INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF GRADUATE STUDIES

MORPHOLOGICAL ANALYSIS FOR AFAAN OROMOO USING DEEP LEARNING APPROACHES

MASTERS OF SCIENCE THESIS

BOKI CHELKEBA CHALI

HAWASSA UNIVERSITY, HAWASSA, SIDAMA, ETHIOPIA

MAY 18, 2024

HAWASA UNIVERSITY

INSTITUTE OF TECHNOLOGY

SCHOOL OF POST-GRADUATE STUDIES

P. O.BOX: 5, HAWASSA, ETHIOPIA

TITLE PAGE

TITLE: MORPHOLOGICAL ANALYSIS FOR AFAAN OROMOO USING DEEP LEARNING APPROACHES

BY: BOKI CHELKEBA CHALI

ADVISOR: ANDARGACHEW MEKONNEN (Dr.)

THIS RESEARCH THESIS IS SUBMITTED TO HAWASSA UNIVERSITY INSTITUTE OF TECHNOLOGY DEPARTMENT OF COMPUTER SCIENCE FOR PARTIAL FULFILLMENT OF THE REQUIREMENTS OF MSC DEGREE IN COMPUTER SCIENCE.

HAWASA UNIVERSITY

INSTITUTE OF TECHNOLOGY

SCHOOL OF POST-GRADUATE STUDIES

P.O.BOX: 5, HAWASSA, ETHIOPIA

APPROVAL SHEET

This is to certify that the thesis entitled, "MORPHOLOGICAL ANALYSIS FOR AFAAN OROMOO USING DEEP LEARNING APPROACHES" submitted in partial fulfilment of the requirements for the degree of Master's with specialization in Computer Science, the Graduate Program of the Faculty of Informatics, and has been carried out by BOKI CHELKEBA CHALI. Therefore we recommend that the student has fulfilled the requirements and hence hereby can submit the thesis to the department.

1.	Andargachew Mekonnen		May 18, 2024
	Name of Major Advisor	**Signature**	**Date**

HAWASSA UNIVERSITY

INSTITUTE OF TECHNOLOGY

SCHOOL OF POST-GRADUATE STUDIES

P.O.BOX: 5, HAWASSA, ETHIOPIA

EXAMINERS' APPROVAL SHEET

SCHOOL OF GRADUATE STUDIES

HAWASSA UNIVERSITY EXAMINERS' APPROVAL SHEET

(Submission Sheet-2)

We, the undersigned, members of the Board of Examiners of the final open defense by BOKI CHELKEBA CHALI have read and evaluated his/her thesis entitled "MORPHOLOGICAL ANALYSIS FOR AFAAN OROMOO USING DEEP LEARNING APPROACHES " , and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfilment of the requirements for the degree.

1.	Andargachew Mekonnen		May 18, 2024
	Name of Major Advisor	**Signature**	**Date**
2.	_____	_____	_____
	Name of Internal Examiner-I	**Signature**	**Date**
3.	_____	_____	_____
	Name of Internal Examiner-II	**Signature**	**Date**
4.	Michael Melese Woldeyohannis (PhD)		21/07/2024
	Name of External examiner	**Signature**	**Date**
5.	_____	_____	_____
	SGS Approval	**Signature**	**Date**

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

Stamp of SGS Date: _____

Dedication

This thesis is dedicated to my father, whose absence has profoundly shaped my journey. Though I never had the chance to know him, and never saw even his final resting place, his memory has been a silent motivator in my life. Despite the countless challenges and the lack of support, I deserved most of my goal. This achievement is a tribute to the power of hope and the human spirit's ability to overcome adversity. I also dedicate this work to the Almighty God, whose unwavering guidance and grace have been my constant companions. Despite the hardships and challenges of growing up on the streets, it was through the help of God that I found the strength and resilience to pursue my education. May this dedication serve as a beacon of hope, illustrating that with God's help, even the most difficult circumstances can be overcome, and dreams can be realized.

Declaration

As I am the writer of the thesis, I affirm that my own work is included in the thesis. I have dually recognized and cited all materials utilized in this work in accordance with globally recognized procedures. I am aware that disregarding the standards of academic honesty and integrity, as well as fabricating or misrepresenting any idea, data, fact, or source, will be sufficient justification for disciplinary action by the university. It may also result in legal action being taken against the source if it is not properly cited or acknowledged.

Boki Chelkeba Chali

Name of the candidate

Signature

May 18, 2024

Date

Acknowledgement

Firstly, I extend my deepest gratitude to my Almighty God, whose boundless assistance has been the cornerstone of my journey. His unwavering guidance and grace have provided me with the strength and resilience to overcome every challenge.

Secondly, I wish to express my sincere appreciation to my Advisor, Dr. Andargachew Mekonnen(PhD). His expertise, support, and mentorship have been instrumental in shaping this work. I am profoundly grateful for his invaluable contributions and unwavering encouragement throughout this process.

Abbreviation

1-D	One Dimensional
AF	Activation Function
AI	Artificial Intelligence
AO	Afaan Oromoo
AOMA	Afaan Oromoo Morphological Analysis
API	Application Programming Interface
ARAS	Activity recognition with ambient sensing
AUC	Area under the Curve
BiGRU	Bidirectional Gated Recurrent Unit
BiLSTM	Bidirectional Long Short Term Memory
BLUE	Bilingual Evaluation Understudy
BS	Batch Size
C	Consonant
CBOW	Continuous Bag of Words
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRF	Continuous Random Forest
CSV	Comma Separated Values
F	Female
DS	Derivative Stem
F1	Formula One
FN	False Negative
FP	False Positive

FST	Finite State Transducer
GB	Giga Byte
GHz	Giga Hertz
GPU	Graphic Processing Unit
GRU	Gated Recurrent Unit
HML	Hypertext Mark-up Language
IB1	Instance based learning 1
IGTREE	Information gain tree
IMDB	Internet movie database
IOS	IPhone operating system
La TeX	Latest Text
LSTM	Long short term memory
M	Male
MAE	Mean Absolute Error
MLB	Machine Learning Based
MS	Micro Soft
MSE	Mean Square Error
MT-DMA	Multi-task deep morphological analyser
NLP	Natural language processing
OCR	Optical Character Recognition
OS	Operating System
PDF	Portable Document Format
RAM	Random Access Memory
ReLUs	Rectified Linear Unit

RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristics
TN	True Negative
TP	True Positive
TPU	Tensor Processing Unit
V	Vowel
VAE	Variational Auto Encoder
WPS	Wireless Presentation System
XML	Extensive Mark-up Language

Table of Contents

List of Tables

List of Figures

List of table in appendix

Abstract

Afaan Oromoo, a widely spoken language in Ethiopia and neighbouring countries, presents unique challenges due to its complex morphological structure. Morphological analysis, which decomposes words into morphemes and assigns grammatical information, is a crucial natural language processing (NLP) task for this language. Previously some researchers conducted Afaan Oromoo morphological analysis using rule-based and traditional machine learning techniques. Rule-based methods are labour-intensive and time-consuming, especially with large datasets, while traditional machine learning approaches struggle with feature extraction and high-dimensional vector spaces, leading to information loss. This study addresses these challenges by employing deep learning architectures, including Convolutional Neural Networks (CNNs), Long Short-Term Memory networks (LSTMs), Gated Recurrent Units (GRUs), and Bidirectional LSTMs (BiLSTMs) which are not applied for Afaan Oromoo morphological analysis yet. In the study, Comprehensive evaluations were conducted on a dataset consisting of 30,636 training words, 10,213 validation words, and 4,539 testing words. Performance metrics such as accuracy, precision, recall, and F1-score were used to evaluate the models. The evaluation results for each model are as follows: Normal CNN-LSTM with 70.94% accuracy, Word2Vec CNN-LSTM with 94.74% accuracy, Fast Text CNN-LSTM with 95.25% accuracy, Normal LSTM with 95.06% accuracy, Word2Vec LSTM with 93.89% accuracy, Fast Text LSTM with 90.02% accuracy, Normal GRU with 92.96% accuracy, Word2Vec GRU with 91.98% accuracy, Fast Text GRU with 91.32% accuracy, Normal BiLSTM with 95.24% accuracy, Word2Vec BiLSTM with 96.21% accuracy, and Fast Text BiLSTM with 96.43% accuracy. The Bidirectional LSTM (BiLSTM) models, particularly those using Word2Vec and Fast Text embeddings, demonstrated the highest accuracies, highlighting the effectiveness of deep learning approaches and neural word embedding techniques in Afaan Oromoo morphological analysis. This research not only advances the state-of-the-art in this domain but also provides a robust methodology for handling the morphological complexity of Afaan Oromoo using deep learning.

Key words: ***Afaan Oromoo, Morphological analysis, Natural Language Processing (NLP), Deep learning architectures, Neural word embedding technique***

Chapter One: Introduction

1.1 Background of the Study

Language is a fundamental aspect of human behaviour, facilitating daily interactions and preserving knowledge passed down through generations. Linguists, psycho-linguists, philosophers, and computational linguists study various linguistic concepts, including phonetics, phonology, morphology, syntax, semantics, pragmatics, and discourse[1][2]. Natural language, which has evolved naturally in humans, can take various forms and be transmitted from generation to generation[3][4][5].

Afaan Oromoo holds a significant place as one of the major languages of East Africa, spoken primarily in Ethiopia, Kenya, Tanzania, Uganda, Djibouti and Somalia. Understanding the morphology of Afaan Oromoo is crucial for comprehending its structure, evolution, and usage patterns[6][7][8].

Morphology, the study of the structure of words and their formation, plays a fundamental role in linguistics, offering insights into how words are created, modified, and combined to convey meaning[9][10]. Through morphological analysis, linguists and language enthusiasts delve into the intricate systems governing word formation processes, shedding light on the rich tapestry of linguistic diversity.

This paper embarks on a journey of morphological analysis of Afaan Oromoo, aiming to unravel the intricate mechanisms that underlie the construction of words in this vibrant language. By examining morphological processes such as affixation, and derivation, the study seek to elucidate the patterns and principles governing word formation in Afaan Oromoo. Furthermore, this study endeavours to explore the implications of morphological analysis for language learning, computational linguistics, and cultural preservation.

Natural Language Processing (NLP) uses computer science methods to understand and interact with natural languages. NLP systems analyse natural language at various complexity levels, including phonemes, words, sentences, and pragmatics[5][11][12]. The goal is to enable human-machine communication, improve human-human communication, and perform useful tasks such as text or speech processing. NLP is used in various applications, such as spelling error correction, machine translation, and automatic knowledge extraction. Morphology is a field of linguistics that studies the internal structure of words and their formation. In computational morphology, there are two main methods

for morphological analysis: rule-based and corpus-based methods. Rule-based methods use less data but yield very accurate results by fusing language and domain expertise.

Two approaches are essential for understanding and enhancing NLP applications: rule-based, corpus based methods[8][13][14]. This study is focused on the extension of machine learning approaches called deep learning approaches. Deep learning is a subset of the machine learning technique that draws inspiration from the workings of the human brain and makes use of artificial neural networks to learn intricate correlations and patterns from vast volumes of data without the need for explicit programming. Language processing, speech comprehension, and picture identification are examples of tasks that can be handled by deep learning[15][16][17].

The intersection of traditional linguistic analysis and modern computational techniques has opened up new avenues for understanding and exploring the structures of languages. Afaan Oromoo, a language spoken by millions in East Africa, stands as a testament to this rich linguistic diversity. This paper presents a pioneering approach to the morphological analysis of Afaan Oromoo, leveraging the power of deep learning methodologies. The goal is to create models that can precisely anticipate and construct morphological structures inside the language by training neural networks on massive datasets of Afaan Oromoo text. This approach not only streamlines the process of linguistic analysis but also enables one to uncover subtle patterns and nuances that may elude traditional methods.

The application of deep learning to Afaan Oromoo morphology holds immense promise for various fields, including language learning, machine translation, and natural language understanding. By building robust computational models of Afaan Oromoo morphology, this study lay the foundation for developing more effective language technologies tailored to this linguistic context. The research contributes to the broader understanding of Afaan Oromoo linguistics, shedding light on its unique morphological features and structures.

1.2 Motivation of the Study

Afaan Oromoo, a widely spoken Cushitic language in Ethiopia and neighbouring countries like Kenya and Somalia, is crucial for Natural Language Processing (NLP) tasks. Morphological analyzers are vital for advancing higher linguistic levels and NLP applications. However, despite the widespread use of Afaan Oromoo, particularly in Ethiopia, there is a lack of research specifically focused on developing deep learning-based

morphological analysis systems for this language. While deep learning techniques have been employed to create numerous morphological analyzers for other languages, no published work exists on using deep learning approaches for Afaan Oromoo morphological analyzers.

1.3 Statement of the Problem

Afaan Oromoo is significant African language spoken in Ethiopia, Kenya, Somalia, Uganda, Tanzania, and Djibouti. It is the working language in the Oromia Regional State and the official language of the Oromia Regional State. The language has a distinctive writing structure and is widely spoken in Ethiopia and neighbouring nations. However, there is no comparable study, including morphological analysis, has been conducted for this language. Morphological analyzers for several languages have been developed using various approaches; for instance, deep learning based morphological analysis at character level was developed by [58] [34].

As Afaan Oromoo has its own unique writing structure, previously conducted morphological analyzer for other languages are not applicable for this language. So the language requires its own robust and unique Afaan Oromoo morphological analyzer for its higher-level linguistic analysis. Previously some Afaan Oromoo researchers like Assefa W/Mariam[18], Michael Gasser[19], Moyka Degefa[7], and Kedir Gena[8] conducted morphological analysis using rule based and traditional machine learning approaches.

However, these systems explored research gaps those used as the input for this study. For example in the work of Assefa W/Mariam even though the title and documentation of his study describes morphological analysis, the researcher used linguistica which is used for word segmenting purpose only. In the work of Michael Gasser, the researcher evaluated the modules of Amharic and Tigrigna left with the module of Afaan Oromoo un evaluated because of the variation in double consonants between Afaan Oromoo writers. In the case of Moyka and Kedir's research works, as extracting features from huge amount of dataset is time consuming, complex and boring, they used limited dataset size to extract features which may not fill the goal of NLP mainly for morphologically rich Afaan Oromoo language.

Recently, researcher Zarihun Wodajo conducted deep learning morphological Analysis for Afaan Oromoo. Though the researcher presented that his system included word embedding techniques like Word2vec, the system lack all word embedding techniques even word2vec.

At all, all previously conducted research works were used rule based and statistical approaches. In rule based morphological analysis, rules are drafted from complex linguistic structure and theories to extract features from the given dataset which is time consuming, tedious and exhausting mainly if large datasets are used. In rule based morphological analysis, if one rule fails it affects entire rules which lead to error. In case of traditional statistical machine learning approaches explicit programming were written to extract features from the dataset which is also complex, time consuming, and boring task if used in un limited dataset size. In traditional machine learning approaches, as the number of dataset size increases the number of features extracted also increases which make the vector dimension too high and affects the calculation efficiency leading to semantic information loss.

In the work of Zarihun there is no neural based word embedding techniques. So the novel morphological analysis is highly demanded by researchers working on Afaan Oromoo higher Natural Language Processing (NLP) tasks due to its multi-purpose nature and lack of publicly annotated corpus for designing and developing NLP tasks. Therefore, every problem in the languages must be solved using all available computational technology and wisdom. This study aims to develop and evaluate deep learning models specifically for the morphological analysis of Afaan Oromoo, leveraging their strengths like feature learning and hierarchical representations.

1.4 Research Question

At the end of this study, the following research question will be answered.

- ✧ How do different deep learning architectures perform morphological analysis of Afaan Oromoo?
- ✧ How do different word embedding techniques affect the accuracy and efficiency of morphological analyzers for Afaan Oromoo?
- ✧ What are the most effective strategies for mitigating overfitting in deep learning models for Afaan Oromoo morphological analysis?

- How can challenges in data collection and annotation for Afaan Oromoo be addressed to create high-quality datasets?

1.5 Objectives of the Study

1.5.1 General Objective

The general objective of this study is to use deep learning techniques to construct and develop Afaan Oromoo morphological analyzer.

1.5.2 Specific Objectives

The specific objectives of this study are:

- To review articles, journals, books, videos, conferences and many other to get better understanding on how to conduct, and accomplish the proposed study,
- To construct a comprehensive corpus of text data, preprocessing, augment, annotate it to match the linguistic characteristics of Afaan Oromoo,
- To investigate and select appropriate deep learning architectures for Afaan Oromoo morphological analysis,
- To develop and train deep learning models, utilizing the created dataset to learn the morphological patterns and structures of Afaan Oromoo,
- To evaluate the performance of the trained models using various metrics such as accuracy, precision, recall, and F1-score with new dataset,
- To compare the performance of the deep learning models with baseline methods and existing approaches for Afaan Oromoo morphological analysis.

1.6 Significance of the Study

This study aims to fill the digital gap and localize current technologies, such as information and communication technology, by focusing on Afaan Oromoo morphological analysis using deep learning techniques.

This research is crucial for researchers working on higher-level NLP applications, such as information retrieval, question-answering, sentiment analysis, and anaphora resolution. It is also beneficial for Afaan Oromoo native speakers and others who trust the language by

providing the category or part of speeches of words in the language. The study will support linguists, educators, technologists, and researchers by enabling them to apply morphological analyzers in NLP applications and develop educational resources for teaching and understanding Afaan Oromoo morphology.

The research contributes to the preservation, documentation, and development of the language, enabling speakers to engage with digital technologies in their native language. It also expands the frontier of computational linguistics by demonstrating the efficacy of deep learning approaches in handling morphological analysis tasks for languages with complex morphology, such as Afaan Oromoo.

Generally, the study contribute to the academic literature in linguistics, computational linguistics, and natural language processing, enriching scholarly discourse and providing a foundation for future research.

1.7 Scope and Limitation of the Study

The study focuses on morphological analysis of Afaan Oromoo, a language spoken by the Oromoo people in Ethiopia and neighbouring country. It aims to evaluate and classify inflected and derivated Afaan Oromoo words belonging to noun, adjective, and verb classes only. Though, Afaan Oromoo words can be formed through various computational morphology techniques like compounding, reduplication and others, the system focuses solely on inflected and derivated Afaan Oromoo nouns, adjectives and verbs. Deep learning algorithms necessitate large datasets, but the lack of publicly available annotated datasets requires time-consuming data collection, preparation, and annotation. This may deter experts and others from contributing, potentially hindering the study's objectives.

1.8 Organization of the Study

This thesis is divided into five chapters: chapter one discusses the highlight of the proposed system, chapter two discusses all terminologies included under the topic literature review and related works, morphology, NLP, and Deep Learning

-concepts, related works, and state-of-the-art morphological analyzers. It also covers the linguistic structures of Afaan Oromoo nominals and verbs, chapter three three discusses the methods and methodology used to design the proposed system, chapter four discusses the experimentation of the proposed system and its result analysis. And finally, the

discussion, conclusion, and recommendation are discussed with in chapter five of the proposed systems.

Chapter Two: Literature Review and Related Work

20.1 Introduction

Morphological analyzers are fundamental for higher-level NLP tasks. They can be developed using rule-based or corpus-based approaches. This thesis aims to develop Afaan Oromoo morphological analyzer using corpus-based deep learning approaches. To achieve this, the research integrates various sources, including previous studies, methodologies, linguistic experts, and resources specific to Afaan Oromoo. The chapter synthesizes insights from the literature, highlighting the critical role of deep learning in morphological analysis, especially for under-represented languages. It identifies gaps and limitations in existing research, providing a rationale for this study and its potential contributions. The literature review establishes a solid foundation, aiming to advance morphological analysis using deep learning.

2.2 Basic Concepts and Terminologies of Morphology

Morphology, which is the branch of linguistics that studies words and the internal structures of words is originated from the Greek word "morph," meaning "shape, or form". It is also the core building element of every human language[11] [13]. The following subsections describe all basic terms and terminologies related to the proposed system.

2.2.1 Stems/roots (lemmas/base)

A stem is the base of an inflected or derivated word with the meaning or not, while roots are the base of an inflected or derivated word carrying the underlying indivisible meaning [4]. For instance: the word availabilities has the stem availability. The stem availability has a meaning called existence and can also be decomposed further into avail and able. From this example, the morphemes avail and able cannot be decomposed further into another morpheme. They are called roots or the foundations of the word availabilities or availability. Lemma is a distinct form within a group of morphologically related forms, such as nominative singular for nouns and infinitive for verbs. Every form has a lemma associated with it. Steal, stole, steals, and stolen are all English words meaning the same thing.

2.2.2 Syllable

Syllables are organizational units for speech sounds, and syllabification is the process that varies based on the language. Every languages have its unique syllable structure, with English permitting more than two consonants to appear consecutively in a single word, such as "screen" [6].

2.2.3 Morpheme and Allomorphs

A morpheme is the smallest semantically meaningful unit in a language, not identical to a word. There are two categories of morphemes: free and bound morphemes. Bound morphemes are unable to occur independently, but free morphemes can stand alone as a word. Bounded morphemes, or affixes, are base that are joined to the root and are unable to exist on their own. Addunyaa Barkessaa[20] divides Afaan Oromoo morphemes into two categories:

Free morphemes (dham-jecha walabaa):- Free morphemes are the morphemes that can stand alone. Individual words with free morphemes include Muka, Farda, Nama, and others.

Bound (dham-jecha hirkataa):- Bound morphemes must be attached to host morphemes before they may be realized as words. They cannot exist without the assistance of other morphemes and are further subdivided into bound root (hundee hirkataa) and affix (fufiilee). Bound roots are parts of a word that left when affixes are removed in Afaan Oromoo. Affixes, which can't stand alone, can alter grammatical information or word classes. They are divided into two categories:

- ✧ **Derivational affixes: (fufiilee yaasaa):-** Derivational affixes modify a word's categories and grammatical information.
- ✧ **Inflectional affixes (fufiilee hortee):-** Inflectional affixes indicate grammatical information such as gender, person, case, aspect, number, and others.

Bound morphemes or affixes can also be classified as: Suffixes which are added to the end of the stem/root word to offer more grammatical information or to modify the word's meaning or class: for instance: - availability. Prefixes are added before or at the beginning of words. For instance, the term "unavailability" has the prefix "un" Circumfixes having two parts, one attached to the beginning of the word and the other to the end of the same

word. It made up of prefixes and suffixes that work together to represent a certain trait. The terms avail and ies in the phrase availabilities. Infixes which is the middle portion of the word where infixes are applied. For example, the German term anzufangen has the infix zu.

Reduplication is a technique used to illustrate repetitive behaviour. Reduplication of verbs are used in the instance of Afaan Oromoo to indicate that the activity has occurred more than once. Adjectives are another tool we may use to indicate the number of the thing we are referring to. Example, the Adjective qal'aa, which means "thin" in masculine, becomes qaqal'aa, meaning "thins one," while the verb mure, which means "he cut," becomes mummure, meaning "He cut something into pieces".

Allomorphs are physical manifestations and different pronunciations of a morpheme. In Afaan Oromoo, there are two primary Allomorphs: "-oota" after a word with a short vowel sound and "-ota" after a word with a long vowel sound. In the language, these morphemes convey plurality [5].

2.2.4 Morphotactic and Morphophonemics

Morphophonemics and Morphotactic are morphological principles that are applied to phoneme replacements and morpheme sequences, respectively. Morphophonemics governs the order of morphemes and distinguishes those that belong to various classes. Analysts utilize morphotactic to determine word structure[15] [20] [23]. The morphology-syntax interaction distinguishes lexical meaning between morphologically related terms. In Afaan Oromoo using words to analyze its grammatical information is ambiguous for some words if used separately. Using phrases or sentences may resolve the problem of ambiguity that is found in single words and creates morph syntactic actions that affect the core meaning of the sentences.

2.2.5 Word

Words are the smallest units with meaning in a language, arranged in a hierarchy of grammatical elements. Word classes or categories are groups of words with similar characteristics, functions, and morphology[12] [28] [29]. Word classifiers are used in NLP systems for voice recognition, machine translation, information extraction, and retrieval. There are so many words used in Afaan Oromoo. For example Nama (man), mana (house), namoonni (human/nom), and many others.

2.2.5.1 Word Classes

Word classes or categories of words are groups of words with similar characteristics, functions, and morphology. Word classifiers are used in NLP systems for voice recognition, machine translation, information extraction, and retrieval[1].

2.2.5.1.1 Open Word Classes

Open word classes, or content words, are constantly expanding due to discoveries and ideas. Noun classes are potentially infinite, with new nouns and verbs introduced in the late twentieth century. Nouns are the words that name People, Places, Things, or Ideas, and can be singular or plural, masculine or feminine and have different cases. Verbs show actions or states of being, with different tenses, aspects, moods, and voices. Adjectives modify or describe nouns, with different degrees of comparison. Adverbs modify or describe verbs, adjectives, or other adverbs, and can show different types of information, such as manner, place, time, frequency, and degree[24].

2.2.5.1.2 Closed Word Classes

Functional words, or closed word classes, are grammatical words that serve as markers for sentence structure and are crucial for linking parts of sentences or creating continuity in text or spoken language. They include pronouns, which replace nouns or phrases, prepositions, which indicate the relationship between nouns or pronouns, and conjunctions, which join words or clauses in a sentence, such as "fi" for "and" or "garuu" for "but"[24].

2.2.6 Sentences

Sentences are constructions made up of important words placed in a certain order to express a whole notion. The subject, verb, object, modifiers, and punctuation all affect the structure. Simple sentences only have one independent component, whereas compound sentences have two or more distinct clauses joined by a comma and coordinating conjunction. Compound-complex sentences comprise elements of both independent and subordinate clauses; complex sentences have one independent clause and one or more subordinate clauses. Sentences can be classified into imperative, declarative, exclamatory, and interrogative categories according to their intended use[25]. Some sentence structures are "The sun rises in the east," "Did you finish your assignment?" "Pass the salt, please,"

and "What a beautiful sunset." It is essential to comprehend linguistic structure in order to write and communicate effectively.

2.3 Natural Language Processing(NLP)

NLP is a rapidly developing field in artificial intelligence that uses deep neural networks to analyse human language and identify important patterns in large text datasets. Numerous applications, including question and answer, machine translation, information retrieval, and others, have made extensive use of NLP. The goal of NLP (NLP), a sub field of linguistics and computer science, is to improve text processing and human-machine interaction. It is used to analyse, produce, and understand human languages in fields like computer science, linguistics, mathematics, electrical and electronic engineering, robotics, artificial intelligence, and psychology. NLP enables users to speak with database management systems and expert systems using natural language by automating translation procedures between human and computer languages. The goal of research is to better understand human language understanding so that computer systems may be developed with appropriate tools and techniques[10] [14]. The following is a detailed discussion of morphological analysis, which is the primary topic of this work and one of the most significant and fundamental NLP applications.

