation with recurrent neural network for English–Amharic translation,” Machine Translation, vol. 35, no. 1, pp. 19-36, 2021.

[39] Tadesse Kassa, “Morpheme-Based Bi-directional Ge’ez Amharic Machine Translation”, unpublished master’s thesis, Addis Ababa University, Ethiopia, 2018.

[40] A. Temesgen and Y. Assabie, “Development of Amharic grammar checker using morphological features of words and N-gram based probabilistic methods,” in Proceedings of the 13th International Conference on Parsing Technologies (IWPT 2013), Nov. 2013, pp. 106-112.

[41] A. Ibrahim and Y. Assabie, “Amharic sentence parsing using base phrase chunking,” in Computational Linguistics and Intelligent Text Processing: 15th International Conference, CICLing 2014, Kathmandu, Nepal, April 6-12, 2014, Proceedings, Part I, vol. 15, pp. 297-306, Springer, Berlin, Heidelberg, 2014.

[42] **ተባባሪ ፕሮፌሰር ጌታሁን አማረ, የአማርኛ ሰዋስው በቀላል አቀራረብ**, 1989

[43] **ብላታ መርስዔ ኀዘን ወልደቂርቆስ, ያማርኛ ሰዋሰው** 1948

[44] **ባዬ ይማም፣ የአማርኛ ሰዋስው**, Addis Ababa, Ethiopia: EMPDA Publications, 1995.

[45] C.H. DAWKINS, The fundamentals of Amharic, SIM Publishing, 1969

[46] **ጌታሁን አማረ፣ ዘመናዊ የአማርኛ ሰዋስው በቀላል አቀራረብ**, Addis Ababa, Ethiopia, 2010

[47] Biruk Abel, “Geez to Amharic Machine Translation using Hybrid approach”, unpublished master’s thesis, Addis Ababa University, Ethiopia, 2018.

[48] Workineh Wogaso, “Attention-based Amharic-to-Wolaita Neural Machine Translation”, unpublished master’s thesis, Addis Ababa University, Ethiopia, 2020.

[49] Abeba Ibrahim, “A Hybrid Approach to Amharic Base Phrase Chunking and Parsing”, unpublished master’s thesis, Addis Ababa University, Ethiopia, 2013.

[50] Eleni Teshome, "Bidirectional English-Amharic Machine Translation: An Experiment using Constrained Corpus", Unpublished Master’s Thesis Department of Computer Science, Addis Ababa University, 2013

[51] B. A. Agajie, “Syntactic Object Representation of Amharic Sentences by Function,” IJOTL-TL: Indonesian Journal of Language Teaching and Linguistics, vol. 5, no. 2, pp. 65-80, 2020.

[52] Yashika, “English Language Overview: A Review Article,” International Journal of Innovative Research in Technology, vol. 9, no. 11, pp. 474-476, 2023

[53] S. Greenbaum, The Oxford English Grammar. Oxford University Press, 1996.

[54] M. Aronoff and K. Fudeman, what is morphology? John Wiley & Sons, 2022.

[55] F. E. S. Rahayu and F. Eka, Introduction to English Morphology. Samarinda: Repository Universitas Mulawarman, 2021.

[56] L. Bauer, Introducing Linguistic Morphology. Edinburgh University Press, 2003.

[57] I. Plag, Word-formation in English. Cambridge University Press, 2018.

[58] G. Booij, “Compounding and derivation,” in Morphology and Its Demarcations, pp. 109-132, 2005.

[59] R. Lieber, Introducing Morphology. Cambridge University Press, 2021.

[60] M. Kolln and R. Funk, Understanding English Grammar. Longman, 2012.

[61] H. Wang, H. Wu, Z. He, L. Huang, and K. W. Church, "Progress in machine translation," Engineering, vol. 18, pp. 143-153, 2022.

[62] J. Hutchins and E. Lovtskii, “Petr Petrovich Troyanskii (1894–1950): A forgotten pioneer of mechanical translation,” Machine Translation, vol. 15, no. 3, pp. 187-221, 2000.

[63] Warren Weaver, Translation. Machine translation of languages. 14:15–23, 1955.

[64] P. Wellburn, “The Routledge Encyclopedia of Translation Technology,” Reference Reviews, vol. 29, no. 5, pp. 30-31, 2015.

[65] M. D. Okpor, “Machine translation approaches: issues and challenges,” *International Journal of Computer Science Issues (IJCSI)*, vol. 11, no. 5, p. 159, 2014.

[66] J. Lu and F. Yin, “Research on improving the quality of Japanese Chinese machine translation based on deep learning,” in DSIE, pp. 267-277, 2023.

[67] J. DeNero, S. Kumar, C. Chelba, and F. J. Och, “Model combination for machine translation,” in Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Jun. 2010, pp. 975-983.

[68] L. Stoimenov, E. Kajan, and S. Djordjevic-Kajan, “Ontology-driven semantic interoperability,” International Review of Computers and Software, vol. 1, no. 2, pp. 132-136, 2006.

[69] M. Nagao, “A framework of a mechanical translation between Japanese and English by analogy principle,” Artificial and Human Intelligence, pp. 351-354, 1984.

[70] S. Tripathi and J. K. Sarkhel, “Approaches to machine translation,” 2010.

[71] J. Oladosu, A. Esan, I. Adeyanju, B. Adegoke, O. Olaniyan, and B. Omodunbi, “Approaches to machine translation: a review,” FUOYE Journal of Engineering and Technology, vol. 1, no. 1, 2016

[72] P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. J. Jelinek, J. Lafferty, and P. S. Roossin, “A statistical approach to machine translation,” Computational Linguistics, vol. 16, no. 2, pp. 79-85, 1990.

[73] S. Vogel, F. J. Och, C. Tillmann, S. Nießen, H. Sawaf, and H. Ney, “Statistical methods for machine translation,” in Verbmobil: Foundations of Speech-to-Speech Translation, pp. 377-393, 2000.

[74] D. Marcu and D. Wong, “A phrase-based, joint probability model for statistical machine translation,” in Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002), Jul. 2002, pp. 133-139.