2.3.1 Computational Morphology

Computational morphology is a sub-field of NLP that analyses and generates word forms based on morphemes and grammatical information. It improves natural language understanding systems, develops morphological analyzers for low-resource languages, and explores linguistic and cognitive aspects. Computational morphology includes morphological processing tasks including affixation, replication, and modification, compounding, and merging. It also incorporates Inflectional morphology, derivational morphology, compounding morphology, Cliticization morphology, Reduplication morphology, morphology, and others. The following subsection briefly discusses the types of computational morphology based on [31][32][33]

2.3.1.1 Derivational Morphology

Derivational morphology is the creation of new words from existing ones by adding prefixes or suffixes. For example, in Afaan Oromoo, the word “namummaa” is a noun

derived from the noun “nama”. Derivational morphology is usually related to the lexical and semantic properties of words and phrases. It changes the classes of the word into another class.

2.3.1.2 Compounding Morphology

Compounding focuses on the combination of two or more word stems to form a new word, such as by concatenation or hyphenation. For example, the Afaan Oromoo word “gara-laafessa” which means kind is a compound word composed of the “garaa” which means stomach and “jabeessa” which means soft. Compounding morphology is usually related to the morphological and phonological properties of words and syllables.

2.3.1.3 Reduplication Morphology

This type of morphology focuses on the repetition of a whole or part of a word to form a new word by doubling or copying. For example, the Afaan Oromoo verb “caccabse” means he has broken frequently. Reduplication morphology is usually related to the semantic and pragmatic properties of words and expressions.

2.3.1.4 Inflectional Morphology

As this subsection of this topic has contributed to the proposed system, it deeply discusses every point that is relevant and important for the study. It focuses on the creation of different forms of the same word, such as plural, past tense, or case endings.

2.3.2 Morphological Analysis

Morphological analysis helps in vocabulary learning, spelling, reading comprehension, and language development by examining the structure and construction of words in a language. It is applicable to mathematics, biology, languages, and problem-solving. Morpheme-based morphology, lexeme-based morphology, and word-based morphology are the three different forms of morphology[13] [27]. In NLP, automatic morphological analysis is a crucial method for comprehending the underlying processes of word form creation. Lemmatization, syntactic parsing, information retrieval, machine translation, and text clustering are just a few of the applications for it. Its work is classified into four areas. Finding morpheme boundaries, employing n-gram grammar, uncovering rules through phonological links, and pursuing succinct language analysis. Accurate morphological

analysis necessitates two steps: the development of morphological rules and the morphological analysis technique. There are four categories of computational morphological analyzers: rule-based, statistical, and neural network-based, and hybrid techniques. Grammatical rules are used in rule-based approaches, whereas neural networks are used in neural network-based methods. Statistical approaches make use of statistical data in a variety of NLP tasks[28] [19]. To boost performance, hybrid models incorporate many methodologies. When rule-based morphological analyzers are combined with a neural network-based model, a more precise comparison of rule-based and neural network-based systems is possible Morphological synthesis involves reconstructing surface forms from morpheme glosses using rule-based, machine-learning, and hybrid methodologies. Rule-based strategies generate acceptable word forms, machine learning learns language output, and hybrid approaches use corpus-based techniques[15][26][34].

2.3.3 Approaches for Morphological Analysis

This section delves into computational techniques for morphology research, such as segmentation, grouping, and labelling. Machine learning or corpus-based approaches encompass a variety of techniques, including rule-based, unsupervised, semi-supervised, and fully-supervised techniques, each tackling distinct subtasks. The following subsections describe all existing approaches used in the development of morphological analysis.

2.3.3.1 Rule-Based Approaches

Rule-based techniques utilize handmade rules derived from complex language theories or computer programs to store morphological, lexical, and syntactical information. These systems have disadvantages such as requiring language expertise, possibly over-generation, and being expensive and time-consuming[7]. They do, however, provide advantages over corpus-based techniques, such as less storage, faster accuracy, and greater speed. These approaches, which may be built using computer programs, may provide advantages over traditional methods. Morphological analysis and generation are procedures that divide words into morphemes and assign grammatical and lexical morphemes to different categories and lexemes. Traditionally, the link between levels has been evaluated through phonological criteria. However, by expressing phonological principles as finite state transducers (FSTs), computational morphology has made considerable improvements, allowing for "two-level" morphology. This enables the creation of complicated finite state

systems, such as ordered alteration rules, morphotactic, and a lexicon. Morphological analyzers are developed as finite state transducers (FSTs) based on a formal description of a language's morphology. These analyzers look for all potential root words and morphemes in a word and return all possible morphological parses. A morphological disambiguator can be used to manage ambiguity at the morphology level[35][36]. Foma, which is a finite state environment tool, can generate a rule-based morphological analyzer for a natural language by generating two sets of two-level morphology rules: orthographic rules for spelling and alternation rules, and morphotactic rules for word order.

2.3.3.2 Corpus-Based Approaches

Corpus-based linguistics studies language as it is used by speakers and writers, using diverse corpora as data sources. These methods aim to understand language variation, distribution, and form-function relationships. Applications include linguistic research, teaching, lexicography, computational linguistics, and linguistics. Challenges include ensuring corpus diversity, manual annotation, and ethical considerations. They offer valuable insights into real-world language functions. The following subsection dives into the discussion of all corpus-based approaches like machine learning and deep learning.

2.3.3.2.1 Machine Learning Approaches

Machine learning is a sub-field of artificial intelligence (AI) and computer science that focuses on using data and algorithms to mimic how people learn, progressively improving its accuracy. It is frequently used in intelligent systems that provide artificial intelligence capabilities nowadays. It also refers to a system's ability to learn from problem-specific training data to automate the process of constructing analytical models and solving associated tasks[15]. The science of creating computer systems that seek to get better over time by learning from their experiences is known as machine learning. It includes tasks including diagnosis, planning, robot control, and prediction. Machine learning systems are built with algorithms, and there are two types of algorithms: supervised and unsupervised.

Supervised Machine Learning:- Supervised machine learning creates predictions and generates general hypotheses by transforming input into desired output. It's expensive yet well-liked. It is predicated on datasets with specific tasks.

Unsupervised machine learning:- This technique builds models without sample outputs by using heuristics or probability information from training corpora. It uncovers

correlations and patterns between instances and reduces the expense of perusing annotated corpora[31] [35] [38] [7].

2.3.3.2.2. Deep Learning Approaches

To reduce summarization time, deep learning techniques, which are based on artificial neural networks, are utilized in the domains of speech recognition, NLP, and medical informatics. It builds high-level abstractions from data using algorithms that mimic the operation of the human brain. Deep learning methodologies come in a variety of forms. Nonetheless, some of them are included and covered in this study in the following ways[18] [20] [38].

A) Convolutional Neural Network

Convolutional neural networks (CNNs), which are composed of Convolutional layers, rectified linear units, and pooling layers, are deep learning techniques used for text and picture categorization. Their superiority over existing methods for large-scale visual identification has been demonstrated, and they find use in generative adversarial networks, object detection, semantic segmentation, style transfer, and picture classification. CNN designs are adaptable based on training, and loss functions are used to determine performance[39][40][41][42].

Convolution Layer:- In Convolutional neural network design, a convolution layer is used to extract features from a training dataset. Gradient descent optimization back propagation modifies learn-able parameters like kernels and weights. Rectified linear units, or ReLUs for short, are a mixture of linear and non-linear operations like convolution and activation functions. Convolution is a linear process that applies a kernel to an input tensor in order to build feature maps. Weight sharing is crucial to create local feature patterns translation-invariant and increase model efficiency. Using non-linear activation functions, such as hyperbolic or sigmoid, is necessary to find kernels for a given job.

Pooling Layer:- A pooling layer reduces the in-plane dimensionality of feature maps by down sampling, which introduces translation in-variance and decreases learn-able parameters. It looks like convolution processes, with filter size, strike, and padding functioning as hyper-parameters. The most popular technique, called Max pooling, is obtaining input feature maps, eliminating patches, and producing the highest value found

in each patch. The most popular technique for pooling is the inverted Convolutional neural network (CNN). These layers increase the efficiency and accuracy of feature maps.

Max pooling:- Selects the maximum value within a local region (usually 2x2 or 3x3).

Fully Connected Layer:- The last convolution or pooling layer transforms the output feature to build a 1D array connected to fully connected layers. These layers relate the final outputs, such classification probabilities, to the retrieved features. The final fully connected layer contains the same number of output nodes as classes, following a non-linear function. For multi-class classification tasks, the activation function used to the last fully connected layer is crucial since it normalizes output real values to target class probabilities. It considers a variety of factors, including weights, padding, stride, and kernel size[41][42].

The process of training a network involves figuring out the kernels and weights in convolution layers in order to reduce the differences between output predictions and ground truth labels. Back-propagation and gradient descent optimization algorithms are essential. Data and ground truth labels are key ingredients in deep learning research. There are three data sets at your disposal: a test set, a validation set, and a training set. Validation is used in machine learning to choose and modify the available data. A model is considered to be over-fitting and performs worse on new datasets when it picks up statistical regularities specific to the training set.

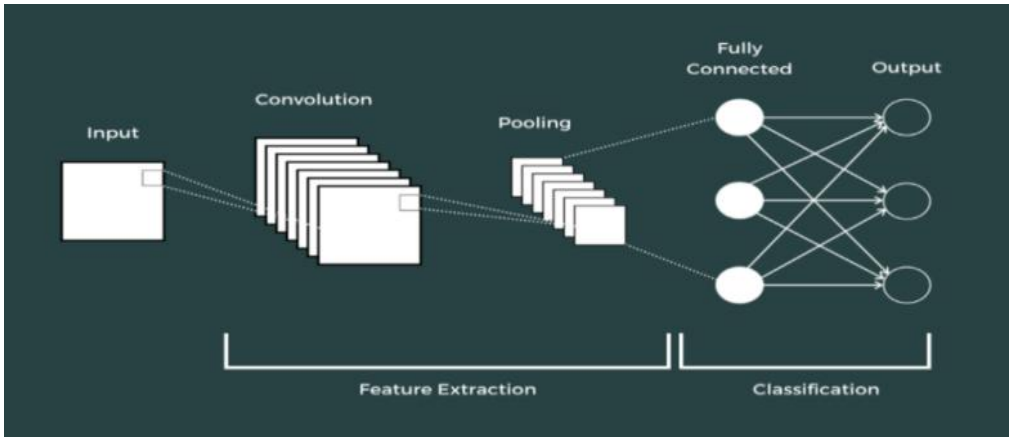


Figure 1: Architecture of CNN[39]

B) Recurrent Neural Network

Recurrent neural networks (RNNs) are artificial neural networks that are made to understand sequential input, such as text, speech, or time series. Information about earlier inputs and outputs is stored in their memory, which affects the input and output. By

sharing parameters across time steps, RNNs increase learning efficiency and are hence useful for machine translation, speech recognition, and natural language processing (NLP) applications. They have shown successful in gathering syntactic and semantic information, showing sequential data, and identifying important information from documents. By providing a non-linear activation function to its hidden layers, RNNs can handle sequential input of varying length[33] [41] [42] [43].

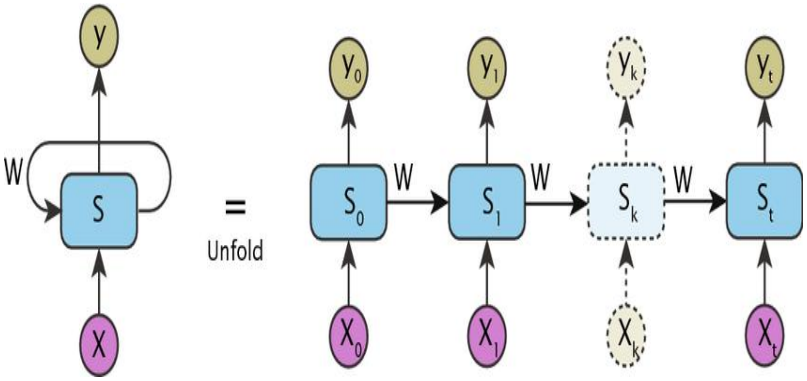


Figure 2: Architecture of RNN[41]

C) Long Short-Term Memory

Hochreiter and Schmidhuber introduced the Long Short-Term Memory (LSTM) unit for the first time in 1997. It is a type of recurrent neural network (RNN) that can process speech, text, and audio information in a sequential fashion. The input layer, the weight layer, the output layer, and the hidden state layer are the four main components of LSTM networks. Layering LSTM networks allows for the construction of deep networks capable of identifying more complex patterns in sequential input. The four gates used by LSTMs are input, forget, output, and input modulating[43] [44] [45]. Compared to ordinary RNNs, LSTMs are superior at capturing long-term dependencies.

Input Gate:- This is a vector that is initialized at random after a series of stages where the input of the current step is the output of the step before. In any scenario where the multiplication result is added to the current memory gate output, the input is subject to element-wise multiplication with the forget gate's output.

Forget Gate:- Forget gate is a neural network with a sigmoid activation function and one layer is called a forget gate. Depending on the sigmoid function, the quantity will determine whether or not the information from the previous state is remembered (when the

sigmoid value is 1) or forgotten (when the sigmoid value is 0). The bias, the input vector, the previous block's output, and the remembered data are the four inputs of the forget gate.

Memory Gate:- The memory gate, which consists of two neural networks, controls the relationship between freshly learns and remembered information. The distinct bias of the first network makes it special. The second neural network's tanh activation function generates the new information.

Output Gates:- The amount of new information supplied to the next LSTM unit is controlled by the sigmoid activation function of the output gates, which consider the input vector, the previous hidden state, the incoming information, and the bias as input. Multiply the output of the sigmoid function by the tanh of the incoming data to obtain the outcome of the current block.

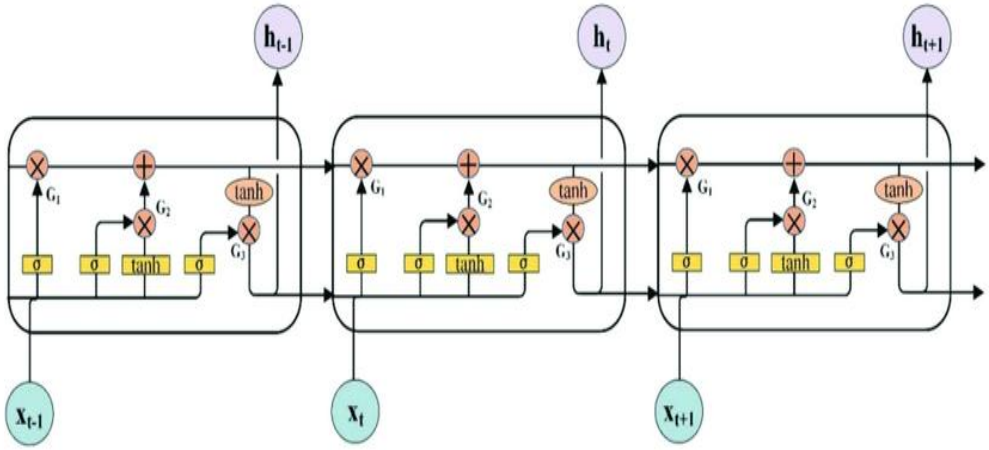


Figure 3: Architecture of LSTM[44]

D) Bidirectional Long Short-Term Memory

Recurrent neural networks (RNNs) like bidirectional LSTM analyse sequential data such as audio, voice, or text signals. It is composed of two LSTM layers that process input in two ways: forward and backward, respectively, capturing dependencies and context from both directions. Machine translation, text categorization, and sentiment analysis are some of its uses. Speech recognition and NLP are two examples of tasks where BiLSTM improves performance[45] [47] [48].

Input sequence:- This is the order in which input, such as a time series, audio signal, or phrase, is supplied into the network. Typically, the input sequence is shown as a matrix of feature vectors, with each row denoting a time step and each column a feature dimension.

Forward LSTM layer:- For each time step, this LSTM layer generates an output vector and a hidden state vector by processing the input sequence from left to right. The output vector shows the prediction or classification for the current time step, whilst the hidden state vector encodes the data from the past ten time steps in a forward direction.

Backward LSTM layer:- The backward LSTM layer generates an output vector and a hidden state vector for each time step by processing the input sequence from right to left. The output vector shows the prediction or classification for the current time step, while the hidden state vector encodes data from the subsequent time steps in a backward direction.

Backward LSTM Output layer:- The output layer generates the final output for each time step by combining the output vectors from the two LSTM layers. Depending on the job, the output layer may consist of a softmax layer, an attention mechanism, a fully linked layer, or any other appropriate layer. The output layer can also combine the output vectors from both directions using a variety of techniques, including element-wise multiplication, concatenation, summation, and averaging.

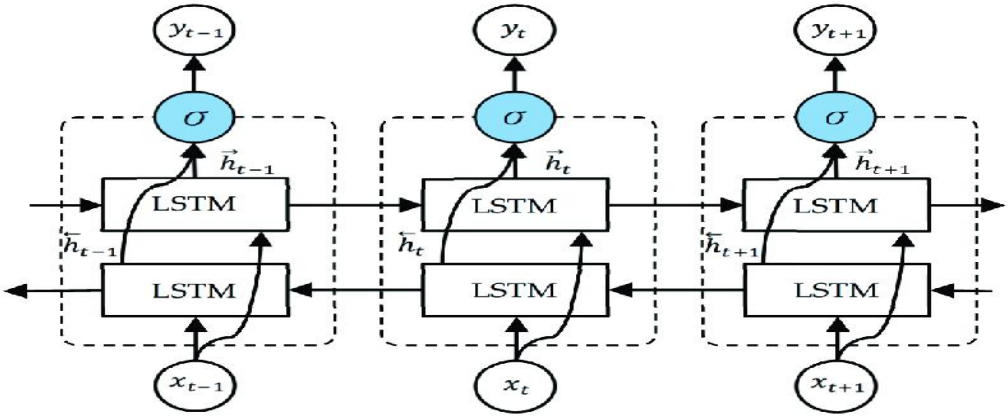


Figure 4: Architecture of BLSTM[47]

E) Gated Recurrent Unit

In order to express long-term connections in input and adapt to diverse time scales, recurrent neural networks (RNNs) known as GRUs use mathematical gates. They are simplified long short-term memory (LSTMs) with an update gate and a reset gate in place of explicit memory. Unlike voice and text, GRUs do not experience disappearing or bursting gradients while processing sequential input, such as time series, and may retain knowledge over long periods of time. They function well in natural language processing (NLP) activities including language modelling, machine translation, and speech

recognition. To update the hidden state selectively at each time step, GRUs use two gating mechanisms: update Gate and reset Gate. GRUs balance complexity and usefulness in a variety of NLP applications. Compared to LSTMs, GRUs have fewer gates and do not maintain an Internal Cell State, making them computationally efficient and effective for capturing dependencies in sequential data[48] [36] [37] [26].

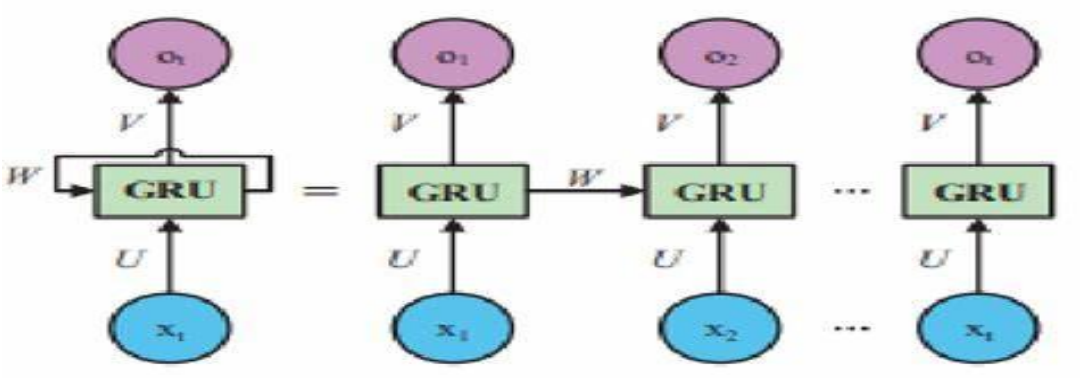


Figure 5: Architecture of GRU[26]

F) Bidirectional GRU

The Bidirectional Gated Recurrent Unit is a well-liked sequence processing paradigm for natural language processing (NLP) and other sequential applications (BiGRU). It is a two-way extension of the original GRU that processes the input sequence in both directions using two GRUs. Words may be interpreted in their context based on words that come before and after them since their bidirectional nature helps to capture context from both sides[36]. BiGRU operates in two phases:

Forward pass:- The forward GRU evaluates the input sequence in a left-to-right manner in the forward pass, modifies its hidden state based on the input at hand and the previous hidden state, and saves context-related data in the final hidden state.

Backward pass:- In this mode, the backward GRU processes the input sequence in a right-to-left direction, updating its hidden state as it goes. This mode's last concealed state contains information about the context that comes next. BiGRU embeddings are useful for downstream applications such as machine translation, sentiment analysis, and sequence labelling. Machine translation, sentiment analysis, relationship extraction, NER, and part-of-speech tagging are among the applications for BiGRU. Because BiGRU includes the power of GRUs in both forward and backward orientations, it is a valuable tool for gathering context in sequential data[36].

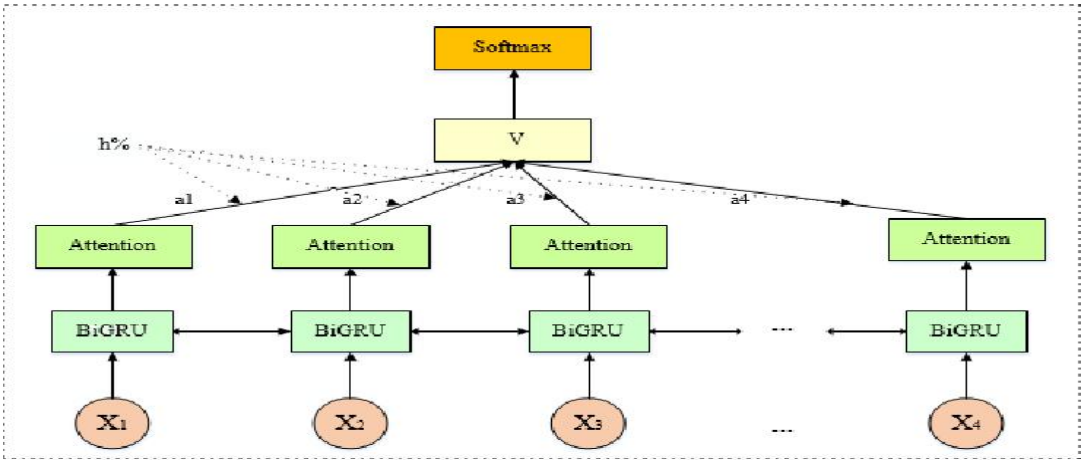


Figure 6: Architecture of BiGRU[36]

G) Transformer

Transformers are a deep learning architecture that has revolutionized NLP (NLP) and other sequence-to-sequence tasks. They address the limitations of traditional sequence models, such as vanishing gradients and parallelization difficulties, by using a novel attention mechanism. Transformers consist of a self-attention mechanism, multiple-head attention, and positional encoding, which allow the model to weigh the importance of different words in the input sequence when predicting an output word. Transformers consist of an encoder stack for input and a decoder stack for output, each containing multiple layers of self-attention and feed-forward neural networks. Residual connections and layer normalization stabilize training and improve gradient flow. The decoder stack creates the output sequence by paying attention to the encoder's hidden states and forecasting the next word, while the encoder stack analyses the input sequence using feed-forward layers and self-attention. Machine translation, text summaries, question answering, audio recognition, language modelling, picture captioning, and more applications require transformers[49]. They get state-of-the-art performance in a variety of NLP tasks by using self-attention and parallelization to capture long-range relationships. Back-propagation is used to train the complete architecture from start to finish.

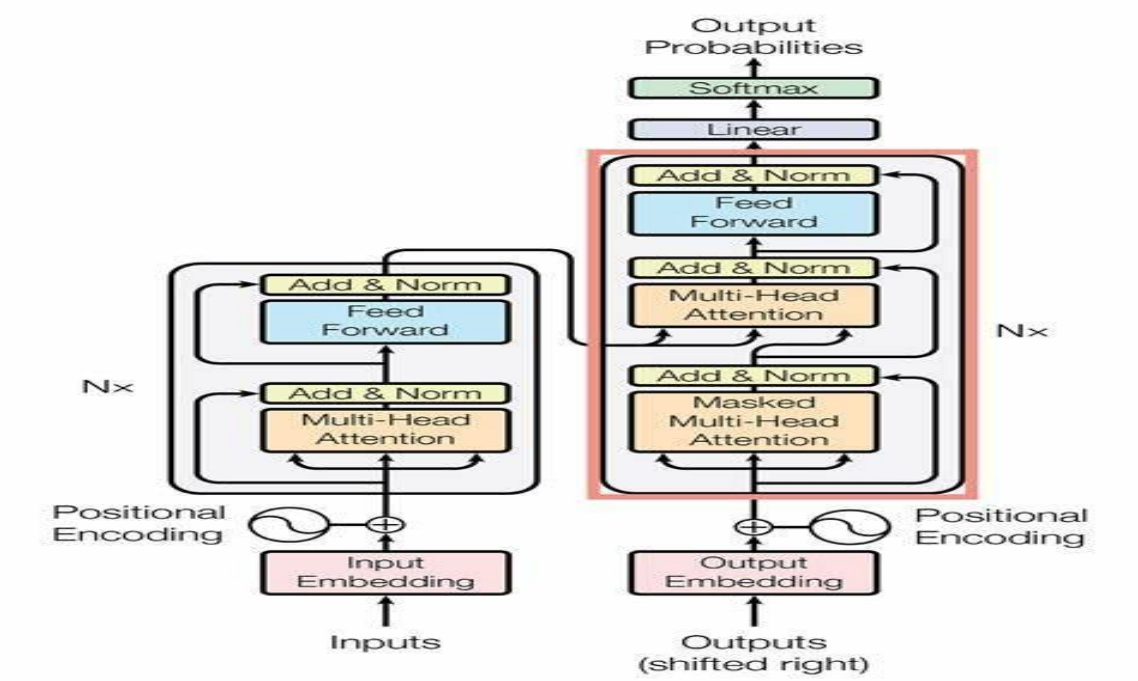


Figure 7: Architecture for Transformer[49]

H) Auto Encoder

An artificial neural network called an auto-encoder is used for unsupervised learning to acquire effective representations of input data. It is made up of two primary parts: a decoder that reconstructs the input data from the encoded representation and an encoder that converts input data into a lower-dimensional representation. Learning a concise and understandable representation of the incoming data is the aim. The input data itself is used to train the auto-encoder, and the difference between the input and the reconstructed output is measured by a loss function. Back-propagation adjusts the encoder and decoder weights to lessen this loss. Auto encoders are used in many applications such as image compression, anomaly detection, noise reduction, Variational auto-encoders (VAEs), denoising auto-encoders, sparse auto-encoders, and feature learning. They can, however, pick up simple fixes like copying input to output. For auto-encoders to work properly, regularization and architectural design must be done correctly. In conclusion, auto-encoders are helpful for dimensionality reduction, noise reduction, and anomaly detection because they can learn compact representations of data [50] [51] [52].

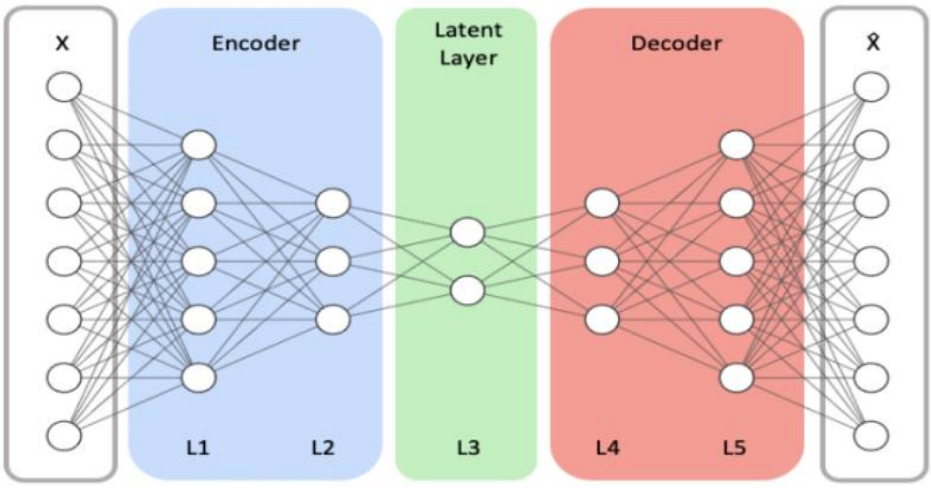


Figure 8: Architecture for Auto-encoder[52]

2.4 Evaluation Methods

The performance, efficacy, and impact of different treatments, programme, models, or processes can be better understood with the use of evaluation techniques. NLP (NLP morphological analysis) is evaluated using a variety of metrics, including word error rate, cosine similarity, Jaccard similarity, and perplexity. Data splitting, cross-validation, manual and automated verification datasets (validation and testing), and visual examination are among the frequently used techniques. Loss functions, accuracy, precision, recall, F1-Score, Mean Absolute Error (MAE), Mean Squared Error (MSE) for regression, and visual inspection to comprehend model behaviour are among the metrics used to measure model performance. The unique setting, task, and data that are accessible all influence the assessment technique selection. Together, these techniques aid in evaluating and raising the calibre of models and actions[16][5] [6] [12].

2.5 Afaan Oromoo

2.5.1 Background

The Cushitic language Afaan Oromoo, sometimes called Oromiffa, is spoken in Ethiopia, Somalia, Sudan, Tanzania, and Kenya. It is a member of the six main branches of the Afro-Asiatic or Harnito-Semitic language family. Ethiopians speak Afaan Oromoo, Somali, Sidama, Hadiyya, and Afar-Sabo as their primary Cushitic languages. After Hausa and Arabic, it is the third most widely spoken Afro-Asian language globally. The Oromia National Regional State is home to the native speakers of Afaan Oromoo, which is taught

in elementary, secondary, preparatory, and university settings. In Ethiopia, it is also often used as a lingua franca and a means of communication between many ethnic groups. Despite its widespread distribution and benefits, Afaan Oromoo is the most resource-scarce language in Africa, including resources for research works[53] [54] [20].

2.5.2 Afaan Oromoo writing System

Since1991, the Afaan Oromoo language has been written using a modified Latin script known as Qubee Afaan Oromoo. This phonetic system has distinct word combinations, syllable structures, and consonants and vowels. There are 33 letters in the alphabet: five vowel letters and seven mixed consonant letters. With capital and tiny characters that resemble the English alphabet, Afaan Oromoo has started to exhibit consonant germination (double consonants). The language has unique morphemes, Allomorphs, morphotactic, word classes, and sentences [6] [20].

2.5.2.1 Afaan Oromoo alphabets

Linguistic experts developed different Afaan Oromoo alphabets using different methods like Arabic, Ethiopic, and Roman characters. Sheikh Bakri Sapalo created an indigenous Oromoo alphabet in 1956, with the help of Oromoo students overseas. In 1991, the Latin alphabet based Qubee was adopted as the official orthography, allowing local languages for education and study. Afaan Oromoo uses 33 fundamental letters, 5 vowels, and 24 consonants, without the common foreign terms 'p', 'v', and 'z'. The language is phonetic with a straightforward Latin script, with no skipped or un-pronounced sounds. Vowels and consonants are represented by five basic letters, with few special combinations[9] [55].

Table 1: Afaan Oromoo alphabets

Alphabets	Sounds	Alphabets	Sounds	Alphabets	Sounds	Alphabets	Sounds	Alphabets	Sounds	Alphabets	Sounds	Alphabets	Sounds
A a	[aa] Like Ask	B b	[baa] Like Bird	C c	[caa] Like Cat	D d	[daa] Like Dam	E e	[ee] Like Ate	F f	[faa] Like fungi	G g	[gaa] Like Gun
Hh	[haa] Like Hat	I i	[ii] Like Indian	J j	[jaa] Like Just	K k	[kaa] Like Cast	L l	[la] Like Life	M m	[ma] Like Man	N n	[naa] Like Nasty
O o	[oo] Like Old	P p	[pee] Like Past	Q q	[quu] Like Quit	R r	[ra] Like Rat	S s	[saa] Like Sad	T t	[taa] Like tally	U u	[uu] Like urge
V v	[vau] Like Vary	W w	[wee] Like Want	X x	[taa] Like xender	Y y	[y] Like You	Z z	[zay] Like That	CH ch	[chaa] Like chat	DH dh	[dhaa] Like
SH sh	[shaa] Like Shy	NY ny	[nyaa] Like …	PH ph	[phaa] Like Pente								

2.5.2.2 Afaan Oromoo syllables

Syllables are groupings of speech sounds used to organize speech, and each language has a unique syllabic structure. In Afaan Oromoo, only digraphs can contain more than two consonants, with seven possible structures: CV, CVV, CVC, CVVC, VC, VVC, and V. Vowels and consonants are represented by five basic letters, with few special combinations[7].

2.5.2.3 Afaan Oromoo Word and Word Classes

The building blocks of a language are words, which are the smallest meaning-carrying units in speech. A set of words having similar morphology, responsibilities in sentence construction, and grammatical traits is called a word class or category. Similar terms include parts-of-speech, morphological class, and semantic tag. Depending on their feature and category, words have varying contributions to sentences. The five primary grammatical heads in Afaan Oromoo, each with extra subclasses, are utilized in phrase construction. An infinite number of subclasses can be created by further subdividing these subclasses. Based on their classification, Afaan Oromoo word classes can be classified as either closed or open [8] [7].

2.5.2.3.1 Open Word Classes

Open word classes, also known as content words or lexical words, are a category of words that can be constantly expanded to include new concepts and words. They are constantly evolving, such as with computer technology advancements, resulting in new nouns and verbs. Closed word classes do not include new conjunctions, prepositions, or determiners. Open word classes are expanded through operations like compounding, derivation, coining, and borrowing[7].

Nouns: are words or lexical items that can refer to any concrete or abstract entity, such as a person, place, thing, idea, or quality (such as gladness). They are known as naming words and are the most prevalent component of speech.

Verbs: are words that refer to an event (walk) or a state of being (happen) (be). A group of words cannot be a clause or sentence without a verb.

Adjectives: are nouns or pronoun modifiers. They clarify the meaning of a different word (noun, pronoun, etc.).

Adverbs: are the modifiers of an adverb, adjective, or verb (e.g. very, quite, frequently, etc.). They improve the precision of language.

2.5.2.3.2 Closed Word classes

Closed word classes, also referred to as functional words, are a group of words with primarily grammatical functions and little inherent meaning. They serve as identifiers or directions for sentence structure. Contrary to open word classes, closed word classes add new members very infrequently, even though this is sometimes necessary because it is extremely rare for new function words to be created during speech. They are very resistant to the introduction of new things and don't readily accept new members. This group of words is largely fixed, and we use them quite frequently to connect various clauses in a sentence or to establish a sense of coherence (continuity) between various clauses in written or spoken language. So, among the most crucial words to learn are those. These classes include:

Pronouns: are appropriate in place of nouns or noun phrases (them, he). Since pronouns take the place of nouns and noun phrases, sentences become shorter and clearer.

Determiners (articles, demonstratives, etc.): are, in some languages, grammatical indicators of definiteness (the) or indefiniteness (a, an). The article can be delivered using inflections, so it is not always included in the list of parts of speech. For example, Afaan Oromoo lacks articles.

Ad-positions (prepositions and post-positions): are words that provide context for other words in a phrase or sentence and help with syntactic structure (in, of). They illustrate how a noun or a pronoun relates to other words in the sentence.

Conjunctions: are syntactic connectors that connect sentences, clauses, or phrases (e.g., and, but, or). They link individual words or word groups.

2.5.2.4 Afaan Oromoo Sentences

Sentences in Afaan Oromoo are composed words that convey a full notion or thinking. They are composed of verbal, noun, and occasionally prepositional phrases. A prepositional phrase alters a verb or noun, a noun phrase expresses a subject, and a verb phrase expresses an action. The Latin writing system is used to end sentences in Afaan Oromoo. Full stops, question marks, exclamation marks, commas, and semicolons are among the options. Based on their structure and purpose, sentences may be divided into four categories: simple, compound, complex, and compound-complex sentences. Compound sentences include two or more independent clauses connected by a coordinating conjunction, whereas simple sentences have a single clause with a subject and a predicate. One independent clause, one or more dependent clauses, and a subordinating conjunction link complex sentences, whereas two or more independent clauses, one or more dependent clauses, and coordinating and subordinating conjunctions combine compound-complex sentences[55] [25].

2.6 Afaan Oromoo word Formation

There are two sorts of words: simple and complex. Simple words are made up of stems, prefixes, and suffixes, while compound words are made up of two or more distinct words. Words are categorized using open and closed classes; nouns, adjectives, and verbs are the three open classes found in the majority of languages. Function words form closed sets of words that are not expandable by common word-formation patterns. Examples of these words include determiners, conjunctions, pronouns, and ad positions. Historically, linguists have categorized words according to their functions by examining two related types of

evidence: the way a word influences a phrase's meaning and the actual syntactic structures in which it may be employed. The initial step in building a morphological analyzer is defining word classes and grammatical data. The five word classes in Afaan Oromoo are conjunction, adjective, adverb, noun, and verb. The word classes in Afaan Oromoo are gochibsa (verb), maqaa (noun), maqibsa (adjective), durduubee (affix), qabsiistuu (a conjugation), and bamaqaa (pronoun). The words in these word classes have either undergone inflection or derivation[13] [30].

2.6.1 Derivation

A root word and stem are combined with one or more affixes during the process of word formation to create a new word known as a derived word.

2.6.1.1 Noun Derivation

Nouns are words that are used to name or identify specific instances of things, people, places, or ideas. Normalization is the process of removing a noun from another word class, and nominalizers are the affixes that are used to do this. There is a sizable stock of nominals in Afaan Oromoo that are derived from adjectival, verbal, and nominal bases. Suffixes involved in the derivation of nouns in Afaan Oromoo are classified into different groups based on the types of word classes they change into nouns.

Group 1) /-eenya/, /-ina/

These suffixes are used to derive nouns from adjectives.

For example adii/white->addeenya/whiteness

Group 2) /-a/, /-aa/, /-ee/, /-ii/, /-sa/, /-aatii/, /-cha/, /-choo/, /-maata/, /-nsa/, /-noo/

These suffixes are used to derive nouns from verbs.

For example:- Bareedi/be beauty-> Bareedaa/handsome

Group 3) /-ummaa/, /-ooma/ -aa/ -a /-aatii

These suffixes are used to derive nouns from other nouns. It changes the concrete noun (maqaa waan qabatamaa) to abstract noun (maqaa waan yaadaan jiruu)

For example: nama/human->namummaa/humanity

2.6.1.2 Verb Derivation

Verbs are words or word combinations that convey action, a state of being, and/or a connection between two objects. Based on the kind of word class they convert into verbs, Afaan Oromoo suffixes that are used in the derivation of verbs (verbilizers) are divided into various groups.

Group 1) /-oom-/, /-aa'-/, /-a'-/

These verbalisers are used to derive verbs from nouns and adjectives.

Diimaa/red->diimate/ he become red

Deemi/go->deemsise/make it to go

Galata/press->galatoomi/thanks

Group 2) /-at-/, /-am-/, /-sis-/, /-siis-/, and /-s-/

These verbalisers are used to derive verbs from adjectives and other verbs.

Gurraacha/black->gurraachate/he become black

Deemi/go->deemsisani/they let them go

2.6.1.3 Adjectives Derivation

Adjectives are noun-descriptive or noun-modifying words (and pronouns). Adjectivization refers to creating adjectives from other lexical categories. The adjective "reddened" can be derived from stative verbs like "diim-at," which means "become red." Adjectives in Afaan Oromoo can be created by taking adjectivizers like /-aa/, /-tuu/, /-eessa/, and /-eettii/ from verbs. Due to the limited number of adjectives in Afaan Oromoo, adjectives are not derivatives of nouns or verbs[16] [20].

2.6.2 Inflection

Inflection is crucial in computational linguistics and natural language processing for tasks like morphological analysis, machine translation, and speech recognition, especially in languages like Afaan Oromoo due to their rich morphological structure.

2.6.2.1 Noun Inflection (Maqaa)

Afaan Oromoo nouns are primarily vowel-ending, with some exceptions ending in consonants. Nouns have inherent inflectional categories that signify various grammatical functions, such as number, gender, definiteness, case, and focus markers. Case markers are relational, while number and gender markers are inherent categories. Inflectional morphology is related to the grammatical and syntactic properties of words and sentences. For example, the word "nama" can be inflected to “namuma”, “namarraa”, “namaarraa”,“namni”, “namoota”, “namaaf”, or “namatu” depending on the number, case, focus, and other factors. Inflectional morphology is often combined with the stem to indicate grammatical functions like number, gender, definiteness, and case. Direct and indirect objects can be optionally followed by the noun, and affixes like -eenya, -ina, -ummaa, and ota are also used[6].

2.6.2.1.1 Number

Afaan Oromoo the most common number features are plural and singular. The language uses different suffixes to form plural nouns. These suffixes are different from dialect to dialect and used alternatively by replacing one with another pluralization suffix or concatenative one after another. Nonetheless, certain collective nouns are exclusively found in the plural, as in the cases of ijoollee (children) and hamaamota (wedding slaves), while other nouns, like ilkaan (tooth/teeth) and bishaan (water), have both single and plural forms. Certain inherently singular nouns, like aduu and Waaqa, become meaningless when they get pluralization suffixes. The following are typical suffix morphemes that indicate a noun's plural form: -oota -lee -wwan - yyii -een -oolii -eetii -ii -oo -iin -an. These plural indicators are grouped according to general use and differ according to the parent noun's semantic make up. Barkessa divided the pluralization suffixes into four categories, taking into account their variations.

Group 1) oota, oolee, oolii, ilee (ota, olee, olii, ilee): These suffixes delete all last vowels of the citation form of nouns and then attach them. The variations between the long form and short form of this group of suffixes are based on the length penultimate syllable (vowel present in the syllable that precedes the last syllable) of a base noun. Thus, when the penultimate syllable contains a short vowel, the long form (oota, oolee, oolii, iilee) is suffixed but when it contains a long vowel short form (ota, olee, olii, ilee) is suffixed for plurality of the nouns[21].

Table 2: Afaan Oromoo group 1: number markers

Pl-Noun	Sg-Noun	Root	Suffix	Gloss
Namoota	Nama	Nam	oota	man → men
Gaalota	Gaala	Gaal	ota	camel →camels
Jaarsolee	Jaarsa	Jaars	olee	old man →old men
kitaabilee	Kitaaba	Kitaab	ilee	book → books
waatilee	Waatii	Waat	ilee	calf→calfs

Group 2) –wwan, lee

These suffixes are used with nouns terminating in long vowels without deleting the final vowels.

Table 3: Afaan Oromoo group 2 number markers

Pl-Noun	Sg-Noun	Root	Suffix	Gloss
Saawwan	sa'a	sa'	wwan	cow → cows
Ergaawwan	ergaa	erg	wwan	message →messages
kennaawwan	Kennaa	Ken	Wwan	gift → gifts
loowwan	loon	loon	wwan	cow →cows
lammiilee	lammii	lammii	lee	citizen→citizens
naannoolee	naannoo	naannoo	lee	region→ regions

Group 3) -an, -een, -iin

All nouns that take this group of plural maker suffixes make plural nouns by doubling the consonant in the last syllable. These nouns mostly end in the consonantal phoneme like l, m, and r in the case of -an. And b, d, g, k, n in the case of -een or -iin followed by a short vowel. -iin and -een reflect only dialectal differences but their function is similar. In Afaan Oromoo, more than two consonants cannot come consecutively in a single word.

Table 4: Afaan Oromoo group3 number markers

Pl-Noun	Sg-Noun	Root	Suffix	Gloss
Eessumman	Eessuma	eessum	An	uncle → uncles
Ilmaan	Ilma	ilm	An	son →sons
Mukkeen	Muka	muk	Een	plant → plants
Manniin	Mana	man	Iin	house →houses

Group 4) -eyyii

This plural maker suffix forms plural by dropping complex endings that form nouns like -(e)essa, (e)ensa, (e)ettii and attaching the plural maker -(e)eyyii. Although the categories of suffixes discussed above are the most common ones, it is difficult to categorize a given noun under one of the above categories as members of different categories can be alternatively attached.

Table 5: Afaan Oromoo group4 number markers

Plural Noun	Singular Noun	Stem	Suffix	Glossary
Dargaggeeyyii	Dargaggeettii	Dargag	eyyii	Youth
Ogeeyyii	Ogeessa	Og	eyyii	Wise men
Buleeyyii	Buleettii	Bul	eyyii	Elders
Gara-laafeyyii	Gara-laafessa	Laaf	eyyii	Kinds

In addition to the above-discussed plural markers, the Afaan Oromoo proper noun uses the associative marker fa'a, which means 'and others', (-faa) to identify a group referring to humans. For instance, 'Caalaa faa' means Chala and others. The morpheme -faa is suffixed to someone's name in the group and suffixed on the interrogative pronoun eenyu 'who' in questions to form eenyuunfaa 'whom and others.

2.6.2.1.2 Gender

There are two types of gender in Afaan Oromoo: Masculine and Feminine. A limited group of nouns differed by using different suffixes for masculine and feminine forms because

some of the nouns are distinct (distinguished semantically) as in Eessuma or Wasiila or Abeeraa ‘uncle’ and adaadaa or haboo ‘aunt’. Some of the nouns are epicene as in hattuu ‘thief’ while some of them are expressed using contrary adjectives or verbs that follow them such as kormaa ‘male’ and dhaltuu ‘female’ as in lukkuu kormaa ‘rooster’ and lukkuu dhaltuu ‘hen’ or sareen dhufe ‘the dog came/masculine’ and sareen dhufte ‘the dog came/for feminine’. Some of the language’s nouns are considered as natural male or female gender. For instance, biiftuu ‘sun’ is considered female while garba ‘ocean’ and Waaqa ‘God’ are considered male gender grammatically. Some of the Afaan Oromoo gender markers suffix pairs are -ssa /-ttii, -aa/ (d) tuu, icha/ittii, etc.

Table 6: Afaan Oromoo gender noun sample

Noun	Root	Suffix	gender	Gloss
Barataa	barat	aa	M	student/M
Barattuu	barat	tuu	F	student/F
Bareedaa	bareed	aa	M	handsome/M
Bareedduu	bareed	tuu	F	beautiful/F
Ogeessa	ogee	ssa	M	wise/M
Ogeettii	ogee	ttii	F	wise/F

2.6.2.1.3 Definiteness

Some languages use articles or other determiners to express definiteness. For instance, English uses the definite article ‘the’ or determiners such as ‘all’, ‘this’, ‘that’, and so on, to express definiteness. English has also an indefinite article a/an. Although Afaan Oromoo has no indefinite articles (corresponding to English a/an), it indicates definiteness (English corresponding to the) with suffixes on the noun: -icha for masculine nouns and - ittii for feminine nouns. Nouns inflected by these suffixes drop the end vowel before adding these suffixes. For example: Nama ‘human being’ becomes namicha 'the man' or namittii ‘the woman’. Afaan Oromoo animate nouns that can take either gender, these definite suffixes may indicate the intended gender: qaalluu 'priest', qaallicha 'the priest (masculine)', qallittii 'the priest (feminine)'. Unlike in English, in Afaan Oromoo definite suffixes do not to co-occur with the plural suffixes. Thus, definiteness suffixes of Afaan Oromoo refer to the

singular markers. Definiteness in Afaan Oromoo is also expressed using demonstrative pronouns like Kun (this), and Sun (that).

Table 7: Afaan Oromoo definiteness noun sample

Noun	Root	Suffix	Gender	Gloss
Namicha	Nam	icha	M	the man
Namittii	Nam	ittii	F	the woman
Saricha	Sar	icha	M	the dog/M
Sarittii	Sar	ittii	F	the dog/F

2.6.2.1.4 Cases

Case is a grammatical category of nouns, pronouns, or determiners that indicates the nature of their relationship to the verb or other function-like expression in sentences[6]. Case marking is implemented in various languages, in various ways specifically word order, inflection, and ad-position. Number of cases varies from language to language. Afaan Oromoo has extensive case systems, with nouns, pronouns, adjectives, and determiners all inflecting (usually using different suffixes) to indicate their case. Fundamentally, when we come to noun cases, the Afaan Oromoo case is based on changes to the noun to indicate the noun's role in the sentence. It is dominated by an inflectional case system. Inflectional patterns may depend on a variety of factors, such as gender, number, and phonological environment. In addition to, the inflection case system Afaan Oromoo is also characterized by the Ad-positional case system. The case is understood by considering the placement of the noun and the syntactic function it conveys. Although some literature shows Afaan Oromoo nominals (nouns, pronouns, and adjectives) are inflected for four to six cases; they are subject to the following case forms[16] [20].

Base form: Ø inflected: NULL inflected Base form (noun base form) is usually given in dictionaries because it is that form of a noun, adjective, or pronoun that does not have any case ending or suffix. It is used for isolated citations, a direct object, a predicate nominal and to express the different oblique cases for the subject in focus. For example ilkaan (teeth), funyaan (nose)

Nominative case or subject form: A nominative case is a form of the noun used for subjects in clauses, consisting of four Allomorphs: '-n', '-ni', -i, and Ø. Because the nouns are phonological, these Allomorphs have different phonological realizations. If the noun has a long vowel at the end, the marker -n comes after it. The last vowel is omitted and -na is inserted if the noun ends in short vowels with a single consonant. The nominative is the same as the base form (Ø is suffixed) if the noun ends in a certain consonant. Ad-positional case systems pair nouns with prepositional and post positional terms that indicate case ad-positions. As seen in Barri and Gaalli, the nominative case Allomorphs /-ni/ transforms phonologically to -ti or doubled final consonant.

Table 8: Afaan Oromoo Nominative case marker

Base form	Inflected form	Glossary
Nama	Namni	Human
Mana	Manni	House
Lafa	Lafti	Land
Maqaa	Maqaan	Name
Dhara	Dharri	Lie

Accusative/Absolutive case: Object form Absolutive case, the unmarked noun in Oromoo is an underlying/inherent noun that occurs in the object position without an inflectional suffix. The absolutive case is considered as the primitive form (base form) of nouns. Sometimes an object seems to be marked if the underlying nouns end with a certain consonant. For example, Caalaan (Caalaa-nom) Galaaniin (abs) waame 'Chala called Galan'.

Dative case: Nouns that denote the beneficiary or the recipient of an event are represented by the indirect object known as the dative case. It serves as semantic criterion by taking the place either before or after the direct object and indicating "for whom" or "to whom" the action is performed.The dative case can be expressed by: Lengthening of a short final

vowel, Lengthening of a short final vowel and adding of a suffix -f, -dhaa, or -dhaaf (to nouns with final long vowel), Adding the suffix -ii to nouns terminating in a consonant,

Adding the suffix (tii) f to a genitive construction, and Adding the suffix -tti (irrespective of the spelling of the noun).

Table 9: Afaan Oromoo dative case sample

Base form	Inflected form	Glossary
Nama	Namaa	For human
Daa'ima	Daa'imaa/f	For child
Jabaa	Jabaadhaa/f	For Jeba
ilkaan	Ilkaanii	For teeth
Mana barumsaa	Mana barumsaatiif	For school
Loon	Loonitti	For cows

There are two fundamental inflectional morphemes to mark dative case in Afaan Oromoo: -f and -tti. All lengthening of the final vowel, dhaa, dhaaf, and tiif are the variations of f. The dative form of a verb infinitive (which acts like a noun in Afaan Oromoo) indicates purpose.

Genetive case: Nouns (nominals) are indicated in the genitive case to indicate possession. Naturally, the genitive case includes purpose, source, reference, and other elements in addition to ownership. The genitive in Afaan Oromoo is defined by the following sequence: Possessed-possessor. It has no unique marking on it. The genitive case is formed in two ways:-

- ✧ By succeeding relative particle kan/tan and lengthening the last vowel of possessor nouns, By suffixing -i to the final consonant of the possessor noun or by leaving a final long vowel of possessor nouns unchanged and
- ✧ By putting side by side the thing possessed and the possessor in that sequence and lengthening the final vowel of the possessor if it is short or suffixing -I after a consonant. If the noun has more than one qualifier normally, only the last part of the

noun phrase has the genitive marker. Vowel length is the marker of the genitive case of a noun.

Table 10: Afaan Oromoo genitive case marker

Base possessed	Base possessor	Glossary
Mana (base)	Namaa	Some one's house
Manni (nominal)	Namaa	The house of some one
Kitaaba (base)	Barataa	Student's textbook
Kitaabni(nominal)	Barataa	The text book of some one

Instrumental case: Nouns that denote the instrument ('with'), the means ('by'), the agent ('by'), the cause, or the moment of an occurrence are all represented by the instrumental case. Instrumental case in Afaan Oromoo is indicated by -n and its Allomorphs. The instrumental case is marked in the following ways: Lengthening of a short final vowel and adding of the suffix -n or adding of the suffix -n to nouns with a final long vowel, Adding the suffix -dhaan to nouns with a final long vowel, Adding the suffix -iin to nouns terminating in consonant, and adding of the suffix -tiin to a genitive construction.