[75] F. J. Och and H. Ney, “The alignment template approach to statistical machine translation,” Computational Linguistics, vol. 30, no. 4, pp. 417-449, 2004.

[76] A. Ahmed and G. Hanneman, “Syntax-based statistical machine translation: A review,” Computational Linguistics, 2005.

[77] K. Yamada and K. Knight, “A syntax-based statistical translation model,” in Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, Jul. 2001, pp. 523-530.

[78] M. Galley et al., “Scalable inference and training of context-rich syntactic translation models,” in Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Jul. 2006, pp. 961-968.

[79] Y. Liu, Q. Liu, and S. Lin, “Tree-to-string alignment template for statistical machine translation,” in Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Jul. 2006, pp. 609-616.

[80] J. Graehl, K. Knight, and J. May, “Training tree transducers,” Computational Linguistics, vol. 34, no. 3, pp. 391-427, 2008.

[81] F. J. Och, “Minimum error rate training in statistical machine translation,” in Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, Jul. 2003, pp. 160-167.

[82] F. J. Och and H. Ney, “Improved statistical alignment models,” in Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics, Oct. 2000, pp. 440-447.

[83] A. Burbank, M. Carpuat, S. Clark, M. Dreyer, P. Fox, D. Groves, and D. Wu, “Final report of the 2005 language engineering workshop on statistical machine translation by parsing,” Johns Hopkins University, 2005.

[84] P. Koehn, F. J. Och, and D. Marcu, “Statistical phrase-based translation,” in 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL 2003), pp. 48-54, 2003.

[85] F. J. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, and D. Radev, “A smorgasbord of features for statistical machine translation,” in Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004, pp. 161-168, 2004.

[86] M. Collins, P. Koehn, and I. Kučerová, “Clause restructuring for statistical machine translation,” in Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL’05), Jun. 2005, pp. 531-540.

[87] C. Wang, M. Collins, and P. Koehn, “Chinese syntactic reordering for statistical machine translation,” in Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Jun. 2007, pp. 737-745.

[88] M. Simard, N. Cancedda, B. Cavestro, M. Dymetman, E. Gaussier, C. Goutte, and A. Mauser, “Translating with non-contiguous phrases,” in Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Oct. 2005, pp. 755-762.

[89] R. E. Frederking and K. B. Taylor, Eds., Machine Translation: From Real Users to Research: 6th Conference of the Association for Machine Translation in the Americas, AMTA 2004, Washington, DC, USA, September 28-October 2, 2004, Proceedings, vol. 3265. Springer, 2004.

[90] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, and E. Herbst, “Moses: Open source toolkit for statistical machine translation,” in Proceedings of the 45th

Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions, Jun. 2007, pp. 177-180.

[91] S. Sharma, M. Diwakar, P. Singh, A. Tripathi, C. Arya, and S. Singh, “A review of neural machine translation based on deep learning techniques,” in 2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Nov. 2021, pp. 1-5.

[92] S. A. Mohamed, A. A. Elsayed, Y. F. Hassan, and M. A. Abdou, “Neural machine translation: past, present, and future,” Neural Computing and Applications, vol. 33, pp. 15919-15931, 2021.

[93] Y. Wu, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” arXiv preprint arXiv:1609.08144, 2016.

[94] M. T. Luong, "Effective approaches to attention-based neural machine translation," arXiv preprint arXiv:1508.04025, 2015.

[95] Z. El Maazouzi, B. E. El Mohajir, and M. Al Achhab, “A technical reading in statistical and neural machines translation (SMT & NMT),” in 2017 8th International Conference on Information Technology (ICIT), May 2017, pp. 157-165.

[96] Z. Niu, G. Zhong, and H. Yu, “A review on the attention mechanism of deep learning,” Neurocomputing, vol. 452, pp. 48-62, 2021.

[97] D. Datta, P. E. David, D. Mittal, and A. Jain, “Neural machine translation using recurrent neural network,” International Journal of Engineering and Advanced Technology, vol. 9, no. 4, pp. 1395-1400, 2020.

[98] I. Sutskever, “Sequence to Sequence Learning with Neural Networks,” arXiv preprint arXiv:1409.3215, 2014.

[99] D. Bahdanau et al., “Neural machine translation by jointly learning to align and translate,” arXiv preprint arXiv:1409.0473, 2014.

[100] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, “Convolutional sequence to sequence learning,” in International Conference on Machine Learning, Jul. 2017, pp. 1243-1252.

[101] S. Hochreiter, “Long Short-term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[102] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” arXiv preprint arXiv:1406.1078, 2014.

[103] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” arXiv preprint arXiv:1412.3555, 2014.

[104] K. Cho, “On the Properties of Neural Machine Translation: Encoder-decoder Approaches,” arXiv preprint arXiv:1409.1259, 2014.

[105] F. Stahlberg, “Neural Machine Translation: A Review and Survey,” arXiv preprint arXiv:1912.02047, 2019.

[106] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673-2681, Nov. 1997.

[107] Y. Jia, “Attention mechanism in machine translation,” in Journal of Physics: Conference Series, vol. 1314, no. 1, p. 012186, Oct. 2019. IOP Publishing.

[108] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in International Conference on Machine Learning, pp. 2048-2057, Jun. 2015.

[109] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” Machine Learning, vol. 8, pp. 229-256, 1992.

[110] N. Kalchbrenner and P. Blunsom, “Recurrent continuous translation models,” in Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1700-1709, Oct. 2013.

[111] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, “A decomposable attention model for natural language inference,” arXiv preprint arXiv:1606.01933, 2016.

[112] J. Devlin, “Bert: Pre-training of deep bidirectional transformers for language understanding,” arXiv preprint arXiv:1810.04805, 2018.

[113] Y. Sun, S. Wang, Y. Li, S. Feng, H. Tian, H. Wu, and H. Wang, “Ernie 2.0: A continual pre-training framework for language understanding,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 05, pp. 8968-8975, Apr. 2020.