Table 11: Afaan Oromoo instrumental case marker

Base form	Inflected form	Gloss
Nama	Namaan	By man
Daa'ima	Daa'imaan	By baby
Jabaa	Jabaadhaan	By jabaa
ilkaan	Ilkaaniin	By teeth/tooth
ulee	Uleedhaan	By stick
Loon	Looniin	By cows

Ablative case: The source of an event is represented by the ablative case. It closely resembles English. It conveys the beginning, origin, or source of a movement. It takes on the following forms: Vowel lengthening is the process of prolonging a word that ends in a short vowel. When a word finishes in a long vowel, -dhaa is added. -ii is added when the

word ends in a consonant (like for the genitive: tii). This is a significant feature since it separates the ablative case from the object form of nouns in their morphology, as in Adaamaa dhufe "he came to Adama" and Adaamaadhaa dhufe "he came from Adama." In the genitive, -tii is appended after a noun.

Table 12: Afaan Oromoo ablative case marker

Base form	Inflected form	Glossary
Jimma	Jimmaa	From Jimma
Adaamaa	Adaamaadhaa	From Adama
Harar	hararii	From Harar
Ambo	Amborraa	From Ambo

Locative case: The locative, which approximately translates to English "at," is used for nouns that denote broad locations of objectives, events, or conditions. Afaan Oromoo employs ad positions for more precise placements. The suffix -tti designates the locative case.

Table 13: Afaan Oromoo locative case marker

Base form	Inflected form	Gloss
Naqamte	Naqamtetti	At nekemte
Lafa	Lafatti	At land
Ala	Alatti	Loonitti
Maqi	Maqitti	At meki
Mana barumsaa	Mana barumsaatiif	For school

Vocative case: Vocative is a verbal means of calling attention or incorporating strong feelings. In Afaan Oromoo, there are various ways of marking the vocative case. The common ones are using the word ‘yaa’ and the suffix ‘na’. The suffix ‘na’ which is our focus case is appended to a noun that is two syllables and ends in a short vowel with

harmonic occurrence of vowels as in namana ‘you guy!’ and jarana ‘you guys!’ Its full word form ‘nana’ is used after nouns that end in long vowels as in gurbaa nana ‘you boy!’.

Table 14: Afaan Oromoo vocative case marker

Base form	Inflected form	glossary
Nama	Namana	You Man
Jara	Jarana	You Guys
Warra	Warrana	You House Men
Intala	Intalannana	You Leddy
Maraattuu	Maraattuunnana	You Mud
Grubaa	Gurbaannana	You Guy

2.6.2.1.5 Focus (Xiyyeeffannoo)

In Afaan Oromoo, focus suffixes are attached to the last constituent of a noun. The focus markers attached nouns are ‘-tu’, ‘-uma’, and ‘-dhuma’. When the ‘-tu’ focus marker is attached to a noun, no case marker is allowed. For example, in nam-a-tu ‘it is a man’, the last marker ‘tu’ shows a focus marker.

2.6.2.2 Adjective Inflections

An adjective is a term that provides further details about a noun. Adjectives in Afaan Oromoo can be either un inflected or inflected; they are morphologically simple and do not require inflectional affixes for any of the grammatical categories. As instances, consider hoffaa, haaraa, diimaa, and so on. An adjective is a word that describes, identifies, or quantifies a noun or pronoun.An adjective in Afaan Oromoo often comes after the noun or pronoun it modifies. Adjectives in Afaan Oromoo can be interpreted using the same rules as nouns. Nominals can be used to define the word classes of adjectives, pronouns, and nouns[15] [58] [5]. Adjectives, like nouns, include inflectional categories for definiteness, number, gender, and case, among other grammatical categories. However, there is an alternative kind of affixation that is only employed for the inflectional function of adjectives. Adjectives, for example, are inflected by reduplication of stem to indicate

plurality, unlike nouns. To indicate plurality, the suffixes /-(o) ota/ are also employed. Adjectives have gender class inflected forms. The markers /-tuu/ and /-oo/ indicate feminine gender, whereas the marker /-aa/ indicates masculine gender. Most adjectives that terminate in long /-aa/ are suffixed with either /-tuu/ or /-oo/ morphemes removed. Adjective grammatical categories have the same case and definiteness as nouns.

2.6.2.3 Verb inflections

When describing acts, events, changes, or states of being, verbs are crucial parts of sentences. They are separated into three categories: state-change verbs, action verbs, and transitive and intransitive verbs. Intransitive verbs do not need objects to express their full meaning, whereas transitive verbs need[20] [6]. In Oromoo, verbs are classified into action/stative verbs, auxiliary verbs, and copula. Verbs undergo inherent and agreement inflection, leading to greater morphological complexity. Afaan Oromoo verbs are distinguished by syntax and morphology. Syntactically, they function as predicates in simple sentences and occur in the final positions of a sentence. Morphology shows agreement with the subject's number, gender, or person, and expression of the tense and aspect of the verb. Generation of syntactically and semantically correct sentences requires appropriate choice among different verb forms.

2.6.2.3.1 Agreement properties of verb inflection

Agreement properties indicate the inflection of a word class for properties out of its members. These properties are dependent on persons (first, second, and third), number (Singular and plural), and gender (muscular and feminine) of subjects. Like in many other languages in Afaan Oromoo Genders are distinctly identified only in the third person singular whereas numbers are indicated in all persons[6]. Verbs are marked by agreement morpheme -t- for 2nd persons (singular and plural) and 3rd person feminine. Plural numbers of persons are marked by the suffix -n (an).

2.6.2.3.2 Inherent properties of verb inflection

The lexical meaning of an Afaan Oromoo verb is represented by the stem, and aspects, mood or voice, and subject agreement are represented by the suffix [7] [16]. Verbs naturally have tenses, moods, and voices with some indicators of aspect. These

characteristics are a word class's fundamental members, which cause inflection on that word class.

Aspect: Since the morphological markers do not discriminate between present and future tense types, Afaan Oromoo verbs have an aspectual characteristic that distinguishes between past and non-past tense. With a focus on the circumstance, moment in time, and length of the event, aspect morphology also makes a distinction between an action's completion and incompleteness. Perfective and imperfective aspects are distinguished in Afaan Oromoo by several suffixes. While imperfective aspects denote acts that can be performed at any moment, perfective aspects denote finished actions. While imperfection denotes ongoing acts, perfection denotes finished actions at a single point in time. The suffixes -e, -ne, -te, and -tan denote the perfective aspect, while the suffixes -a, -na, -ta, -ti, -tu, and -u denote the imperfective aspect.

Mood: Afaan Oromoo is a language with several types of moods, including indicative, imperative, and jussive. Indicative moods are used for making statements and asking questions, with the final vowel lengthened for intonational relevance. Imperative moods are used to give order or command for second-person singular and plural verbs, marked by dependent morphemes i, u, and aa. Imperative verbs are inflected by superlative form using a different word of inflection. Imperative singular stems are formed using suffixes -i and -u, while imperative plural stems use -aa. Negative imperatives are formed using suffixes -in for singular and -inaa for plural preceded by the pre verbal particle hin. The jussive mood is used to give permission or command for actions done by third persons (inni, ishiin, and isaan) or first person plural nuti. In Afaan Oromoo, jussive mood shares semantic and morphological features with imperative mood but is marked by the preverbal particle haa and dependent morphemes -u or -n on the verb. Negation of jussive form is formed by a preverbal particle with the connection of suffix -in for all third persons, but no negation jussive form for first person plural.

Voice: The voice of the verb indicates whether the subject gives or receives the action. When the subject conducts the action, the voice is active; when the subject is the recipient of the action, the voice is passive. The verb form is altered in inflectional sentence structures for grammatical purposes. The morpheme -am- is added to transitive verbs in Afaan Oromoo to create the passive form. With the morpheme -am- designating the passive voice in both perfective and imperfection aspects, this passive form functions similarly to the English passive.

2.6.3 Morphophonemic Processes

The change that occurs between the boundary of stems and inflectional or derivational suffixes is known as a morpho-phonemic change[57]. The word may be reduplicated, assimilated, epenthesized, metathesized, or deleted in Afaan Oromoo as a result of the change. The sections that follow each briefly discuss each of them[7].

2.6.3.1 Reduplication

Reduplication is produced by duplicating the verb stem's initial consonant and vowel and geminating the first consonant's second occurrence. The term that emerges from the verb's action indicates intense performance or repetition. If the stem starts with a consonant, reduplication usually takes the form CV(C) + stem, where C is for consonant and V is for vowel. On the other hand, the stem becomes V (') + stem if the stem starts with a vowel. For example consider the phrases kufuu/failing->kukkufuu/repeatedly failing, fiiguu/running->fiffiiguu/repeatedly running, and utaaluu/jumping->u'utaaluu/repeatedly jumping.

2.6.3.2 Deletion

Inflections or derivations always result in deletion. For instance:

Mana/house->Manoota/houses, and Nama/Man->Namicha/The Man. In verb stems ending with h, dh, hudhaa ('), deletion usually occurs. For example: baate/She left->Bah-+te hoote/she fed the breast->hodh-+te.

2.6.3.3 Assimilation

The preceding or following phoneme may occur at morpheme or word boundaries between two phonemes. As a result, a variety of stem-final consonants are combined with t (third-person singular feminine, second-person singular, and second-person plural), n (first-person plural, neutral common), s (common, causative-common singular), and other consonants. Prefix and stem relationships can change, as can stem and suffix relationships. For example: duucha/he will make deaf->duud+sa.

2.6.3.4 Epenthesis

More than two consecutive consonants cannot coexist in Afaan Oromoo. When there are more than two consonants in a row, a /i/ or other sound will be placed in between them. For instance, sirbita/you will sing->sirb-+ta, and elmina->elm-+-na.

2.7 Related Work

There were so many developed morphological analyzers for both local and foreign languages using different approaches based on their language complexity. In the subtopics of this chapter, we tried to mention the work of researchers based on morphological analyzers mainly on deep learning approaches as the proposed system uses these approaches.

2.7.1 Morphological Analyzer for Foreign Languages

2.7.1.1 Morphological analysis for Malayalam language

A deep learning Malayalam morphological analysis at the character level was developed by [58] [34]. Their study used a deep learning method for automatically recognizing morphemes and separating them from the original word. Three systems with respective accuracy levels of 98.08%, 97.88%, and 98.16% were built employing recurrent neural networks, long short-term memory, and gated recurrent units.

2.7.1.2 Morphological analysis for the Japanese language

Jun Izutsu and Kanako Komiya [59] created a neural Bi-LSTM CRF-based morphological analyzer for Japanese Hiragana phrases by using the BiLSTM model. They stated that their system was crucial for Japanese NLP systems and is competitive for forceful languages such as German and Spanish. They evaluated their system based on Kanji-Kana and Wiki+ Hiragana sentences. In their model comparison based on the data source they used, the Wiki model has greater macro and micro-averaged accuracy (56.90% and 58.80%) than the Hiragana Wiki model (56.72% and 58.67%). The Kanji-Kana Wiki+ Hiragana Wiki+ Hiragana Yahoo! model (61.39% and 62.44%) increased accuracy by 4.49 and 3.64 points, respectively, demonstrating that fine-tuning using Hiragana Yahoo! data considerably increases permanence. When utilizing Yahoo! Answers, the model performed better, probably owing to greater corpus quality and similarity of training and test data.

2.7.1.3 Morphological analysis based on artificial Neural-Net-XMOR

Ayla KAYABAŞ1, Ahmet E. TOPCU2, and Özkan KILIÇ [60] developed artificial Neural-Net-XMOR, A novel hybrid algorithm for morphological analysis: Their paper provides a unique approach for morphological analysis that blends rule-based and artificial neural network-based methodologies. As they stated the hybrid approach increases performance by building an artificial neural network based on two-level phonological principles. The hybrid algorithm was evaluated on optical character recognition (OCR) and social media data, with an OCR accuracy of 99.91% and a social media accuracy of 99.82%. Their research work assesses the efficiency of hybrid models in morphological analysis.

2.7.1.4 Morphological analysis for Hind language

Researchers Saurav Jha and Akhil Sudhakar have developed a Multi-Task Deep Morphological Analyzer (MT-DMA) for Hindi morphology [61]. The study compared its performance to two functional analyzers from IIT-Bombay and IIIT-Hyderabad. The baseline model, consisting of a bidirectional GRU layer and a dense layer with softmax activation, was used. Two additional neural architectures (char-CNN and Bi-GRU) were used to modify MT-DMA based on the tag predictor. The results showed that MT-DMA outperformed current analyzers and baselines for Hindi. The study also compared the Bilingual Evaluation Understudy Score (BLEU) and Levenshtein distance.

2.7.2 Morphological Analyzer for Local Languages

2.7.2.1 Morphological Analyzer for Amharic

Mesfin Abate and Yaregal Assabie [28] used memory-based machine learning techniques to create a morphological analyzer for Amharic. They approached morphological analysis as a classification job using a memory-based supervised machine learning technique.Their model was evaluated by a 10-fold cross-validation using the IB1 and IGtree methods. The results showed an overall accuracy of 93.6% and 82.3%, respectively.Their model is designed to analyse vowel-led inflected Amharic words along with their morpheme grammatical functions.

2.7.2.2 Morphological Analyzer for Tigrigna

Yemane and Kazuhide [45] performed morphological segmentation for Tigrigna using LSTM neural networks: they built a new corpus with 45,127 manually segmented tokens in order to apply LSTM approaches to morphological segmentation for the Tigrigna language. They achieved a 94.67% F1 score for morpheme boundary recognition by utilizing Conditional Random Fields (CRF) in conjunction with Window-based LSTM neural networks.

2.7.2.3 Morphological Analyzer for Afaan Oromoo

HornMorpho is a language analysis programme that employs web crawlers to extract unique word forms in Amharic and Tigrigna and Afaan Oromoo. It was developed by Michael Gasser [19] as a system for morphological processing of Amharic, Oromoo, and Tigrigna. A human reader evaluates it. The program picks 200 words from a list and uses an anal word function to identify grammatical structures. However, it made 8 mistakes in Tigrigna verbs (96% accuracy), 2 in Amharic verbs (99% accuracy), and 9 in Amharic nouns and adjectives (95.5% accuracy).

A morphological analyzer for Afaan Oromoo using machine learning was created by Moyka Degefa [7] and submitted to Addis Ababa University's Department of Computer Science in March 2020. Moyka created a system in this thesis by employing a memory-based learning methodology. He used the algorithms IB1 and IGTREE to test his system, and the results showed an overall accuracy of 98.86% and 94.36%, respectively.The accuracy of his work was considered high, compared to other existing morphological analyzers for Ethiopian languages, such as HornMorpho and OroRoots, which do not report their accuracy in their sources. However, his work can also be improved by using a larger and more diverse corpus of Afaan Oromoo words, and by comparing his system with other data-driven or hybrid methods.

Kedir Gena [8] developed Afaan Oromoo Morphological Analysis using a Hybrid Approach: his paper proposes a hybrid method that combines rule-based and data-driven techniques for analyzing words in Afaan Oromoo. The paper reports that the analyzer has an accuracy of 84.6% for nouns and 82.9% for verbs.

2.7.3 Summary of Related Work

There were so many previously conducted research works on morphological processing and some for the Afaan Oromoo language. In this subtopic, some of them were reviewed and discussed to get knowledge to deal with this study. The following table shows the summarization of the related work of this study.

Table 15: Table to summarize related work

Author citation	Title	Method	Result	Gap
[34]	Malayalam morphological analysis at character level	RNN, LSTM, GRU	98.08%, 97.88%, 98.16%	The system is not applicable for Afaan Oromoo
[59]	Morphological analyser for Japanese Hiragana phrases	BiLSTM CRF-	WM=56.90%, 58.80% HWM=56.72%,58.67 WMHY=61.39%,62.44 % Based on micro, macro Averaged accuracy	The system is not applicable for Afaan Oromoo
[61]	Multi Task Deep Morphological Analyzer (MT-DMA) for Hindi morphology	CNN, Bi-GRU	-	The system is not applicable for Afaan Oromoo
[28]	Morphological analyzer for Amharic language	ML(IB1, IGtree)	93.6% and 82.3%, respectively	The system is not applicable for Afaan Oromoo
[45]	Morphological segmentation with LSTMs neural networks for Tigrinya:	LSTM with (CRF)	94.67% F1	The system is not applicable for Afaan Oromoo
M.Gasser	HornMorpho: a system for morphological processing of Amharic, Oromoo, and Tigrinya;	FST	Tigrinya verb (96%), Amharic verb (99%), Amharic N & A 95.5% None for A/Oromoo	Afaan Oromoo module was not evaluated
M.Degefa	morphological Analyzer for Afaan Oromoo	IB1 IGtree	98.86% , 94.36%	Require larger dataset to be trained and need to be compared with other related models
K.Gena	Afaan Oromoo Morphological Analysis	RB, HMM	84.6% for nouns and 82.9% for verbs	more corpus size in the gold standard is needed

2.7.4. Gaps

This study identifies gaps in previous research on morphological analysis of Afaan Oromoo, a language with unique language structure, patterns, and writing system. Previous studies have used deep learning and rule-based approaches for morphological analysis in foreign languages and Ethiopian languages like Tigrigna. However, some reviewed works have limited methodology, such as those for Afaan Oromoo, which were developed using rule-based and machine-learning approaches. The proposed system aims to address these gaps and improve understanding of morphological analysis in the language by using deep learning approaches. Another specific gaps in the literature include the variation between Afaan Oromoo writers, difficulty in extracting morphological features, limited dataset size, and the focus area of Afaan Oromoo researchers on simple words. While efforts have been made to develop morphological analyzers for Afaan Oromoo, there is still room for improvement in terms of accuracy, scalability, and generalization. The proposed system can address the lack of a deep learning-based morphological analyzer specifically designed for Afaan Oromoo with comprehensive evaluation metrics and comparisons to existing systems. The proposed system, utilizing deep learning methodologies such as CNN, LSTM, GRU, and BiLSTM, can potentially offer superior performance compared to existing approaches. By training on a larger and more diverse corpus of Afaan Oromoo words, the proposed system can achieve higher accuracy levels and better generalization to handle various linguistic variations and dialects.

2.8 Summary

This chapter reviews the proposed system for morphological analysis of Afaan Oromoo using deep learning approaches. It discusses terminologies related to the study, methods used in morphological processes, Afaan Oromoo's writing structure, and specific terms used. The chapter also reviews previous research on foreign and local languages, their methods, results, and gaps for solutions. The aim is to provide a comprehensive understanding of Afaan Oromoo's writing and its linguistic context.

Chapter Three: Research Methodology

3.1 Introduction

This chapter provides a comprehensive methodology for designing and implementing a deep learning-based morphological analyzer for Afaan Oromoo. The system's architecture is a conceptual framework that defines its structure, behaviour, and various components. The development process includes data collection, preprocessing, construction, and training of deep learning models. Sequence-to-sequence models like CNN, LSTM, BLSTM, and GRU are used, along with advanced word embedding techniques like Word2Vec and FastText.

The data collection process involves gathering a substantial corpus of Afaan Oromoo text from various sources. Preprocessing techniques are used to clean and prepare the data for analysis, while data augmentation enhances the data set's size and diversity. Data annotation is crucial for training the models effectively.

Word embedding techniques are introduced to transform textual data into numerical representations. The core of the chapter focuses on the encoder-decoder framework and specific neural network models used. The encoder and decoder components are explained, along with the training algorithm, optimization techniques, and evaluation metrics used to assess the model's performance.

The prediction architecture is presented, outlining how the trained model predicts morphological tags for new Afaan Oromoo words. Performance evaluation metrics are used to measure the accuracy and effectiveness of the system.By detailing each component and phase of the system's development, this chapter establishes a clear and replicable methodology for creating a state-of-the-art morphological analyzer for Afaan Oromoo, leveraging deep learning techniques to handle the complexities of this rich and under-represented language.

3.2 Architecture of the Proposed System

The system architecture is a conceptual model that defines a system's structure, behavior, and views. The following architecture describes a process for constructing a deep learning model for morphological analysis using Afaan Oromoo text data. The process involves collecting and preprocessing data, tokenizing sentences and words, and using keras

embedding and pre-trained word embeddings. The model is designed using sequence-to-sequence CNN, LSTM, BLSTM, and GRU with neural word embedding techniques such as word2vec and FastText. The system was trained on labelled data with morphological annotations and is optimized using loss functions.

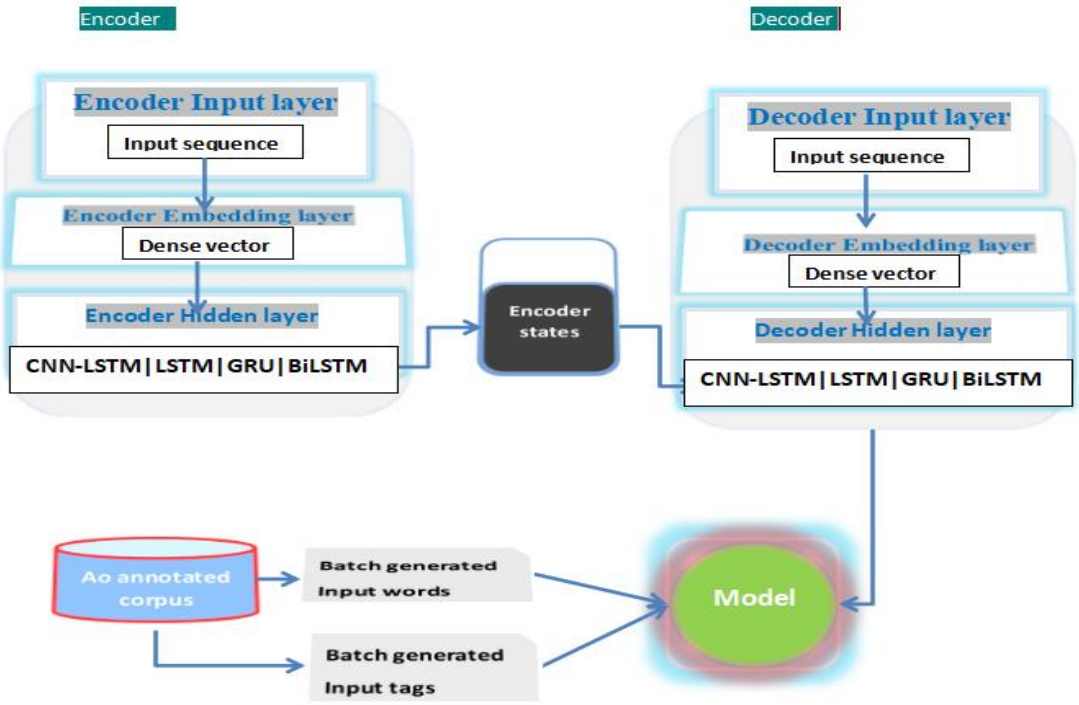


Figure 9: Architecture of the proposed system

3.2.1 Data Collection

For this study, a total of 27043 sentences were collected from five different Afaan Oromoo books available on a Telegram group channel called Kitaaba Afaan Oromoo (https://t.me/Freeorobook) [67]. These books are kitaaba aarsaa walaloo by Sabboonaa Tolosaa, Falmii Dhugaa by Amaanu'eel Oljirraa, kitaaba siyaasaa fi ummataa by Qalbeessaa Dhanga'aa, Qorii Qaroomaa by Abdussalaam Idris, Seenicha sukkaneessaa barri dhokse by Wolde Yohaannis and Gammachuu Malkaa. The texts of a ll these books are combined and saved as Ao_rawtext.

3.2.2 Data pre-processing

Textual data pre-processing techniques are essential for preparing raw text data for analysis or modelling. Pre-processing techniques help to reduce noisy data, improve efficiency, and ensure consistent and accurate text analysis. For the proposed system, the following pre-

processing steps are used: Tokenizing, normalizing, and removing numerical data, punctuation marks, and extra spaces, stop word removal are done by storing all stop words of Afaan Oromoo as Ao_stop words. Stop words are taken from Hawi Tamiru, Moyka Degefa, and Kedir Gena's research works.

3.2.3 Data Augmentation

The study pre-processed 1,367,981 Afaan Oromoo words and gathered 27,043 sentences with noisy data from the books and 33,593 unique words are preprocessed. All pre-processed unique words are not used in this study. consequently Afaan Oromoo inflected and derivated words are selected based on affixes they attached with. For this study, 13,460 inflected and derived nouns, adjectives, and verbs are selected and are not large enough to train deep learning approaches. So, via the use of data augmentation techniques, the dataset size is increased to 27,931 Afaan Oromoo inflected and derivated nouns, adjectives, and verbs. Augmentation method is selected to increase the size of dataset and enhance the deep learning models' performance.

3.2.4 Dataset Annotation

To design and develop deep learning based morphological analyzer for Afaan Oromoo language, dataset must be morphologically annotated in a way that the model can understand and analyze the words and assign correct grammatical information. For this study, all pre-processed and augmented Afaan Oromoo inflected and derivated nouns, adjectives, and verbs are annotated by three Afaan Oromoo teachers.

3.2.5 Word Embedding

To extract features from a dataset, the dataset must be pre-processed and annotated based on the specific task. The annotated dataset also must be converted to numbers either by one-hot encoding, word embedding techniques, or other vectorization techniques. In this study, keras embedding, word2vec and FastText word embedding techniques are employed to extract the features. Word2Vec and FastText are popular methods for word embedding, which is a technique used to map words or phrases to vectors of real numbers in a continuous vector space. These methods capture semantic and syntactic similarities between words based on their usage in large text corpora.

3.2.5.1. Word2Vec Word Embedding

Word2Vec is one of the most commonly used word embedding techniques used in this study. For this study, the skip-gram model of word2vec is implemented to extract features from the given input datasets. The model contains neural word embedding techniques used to embed input words. The model was trained based on annotated dataset containing 27,931 unique words with an embedding dimension of 64 and two window size.

3.2.5.2. FastText Word Embedding

FastText is also another widely used word embedding technique used in this study. The model contains neural word embedding techniques used to embed input words based on n-gram character embedding. It is implemented to extract features from the annotated dataset and trained on annotated dataset of 27,931 input words with 64 embedding dimensions and window size 2.