[114] G. Zhao, J. Lin, Z. Zhang, X. Ren, Q. Su, and X. Sun, “Explicit sparse transformer: Concentrated attention through explicit selection,” arXiv preprint arXiv:1912.11637, 2019.

[115] H. Sawaf, B. Gaskill, and M. Veronis, “Hybrid machine translation applied to media monitoring,” in Proceedings of the 8th Conference of the Association for Machine Translation in the Americas: Government and Commercial Uses of MT, pp. 440-447, 2008.

[116] M. R. Costa-jussà, R. Rapp, P. Lambert, K. Eberle, R. E. Banchs, and B. Babych, Eds., Hybrid Approaches to Machine Translation. Springer, 2016.

[117] J. Daba and Y. Assabie, “A hybrid approach to the development of bidirectional English-Oromiffa machine translation,” in Advances in Natural Language Processing: 9th International Conference on NLP, PolTAL 2014, Warsaw, Poland, September 17-19, 2014. Proceedings 9, pp. 228-235, Springer International Publishing, 2014.

[118] M. R. Costa-Jussa and J. A. Fonollosa, “Latest trends in hybrid machine translation and its applications,” Computer Speech & Language, vol. 32, no. 1, pp. 3-10, 2015.

[119] C. España Bonet, L. Màrquez Villodre, G. Labaka, A. Díaz de Ilarraza Sánchez, and K. Sarasola Gabiola, “Hybrid machine translation guided by a rule-based system,” in Machine Translation Summit XIII: Proceedings of the 13th Machine Translation Summit, September 19-23, 2011, Xiamen, China, pp. 554-561, 2011.

[120] C. Federmann, “Hybrid machine translation using joint, binarised feature vectors,” in Proceedings of the 10th Conference of the Association for Machine Translation in the Americas: Research Papers, 2012.

[121] J. Dugonik, M. Sepesy Maučec, D. Verber, and J. Brest, “Reduction of neural machine translation failures by incorporating statistical machine translation,” Mathematics, vol. 11, no. 11, p. 2484, 2023.

[122] B. E. Batsukh, "Hierarchical triple model of hybrid neural machine translation," International Journal of Applied Linguistics and English Literature, vol. 11, no. 2, pp. 38-42, 2022.

[123] V. Khenglawt, “Machine translation and its approaches,” in Mizoram Science Congress 2018 (MSC 2018), pp. 141-145, Dec. 2018. Atlantis Press.

[124] H. Lu, H. Huang, D. Zhang, F. Wei, and W. Lam, “Revamping multilingual agreement bidirectionally via switched back-translation for multilingual neural machine translation,” in Findings of the Association for Computational Linguistics: EACL 2024, pp. 264-275, Mar. 2024.

[125] K. D. Garg, V. M. Sood, S. K. Narang, and R. Bhandari, “Bidirectional machine translation for Punjabi-English, Punjabi-Hindi, and Hindi-English language pairs,” in Proceedings of International Conference on Recent Innovations in Computing: ICRIC 2022, Volume 1, pp. 621-632, May 2023. Singapore: Springer Nature Singapore.

[126] T. R. Chiang, Y. P. Chen, Y. T. Yeh, and G. Neubig, “Breaking down multilingual machine translation,” arXiv preprint arXiv:2110.08130, 2021.

[127] E. Chatzikoumi, “How to evaluate machine translation: A review of automated and human metrics,” Natural Language Engineering, vol. 26, no. 2, pp. 137-161, 2020.

[128] G. S. K. KSU, P. Fields, D. H. BYU, A. Lommel, and A. Melby, “Defining translation quality.”

[129] M. S. Maučec and G. Donaj, “Machine translation and the evaluation of its quality,” in Recent Trends in Computational Intelligence, vol. 143, 2019.

[130] B. J. Dorr, J. Olive, J. McCary, and C. Christanson, “Chapter 5: Machine translation evaluation and optimization,” in Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation, Springer, New York, pp. 745-843, 2011.

[131] S. K. Mondal, H. Zhang, H. D. Kabir, K. Ni, and H. N. Dai, “Machine translation and its evaluation: A study,” Artificial Intelligence Review, vol. 56, no. 9, pp. 10137-10226, 2023.

[132] M. B. Bakaric, K. Tonkovic, and L. N. Prskalo, “Clash between segment-level MT error analysis and selected lexical similarity metrics,” International Journal of Advanced Computer Science and Applications, vol. 11, no. 5, 2020.

[133] A. L. F. Han, D. F. Wong, and L. S. Chao, “Machine translation evaluation: A survey,” arXiv preprint arXiv:1605.04515, 2016.

[134] S. Lee, J. Lee, H. Moon, C. Park, J. Seo, S. Eo, and H. Lim, “A survey on evaluation metrics for machine translation,” Mathematics, vol. 11, no. 4, p. 1006, 2023.

[135] S. Chauhan and P. Daniel, “A comprehensive survey on various fully automatic machine translation evaluation metrics,” Neural Processing Letters, vol. 55, no. 9, pp. 12663-12717, 2023.

[136] K. Y. Su, M. W. Wu, and J. S. Chang, “A new quantitative quality measure for machine translation systems,” in COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics, 1992.

[137] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 311-318, Jul. 2002.

[138] G. Doddington, “Automatic evaluation of machine translation quality using n-gram co-occurrence statistics,” in Proceedings of the Second International Conference on Human Language Technology Research, pp. 138-145, Mar. 2002.

[139] J. Turian, L. Shen, and I. D. Melamed, “Evaluation of machine translation and its evaluation,” in Proceedings of Machine Translation Summit IX: Papers, 2003.

[140] S. Banerjee and A. Lavie, “METEOR: An automatic metric for MT evaluation with improved correlation with human judgments,” in Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, pp. 65-72, Jun. 2005.

[141] A. Agarwal and A. Lavie, “METEOR, m-BLEU and m-TER: Evaluation metrics for high-correlation with human rankings of machine translation output,” in Proceedings of the Third Workshop on Statistical Machine Translation, pp. 115-118, Jun. 2008.

[142] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[143] C. M. Bishop, Pattern Recognition and Machine Learning. New York, NY, USA: Springer, 2006.

[144] G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets,” Neural Computation, vol. 18, no. 7, pp. 1527–1554, 2006.

[145] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction. New York, NY, USA: Springer, 2009.

[146] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, pp. 436–444, May 2015.

[147] F. Chollet, Deep Learning with Python. Shelter Island, NY, USA: Manning Publications, 2017.

[148] M. Mohri, A. Rostamizadeh, and A. Talwalkar, Foundations of Machine Learning, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[149] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 770–778.

[150] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning,” in Proc. 33rd Int. Conf. Mach. Learn. (ICML), 2016, pp. 1050–1059.

[151] R. Tibshirani, “Regression shrinkage and selection via the lasso,” J. Roy. Stat. Soc. B, vol. 58, no. 1, pp. 267–288, 1996.

[152] A. Y. Ng, “Feature selection, L1 vs. L2 regularization, and rotational invariance,” in Proc. 21st Int. Conf. Mach. Learn. (ICML), 2004, pp. 78–85.

[153] A. E. Hoerl and R. W. Kennard, “Ridge regression: Biased estimation for nonorthogonal problems,” Technometrics, vol. 12, no. 1, pp. 55-67, 1970.

[154] A. Ng, M. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in Proc. 15th Int. Conf. Neural Inf. Process. Syst. (NIPS), 2001, pp. 849–856.

[155] T. Zhang, “Solving large scale linear prediction problems using stochastic gradient descent algorithms,” in Proc. 21st Int. Conf. Mach. Learn. (ICML), 2004, pp. 919–926.

[156] R. A. Willoughby, “Solutions of ill-posed problems (AN Tikhonov and VY Arsenin),” Siam Review, vol. 21, no. 2, p. 266, 1979.

[157] J. Friedman, T. Hastie, and R. Tibshirani, “A note on the group lasso and a sparse group lasso,” J. Roy. Stat. Soc. B, vol. 73, no. 2, pp. 141–161, Jan. 2011.

[158] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” J. Roy. Stat. Soc. B, vol. 67, no. 2, pp. 301–320, Apr. 2005.

[159] A. R. Ribeiro, L. A. Amaral, and L. A. N. de Azevedo, “Regularization in machine learning: A review,” in Advances in Machine Learning, vol. 2, I. Arora, Ed. Cambridge, UK: Cambridge University Press, 2017, pp. 1–34.

[160] J. Y. Lee and G. E. Hinton, “Regularization techniques for neural networks,” in Machine Learning: A Probabilistic Perspective, M. I. Jordan, Ed. Cambridge, MA, USA: MIT Press, 2012, pp. 611–622.

[161] C. D. Sutton and A. McCallum, An Introduction to Conditional Random Fields. Cambridge, MA, USA: MIT Press, 2012.

[162] G. W. Taylor and R. Hinton, “A comparison of dropout and dropconnect for regularization of deep neural networks,” in Proc. 25th Int. Conf. Machine Learning (ICML), 2016, pp. 2115–2123.

[163] J. M. R. Williams and D. G. Hinton, “Training recurrent neural networks,” in Neural Networks for Pattern Recognition, C. M. Bishop, Ed. Oxford, UK: Oxford University Press, 1995, pp. 127–146.

[164] H. A. Simon and G. W. Richardson, “Generalization of neural network architecture for learning,” IEEE Trans. Neural Networks, vol. 22, no. 8, pp. 1248–1258, Aug. 2011.

[165] X. Chen, X. Xu, and X. Hu, “A regularization technique for deep neural networks based on dropout and dropout training,” in IEEE Trans. Neural Networks Learn. Syst., vol. 25, no. 2, pp. 248–258, Feb. 2014.

[166] M. S. M. A. Al-Amin, F. S. H. Khan, and H. C. Lam, “Regularization of deep neural networks: A survey,” in Advances in Neural Information Processing Systems (NeurIPS), vol. 32, 2019, pp. 3155–3166.

[167] L. Song, W. Choi, and J. Y. Lee, “Regularization for deep learning in neural networks,” in Machine Learning for Deep Neural Networks, T. F. L. Nguyen, Ed. Cambridge, MA, USA: MIT Press, 2018, pp. 229–250.

[168] T. F. Smith and J. G. Goll, “Analysis of dropout and other regularization methods in deep learning,” in Proc. 34th Int. Conf. Machine Learning (ICML), 2017, pp. 295–304.

[169] A. G. Kim and R. E. Prasad, “Regularization methods for low-resource languages in neural networks,” in IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI), vol. 42, no. 3, pp. 345–356, Mar. 2020.

[170] Y. Zhang, S. Ren, and R. Xu, “Regularization strategies for low-resource language models,” in Proc. 2021 Int. Conf. Computational Linguistics (COLING), 2021, pp. 2897–2906.

[171] N. Banar, W. Daelemans, and M. Kestemont, “Character-level transformer-based neural machine translation,” in Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval, pp. 149-156, 2020.

[172] L. H. Baniata, I. K. Ampomah, and S. Park, “A transformer-based neural machine translation model for Arabic dialects that utilizes subword units,” Sensors, vol. 21, no. 19, p. 6509, 2021.

[173] M. K. Negia, R. M. Tamiru, and M. Meshesha, “Amharic-Kistanigna Bi-directional Machine Translation using Deep Learning,” in 2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA), Nov. 2022, pp. 61-65. IEEE.

[174] D. Variš and O. Bojar, “Unsupervised pretraining for neural machine translation using elastic weight consolidation,” arXiv preprint arXiv:2010.09403, 2020.

[175] P. Gao, L. Zhang, Z. He, H. Wu, and H. Wang, “Improving zero-shot multilingual neural machine translation by leveraging cross-lingual consistency regularization,” arXiv preprint arXiv:2305.07310, 2023.