3.2.6 Encoder Decoder Model

Sequence-to-sequence transformations such as machine translation, text summarization, speech recognition, and morphological analysis are performed by neural networks with an encoder-decoder architecture. It is made up of an encoder that converts an input sequence into a context vector or fixed-size representation while preserving the most important aspects.An output sequence is generated by the decoder of the encoded representation by iteratively adding context vector information. Pairs of the input word sequences and tag sets are provided to the architecture, which uses this training to precisely map input sequences to output sequences. Since it can handle sequences of varying lengths for input and output and capture long-range relationships within data, it is useful for jobs involving sequences that vary from one to the other.

3.2.6.1 Encoder

An encoder is a crucial component in neural network architectures, particularly in sequence-to-sequence models like machine translation or text summarization. It converts the input sequence into a fixed-size vector representation, capturing the semantic and contextual information. The encoder typically consists of recurrent neural network layers, such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) cells, which process the input sequence step by step, updating their internal states at each time step. The

encoder's final hidden state, the context vector or thought vector, represents the entire input sequence, which is then passed on to the decoder component for further processing, such as generating the output sequence in machine translation tasks.

3.2.6.1.1 Encoder Embedding Layer

The Encoder Embedding Layer is a crucial component in neural network architectures, transforming input sequences into continuous vector representations that capture semantic meaning and contextual information. It maps each token in the input sequence to a high-dimensional vector space, allowing the neural network to understand the relationships between different tokens. The embedding layer is usually initialized with random weights and trained using back propagation, but can also be initialized using pre-trained word embeddings like Word2Vec, or FastText. This layer plays a crucial role in the encoder's ability to process and represent the input sequence effectively, enabling downstream neural network tasks like machine translation, text summarization, and sentiment analysis.

3.2.6.1.2 Encoder recurrent layer

The Encoder recurrent layer is a crucial part of sequence-to-sequence models, particularly in tasks like machine translation and text summarization. It processes the input sequence and generates a fixed-size representation, known as a context vector or hidden state, which captures the semantic meaning and contextual information of the input sequence. The Encoder recurrent layer consists of recurrent neural network (RNN) cells, such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) cells, which process the input sequence one token at a time while maintaining an internal state that captures previous tokens. This recurrent nature allows the model to capture dependencies and sequential patterns within the input sequence. During the forward pass, each token of the input sequence is fed into the recurrent layer sequentially, updating its hidden state based on the current token and previous hidden state. The final hidden state serves as the context vector, which contains aggregated information from the entire input sequence and is passed on to the decoder for further processing in sequence-to-sequence tasks.

3.2.6.2 Decoder

The decoder is a crucial part of sequence-to-sequence models, working alongside the encoder to generate an output sequence based on the encoder's context. In tasks like

machine translation or text summarization, the decoder uses the encoder's context vector to generate the target sequence one token at a time. It employs recurrent neural network (RNN) cells to model the conditional probability distribution of the next token in the output sequence. The decoder RNN generates tokens iteratively, predicting the next token in the output sequence based on the current hidden state, the encoder's context vector, and previous tokens. This process continues until an end-of-sequence token is generated or a predefined maximum sequence length is reached. The decoder's initial hidden state is typically initialized with the encoder's final hidden state, allowing it to generate the output sequence based on the input sequence's information. The decoder's recurrent nature allows it to model dependencies between tokens in the output sequence, ensuring coherent and contextually relevant predictions.

3.2.6.2.1 Decoder Embedding Layer

The decoder embedding layer is a crucial component of sequence-to-sequence models, particularly in Transformer or LSTM-based architectures. It converts discrete input tokens, usually words or sub words, into continuous vector representations, or embeddings. The decoder receives input tokens, which are usually generated from the target language or output sequence. These tokens are represented as one-hot vectors or integer indices, with each token corresponding to a unique position in the vocabulary. The decoder embedding layer consists of an embedding matrix, which is initialized randomly or with pre-trained embeddings and has dimensions equal to the size of the vocabulary. During the forward pass, each input token is mapped to its corresponding embedding vector in the matrix, resulting in a continuous vector representation for each input token. Embedding dimensionality is a hyperparameter that determines the size of the embedding vectors, typically higher-dimensional embeddings can capture more nuanced semantic relationships between tokens but require more computational resources. The embedding matrix's entries are learnable parameters, updated during the training process via back propagation. The model learns to adjust the embeddings to minimize the discrepancy between predicted and target sequences, effectively learning meaningful representations for the input tokens.

3.2.6.2.2 Decoder Recurrent Layer

The decoder recurrent layer is a crucial part of sequence-to-sequence models, responsible for generating output sequences based on the encoded input sequence. It captures

sequential dependencies and generates contextually relevant output tokens one step at a time. The decoder recurrent layer takes input from the previous time step, maintaining an internal state representation called the hidden state. This allows the model to incorporate context from earlier time steps into its predictions. The recurrent computation is performed using a recurrent neural network cell, such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU), updating the hidden state based on the current input and previous hidden state. This process captures dependencies between sequential tokens in the output sequence.The output prediction is generated for the current time step, typically projecting the hidden state onto the vocabulary space using a linear transformation followed by a softmax activation function. During training, the decoder recurrent layer generates output tokens by sampling from the predicted probability distribution, which becomes the input for the next time step. The decoder recurrent layer is trained end-to-end using back propagation and optimization techniques like stochastic gradient descent (SGD) or its variants. The model learns to adjust its parameters to minimize discrepancies between predicted and target sequences, optimizing the overall sequence generation process.

3.2.6.2.3 Decoder output layer

The decoder output layer is the final component of a sequence-to-sequence model, generating the output sequence based on the encoder's context and previously generated tokens. It consists of a dense neural network layer and a softmax activation function, which project the hidden state of the decoder onto the vocabulary space. The model predicts the next token in the output sequence using the probability distribution, comparing it with the actual target token in the ground truth sequence. The decoder output layer also samples the next token from the probability distribution generated by the softmax function during inference, allowing the model to generate diverse and contextually relevant output sequences. The output layer is trained using back propagation and optimization techniques Adam optimization to optimize the sequence generation process. In tasks with variable output sequence lengths, the decoder output layer may incorporate a termination mechanism to ensure the model produces sequences of appropriate length and avoids generating unnecessary tokens. Overall, the decoder output layer is crucial for generating coherent and contextually appropriate output sequences in sequence-to-sequence models.

3.2.7. Algorithm to train and validate AOMA

The system designed for Afaan Oromoo morphological analysis has two main parts: encoder and decoder. The following steps are the way the designed models interact with the training, and validation dataset.

✓ Input: Afaan Oromoo batch generated words and their corresponding morphemes are fed in to encoder and decoder respectively.

✓ Output: Saved Afaan Oromoo Morphological Analysis system model

Step 1: Receive the variable length input sequences from the input layer of the models.

Step 2: The embedding layers of the model calculate word vectors and convert to fixed length dense vector.

Step 3: The hidden layers of the model like CNN, LSTM, GRU, and BILSTM accept fixed length dense vector from embedding layer and process it.

Step 4: Decoder part of the model uses encoder's hidden vector and fixed length dense vectors from embedding layer to calculate the hidden state.

Step 5: The output layer of the decoder part of the model generates the probability of output from hidden vectors

Step 6: Model is trained with:

Loss function = Categorical cross entropy

Optimizer= Adam

Batch size=64

Step 7: Save model

3.2.8 Algorithm to Predict Morphemes of Words

The following algorithms aims to describe how to use its trained model to predict Afaan Oromoo words with their morpheme tags. Pre-processing removes superfluous capitalization and symbols from Afaan Oromoo words. The pre-processed word is segmented by the decoder based on the training model and the hidden state of the input

word. It gives a set of morpheme tags if the word is present in the dataset; if not, it divides the corpus into terms that are very similar to the missing term.

➢ Input: Afaan Oromoo word

➢ Output: stem with its morpheme tags separated by the ";" sign.

Step 1: Input Afaan Oromoo new words

Step 2: Pre-process the inserted words

Step 3: Calculate the character vector from the preprocessed words

Step 4: Analyse the calculated character of words based on a trained model

3.3 Performance Evaluation

The measurement that will be employed to assess the trained model's performance are Metrics like accuracy, recall,precision, and f1-score are frequently used to assess machine learning models. The proportion of accurate borders discovered relative to all correct boundaries is assessed by precision, while the percentage of correct boundaries found relative to all correct boundaries is measured by recall. The f1-score, which may be thought as their weighted average, is the harmonic mean of recall and accuracy. When assessing classification accuracy, one looks at the proportion of properly predicted occurrences relative to all instances. It is determined by dividing the total number of predictions generated by the algorithm by the number of accurate forecasts.

$$\textbf{Accuracy} = \frac{TP+TN}{TN+TP+FN+FP} \text{ ----------(1)}$$

Precision is the total estimate of the class labels accurately predicted for each class. The precision measure is calculated using the equation:

$$\textbf{Precision} = \frac{TP}{TP+FP} \text{ ----------(2)}$$

The recall value is the weighted average of the correct labels, correctly classified for each class. Calculated according to equation:

$$\textbf{Recall} = \frac{TP}{TP+FN} \text{ ----------(3)}$$

The F1 score is a function of precision and recall. It is used to find the correct balance between the two metrics. It is calculated as follows:

$$\mathbf{F1} = \frac{2 * Recall * Precision}{Recall + Precision} \quad \text{----------(4)}$$

Where: The accuracy of a model gauges how accurate its predictions are overall. The percentage of accurately predicted morphological annotations to all predictions is computed. The percentage of accurately predicted positive instances (morphological annotations) among all anticipated positive instances is known as precision. It measures how well the model avoids producing false positive results. The percentage of accurately anticipated positive instances among all actual positive instances is measured by recall, which is sometimes referred to as sensitivity or true positive rate. It measures the model's resistance to erroneous negative results. Recall and precision are combined into a single statistic called the F1 score. It is the harmonic mean of recall and accuracy that offers a fair assessment of the model's functionality.

3.4 Summary

In general sequence to sequence Afaan Oromoo morphological analyzer is designed using deep learning techniques to handle annotated text data of Afaan Oromoo language. The system consists of an encoder and a decoder, with the encoder preparing inputs for deep learning models and the decoder extracting features and producing outputs. The system uses CNN, LSTM, GRU, and BLSTM models for morphological analysis. Data collection involved gathering many sentences from various Afaan Oromoo books, pre-processed to remove noise, tokenize words, and remove numerical data and punctuation marks. Word embedding techniques like Word2Vec and FastText are used to extract features. The model's training algorithm uses categorical cross-entropy loss and the Adam optimizer, and its performance is evaluated using metrics like accuracy, precision, recall, and F1 score.

Chapter Four: Experimentation and Result Analysis

4.1 Introduction

In this chapter, we detail the experimentation and result analysis of the system, aimed at addressing the research questions posed at the beginning of the study. This section outlines the methods and processes used to develop and evaluate the morphological analysis system for Afaan Oromoo using deep learning approaches.

The experimentation section covers several key areas: dataset collection and preparation, the experimental setup, the training components of the models, and the hyper parameters employed during system design and development. Each of these components is essential for building a robust and effective system capable of accurately performing morphological analysis on Afaan Oromoo.

Following the experimentation, the result analysis section presents the findings of our experiments. This includes the performance evaluation metrics and their outputs, which were used to assess the system's effectiveness during development. Through this analysis, we aim to understand the strengths and limitations of the various models and techniques applied, providing insights that will guide future improvements.

Systematic experimentation and thorough result analysis are crucial for developing a high-performance morphological analysis system. By rigorously testing different approaches and critically evaluating their results, we ensure that the final system is robust, accurate, and capable of effectively analyzing the complex morphology of Afaan Oromoo. This chapter thus plays a pivotal role in demonstrating the efficacy of the deep learning models and embedding techniques used in this study.

4.2 Experimental setup and setting

Hardware/Software	Specification
Manufacturer	HP
Processor	Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz
System type	64-bit operating system, x64-based processor
Installed RAM	8.00 GB (7.89 GB usable)
Google Colab	GPU and TPU
Programming Language	Python
Deep Learning Frameworks	Tensor Flow, Keras
Libraries	NumPy, Pandas, Scikit-learn, Gensim
Experimental setting	4 Deep learning models with 3 different word embedding Totally 12 different experimentations will be performed

4.3 Data Collection

For this study, total of 27043 sentences were collected from five different Afaan Oromoo books available on telegram group channel called Kitaaba Afaan Oromoo (https://t.me/Freeorobook)[67]. These books are Kitaaba Aarsaa walaloo by Sabboonaa Tolosaa, Falmii Dhugaa by Amaanu'eel Oljirraa, Kitaaba siyaasaa fi ummataa by Qalbeessaa Dhanga'aa, Qorii Qaroomaa by Abdussalaam Idris, Seenicha sukkaneessaa barri dhokse by Wolde Yohaannis and Gammachuu Malkaa, The texts of all these books were combined and saved as Ao_rawtext.

```
filename = "/content/drive/MyDrive/Colab Notebooks/AO_MABOKI/DOCUMENT/training_data/Ao_rawtext.TXT"
stopwords = open("/content/drive/MyDrive/Colab Notebooks/AO_MABOKI/DOCUMENT/training_data/stopwords.txt").read().split()
```

```
Ao_rawtext=open(filename).readlines()

len(Ao_rawtext), Ao_rawtext
```

```
(27043,
 ['AARSAA \n',
  'Sabboonaa Tolasaa Darasaa \n',
  'Fuula 0 \n',
  'AARSAA \n',
  'AARSAA \n',
  '(WALALOO) \n',
  'Yaadannoo guyyaa 40ffaa wareegamuu \n',
```


Figure 10: Collected Afaan Oromoo raw texts

4.4 Data pre-processing

Textual data pre-processing techniques are essential for preparing raw text data for analysis or modelling. Pre-processing techniques help to reduce noisy data, improve efficiency, and ensure consistent and accurate text analysis. For the proposed system the following pre-processing steps are used. To begin with text pre-processing raw text loading is the mandatory steps used in research work. Ao_rawtext is loaded using the following snipped python code. Tokenizing, normalizing, and removing numerical data, punctuation mark and extra spaces, stop word removal all are done by storing all stop words of Afaan Oromoo as Ao_stopwords. These stop words are taken from Hawi Tamiru, Moyka Degefa, and Kedir Gena research works and illustrated as.

Figure 11: Pre-processing steps

Finally, 1367981 Afaan Oromoo words were compiled and saved as Ao_preprocessedtext as shown below.

4.5 Data Augmentation

As stated above from the collected documents, 27,043 sentences with full of noisy data are collected. After pre-processing 1367981 Afaan Oromoo words are compiled and saved as preprocessed_text. Out of 1,367,981 compiled words 33,593 unique words are screened

out. The unique words are also the combination of words from other languages like English and Amharic, and words with small sized. These like unwanted words are screened out manually by looking at their meaning and Afaan Oromoo inflected words are selected depending up on the affix they attached with. As a result 13,460 inflected nouns, adjectives and verbs of Afaan Oromoo are selected for this study. As deep learning models require large amount of dataset, it is mandatory either to find out other raw text or to increase the amount of the existing data by data augmentation techniques. So, data augmentation is preferred to increase the size of dataset. Using this technique the size of the dataset became 27,931 unique inflected nouns, adjectives and verbs. For example the following snipped python code is used to convert suffix a to onni, otaan, otaaf representing nominative case nouns instrumental case, and dative case respectively.

4.6 Dataset Annotation

To design and develop deep learning based morphological analyzer for Afaan Oromoo, datasets must be morphologically annotated in the way that the model could understand and analyse the words and assign correct grammatical information. For this study, out of all pre-processed, inflected nouns and verbs with augmented words 27,931unique words are divided among three Afaan Oromoo teachers. The teachers discussed on the way that would be suitable to annotate words to make similar annotation style. They shared comman annotating tagsets. For example all are agreed to annotate based on the agreement:

Wi S P|G1;G2;Gn where: Wi represent input word and separated from other by tab, S represents stem of the word separated by single space, P represents part of speech and G represents grammatical information.

4.6.1 Afaan Oromoo morpheme tagsets

Morphological analysis are NLP task that analysis the structures of words, break down the words in to morphemes and assign the language's grammatical tags for the morphemes as they provides the meaning. The following table describes all tagsets of Afaan Oromoo language which are used in the proposed system.

Table 16: Afaan Oromoo Tag sets and Description

No	Nouns/Adjectives	Description	Verbs	Description
1	N	Noun	V	Verb
2	ADJ	Adjectives	1	First person
3	PL	Plural	2	Second person
`4	RD	Reduplication	3	Third person
5	M	Male	INF	Infinitive verb
6	F	Female	FUT	Future verb
7	DEF	Definitive	IMPR	Imperative
8	NOM	Nominative	IMPRF	Imperfect
9	GEN	Genetive	PRF	Perfect
10	DAT	Dative	JSV	Jussive
11	ACC	Accusative	IND	Indicative
12	ABL	Ablative	AF	Affirmative
13	LOC	Locative	NEG	Negative
14	VOC	Vocative	PST	Past
15	INST	Instrumental	SG	Singular
16	FOC	Focus	DS	Derivated Stem
17	DS	Derivative Stem		

4.6.2. Annotation for Nouns

Afaan Oromoo nouns have eight suffix slots and no prefix, connected to the stem in a specific sequence. Eight suffix slots are derivation, direct object (accusative case), plurality, definiteness, nominative case, dative case, instrumental case, and focus are the suffixes that are connected to the stems.In Afaan Oromoo adjectives are also termed as noun and have nine suffixes and one prefix, with all suffixes appended to the stem being similar to nouns. Adjective stems can have affixes added, including reduplication, gender, plurality, definiteness, nominative case, accusative case, dative case, instrumental case, and focus marker. Adjectives should be manually annotated according to their grammatical function to generate a dataset. The prefix-stem-suffix structure is followed when annotating.

Table 17: Annotation sample for noun

Rd	Stem	Der	Num	Gender	Def	Nom	Acc	Dat	Inst	Foc
Qa-	qal’	=	=	-aa	=	=	=	=	=	-tu
Qa-	qal’	-in	=	=	=	-ni	-a	=	-an	=
=	qal’	=	=	=	-ich	-i	-a	=	=	-tu
-	Nam-	-	-oota	=	=	=	=	=	-an	=

Where: The dataset is generated by manually annotating a collection of noun and adjective terms based on their grammatical function. In this study, as adjectives are termed as nouns the words are annotated manually using the prefix-stem-suffix structure. And the abbreviated character like Red for reduplication, Stem for stem, Der for derivation stem, Num for number, Def for definitiveness, Nom for nominative case, Acc for accusative case, Dat for dative case, Inst for instrumental, = for possible, - for not possible, and Foc is for focus used in noun and adjective annotation.

Table 18: Noun annotation samples

Word	Prefix	Stem	Suffix
Qaqal’aa	Qa-[RD]	Qal'-[N]	-aa[M]
Qaqal’inaan	Qa-[RD]	Qal'-[N]	-in-[DS]-a-[ACC]-an[INST]
Qal’ichatu	=	Qal’-[N]	-ich-[DEF]-a-[ACC]-tu[FOC]
Namootaan	=	Nam-[N]	-oota-[PL]-an[INST]

Where: RD, N, DS, PL, ACC, INST, M, and F denoting Reduplication, Noun, Derivative stem, Plural, Accusative, Instrumental, Male, and Female respectively.

4.6.3 Annotation for Verbs

Afaan Oromoo verbs contain various information on their affixes, including person, gender, and number. They consist of base stems and derived stems, with the infinitive forms being the root stems. Derivative stems are created by adding markers to verb roots or other derived stems, such as passive, causative, auto-benefactive, middle voice, and intensive. The affixes attached to verb stems include subject, aspect-mood, negative marker as a suffix, negative/jussive, and affirmative as a prefix and derivation. The prefix-stem suffix structure is followed when annotating verbs.

Table 19: Verb annotation samples

Ng/Jsv/Af	Stem	DS	Ds	Subj	Ng	AM
hin-	deem-	=	=	=	in-	-aa
haa-	deem-	=	=	-t-	-	-u
ni-	deem-	-sis-	=	-n-	-	-a
=	deem-	sis-	=	=	-	-e
=	fil-	-at-	-noo	=	-	-
=	yaad-	-at-	-noo	=		-

Where: Ng/Jsv/Aff, Stem, DS, Ds, subj, Ng, and AM denote Negative prefix, jussive marker, affirmative marker, stem, derivation to verb, derivation to noun, subject, negative marker suffix, aspect mood marker respectively. The verb follows prefix-stem suffix ([P]-[S]-[S]).

Table 20: Verb annotation samples

Word	Prefix	Stem	Suffix
hin deeminaa	hin-[NG]	-deem-[V]	in-[NG] -aa[IMPR]
Haadeemu	haa-[JSV]	-deem-[V]	-u[IMPRF]
Nideema	ni-[AF]	-deem-[V]	-a[IMPRF]
Deemsifne	-	deem-[V]	-sis-[DSV] -n[1;PL;A] -e[PRF]
Filannoo	-	fil-[V]	-at-[DSV] -noo[DS]
Yaadannoo	-	yaad-[V]	-at-[DSV] -noo[DS]

Where V, PRF, 1, PL, AF and IMPR, IMPRF, DS, DS, indicating base stem, normal perfective aspect marker, number marker, plural, affirmative marker, imperative marker, imperfect marker, derivation stem for verb, derivation stem for noun, first person plural for any gender.

4.7 Training, validation and testing dataset

After all processes of the experimental result related to the data collection, data pre-processing, generating task specific dataset is mandatory step to train, validate and test the models. As a result python code based method which is used to generate and enumerate the context and target dataset is created. Before using this method the annotated dataset must be splitted into training, validation, and testing datasets. So, 45349 annotated words are splitted in to 40814 for training and validating purpose and 4535 for testing purpose. Then 40814 words are further splitted into training dataset (30610), and validation dataset (10204) which is the 25% percent of the total words. To train, validate or test the model the batch generating method is used alongside with the splitted dataset based on their tasks. The following snipped python code is used to generate input words and target words for the models.

```
def generate_batch(X = x_train, y = y_train, batch_size = 128):
    ''' Generate a batch of data '''
    while True:
        for j in range(0, len(X), batch_size):
            encoder_input_data = np.zeros((batch_size, max_length_src),dtype='float32')
            decoder_input_data = np.zeros((batch_size, max_length_tar),dtype='float32')
            decoder_target_data = np.zeros((batch_size, max_length_tar, num_decoder_tokens),dtype='float3
            for i, (input_text, target_text) in enumerate(zip(X[j:j+batch_size], y[j:j+batch_size])):
                for t, word in enumerate(input_text.split()):
                    encoder_input_data[i, t] = input_token_index[word] # encoder input seq
                for t, word in enumerate(target_text.split()):
                    if t<len(target_text.split())-1:
                        decoder_input_data[i, t] = target_token_index[word] # decoder input seq
                    if t>0:
                        # decoder target sequence (one hot encoded)
                        # does not include the START_ token
                        # Offset by one timestep
                        decoder_target_data[i, t - 1, target_token_index[word]] = 1.
            yield([encoder_input_data, decoder_input_data], decoder_target_data)
```

Figure 12: batch generating for datasets

4.8 Hyper Parameter Settings

Each of our deep learning models included a variety of hyper parameter tunings during training depending on the intended use. Accordingly, tests are carried out by altering the values of each hyper parameter, which are described in more detail below.

4.8.1 Activation Functions (AF)

In neural networks, the activation function determines whether a neuron should be actively involved by calculating the weighted sum and applying bias. It can be either linear or non-linear, with the latter being more common and widely used. Neural networks often use ReLUs and softmax in their hidden layers. ReLUs is a non-linear AF that requires less computing power. A different kind of activation function known as Softmax is utilized in neural computing to determine the probability distribution from a vector of real values. Softmax produces a result that goes from 0 to 1, where 1 represents the complete probability. In its hidden layers, every model in this study uses the softmax activation function.

4.8.2 Batch Size (BS)

In this study's experiment, 64 batch sizes are used to train classification models using training datasets. Batch size refers to the number of dataset handled during each iteration, less than the training dataset size and greater than zero. This ensures that 64 words are seen in all epochs before model updates.

4.8.3 Epoch Number (EN)

Epoch number refers to the number of times the entire dataset is sent to a model, and in our demonstration, all models were trained with epochs 50.

4.8.4 Loss Function (LF)

Categorical cross-entropy (CCE) is the most suitable loss function for multi-class classification, where classes are encoded as one-hot encoded vectors. In the case of Afaan Oromoo Morphological Analysis, the proposed models use CCE loss function, as embedded and encoded labelling systems are applied.

4.8.5 Initial Learning Rate (ILR)

Initial learning rates is the rate at which the learning process will start and the optimization algorithm ought to include it. In our study experiment, ILR values between 0.1 and 0.0001 are tried, with 0.01 being selected due to its better efficiency.

4.8.6 Optimization Algorithm (OA)

The choice of deep learning optimization techniques can be influenced by the structure of the models. Studies have shown that adaptive methods are generally superior and computationally efficient. Further research is needed to find even better adaptive methods. Adam, a superior deep learning optimization technique, is also used in these investigations. The Adam optimization algorithm with 0.01 initial learning rates is employed in these investigations.

4.8.7 Early stopping

Some times the models train for specified epochs and might start to learn training dataset which eventually lead to overfitting. To prevent overfitting early stopping mechanism is used in this study. Which is used to stop the model training as soon as the start overfitting.

4.9 Training components of the Models

As shown in chapter 3 architecture, system encoder and decoder part incorporates deep learning approaches of sequence to sequence composed of CNN, LSTM, GRU, and BiLSTM. The system was designed and developed based on these models and the training component of these models are described as:

4.9.1 Normal CNN-LSTM Model

The CNN-LSTM Encoder is a machine learning algorithm that takes input sequences and encodes them into a latent space representation. It has several layers, including the input layer, embedding layer, Convolutional layer, max-pooling layer, LSTM layer, and encoder states. The input layer accepts input sequences with variable lengths, the embedding layer converts the input sequences into dense vectors, the drop-out layer applies drop-out regularization to prevent overfitting, the LSTM layer processes the embedded input sequences and encoder states, and the dense layer applies a softmax activation function to generate the final probability distribution over the target language's vocabulary. The

decoder layer processes the output sequences and updates the decoder states. The CNN-LSTM model is defined by specifying the inputs (encoder and decoder inputs) and outputs (decoder outputs), creating a functional model that can be trained end-to-end. The CNN-LSTM Encoder and CNN-LSTM Decoder are essential components of this machine learning process.