[176] M. Yang, M. K. Lim, Y. Qu, X. Li, and D. Ni, “Deep neural networks with L1 and L2 regularization for high dimensional corporate credit risk prediction,” Expert Systems with Applications, vol. 213, p. 118873, 2023.

Annexes

Annex I - Sample Source Code

Initialization of Required Packages and Dependencies:

```
!pip install tensorflow==2.15.1
import pathlib
import random
import string
import re
import os
import time
import tensorflow as tf
import numpy as np
import unicodedata
from collections import Counter
from tensorflow.keras import layers, models
from tensorflow.keras.layers import TextVectorization
from tensorflow.keras.optimizers.schedules import PolynomialDecay
from tensorflow.keras.optimizers import Adam
import pandas as pd
import matplotlib.pyplot as plt
from nltk.translate.bleu_score import sentence_bleu, SmoothingFunction
from google.colab import drive
from sklearn.model_selection import train_test_split

# Mount Google Drive
drive.mount('/content/drive')
```

Preprocessing Functions for Amharic and English Text:

```python
def remove_special_chars(text):
    return re.sub(r'[^a-zA-Z0-9\s.,;:!?\'"()\[\]{}\\/-]', '', text) #remove any character that is not a letter, digit, whitespace, punctuation, bracket, or slash
def handle_unicode(text):
    return unicodedata.normalize('NFKD', text).encode('ascii', 'ignore').decode('utf-8')
def expand_contractions(text):
    contractions = {
        "n't": " not",
        "'s": [" is", " was"],  # Can be either "is" or "was"
        "'m": " am",
        "'re": " are",
        "'ll": " will",
        "'ve": " have",
        "'d": [" would", " had"]  # Can be either "would" or "had"
    }
    for contraction, expansion in contractions.items():
        if isinstance(expansion, list):
            for exp in expansion:
                text = text.replace(contraction, exp)
        else:
            text = text.replace(contraction, expansion)
    return text
def remove_excess_whitespace(text):
    return ' '.join(text.split())
# Preprocess Amharic sentences
def preprocess_amharic(text):
    # Remove any non-Amharic characters (keeping spaces, Amharic/Arabic numbers, and Amharic punctuation)
    text = re.sub(r'[^\u1200-\u137F\u1361-\u1368\u1369-\u1371\u0030-\u0039\s\[\]{}\\/-]', '', text)
     # \u1200-\u137F matches Amharic characters/Scripts.
     # \u1361-\u1368 matches Amharic punctuation marks.
     # \u1369-\u1371 matches Amharic numbers.
     # \u0030-\u0039 matches Arabic digits.
     # Remove excess whitespac

    # Remove excess whitespace
    text = remove_excess_whitespace(text)
    return list(text)  # Return a list of characters

# Preprocess English sentences
def preprocess_english(text):
    text = handle_unicode(text)
    text = expand_contractions(text)
    text = remove_special_chars(text)
```

Preprocessing Amharic and English Sentences and Calculating Sentence Lengths:

```
# Preprocess sentences and calculate lengths
am_lengths = []
en_lengths = []
text_pairs = []
for am_sentence, en_sentence in zip(amharic_sentences, english_sentences):
    am_processed = preprocess_amharic(am_sentence)
    en_processed = preprocess_english(en_sentence)
    am_length = len(am_processed)  # list of Amharic characters
    en_length = len(en_processed)  # list of English characters

    # Filter out zero-length sentences
    if am_length > 0 and en_length > 0:
        am_lengths.append(am_length)
        en_lengths.append(en_length)
        text_pairs.append((''.join(am_processed), ''.join(en_processed)))  # Join characters back into a string
# Calculate min and max lengths for each language
am_min_length = min(am_lengths)
am_max_length = max(am_lengths)
en_min_length = min(en_lengths)
en_max_length = max(en_lengths)

# Calculate overall min and max lengths
min_length = min(am_min_length, en_min_length)
max_length = max(am_max_length, en_max_length)

print(f"Amharic minimum sentence length: {am_min_length}")
print(f"Amharic maximum sentence length: {am_max_length}")
print(f"English minimum sentence length: {en_min_length}")
print(f"English maximum sentence length: {en_max_length}")
print(f"Overall minimum sentence length: {min_length}")
print(f"Overall maximum sentence length: {max_length}")
```

Filtering, Stratifying, and Splitting Sentence Pairs into Train, Validation, and Test Sets:

```
# Filter sentences based on length
filtered_pairs = []
filtered_am_lengths = []
filtered_en_lengths = []

for (am_sentence, en_sentence), am_length, en_length in zip(text_pairs, am_lengths, en_lengths):
    if min_length <= am_length <= max_length and min_length <= en_length <= max_length:
        am_sentence = "[start] " + am_sentence + " [end]"
        en_sentence = "[start] " + en_sentence + " [end]"
        filtered_pairs.append((am_sentence, en_sentence))
        filtered_am_lengths.append(am_length)
        filtered_en_lengths.append(en_length)

# Print the number of pairs before and after filtering
print(f"Number of pairs before filtering: {len(text_pairs)}")
print(f"Number of pairs after filtering: {len(filtered_pairs)}")

# Calculate combined lengths and handle single occurrences
combined_lengths = [am + en for am, en in zip(filtered_am_lengths, filtered_en_lengths)]
counts = Counter(combined_lengths)
# Filter out lengths that occur only once
filtered_pairs = [pair for pair, length in zip(filtered_pairs, combined_lengths) if counts[length] > 1]
filtered_lengths = [length for length in combined_lengths if counts[length] > 1]

# Split the data into train, validation, and test sets with stratification
train_pairs, test_pairs = train_test_split(filtered_pairs, test_size=0.2, random_state=42, stratify=filtered_lengths)
train_lengths, test_lengths = train_test_split(filtered_lengths, test_size=0.2, random_state=42, stratify=filtered_lengths)

train_pairs, val_pairs = train_test_split(train_pairs, test_size=0.2, random_state=42, stratify=train_lengths)
train_lengths, val_lengths = train_test_split(train_lengths, test_size=0.2, random_state=42, stratify=train_lengths)

# Print the size of each set
print(f"{len(filtered_pairs)} total pairs")
print(f"{len(train_pairs)} training pairs")
print(f"{len(val_pairs)} validation pairs")
print(f"{len(test_pairs)} test pairs")
```

Tokenization, Frequency Calculation, and Analysis of Token Counts for Amharic and English Texts:

```
# Custom standardization function
def custom_standardization(input_string):
    return tf.strings.lower(input_string)

# Tokenize and count frequency
def tokenize_and_count(texts):
    all_tokens = []
    for text in texts:
        tokens = text.split()
        all_tokens.extend(tokens)
    return Counter(all_tokens)

train_am_texts = [pair[0] for pair in train_pairs]
train_en_texts = [pair[1] for pair in train_pairs]

am_token_freq = tokenize_and_count(train_am_texts)
en_token_freq = tokenize_and_count(train_en_texts)

# Calculate and print max tokens for both languages
am_max_tokens = max(len(text.split()) for text in train_am_texts)
en_max_tokens = max(len(text.split()) for text in train_en_texts)

print(f"Max tokens in Amharic: {am_max_tokens}")
print(f"Max tokens in English: {en_max_tokens}")

# Calculate the 95th percentile of token counts
am_token_counts = [len(text.split()) for text in train_am_texts]
en_token_counts = [len(text.split()) for text in train_en_texts]
am_95th_percentile = int(np.percentile(am_token_counts, 95))
en_95th_percentile = int(np.percentile(en_token_counts, 95))
print(f"95th percentile of token counts in Amharic: {am_95th_percentile}")
print(f"95th percentile of token counts in English: {en_95th_percentile}")
```

Sequence Length Calculation and Character Vectorization for Amharic and English Texts:

```python
def get_max_sequence_length(texts):
    return max([len(text) for text in texts])

# Preprocess and get max lengths
am_texts = [' '.join(preprocess_amharic(text)) for text, _ in train_pairs + val_pairs + test_pairs]
en_texts = [' '.join(preprocess_english(text)) for _, text in train_pairs + val_pairs + test_pairs]

max_am_sequence_length = get_max_sequence_length(am_texts)
max_en_sequence_length = get_max_sequence_length(en_texts)

print(f"Max Amharic sequence length: {max_am_sequence_length}")
print(f"Max English sequence length: {max_en_sequence_length}")

# TextVectorization layers with max sequence lengths
am_vectorization = layers.TextVectorization(
    max_tokens=None,
    output_mode="int",
    output_sequence_length=max_am_sequence_length,
    standardize=custom_standardization,
    split='character'
)
en_vectorization = layers.TextVectorization(
    max_tokens=None,
    output_mode="int",
    output_sequence_length=max_en_sequence_length,
    standardize=custom_standardization,
    split='character'
)

# Adapt vectorization layers on training data
am_vectorization.adapt(train_am_texts)
en_vectorization.adapt(train_en_texts)

# Vocabulary sizes
amharic_vocab_size = len(am_vectorization.get_vocabulary())
english_vocab_size = len(en_vectorization.get_vocabulary())
print(f"Amharic vocabulary size: {amharic_vocab_size}")
print(f"English vocabulary size: {english_vocab_size}")
```

Proposed Model Parameters and Dataset Formatting and Creation for Training:

```python
# Parameters
batch_size = 64
d_model = 256
latent_dim = 524
num_heads = 8
num_layers = 5
dropout_rate = 0.3

# Dataset formatting function
def format_dataset(am, en):
    am = am_vectorization(am)
    en = en_vectorization(en)
    return ({"encoder_inputs": am, "decoder_inputs": en[:, :-1]}, en[:, 1:])

def make_dataset(pairs, batch_size):
    am_texts, en_texts = zip(*pairs)
    am_texts = [' '.join(preprocess_amharic(text)) for text in am_texts]
    en_texts = [' '.join(preprocess_english(text)) for text in en_texts]
    dataset = tf.data.Dataset.from_tensor_slices((am_texts, en_texts))
    dataset = dataset.batch(batch_size)
    dataset = dataset.map(format_dataset, num_parallel_calls=tf.data.experimental.AUTOTUNE)
    return dataset.shuffle(1024).prefetch(tf.data.experimental.AUTOTUNE).cache()

# Create the datasets
train_ds = make_dataset(train_pairs, batch_size)
val_ds = make_dataset(val_pairs, batch_size)
test_ds = make_dataset(test_pairs, batch_size)
```

Positional Encoding Implementation:

```python
def get_angles(pos, i, d_model):
    angle_rates = 1 / tf.pow(10000.0, (2 * (i // 2)) / tf.cast(d_model, tf.float32))
    return tf.cast(pos, tf.float32) * angle_rates

def positional_encoding(position, d_model):
    angle_rads = get_angles(
        tf.range(position, dtype=tf.float32)[:, tf.newaxis],
        tf.range(d_model, dtype=tf.float32)[tf.newaxis, :],
        d_model
    )

    # Apply sin to even indices in the array; 2i
    sines = tf.math.sin(angle_rads[:, 0::2])

    # Apply cos to odd indices in the array; 2i+1
    cosines = tf.math.cos(angle_rads[:, 1::2])

    pos_encoding = tf.concat([sines, cosines], axis=-1)
    pos_encoding = pos_encoding[tf.newaxis, ...]

    return tf.cast(pos_encoding, tf.float32)
```