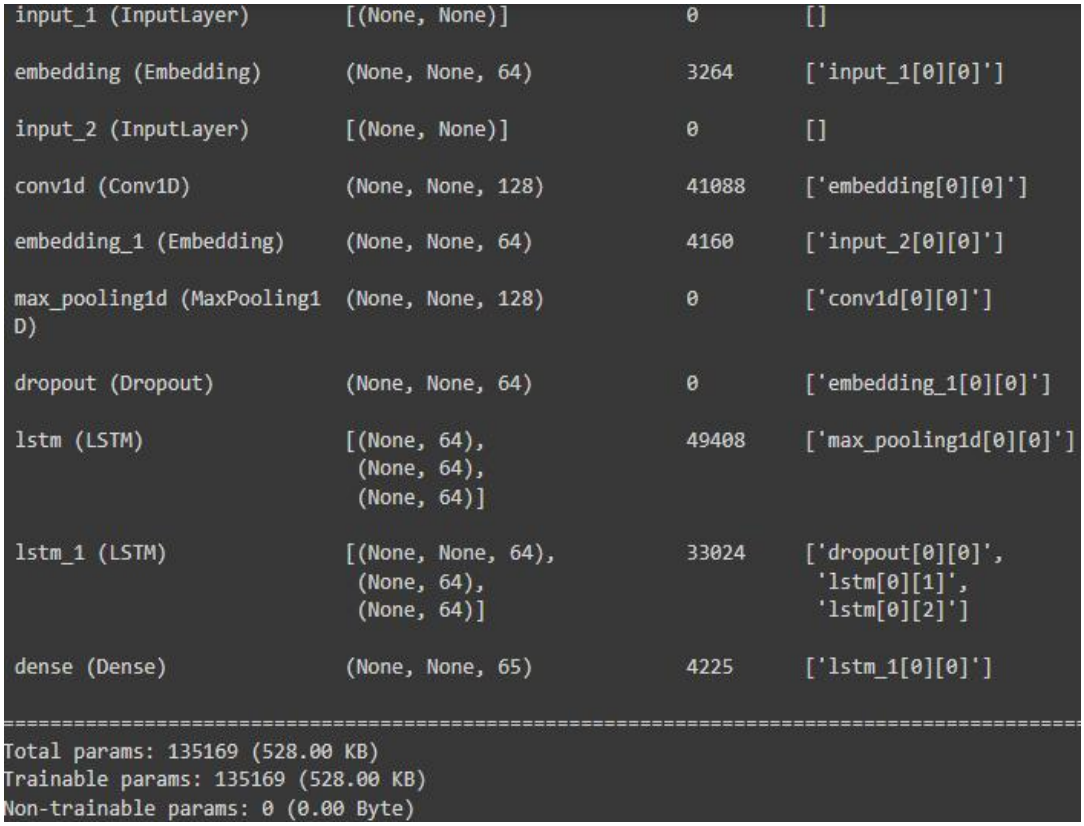
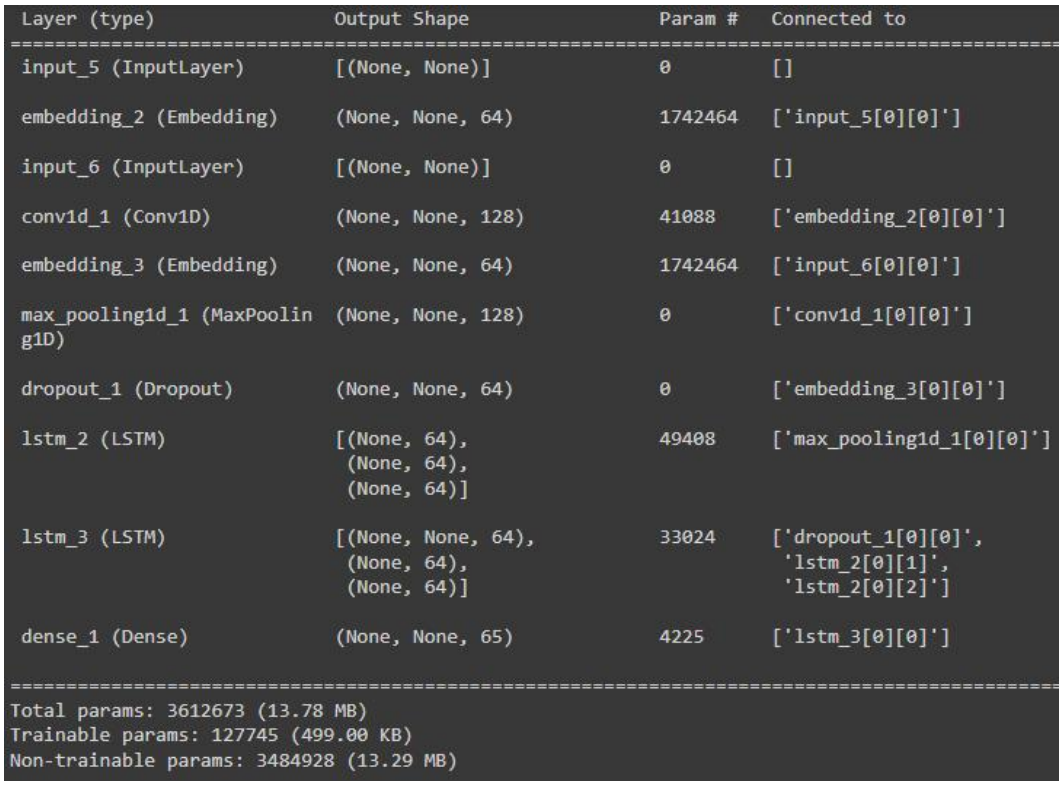


Figure 13: Training Components of normal CNN Model

4.9.2 Word2Vec CNN-LSTM Model

The CNN-LSTM Encoder uses a pre-trained word embedding model (Word2Vec) to encode input sequences and extract features using Convolutional layers. The input layer accepts variable-length input sequences, while the embedding layer converts them into dense vectors using pre-trained Word2Vec embeddings. The Convolutional layer applies Convolutional operations to extract features, using filters and padding to maintain sequence length. The max-pooling layer reduces the dimensionality of the Convolutional layer's output by taking the maximum value within each window of size 2. The LSTM layer processes the output, capturing temporal dependencies within the data, and returns the encoded representation like encoder outputs, hidden and cell states. The CNN-LSTM Decoder follows a similar architecture but in reverse order, using LSTM layers to decode

the encoded representation. The input layer accepts input sequences, the embedding layer converts them into dense vectors, the drop-out layer applies drop-out regularization to prevent overfitting, and the LSTM layer processes the embedded input sequences and encoder states, producing output sequences and updating the decoder states. The final probability distribution over the vocabulary is generated using a softmax activation function.



Layer (type)	Output Shape	Param #	Connected to
input_5 (InputLayer)	[(None, None)]	0	[]
embedding_2 (Embedding)	(None, None, 64)	1742464	['input_5[0][0]']
input_6 (InputLayer)	[(None, None)]	0	[]
conv1d_1 (Conv1D)	(None, None, 128)	41088	['embedding_2[0][0]']
embedding_3 (Embedding)	(None, None, 64)	1742464	['input_6[0][0]']
max_pooling1d_1 (MaxPooling1D)	(None, None, 128)	0	['conv1d_1[0][0]']
dropout_1 (Dropout)	(None, None, 64)	0	['embedding_3[0][0]']
lstm_2 (LSTM)	[(None, 64), (None, 64), (None, 64)]	49408	['max_pooling1d_1[0][0]']
lstm_3 (LSTM)	[(None, None, 64), (None, 64), (None, 64)]	33024	['dropout_1[0][0]', 'lstm_2[0][1]', 'lstm_2[0][2]']
dense_1 (Dense)	(None, None, 65)	4225	['lstm_3[0][0]']

Total params: 3612673 (13.78 MB)
Trainable params: 127745 (499.00 KB)
Non-trainable params: 3484928 (13.29 MB)

Figure 14: Training Components of word2vec CNN Model

4.9.3 FastText CNN-LSTM Model

The CNN-LSTM Encoder is a code snippet that uses pre-trained word embeddings from a FastText model to encode input sequences. It consists of an input layer that accepts variable-length input sequences, an embedding layer that converts the input sequences into dense vectors, a Convolutional layer that applies Convolutional operations, a max-pooling layer that reduces the dimensionality of the Convolutional layer's output, an LSTM layer that processes the output, and an encoder states that represent the encoded input sequence.The CNN-LSTM Decoder follows a similar architecture, but in reverse order, using LSTM layers to decode the encoded representation. It starts with an input layer that accepts input sequences for decoding, then an embedding layer that converts the input sequences into dense vectors using pre-trained FastText embeddings. The drop-out layer

applies drop-out regularization to prevent overfitting, and the LSTM layer processes the embedded input sequences and encoder states, producing output sequences and updating the decoder states. The final probability distribution over the vocabulary is generated by applying a softmax activation function to the output sequences.This model is defined by specifying the inputs and outputs, creating a functional model that can be trained end-to-end.

Layer (type)	Output Shape	Param #	Connected to
input_15 (InputLayer)	[(None, None)]	0	[]
embedding_6 (Embedding)	(None, None, 64)	1742464	['input_15[0][0]']
input_16 (InputLayer)	[(None, None)]	0	[]
conv1d_3 (Conv1D)	(None, None, 128)	41088	['embedding_6[0][0]']
embedding_7 (Embedding)	(None, None, 64)	1742464	['input_16[0][0]']
max_pooling1d_3 (MaxPooling1D)	(None, None, 128)	0	['conv1d_3[0][0]']
dropout_3 (Dropout)	(None, None, 64)	0	['embedding_7[0][0]']
lstm_6 (LSTM)	[(None, 64), (None, 64), (None, 64)]	49408	['max_pooling1d_3[0][0]']
lstm_7 (LSTM)	[(None, None, 64), (None, 64), (None, 64)]	33024	['dropout_3[0][0]', 'lstm_6[0][1]', 'lstm_6[0][2]']
dense_3 (Dense)	(None, None, 65)	4225	['lstm_7[0][0]']

Total params: 3612673 (13.78 MB)
Trainable params: 127745 (499.00 KB)
Non-trainable params: 3484928 (13.29 MB)

Figure 15: Training Components of FastText CNN Model

4.9.4 Normal LSTM Model

The Encoder-Decoder architecture is a sequence-to-sequence task that is employed in machine translation and natural language processing. It comprises of a decoder, an LSTM layer, an embedding layer, a drop-out layer, and an encoder that takes input sequences of varying lengths.The encoder input layer receives input sequences of varying lengths, transforms the inputs into dense vectors, and sends the output sequence back.Together with the hidden state and cell state of the most recent time step, the decoder LSTM layer analyses the embedded input sequences and provides the output sequence.In order to prevent overfitting, the drop-out layer performs drop-out regularization to the embedded decoder input.The LSTM layer, a decoder, receives inputs from the embedded decoder and its starting states. It then processes them, delivers the output sequences, and updates the states of the decoder. The final probability distribution across the vocabulary is produced

by the decoder dense layer by applying a four softmax activation function to the decoder outputs. By defining inputs and outputs, the LSTM model may be created and trained from start to finish.The encoder processes the input sequence, while the decoder creates the output sequence based on the encoded information. This design is often used for tasks such as machine translation.

```
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
input_20 (InputLayer)        [(None, None)]            0         []

input_19 (InputLayer)        [(None, None)]            0         []

embedding_9 (Embedding)      (None, None, 64)          4160      ['input_20[0][0]']

embedding_8 (Embedding)      (None, None, 64)          3264      ['input_19[0][0]']

dropout_4 (Dropout)          (None, None, 64)          0         ['embedding_9[0][0]']

lstm_8 (LSTM)                [(None, 64),              33024     ['embedding_8[0][0]']
                              (None, 64),
                              (None, 64)]

lstm_9 (LSTM)                [(None, None, 64),        33024     ['dropout_4[0][0]',
                              (None, 64),                        'lstm_8[0][1]',
                              (None, 64)]                        'lstm_8[0][2]']

dense_4 (Dense)              (None, None, 65)          4225      ['lstm_9[0][0]']

==================================================================================================
Total params: 77697 (303.50 KB)
Trainable params: 77697 (303.50 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 16: Training Components of normal LSTM Model

4.9.5 Word2Vec LSTM Model

This component defines an LSTM Encoder-Decoder architecture, a variant of the sequence-to-sequence model used for tasks like machine translation or sequence generation. The architecture consists of an LSTM Encoder, which accepts input sequences with variable lengths, an Embedding Layer, an LSTM Layer, a Drop-out Layer, and a Dense Layer. The LSTM Encoder converts the input sequences into dense vectors using pre-trained Word2Vec embeddings, while the LSTM Decoder processes the input sequences and returns the output sequences, hidden states, and cell states of the last time step. The LSTM Decoder applies drop-out regularization to the embedded decoder inputs to prevent overfitting. The LSTM Layer processes the embedded decoder inputs and initial states obtained from the encoder, returning the output sequences and updating the decoder states. The Dense Layer applies a softmax activation function to the decoder outputs, producing

the final probability distribution over the vocabulary. The model definition specifies the inputs and outputs, creating a functional model that can be trained end-to-end. This architecture leveraged pre-trained word embeddings and LSTM layers to encode and decode input sequences, generating output sequences based on learned representations.

```
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
input_24 (InputLayer)        [(None, None)]            0         []

input_23 (InputLayer)        [(None, None)]            0         []

embedding_11 (Embedding)     (None, None, 64)          1742464   ['input_24[0][0]']

embedding_10 (Embedding)     (None, None, 64)          1742464   ['input_23[0][0]']

dropout_5 (Dropout)          (None, None, 64)          0         ['embedding_11[0][0]']

lstm_10 (LSTM)               [(None, 64),              33024     ['embedding_10[0][0]']
                              (None, 64),
                              (None, 64)]

lstm_11 (LSTM)               [(None, None, 64),        33024     ['dropout_5[0][0]',
                              (None, 64),                         'lstm_10[0][1]',
                              (None, 64)]                         'lstm_10[0][2]']

dense_5 (Dense)              (None, None, 65)          4225      ['lstm_11[0][0]']

==================================================================================================
Total params: 3555201 (13.56 MB)
Trainable params: 70273 (274.50 KB)
Non-trainable params: 3484928 (13.29 MB)
```

Figure 17: Training Components of word2vec LSTM Model

4.9.6 FastText LSTM Model

This component defines an LSTM Encoder-Decoder architecture using pre-trained FastText word embeddings. The architecture consists of two components: an LSTM Encoder using FastText, which accepts input sequences with variable lengths, and an LSTM Decoder using FastText. The LSTM Encoder uses pre-trained FastText embeddings to convert input sequences into dense vectors of fixed size, while the LSTM Decoder uses FastText embeddings to process the input sequences and return the output sequences. The LSTM Decoder uses the same embedding dimension as the encoder and applies drop-out regularization to prevent overfitting. The LSTM Decoder processes the embedded decoder inputs and initial states obtained from the encoder, returning the output sequences and updating the decoder states. The LSTM Decoder uses a softmax activation function to produce the final probability distribution over the vocabulary. The model definition specifies the inputs (encoder and decoder inputs) and outputs, creating a functional model that can be trained end-to-end. This architecture uses pre-trained FastText word

embeddings and LSTM layers to encode and decode input sequences, generating output sequences based on learned representations.

```
Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
input_10 (InputLayer)       [(None, None)]            0         []

input_9 (InputLayer)        [(None, None)]            0         []

embedding_5 (Embedding)     (None, None, 64)          1742464   ['input_10[0][0]']

embedding_4 (Embedding)     (None, None, 64)          1742464   ['input_9[0][0]']

dropout_2 (Dropout)         (None, None, 64)          0         ['embedding_5[0][0]']

lstm_4 (LSTM)               [(None, 64),              33024     ['embedding_4[0][0]']
                             (None, 64),
                             (None, 64)]

lstm_5 (LSTM)               [(None, None, 64),        33024     ['dropout_2[0][0]',
                             (None, 64),                         'lstm_4[0][1]',
                             (None, 64)]                         'lstm_4[0][2]']

dense_2 (Dense)             (None, None, 65)          4225      ['lstm_5[0][0]']

==================================================================================================
Total params: 3555201 (13.56 MB)
Trainable params: 70273 (274.50 KB)
Non-trainable params: 3484928 (13.29 MB)
```

Figure 18: Training Components of FastText LSTM Model

4.9.7 Normal GRU Model

This component implements a GRU Encoder-Decoder architecture for sequence-to-sequence operations. This architecture is extensively used in machine translation and natural language processing. The GRU Encoder, which takes input sequences with variable lengths; the Embedding Layer, which transforms the input sequences into dense vectors of fixed size; the GRU Layer, which processes the embedded input sequences; and the Decoder States, which represent one of the encoded input sequence. This is the architecture.Conversely, the GRU Decoder employs the same embedding dimension as the encoder and takes input sequences for decoding.In order to prevent overfitting, the Drop-out Layer performs drop-out regularization to the embedded decoder input.The decoder, GRU Layer, receives inputs from the embedded decoder and its starting states. It then processes them, delivers the output sequences, and updates the states of the decoder. The final probability distribution across the vocabulary is produced by the Decoder States layer, which applies a four-softmax activation function to the decoder outputs. The model definition is defined by specifying the inputs (encoder and decoder inputs) and the output

(decoder outputs), creating a functional model that can be trained end-to-end. This architecture leveraged GRU layers for both encoding and decoding, generating output sequences based on learned representations.

Layer (type)	Output Shape	Param #	Connected to
input_32 (InputLayer)	[(None, None)]	0	[]
input_31 (InputLayer)	[(None, None)]	0	[]
embedding_15 (Embedding)	(None, None, 64)	4160	['input_32[0][0]']
embedding_14 (Embedding)	(None, None, 64)	3264	['input_31[0][0]']
dropout_7 (Dropout)	(None, None, 64)	0	['embedding_15[0][0]']
gru (GRU)	[(None, 64), (None, 64)]	24960	['embedding_14[0][0]']
gru_1 (GRU)	[(None, None, 64), (None, 64)]	24960	['dropout_7[0][0]', 'gru[0][1]']
dense_7 (Dense)	(None, None, 65)	4225	['gru_1[0][0]']

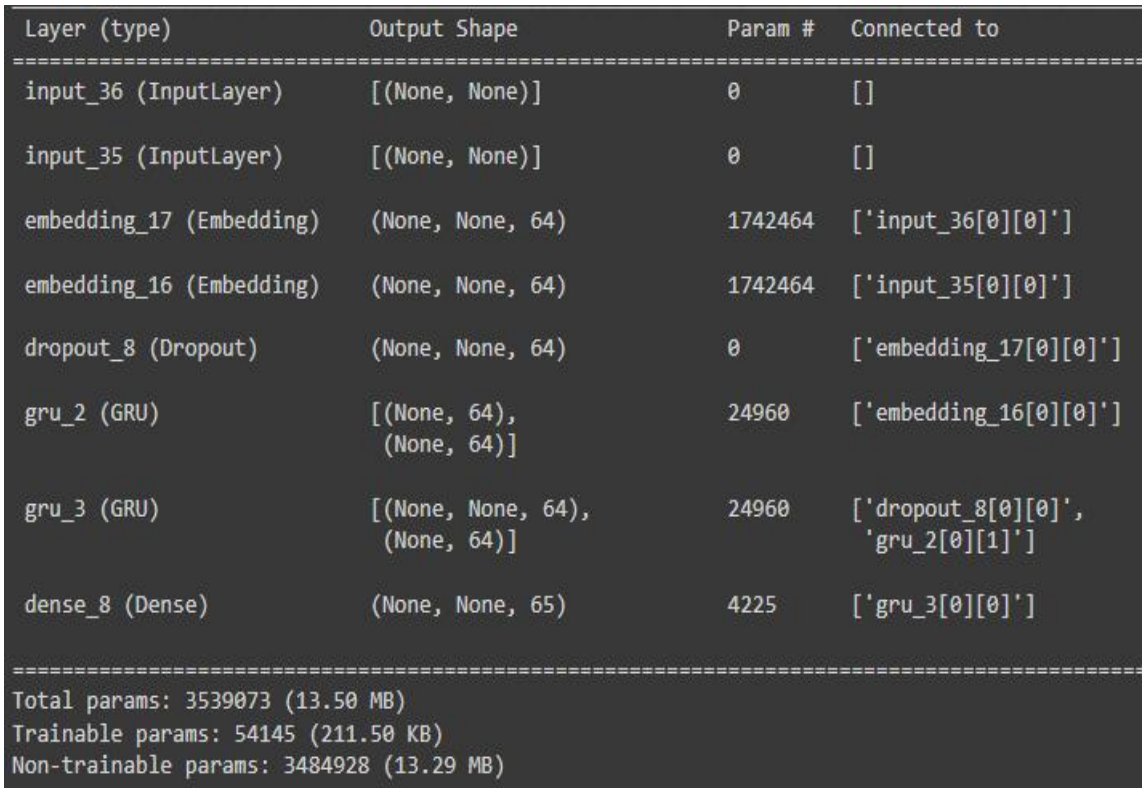
Total params: 61569 (240.50 KB)
Trainable params: 54145 (211.50 KB)
Non-trainable params: 7424 (29.00 KB)

Figure 19: Training Components of GRU Model

4.9.8 Word2Vec GRU Model

This component defines a GRU Encoder-Decoder architecture using pre-trained Word2Vec embeddings. The architecture consists of an input layer, a decoder layer, and a drop-out layer. The encoder accepts input sequences with variable lengths and converts them into dense vectors using pre-trained Word2Vec embeddings. The encoder GRU layer, processes the input sequences and returns the output sequence and hidden state. The assigning `return state parameter to true ensures that the GRU layer returns these states. The encoder states list contains the final hidden state of the GRU layer, representing the encoded input sequence.The decoder layer accepts input sequences for decoding and converts them into dense vectors using the same embedding dimension as the encoder. The drop-out layer applies drop-out regularization to the embedded decoder inputs to prevent overfitting. The decoder GRU layer, processes the decoder inputs and initial states obtained from the encoder, returning the output sequences and updating the decoder states. The decoder dense layer applies a softmax activation function to the decoder outputs,

producing the final probability distribution over the vocabulary. The word2vec gru model is defined by specifying the inputs and outputs, allowing for end-to-end training. This architecture leveraged pre-trained Word2Vec embeddings and GRU layers for encoding and decoding, generating output sequences based on learned representations.



Layer (type)	Output Shape	Param #	Connected to
input_36 (InputLayer)	[(None, None)]	0	[]
input_35 (InputLayer)	[(None, None)]	0	[]
embedding_17 (Embedding)	(None, None, 64)	1742464	['input_36[0][0]']
embedding_16 (Embedding)	(None, None, 64)	1742464	['input_35[0][0]']
dropout_8 (Dropout)	(None, None, 64)	0	['embedding_17[0][0]']
gru_2 (GRU)	[(None, 64), (None, 64)]	24960	['embedding_16[0][0]']
gru_3 (GRU)	[(None, None, 64), (None, 64)]	24960	['dropout_8[0][0]', 'gru_2[0][1]']
dense_8 (Dense)	(None, None, 65)	4225	['gru_3[0][0]']

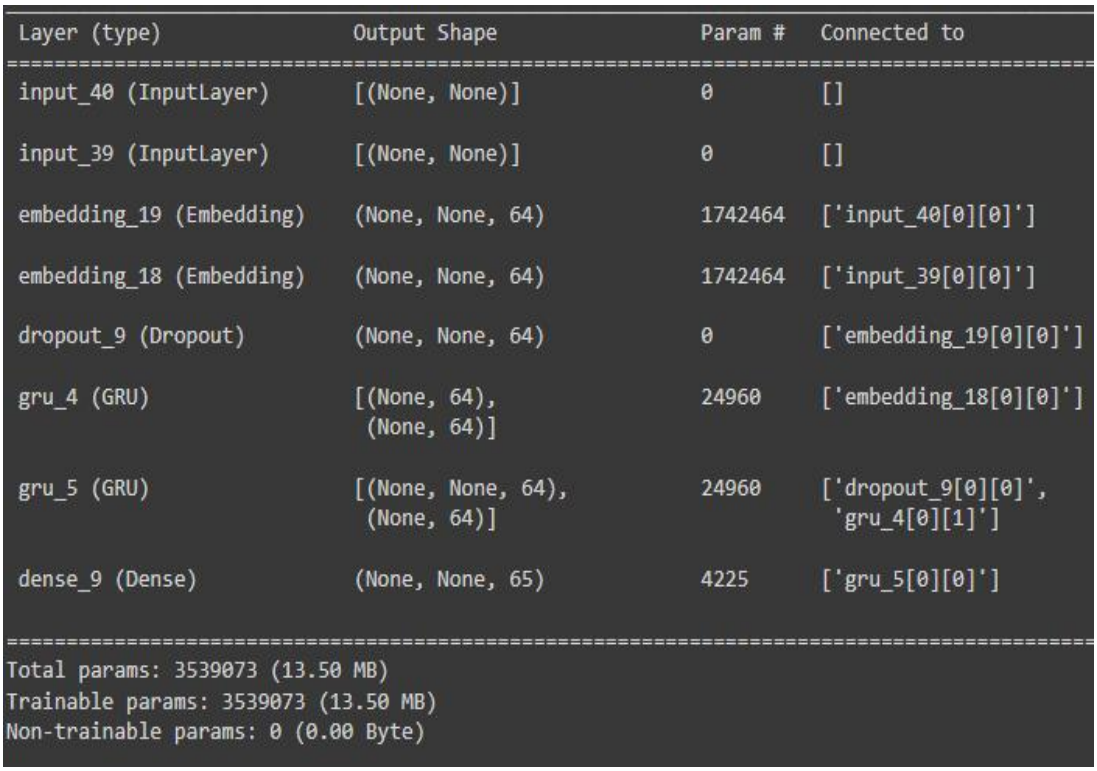
Total params: 3539073 (13.50 MB)
Trainable params: 54145 (211.50 KB)
Non-trainable params: 3484928 (13.29 MB)

Figure 20: Training Components of fword2vec GRU Model

4.9.9 FastText GRU Model

This component defines an Encoder-Decoder architecture with GRU (Gated Recurrent Unit) layers, utilizing pre-trained FastText embeddings for the encoder. The encoder accepts input sequences with variable lengths and converts them into dense vectors using pre-trained FastText embeddings. The encoder GRU layer, processes the input sequences and returns the output sequence and hidden state. The encoder states are a list of the final hidden state of the GRU layer.The decoder accepts input sequences for decoding and converts them into dense vectors using the same embedding dimension as the encoder. The drop-out layer applies drop-out regularization to the embedded decoder inputs to prevent overfitting. The decoder GRU layer, processes the decoder inputs and initial states obtained from the encoder, returning the output sequences and updating the decoder states. The decoder dense layer applies a softmax activation function to the decoder outputs, producing the final probability distribution over the vocabulary. The model definition,

fastText gru model, specifies the inputs and outputs, creating a functional model that can be trained end-to-end. This architecture utilizes pre-trained FastText embeddings and GRU layers for both encoding and decoding, enabling the generation of output sequences based on learned representations.



Layer (type)	Output Shape	Param #	Connected to
input_40 (InputLayer)	[(None, None)]	0	[]
input_39 (InputLayer)	[(None, None)]	0	[]
embedding_19 (Embedding)	(None, None, 64)	1742464	['input_40[0][0]']
embedding_18 (Embedding)	(None, None, 64)	1742464	['input_39[0][0]']
dropout_9 (Dropout)	(None, None, 64)	0	['embedding_19[0][0]']
gru_4 (GRU)	[(None, 64), (None, 64)]	24960	['embedding_18[0][0]']
gru_5 (GRU)	[(None, None, 64), (None, 64)]	24960	['dropout_9[0][0]', 'gru_4[0][1]']
dense_9 (Dense)	(None, None, 65)	4225	['gru_5[0][0]']

Total params: 3539073 (13.50 MB)
Trainable params: 3539073 (13.50 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 21: Training Components of FastText GRU Model

4.9.10 Normal BiLSTM Model

This component defines a sequence-to-sequence model using a Bidirectional LSTM (BiLSTM) encoder and LSTM decoder. The model includes an input sequence, an embedding layer, a bidirectional LSTM encoder, a decoder input sequence, a decoder embedding layer, an LSTM decoder, and an output layer. The input sequences are accepted with variable lengths, and the embedding layer converts the input sequences into dense vectors. The bidirectional LSTM encoder processes the embedded input sequences, returning the combined output and final hidden states of both forward and backward LSTMs. The final hidden states are concatenated to form the final hidden state and cell state. The decoder input sequence accepts input sequences for decoding, and the decoder embedding layer converts the input sequences into dense vectors for the decoder. The LSTM decoder processes the embedded decoder inputs, returning output sequences and updating the decoder states. The output layer applies a softmax activation function to the

decoder outputs, producing the final probability distribution over the vocabulary. The model is defined by specifying the inputs and outputs, allowing for end-to-end training. This architecture leveraged BiLSTM for capturing bidirectional context information and an LSTM decoder for generating output sequences based on learned representations.

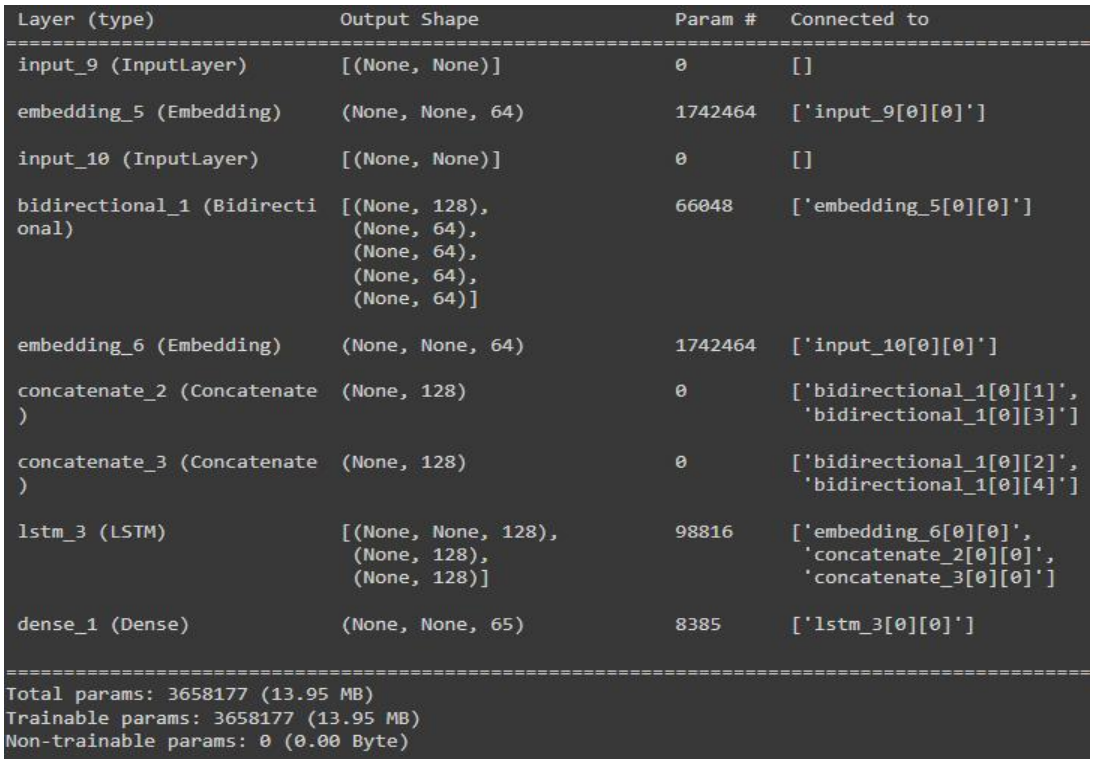
```
Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
input_23 (InputLayer)       [(None, None)]            0         []

embedding_13 (Embedding)    (None, None, 64)          3264      ['input_23[0][0]']

input_24 (InputLayer)       [(None, None)]            0         []

bidirectional_5 (Bidirecti  [(None, 128),             66048     ['embedding_13[0][0]']
onal)                        (None, 64),
                             (None, 64),
                             (None, 64),
                             (None, 64)]

embedding_14 (Embedding)    (None, None, 64)          4160      ['input_24[0][0]']

concatenate_8 (Concatenate  (None, 128)               0         ['bidirectional_5[0][1]',
)                                                               'bidirectional_5[0][3]']

concatenate_9 (Concatenate  (None, 128)               0         ['bidirectional_5[0][2]',
)                                                               'bidirectional_5[0][4]']

lstm_7 (LSTM)               [(None, None, 128),       98816     ['embedding_14[0][0]',
                             (None, 128),                       'concatenate_8[0][0]',
                             (None, 128)]                       'concatenate_9[0][0]']

dense_5 (Dense)             (None, None, 65)          8385      ['lstm_7[0][0]']

==================================================================================================
Total params: 180673 (705.75 KB)
Trainable params: 180673 (705.75 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 22: Training Components of normal BiLSTM Model

4.9.11 Word2Vec BiLSTM Model

This part defines a sequence-to-sequence model using pre-trained Word2Vecembeddings and a Bidirectional LSTM (BiLSTM) encoder and LSTM decoder.The model is composed of an output layer, a bidirectional LSTM encoder, an input sequence for the decoder, a decoder embedding layer, and an output sequence.The encoder, which has 38 layers, analyses the input sequences and outputs the combined output along with the final hidden states of both forward and backward LSTMs.The forward and backward hidden states are concatenated by the state concatenation procedure to create the final hidden state. The input sequence of the decoder receives input sequences for encoding, and the input sequences are converted into dense vectors for the decoder by the decoder embedding layer.The LSTM layer decoder processes the inputs from the embedded decoder, returning

output sequences and changing the decoder states.The final probability distribution across the vocabulary is produced by the output layer by applying a softmax activation function to the decoder outputs. The model can be trained end-to-end, leveraging BiLSTM for the encoder to capture bidirectional context information and an LSTM decoder to generate output sequences based on learned representations, with Word2Vec embeddings providing the initial representations.



Layer (type)	Output Shape	Param #	Connected to
input_9 (InputLayer)	[(None, None)]	0	[]
embedding_5 (Embedding)	(None, None, 64)	1742464	['input_9[0][0]']
input_10 (InputLayer)	[(None, None)]	0	[]
bidirectional_1 (Bidirectional)	[(None, 128), (None, 64), (None, 64), (None, 64), (None, 64)]	66048	['embedding_5[0][0]']
embedding_6 (Embedding)	(None, None, 64)	1742464	['input_10[0][0]']
concatenate_2 (Concatenate)	(None, 128)	0	['bidirectional_1[0][1]', 'bidirectional_1[0][3]']
concatenate_3 (Concatenate)	(None, 128)	0	['bidirectional_1[0][2]', 'bidirectional_1[0][4]']
lstm_3 (LSTM)	[(None, None, 128), (None, 128), (None, 128)]	98816	['embedding_6[0][0]', 'concatenate_2[0][0]', 'concatenate_3[0][0]']
dense_1 (Dense)	(None, None, 65)	8385	['lstm_3[0][0]']

Total params: 3658177 (13.95 MB)
Trainable params: 3658177 (13.95 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 23: Training Components of word2vec BiLSTM Model

4.9.12 FastText BiLSTM Model

This part defines a sequence-to-sequence model using pre-trained Fast Text Model embeddings and a Bidirectional LSTM (BiLSTM) encoder and LSTM decoder.The model is composed of an output layer, a bidirectional LSTM encoder, an input sequence for the decoder, a decoder embedding layer, and an input sequence for the decoder.The encoder, which has a layers, analyses the input sequences and outputs the combined output along with the final hidden states of both forward and backward LSTMs.The forward and backward hidden states are concatenated by the state concatenation procedure to create the final hidden state. The input sequence of the decoder receives input sequences for encoding, and the input sequences are converted into dense vectors for the decoder by the decoder embedding layer.The LSTM layer decoder processes the inputs from the embedded