Scaled Dot-Product and Multi-Head Attention Implementation:

```python
# Scaled Dot-Product Attention
def scaled_dot_product_attention(q, k, v, mask):
    matmul_qk = tf.matmul(q, k, transpose_b=True)
    dk = tf.cast(tf.shape(k)[-1], tf.float32)
    scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)
    if mask is not None:
        scaled_attention_logits += (mask * -1e9)
    attention_weights = tf.nn.softmax(scaled_attention_logits, axis=-1)
    output = tf.matmul(attention_weights, v)
    return output, attention_weights
# Multi-Head Attention
class CustomMultiHeadAttention(tf.keras.layers.Layer):
    def __init__(self, d_model, num_heads):
        super(CustomMultiHeadAttention, self).__init__()
        self.num_heads = num_heads
        self.d_model = d_model
        assert d_model % self.num_heads == 0
        self.depth = d_model // self.num_heads
        self.wq = tf.keras.layers.Dense(d_model)
        self.wk = tf.keras.layers.Dense(d_model)
        self.wv = tf.keras.layers.Dense(d_model)
        self.dense = tf.keras.layers.Dense(d_model)
        self.dropout = tf.keras.layers.Dropout(0.3)  # Adjust dropout rate as needed
        self.layernorm = tf.keras.layers.LayerNormalization(epsilon=1e-6)
    def split_heads(self, x, batch_size):
        x = tf.reshape(x, (batch_size, -1, self.num_heads, self.depth))
        return tf.transpose(x, perm=[0, 2, 1, 3])
    def call(self, v, k, q, mask):
        batch_size = tf.shape(q)[0]
        q = self.wq(q)
        k = self.wk(k)
        v = self.wv(v)
        q = self.split_heads(q, batch_size)
        k = self.split_heads(k, batch_size)
        v = self.split_heads(v, batch_size)
        scaled_attention, attention_weights = scaled_dot_product_attention(q, k, v, mask)
        scaled_attention = tf.transpose(scaled_attention, perm=[0, 2, 1, 3])
        concat_attention = tf.reshape(scaled_attention, (batch_size, -1, self.d_model))
        output = self.dense(concat_attention)
        original_q = tf.reshape(tf.transpose(q, perm=[0, 2, 1, 3]), (batch_size, -1, self.d_model))
        output = self.layernorm(output + original_q)
        return output, attention_weights
```

Encoder and Decoder Stacks Implementation:

```
class Encoder(tf.keras.layers.Layer):
    def __init__(self, num_layers, d_model, num_heads, dff, amharic_vocab_size, rate=0.3):
        super(Encoder, self).__init__()
        self.d_model = d_model
        self.num_layers = num_layers
        self.embedding = tf.keras.layers.Embedding(amharic_vocab_size, d_model)
        self.enc_layers = [EncoderLayer(d_model, num_heads, dff, rate) for _ in range(num_layers)]
        self.dropout = tf.keras.layers.Dropout(rate)

    def call(self, x, training, mask):
        seq_len = tf.shape(x)[1]
        x = self.embedding(x)
        x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
        x += positional_encoding(seq_len, self.d_model)
        x = self.dropout(x, training=training)
        for i in range(self.num_layers):
            x = self.enc_layers[i](x, training, mask)
        return x

class Decoder(tf.keras.layers.Layer):
    def __init__(self, num_layers, d_model, num_heads, dff, english_vocab_size, rate=0.3):
        super(Decoder, self).__init__()
        self.d_model = d_model
        self.num_layers = num_layers
        self.embedding = tf.keras.layers.Embedding(english_vocab_size, d_model)
        self.dec_layers = [DecoderLayer(d_model, num_heads, dff, rate) for _ in range(num_layers)]
        self.dropout = tf.keras.layers.Dropout(rate)

    def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
        seq_len = tf.shape(x)[1]
        attention_weights = {}
        x = self.embedding(x)
        x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
        x += positional_encoding(seq_len, self.d_model)
        x = self.dropout(x, training=training)
        for i in range(self.num_layers):
            x, block1, block2 = self.dec_layers[i](x, enc_output, training, look_ahead_mask, padding_mask)
            attention_weights[f'decoder_layer{i+1}_block1'] = block1
            attention_weights[f'decoder_layer{i+1}_block2'] = block2
        return x, attention_weights
```

Each Encoder and Decoder Layers Implementation:

```
# Encoder Layer
class EncoderLayer(tf.keras.layers.Layer):
    def __init__(self, d_model, num_heads, dff, rate=0.3):
        super(EncoderLayer, self).__init__()
        self.mha = CustomMultiHeadAttention(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):
        attn_output, _ = self.mha(x, x, x, mask)
        attn_output = self.dropout1(attn_output, training=training)
        out1 = self.layernorm1(x + attn_output)
        ffn_output = self.ffn(out1)
        ffn_output = self.dropout2(ffn_output, training=training)
        out2 = self.layernorm2(out1 + ffn_output)
        return out2
# Decoder Layer
class DecoderLayer(tf.keras.layers.Layer):
    def __init__(self, d_model, num_heads, dff, rate=0.3):
        super(DecoderLayer, self).__init__()
        self.mha1 = CustomMultiHeadAttention(d_model, num_heads)
        self.mha2 = CustomMultiHeadAttention(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.layernorm3 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.dropout1 = tf.keras.layers.Dropout(rate)
        self.dropout2 = tf.keras.layers.Dropout(rate)
        self.dropout3 = tf.keras.layers.Dropout(rate)
    def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
        attn1, attn_weights_block1 = self.mha1(x, x, x, look_ahead_mask)
        attn1 = self.dropout1(attn1, training=training)
        out1 = self.layernorm1(x + attn1)
        attn2, attn_weights_block2 = self.mha2(enc_output, enc_output, out1, padding_mask)
        attn2 = self.dropout2(attn2, training=training)
        out2 = self.layernorm2(out1 + attn2)
        ffn_output = self.ffn(out2)
        ffn_output = self.dropout3(ffn_output, training=training)
        out3 = self.layernorm3(out2 + ffn_output)
        return out3, attn_weights_block1, attn_weights_block2
```

Point-wise FFN Implementation with Combined Regularization Techniques: Dropout and Elastic Net:

```
# Point-Wise Feed Forward Network with dropout, elastic net regularizer and layer normalization
# Define Elastic Net regularization
elastic_net = tf.keras.regularizers.L1L2(l1=0.01, l2=0.01)
def point_wise_feed_forward_network(d_model, dff):
    return tf.keras.Sequential([
        tf.keras.layers.Dense(dff, activation='relu', kernel_regularizer=elastic_net),
        tf.keras.layers.Dense(d_model, kernel_regularizer=elastic_net),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.LayerNormalization(epsilon=1e-6)
    ])
```

Point-wise FFN Implementation with Combined Regularization Techniques: Dropout and L1:

```
# Point-Wise Feed Forward Network with dropout, L1 and layer normalization
# Define L1 regularization
l1 = tf.keras.regularizers.l1
def point_wise_feed_forward_network(d_model, dff):
    return tf.keras.Sequential([
        tf.keras.layers.Dense(dff, activation='relu', kernel_regularizer=l1(0.01)),
        tf.keras.layers.Dense(d_model, kernel_regularizer=l1(0.01)),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.LayerNormalization(epsilon=1e-6)
    ])
```

Point-wise FFN Implementation with Combined Regularization Techniques: Dropout and L2:

```
# Point-Wise Feed Forward Network
#Define L2 regularization
l2 = tf.keras.regularizers.l2
def point_wise_feed_forward_network(d_model, dff):
    return tf.keras.Sequential([
        tf.keras.layers.Dense(dff, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(l2=0.01)),
        tf.keras.layers.Dense(d_model, kernel_regularizer=tf.keras.regularizers.l2(l2=0.01)),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.LayerNormalization(epsilon=1e-6)
    ])
```

Annex III – Sample Translated Sentences from Each Experiment

Baseline Model:

Sample	Amharic Sentence	Reference English Sentence	Predicted English Sentence	BLEU Score
1	ሁለቱን ኩላሊቶችና በእነሱ ላይ ማለትም በሽንጡ አካባቢ ያለውን ስብ ያቀርባል። በጉበቱ ላይ ያለውንም ሞራ ከኩላሊቶቹ ጋር አብሮ ያነሳል።	and the two kidneys with the fat on them that is near the loins. he will also remove the appendage of the liver along with the kidneys. [end]	and the two kidneys with them on the fat that is on them, and the fat will be on the loins. he will remove the appendage of the liver along with the loins.	0.4461
2	ከጊዜ በኋላ የእስራኤል ነገዶች በሙሉ በኬብሮን ወደሚገኘው ወደ ዳዊት መጥተው እንዲህ አሉት፦ እንግዲህ እኛ የአጥንትህ ፍላጭ፣ የሥጋህም ቁራጭ ነን።	in time all the tribes of israel came to david at hebron and said: look! we are your own bone and flesh.	in time all the tribes of israel went to hebron and said: here we are your own bone and flesh.	0.5699
3	እባብም ለሴቲቱ አላት። ሞትን አትሞቱም፤	and the serpent said unto the woman, ye shall not surely die:	and the woman said unto the woman, if thou surely surely have died for ever.	0.1892
4	ከዚያም ከእስራኤል ሕዝብ ፊት ፊት ይሄድ የነበረው የእውነተኛው አምላክ መልአክ ከዚያ ተነስቶ ወደኋላቸው ሄደ፤ ከፊታቸው የነበረውም የደመና ዓምድ ወደኋላቸው ሄዶ በስተ ጀርባቸው ቆመ።	then the angel of the true god who was going ahead of the camp of israel departed and went to their rear, and the pillar of cloud that was in front of them moved to the rear and stood behind them.	then the angel of the true god went out before the ark of the true god, and the pillar of the pillar of the cloud stood before them, and the pillar of the pillar of cloud went around them to the rear guard reached them at the rear guard reached them and stood by the rear guard reached them and stood by the rear	0.1828
5	ይህን ሰው ወደ ሆስፒታል በፍጥነት ለመውሰድ አምቡላንስ የሚገኝበት ምንም መንገድ የለም።	there is no way to get an ambulance in quickly to move him to a hospital.	there is no way to get to a hospital. there is no way to be no way to be required.	0.2924

Proposed Model: Character Embedding with Combined Regularization:

Sample	Amharic Sentence	Reference English Sentence	Predicted English Sentence	BLEU Score
1	“ሁን” የኢራቅ አገዛዝ በከፍተኛ የአየር ድብደባ በደረሰበት ጉዳት ምክንያት እየተፍረከረከ መሆኑን አብራርቷል።	“hoon” said Iraq’s regime was crumbling under the pressure of a huge air assault.	“Hoon” said Iraq’s regime was crumbling under the pressure of the air assault.	0.7005
2	ሆኖም የምናሴ ልጅ፣ የማኪር ልጅ፣ የጊልያድ ልጅ፣ የሄፌር ልጅ ሰለጰአድ ሴቶች እንጂ ወንዶች ልጆች አልነበሩትም፤ የሴቶች ልጆቹም ስም ማህላ፣ ኖኅ፣ ሆግላ፣ ሚልካ እና ቲርጻ ነበር።	but zelophehad the son of hepher, the son of gilead, the son of machir, the son of manasseh, did not have sons, only daughters, and these were the names of his daughters: mahlah, noah, hoglah, milcah, and tirzah.	but zelophehad the son of gilead, the son of machir, the son of manasseh, was not only of the daughters of manasseh, but the names of the daughters of manasseh were mahlah, noah, hoglah, milcah, and tirzah.	0.5101
3	ዕጣንም የተሞላ ሚዛኑ አሥር ሰቅል የሆነ አንድ የወርቅ ጭልፋ፤	one golden spoon of ten shekels, full of incense:	one golden spoon of ten shekels, full of incense:	0.8801
4	እንባውን የሚያብስለት፣ ተስፋውን የሚያቃናለት ሌላ ምድራዊ ሃይል የለም።	there is no other mundane fore that wipes his tears, amend his hope.	there is no other mundane fore that wipes his earthly hope.	0.6173
5	በመሆኑም የወይራ ዛፍን በእኛ ላይ ንገስ አሉት።	so they said to the olive tree, “rule over us.”	so they said to the olive tree, “rule over us.”	0.4137

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: Surafiel Habib Asefa

Signature: _____

Date: _____

Confirmed by advisor:

Name: Yaregal Assabie (PhD)

Signature: _____

Date: _____