decoder, returning output sequences and changing the decoder states.The final probability distribution across the vocabulary is produced by the output layer by applying a softmax activation function to the decoder outputs. The model can be trained end-to-end, leveraging BiLSTM for the encoder to capture bidirectional context information and an LSTM decoder to generate output sequences based on learned representations. FastText embeddings provide the initial representations.

```
Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
input_13 (InputLayer)       [(None, None)]            0         []

embedding_7 (Embedding)     (None, None, 64)          1742464   ['input_13[0][0]']

input_14 (InputLayer)       [(None, None)]            0         []

bidirectional_2 (Bidirecti  [(None, 128),             66048     ['embedding_7[0][0]']
onal)                        (None, 64),
                             (None, 64),
                             (None, 64),
                             (None, 64)]

embedding_8 (Embedding)     (None, None, 64)          1742464   ['input_14[0][0]']

concatenate_4 (Concatenate  (None, 128)               0         ['bidirectional_2[0][1]',
)                                                               'bidirectional_2[0][3]']

concatenate_5 (Concatenate  (None, 128)               0         ['bidirectional_2[0][2]',
)                                                               'bidirectional_2[0][4]']

lstm_5 (LSTM)               [(None, None, 128),       98816     ['embedding_8[0][0]',
                             (None, 128),                       'concatenate_4[0][0]',
                             (None, 128)]                       'concatenate_5[0][0]']

dense_2 (Dense)             (None, None, 65)          8385      ['lstm_5[0][0]']

==================================================================================================
Total params: 3658177 (13.95 MB)
Trainable params: 3658177 (13.95 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 24: Training Components of FastText BiLSTM Model

4.10 Result Analysis

Result analysis is a crucial process in the development of sequence-to-sequence models. It involves evaluating the model's performance using various metrics and examining the generated outputs. Common aspects of result analysis include quantitative evaluation, qualitative evaluation, comparison analysis, hyper parameter tuning, generalization and robustness, error correction and refinement, and feedback loop. In this study; performance analysis based on qualitative, quantitative, drop-out and comparison criteria are conducted.

4.10.1 Performance Analysis of CNN-LSTM models

The study analyzes performance metrics for three different models: Normal CNN-LSTM, Word2vec CNN-LSTM, and FastText CNN-LSTM models. Normal CNN-LSTM model has a moderate accuracy but high precision, with a 70.94% accuracy rate. It often predicts a class correctly, but has lower recall, suggesting it misses a significant portion of relevant instances. The F1-score balances precision and recall. Word2vec CNN-LSTM model has a high accuracy rate of 94.74%, with a 96.33% precision rate and a 93.64% recall rate. Its F1-score is 94.56%. Fast Text CNN-LSTM model, on the other hand, has a high accuracy rate of 95.25%, a 96.46% precision rate, a 94.38% recall rate, and a 94.74% F1-score. Both word2vec and fast text based models outperform Normal CNN-LSTM in terms of accuracy, precision, recall, and F1-score. However, Normal CNN-LSTM has the lowest performance metrics, suggesting it may benefit from improvements in feature representation or architecture. The study recommends exploring more sophisticated architectures or pre-trained embeddings for Normal CNN-LSTM, investigating additional data or data augmentation techniques to enhance model generalization, and further analyzing mis classified instances to identify patterns and potential improvement areas.

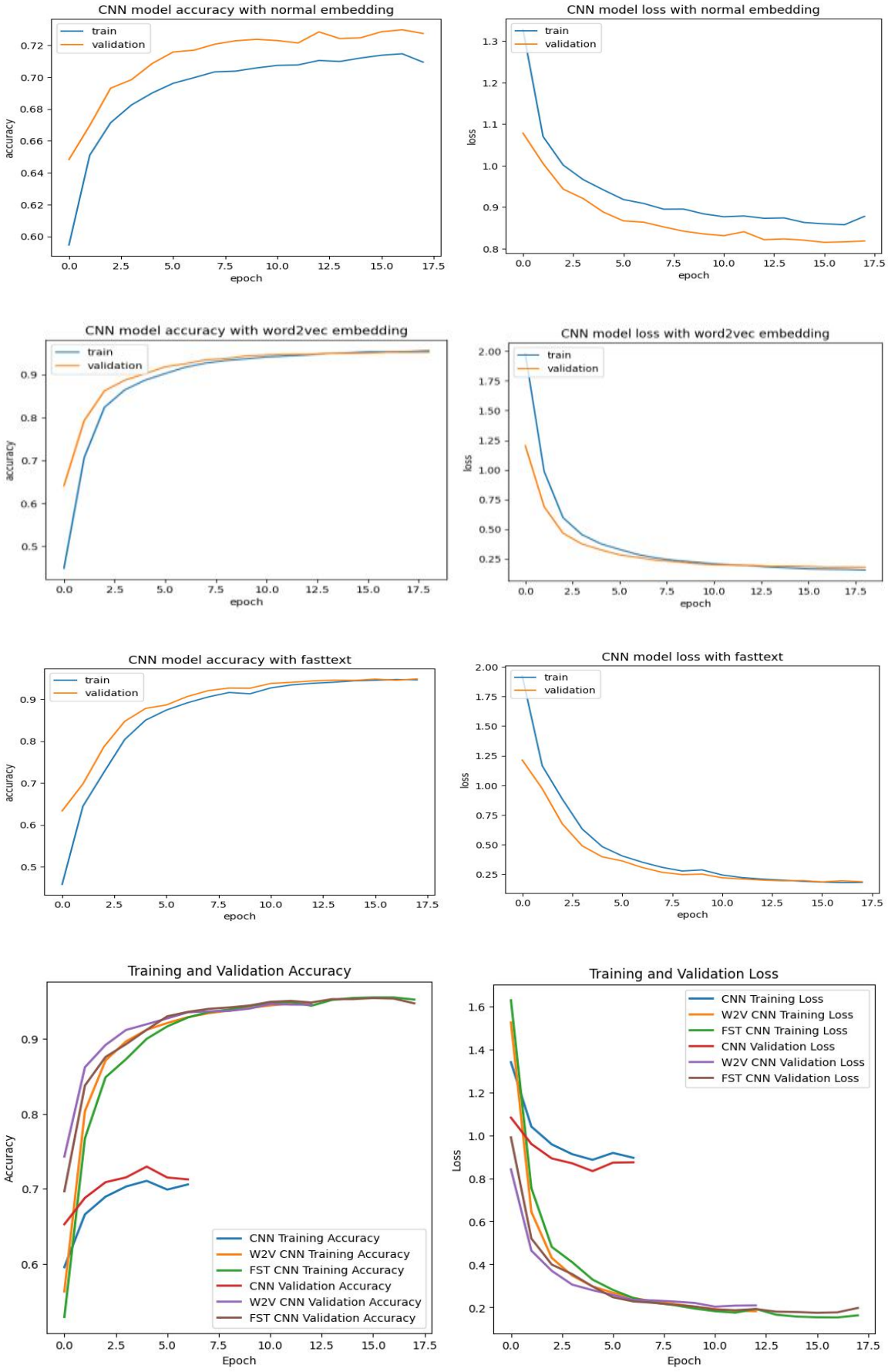


Figure 25: Training, validation accuracy and loss of all CNN Models

4.10.2 Performance Analysis of LSTM models

The performance analysis of Normal LSTM, Word2vec LSTM, and FST-LSTM models reveals that Normal LSTM outperforms FST-LSTM in all metrics, indicating the effectiveness of traditional LSTM and Word2Vec embeddings over Fast Text embeddings for this task. Normal LSTM achieves high Accuracy with 95.06%, Precision with 96.09%, Recall with 94.32%, and F1-Score with 95.41% indicating its effectiveness in correctly classifying instances while maintaining a good balance between precision and recall. Word2vec LSTM achieved Accuracy with 93.89%, Precision with 95.72%, Recall with 92.59%, and F1-Score with 93.84%. It also performs well, though slightly lower than Normal LSTM, indicating the effectiveness of Word2Vec embeddings in capturing semantic information. However, there is a slight decrease in performance compared to Normal LSTM across all metrics. FST-LSTM achieved Accuracy with 90.02%, Precision with 93.26%, Recall with 87.61%, F1-Score with 90.62% shows the lowest performance among the three LSTM models, indicating that Fast Text embeddings might not be as effective for this particular task or that the model architecture needs further optimization. To improve LSTM models' performance, further analysis should be conducted to identify areas where FST-LSTM under performs compared to Normal LSTM and Word2vec LSTM. Model tuning should be explored to optimize FST-LSTM performance and potentially bridge the performance gap with Normal LSTM and Word2vec LSTM. Embedding techniques and ensemble methods should be explored to find alternative techniques and pre-trained embeddings suitable for the task. These recommendations can help users make informed decisions about embeddings and model architectures for similar classification tasks.

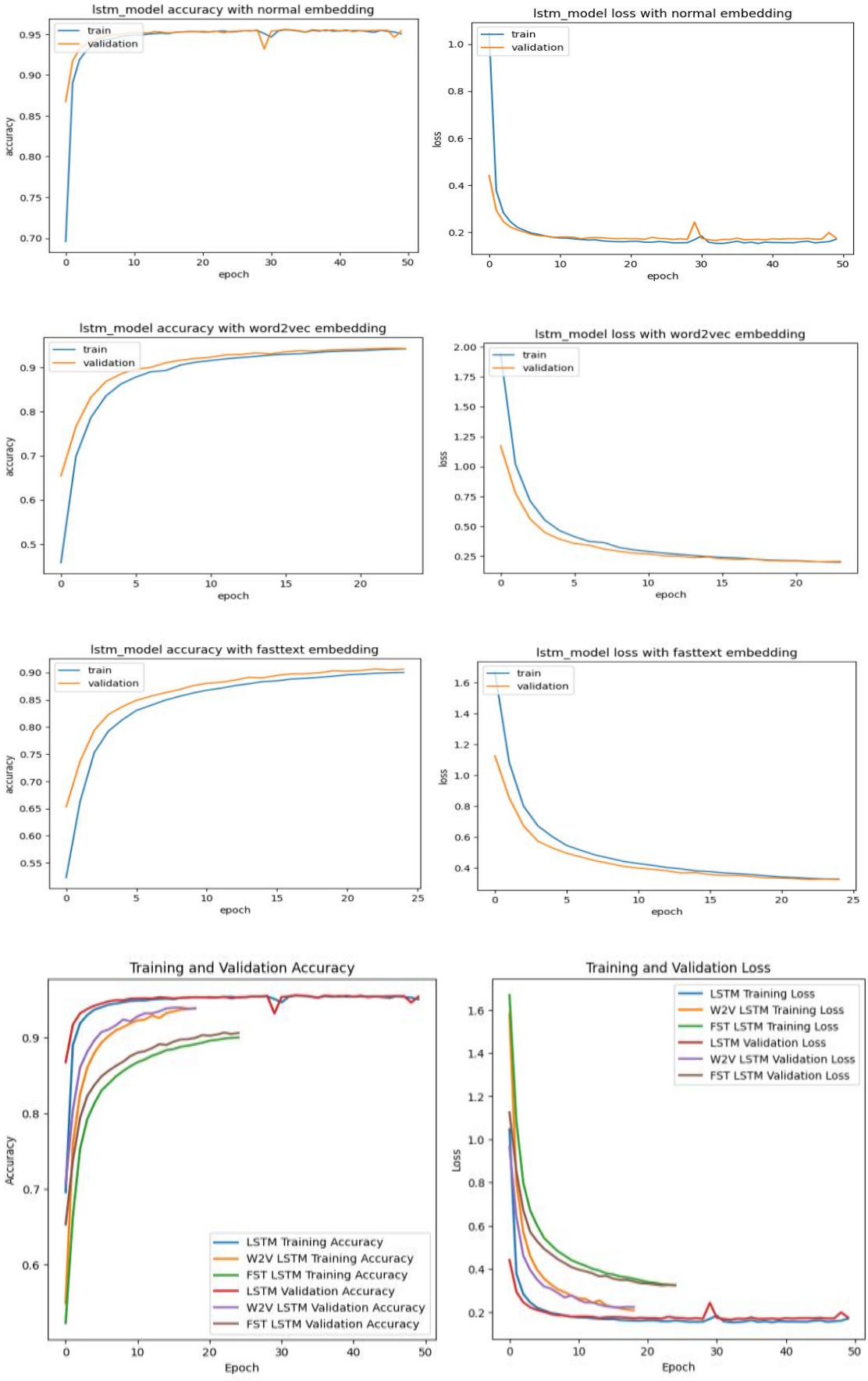


Figure 26: Training, validation accuracy and loss of all LSTM models

4.10.3 Performance Analysis of GRU models

The performance analysis of Normal GRU, Word2vec GRU, and Fast Text GRU models reveals moderate accuracy across all datasets, with training accuracy of 92.96%, validation accuracy of 91.59%, and testing accuracy of 83.6%. The model shows decent precision and recall, with slightly higher values in training and validation compared to testing. The F1-Score is a fairly balanced score, indicating a good balance between precision and recall. Word2vec GRU exhibits consistent accuracy across all datasets, slightly lower than Normal GRU, with training accuracy of 91.98%, validation accuracy of 92.73%, and testing accuracy of 88.8%. The model slightly outperforms Normal GRU in precision and recall. The F1-Score is a less balanced score, indicating a compromise between precision and recall. All three models exhibit some degree of Overfitting, with Normal GRU and Word2vec GRU showing relatively less overfitting compared to FastText GRU. Word2vec GRU is the most balanced performer, with consistently high accuracy, precision, recall, and F1-Score. Normal GRU follows closely, with slightly higher accuracy but marginally lower precision and recall. FastText GRU lags behind both Normal GRU and Word2vec GRU in terms of overall performance, showing lower accuracy, precision, recall, and F1-Score.

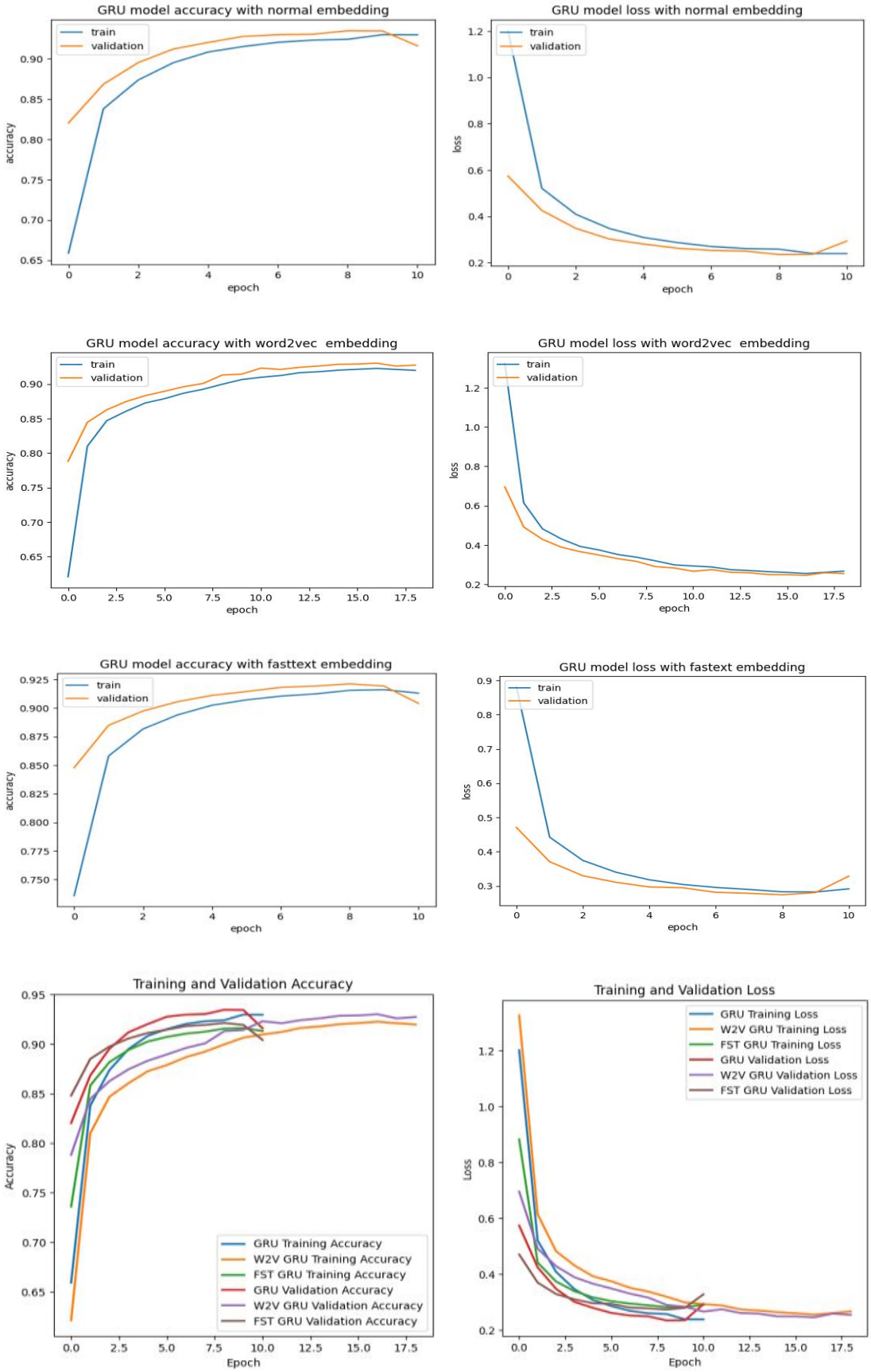


Figure 27: Training, validation accuracy and loss of all GRU models

4.10.4 Performance Analysis of BiLSTM Models

The Bidirectional LSTM models with different embeddings - normal embedding, Word2Vec embedding, and Fast Text embedding - were compared based on their training performance, validation performance, ability to generalize to unseen data, and prevention of overfitting. Normal Embedding BiLSTM: Moderate training performance initially with a loss of 0.7746 and accuracy of 0.7738 in the first epoch. Steady improvement throughout training, reaching an accuracy of 0.9527 and a loss of 0.1637 by the ninth epoch. Validation performance consistently increases, reaching an accuracy of 0.9563. Generalizes well to unseen data with successful predictions on Afaan Oromoo words. Overfitting is not a significant issue, and early stopping prevents it after the twelfth epoch. Word2Vec Embedding BiLSTM: Demonstrates decreasing loss and increasing accuracy, precision, and recall over 11 epochs. Validation loss decreases from 0.7039 to 0.1356, indicating improved performance. High precision, recall, accuracy, and F1-score, suggesting effectiveness in predicting Afaan Oromoo word roots. Generalizes well to unseen data, with early stopping preventing overfitting after the 11th epoch. FastText Embedding BiLSTM: Shows steady improvement over epochs with decreasing loss and increasing accuracy, precision, and recall. High validation metrics (accuracy, precision, recall, F1-score) indicating good generalization to unseen data. It Successfully predicts word roots and their features in Afaan Oromoo words. No significant gap between training and validation metrics, indicating good generalization and no overfitting. Early stopping further mitigates the risk of overfitting. In summary, all three models show strong performance in predicting Afaan Oromoo word roots and their features. They demonstrate effectiveness in generalizing to unseen data and preventing overfitting through the use of early stopping. However, the FastText embedding BiLSTM appears to have slightly higher validation metrics, suggesting it may offer a slight advantage in this specific task.

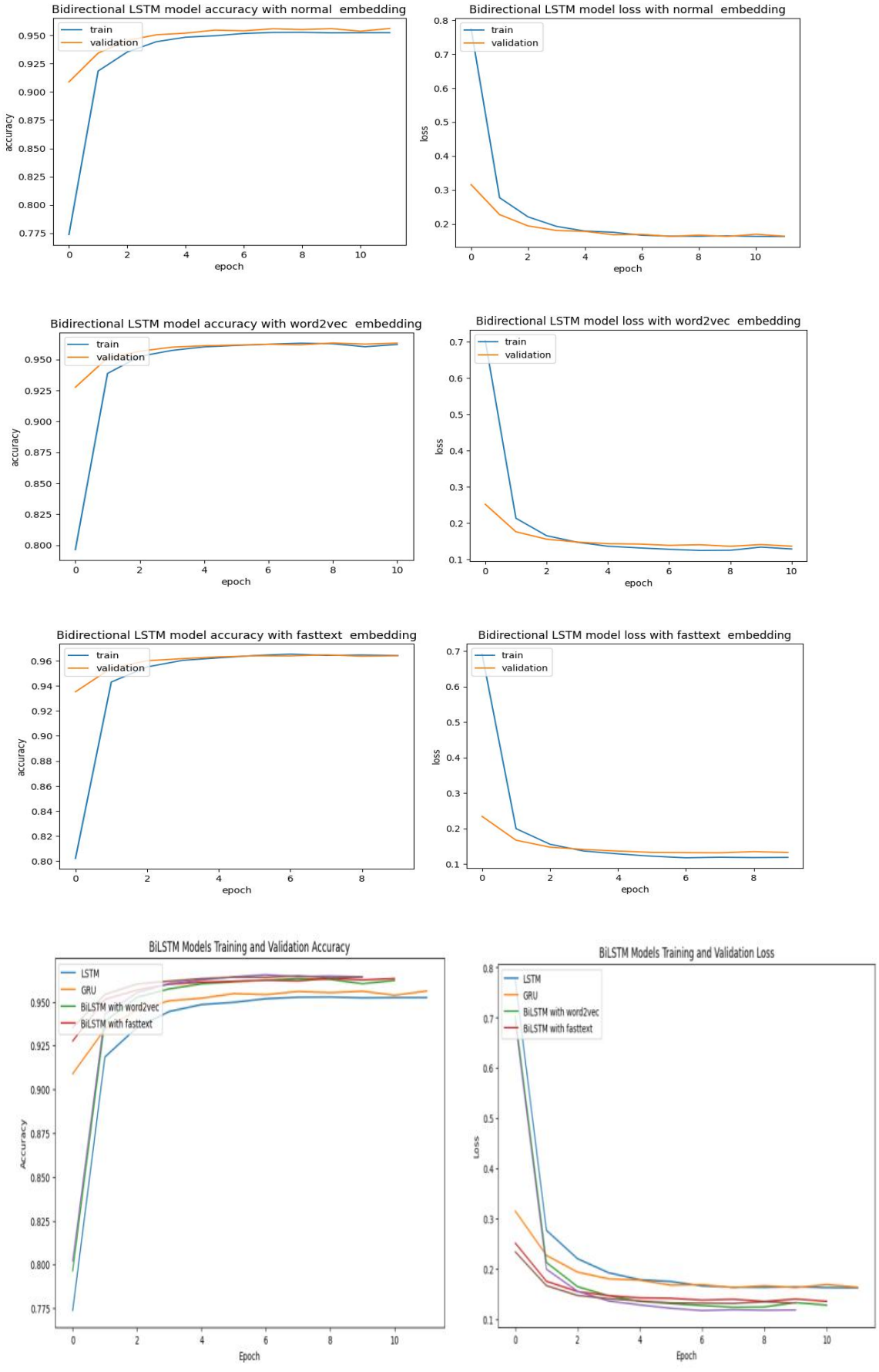


Figure 28: Training, validation accuracy and loss of all BiLSTM models

4.11 Predicted result analysis

This section examines the prediction of Afaan Oromoo words using CNN-LSTM models with different embedding techniques. Three primary embedding methods are evaluated: normal embedding, Word2Vec, and FastText. Normal embedding represents words as dense vectors without capturing linguistic semantics or relationships between words. Word2Vec captures semantic relationships by placing similar words closer in the vector space, outperforming normal embeddings due to its understanding of the context and relationships between words. FastText breaks words into character n-grams and creates embeddings for these sub-word units, which is particularly beneficial for morphologically rich languages like Afaan Oromoo.The combination of normal embedding with LSTM achieved the highest precision, recall, accuracy, and F1 score, suggesting that even simple word representations can be highly effective when combined with the sequential processing power of LSTMs. Word2Vec embedding with LSTM captured semantic relationships and enhanced the model's ability to understand context, although its performance in terms of precision, recall, accuracy, and F1 score was slightly lower than the normal embedding. FastText embedding with LSTM also demonstrated its capability in handling morphologically rich languages and unseen words by providing detailed sub-word information. GRU models with the three embedding methods for predicting word roots from Afaan Oromoo words performed well, demonstrating that GRUs can effectively process simple dense vector representations. Word2Vec GRU Model had an improved capability to capture semantic information, enhancing its prediction performance. FastText GRU Model provided richer word representations, particularly useful in predicting word roots in morphologically rich languages. In conclusion, embedding methods play a crucial role in prediction accuracy across various models, with FastText often providing superior performance in morphologically rich languages due to its ability to capture sub-word information. The following figure illustrates Afaan Oromoo new words with their actual prediction and model prediction.

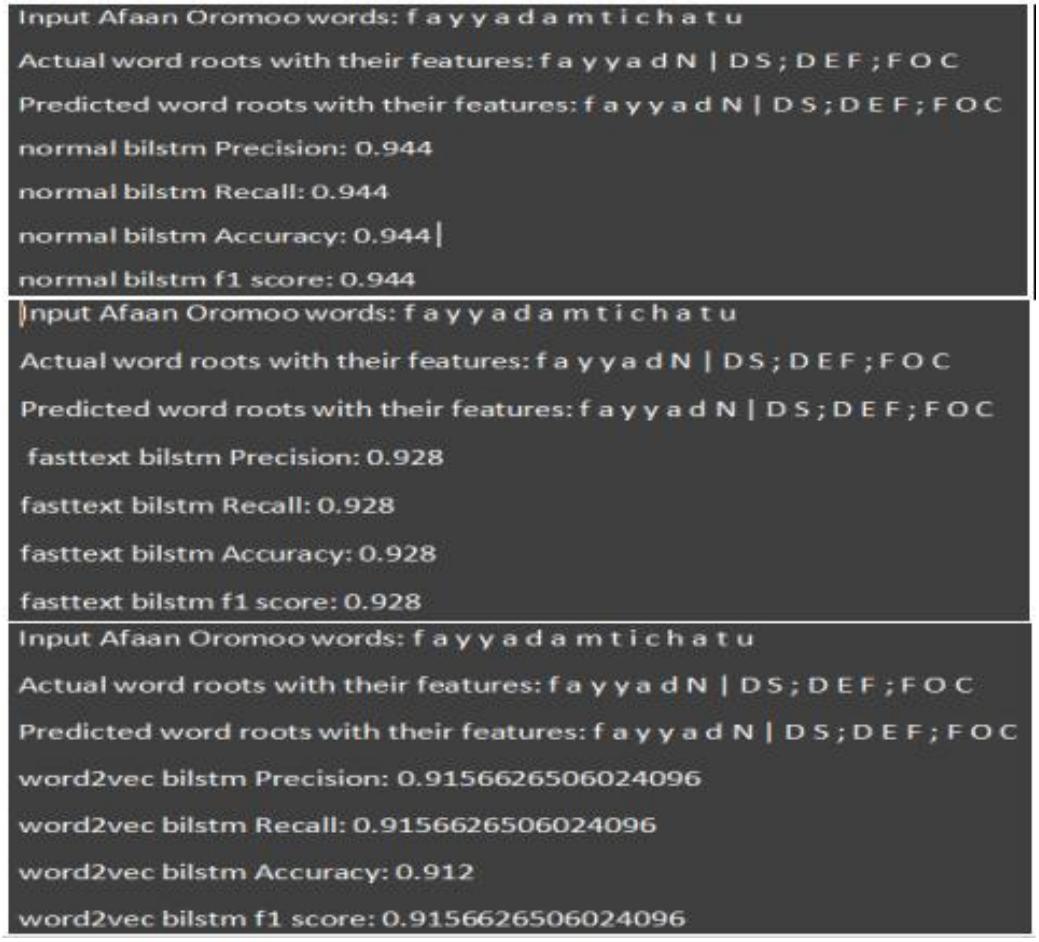


Figure 29: Actual and predicted features with input words by All BiLSTM Models

4.12 Summary of the result Analysis

The result analysis highlights the performance of various models, including CNN-LSTM, LSTM, GRU, and BiLSTM, each utilizing different embedding methods. It identifies strengths and weaknesses across models and embeddings, suggesting avenues for improvement. Notably, while Normal CNN-LSTM shows moderate accuracy with high precision, Word2vec and FastText CNN-LSTM models outperform it, indicating the importance of embedding techniques. Similar trends are observed in LSTM and BiLSTM models, with Word2vec embeddings consistently yielding the highest accuracy and performance metrics. GRU models show moderate accuracy, with Word2vec GRU exhibiting the most balanced performance. Overall, the analysis underscores the significance of embedding methods and model architectures in achieving optimal performance for Afaan Oromoo word prediction tasks.

Table 21: Table Summary of all result analysis of the models

Results	Training			Validation			Testing			
Models	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall	F1-Score
N-CNN	70.94	86.31	61.89	72.74	87.81	64.09	44.44	54.54	54.54	54.54
W2V-CNN	94.74	96.33	93.64	94.56	96.00	93.64	90.74	91.41	91.41	91.41
FST-CNN	95.25	96.46	94.38	94.74	95.92	94.00	87.40	89.39	89.39	89.39
N-LSTM	95.06	96.09	94.32	95.41	96.31	94.77	95.55	95.55	95.55	95.55
W2V-LSTM	93.89	95.72	92.59	93.84	95.62	92.72	90.0	94.18	94.18	94.18
FST-LSTM	90.02	93.26	87.61	90.62	93.60	88.48	85.55	85.55	85.55	85.55
N-GRU	92.96	94.94	91.53	91.59	93.85	89.96	83.6	84.95	84.95	84.95
W2V-GRU	91.98	94.17	90.38	92.73	94.69	91.40	88.8	90.61	90.61	90.61
FST-GRU	91.32	93.47	89.73	90.42	92.67	88.71	80.0	82.37	82.37	82.37
N-BiLSTM	95.24	96.21	94.57	95.61	96.33	95.10	92.76	98.60	98.60	98.60
W2V-BiLSTM	96.21	96.89	95.72	96.32	96.93	95.92	90.46	94.17	94.17	94.17
FST-BiLSTM	96.43	97.07	95.98	96.41	97.00	96.01	83.22	86.05	86.05	86.05

4.13 Comparison with Baseline Experiments

The research introduces significant advancements in Afaan Oromoo morphological analysis compared to existing tools. It focuses on the language and provides a rigorous evaluation framework to assess its performance. The deep learning-based system surpasses Gena's hybrid approach in accuracy, demonstrating competitive or superior performance. Moyka's system achieves high accuracy (98.86%), but further improvement is possible by incorporating a larger and more diverse corpus of Afaan Oromoo words and comparing with other data-driven or hybrid methods. Comparing the results with baseline models across different languages, the deep learning-based approach consistently achieved accuracies exceeding 90% across various models like CNN, LSTM, GRU, and BiLSTM except for normal keras embedding CNN with 70.94 accuracy. This highlights the effectiveness of the approach in addressing morphological analysis challenges in Afaan Oromoo and underscores the significance of the research in providing a state-of-the-art solution for this language.

Table 22: Comparison with baselines

Author	Title	Method	Dataset	Accuracy
M.Gasser [19]	MA for Amharic,Tigrigna, and A/O	Rule Based(FST)	-	-
M.Degefa [7]	MA for A/Oromoo	Machine Learning Memory based(IB1 &IGtree)	2,700	98.86% IB1 94.14% IGTREE
K.Gena [8]	MA for A/Oromoo	Rule based and machine learning techniques	9000	84.6%=Noun 82.9%= Verb
We	MA for A/Oromoo	Deep Learning techniques	45,388	>90%

4.14 Summary

Generally, this chapter briefly discussed the experimentation on how the data are collected, how the collected data are pre-processed, how the pre-processed data are augmented and annotated, how the features are extracted from annotated dataset, training components of deep learning models and the result analysis of each models developed for Afaan Oromoo morphological analysis are also briefly carried out based on the performance metrics like accuracy, precision, recall, and f1- score on train-validation-test splitted dataset. The analyzing process is also incorporated qualitative, quantitative, drop-out, and comparison analysis.

Chapter Five: Discussion, Conclusion, and Recommendations

5.1 Introduction

This chapter consolidates the findings from this research on the morphological analysis of Afaan Oromoo using deep learning approaches, specifically focusing on Keras, Word2Vec, and FastText embeddings. The chapter revisits the research questions, analyzes the results, and discusses their implications. The goal is to provide a comprehensive understanding of how these embeddings and various neural network architectures perform in the context of Afaan Oromoo, a morphologically rich and low-resource language.

5.2 Discussion

The analysis of CNN, LSTM, GRU, and BiLSTM models provides valuable insights into their performance across different embedding techniques and architectures. This section delves deeper into these findings, discussing the implications and addressing the research questions in detail.

Research Questions

How do different deep learning architectures perform morphological analysis of Afaan Oromoo?

How do different word embedding techniques affect the accuracy and efficiency of morphological analyzers for Afaan Oromoo?

What are the most effective strategies for mitigating overfitting in deep learning models for Afaan Oromoo morphological analysis?

How can challenges in data collection and annotation for Afaan Oromoo be addressed to create high-quality datasets?

These questions guided the exploration of the efficacy of various deep learning models and embedding techniques for Afaan Oromoo, a language with a complex morphological structure but limited computational resources and datasets. The study successfully addressed these questions by exploring various deep learning models, including CNN-LSTM, LSTM, GRU, and BiLSTM, combined with different embedding techniques like Normal, Word2Vec, and FastText.

Performance of Deep Learning Models

The study examines the performance of deep learning models in analyzing the complex morphological structure of Afaan Oromoo, a language with rich morphology. The analysis was based on the unique linguistic features of the language and the challenges posed by limited computational resources and datasets. The results showed that combining Convolutional Neural Networks (CNNs), Long Short-Term Memory networks (LSTMs), Gated Recurrent Units (GRUs), and their derivatives can significantly improve the morphological analysis of Afaan Oromoo. CNNs capture local patterns and are computationally efficient, while LSTMs handle sequential data and capture long-term dependencies between words and morphemes. GRUs offer a simpler alternative to LSTMs, making them computationally efficient while still capturing essential sequential dependencies. Transformer architectures can handle long-range dependencies and parallelize computations, making them effective for morphological analysis. Practical applications for Afaan Oromoo include part-of-speech tagging, NER, lemmatization and stemming, and morphological parsing.

Evaluation Metrics

The study utilized performance metrics such as accuracy, precision, recall, and F1-score to assess the models' effectiveness. These metrics provided a comprehensive understanding of how well the models performed in terms of correct predictions, balancing true positives, true negatives, precision, and recall. The F1-score, in particular, was crucial as it balanced precision and recall, offering a holistic view of model performance.

Overfitting Concerns

Overfitting, a common issue in machine learning, was mitigated through techniques like early stopping and monitoring the gap between training and validation metrics. Models with lower validation loss and higher accuracy were better at generalizing, indicating their effectiveness on unseen data. This was particularly evident in models such as LSTM and BiLSTM, where early stopping was instrumental in preventing overfitting.

Impact of Embedding Techniques

Word2Vec and FastText are powerful tools for morphological analysis of Afaan Oromoo, a language with rich morphological structures. Word2Vec uses Continuous Bag of Words (CBOW) and Skip-gram models to understand the relationship between different morphemes and their contexts. The CBOW model helps understand the relationship

between different morphemes and their contexts, while the Skip-gram model predicts context words given a target word. FastText extends Word2Vec by incorporating sub word information, which is particularly useful for languages with rich morphology like Afaan Oromoo. It represents each word as a bag of character n-grams, learning embeddings for these n-grams and combining them to form the final word vector. Fast Text's handling of out-of-vocabulary words ensures more comprehensive coverage of the language's morphology. Practical applications for Afaan Oromoo morphological analysis include part-of-speech tagging, named entity recognition, lemmatization and stemming, morphological parsing, and machine translation.

Word2Vec Embeddings: These embeddings captured semantic relationships effectively, enhancing the model's understanding of context. They performed exceptionally well in LSTM and BiLSTM models.

FastText Embeddings: These embeddings, which break words into character n-grams, provided superior performance, particularly in CNN-LSTM and BiLSTM models. Their ability to handle sub-word information made them highly effective for morphologically rich languages like Afaan Oromoo.

Model-Specific Insights

The morphological analysis of Afaan Oromoo, a language with complex and rich morphology, can be effectively achieved using various models such as Convolutional Neural Networks (CNNs), Long Short-Term Memory Networks (LSTMs), Gated Recurrent Units (GRUs), their derivatives, and combinations. CNNs are effective in capturing local patterns such as prefixes, suffixes, and root words, which are crucial for Afaan Oromoo, where words often undergo various inflectional and derivational changes. They also process text at the character level to detect sub-word features, making them adept at handling morphological variations. LSTMs excel at capturing long-term dependencies in sequential data, which is essential for understanding the morphological structure of words in context. They have input, forget, and output gates that help retain important morphological information over long sequences. They are particularly good at modelling the dependencies between morphemes and word forms, aiding in tasks such as part-of-speech tagging and morphological parsing. GRUs offer a simpler architecture, efficiency, and effective memory, making them a viable option for tasks that require understanding sequences and dependencies in morphological structures. They often perform on par with LSTMs, making them a viable option for tasks that require

understanding sequences and dependencies in morphological structures. Combined and derivative models contribute to enhanced feature extraction, robust analysis, and improved accuracy by integrating CNNs with LSTMs or GRUs. Advanced architectures and mechanisms include attention mechanisms, bidirectional RNNs, transformers, and rich morphological features. Practical applications for Afaan Oromoo include part-of-speech tagging, named entity recognition (NER), lemmatization and stemming, morphological parsing, and machine translation. These models provide a comprehensive tool-kit for capturing the rich morphological features of the language, enabling accurate and efficient processing for various NLP tasks. By leveraging their strengths, robust systems capable of understanding and analyzing the complex morphological structures of Afaan Oromoo can be built. The following are some these models' specific-insights:

CNN-LSTM Models: Showed steady improvement but were prone to overfitting. Word2Vec and FastText embeddings improved performance significantly.

LSTM Models: Demonstrated steady improvement with minimal overfitting. Traditional embeddings performed exceptionally well, with Word2Vec and FastText embeddings also yielding high performance.

GRU Models: Improved significantly in both qualitative and quantitative metrics without overfitting, especially with FastText embeddings. However, there was a slight indication of overfitting, suggesting the need for further refinement and possibly more extensive datasets.

BiLSTM Models: Showed significant improvement, with Word2Vec and FastText embeddings being highly effective. Despite the potential for overfitting, these models demonstrated strong capabilities in capturing sequential dependencies and generalizing to unseen data.

Challenges and Errors

The study's scope was limited, focusing solely on inflected and few derived nouns, adjectives, and verbs in Afaan Oromoo, neglecting other word formation techniques like compounding and reduplication. The absence of publicly available annotated datasets posed a significant challenge, making data collection, preparation, and annotation time-consuming and exhausting. The heavy reliance on deep learning algorithms highlighted the challenge of data availability for training and developing models. The incomplete error analysis lacked specific details or insights, indicating a need for a more thorough examination of model errors to identify linguistic patterns and challenges.

Addressing Research Gaps

This study addressed several gaps in previous research:

Advanced Deep Learning Techniques: By implementing and comparing multiple deep learning architectures, the study provided a comprehensive evaluation that was lacking in prior studies.

Embedding Comparisons: The study systematically compared different embedding techniques, highlighting the importance of sub-word information for morphological analysis in Afaan Oromoo.

Rigorous Evaluation: The use of both quantitative and qualitative evaluations, along with a focus on preventing overfitting, ensured a robust assessment of model performance.

Practical Implications and Future Directions

The findings have significant implications for developing computational tools for Afaan Oromoo and other low-resource languages. The demonstrated effectiveness of deep learning techniques can inform future research and applications in natural language processing.

5.3 Contribution of the Work

The research focuses on the morphological analysis of Afaan Oromoo, a low-resource language. It demonstrates the effectiveness of deep learning techniques in handling the rich morphological structures of Afaan Oromoo. The study also incorporates embedding techniques such as Word2Vec and FastText, which are essential for capturing semantic relationships and sub-word information.

Comprehensive Evaluation: The research conducted a comprehensive evaluation of different deep learning architectures, identifying their strengths and weaknesses in morphological analysis. The findings indicate that LSTM and BiLSTM models generally outperform CNN-LSTM and GRU models, with BiLSTM models excelling in capturing sequential dependencies and generalizing to unseen data.

High Accuracy and Performance: The research advances the field by achieving high accuracy, precision, recall, and F1-scores, surpassing previous methods. A comprehensive

evaluation framework was developed, incorporating quantitative and qualitative metrics to assess model performance.

Annotated Dataset Creation: The study created and annotated a diverse dataset of Afaan Oromoo text, addressing the scarcity of computational resources for this language.

Embedding Techniques: The research underscores the importance of embedding techniques in morphological analysis, with FastText embeddings generally providing superior performance due to their ability to handle sub-word information. Traditional embeddings, when well-optimized, also demonstrated competitive performance. The research recommends the use of FastText embeddings for morphologically rich languages while considering traditional embeddings for specific tasks or architectures based on the findings.

Future Research Guidelines: The study offers guidelines for future research, including dataset expansion, annotation tool development, model optimization, and cross-language validation. By addressing challenges and limitations of previous research, the work contributes significantly to the broader field of natural language processing (NLP), particularly for low-resource languages. Specific contributions to the morphological analysis of Afaan Oromoo include system accuracy, a comprehensive evaluation framework, importance of data diversity, handling linguistic variations, advancing deep learning techniques, sequence-to-sequence modelling, detailed analysis, and accessibility of findings. In conclusion, the research significantly advances the state-of-the-art in morphological analysis of Afaan Oromoo, providing valuable insights, resources, and guidelines for future research in NLP for low-resource languages.

5.4 Future Work

The thesis emphasizes the need for optimizing deep learning models for the morphological analysis of the Afaan Oromoo language, which is hampered by limited labelled data. Key recommendations include dataset expansion, development of annotation tools, model optimization, and cross-language validation. These areas will help identify areas of improvement and advance the field of natural language processing (NLP) for low-resource languages.

Exploring New Architectures: Investigate new deep learning architectures and hyperparameter adjustments.

Advanced Techniques: Apply advanced techniques like attention mechanisms to enhance model performance.

Dataset Enhancement: Enhance the training corpus size and diversity to improve generalization and performance and as this system rely on fiction based books researchers can also check the system by using datasets from different social media.

Cross-Linguistic Transfer Learning: Use cross-linguistic transfer learning techniques to bootstrap model training and improve domain adaptation in real-world scenarios.

Error Analysis and Interpretability: Develop error analysis and Interpretability techniques to identify areas for improvement and understand model decisions.

NLP Pipeline Integration: Integrate NLP pipelines to extend their utility and gather user feedback.

Engage with Specialists and Native Speakers: Engage with specialists and native speakers to provide valuable insights.

Overfitting Concerns: Address overfitting concerns through regularization techniques, hyperparameter tuning, and evaluating additional embedding techniques.

Real-World Evaluation: Conduct real-world evaluations to assess practical applicability and effectiveness.

Documentation and Reproducibility: Ensure comprehensive documentation and reproducibility to facilitate knowledge sharing and advancement in the field. By implementing these recommendations, the effectiveness and practical applicability of sequence-to-sequence models for morphological analysis can be significantly improved, benefiting the field of NLP for low-resource languages like Afaan Oromoo.

5.5 Conclusion

This study aimed to enhance the accuracy and efficiency of morphological analysis for Afaan Oromoo, an under-represented language, by employing deep learning approaches. Through the development and evaluation of sequence-to-sequence models, the effectiveness of different embedding techniques (Normal, Word2Vec, FastText) was compared, and the optimal model architecture for accurate and efficient morphological analysis was identified. Utilizing multiple deep learning models, including CNN-LSTM,

LSTM, GRU, and BiLSTM, combined with Normal, Word2Vec, and FastText embeddings, the performance

Reference

[1] A. Zeynep, “THE ROLE OF MORPHOLOGICAL ANALYSIS IN NATURAL LANGUAGE PROCESSING.”

[2] J. W. Lee, A. Wolters, and Y.-S. Grace Kim, “The Relations of Morphological Awareness with Language and Literacy Skills Vary Depending on Orthographic Depth and Nature of Morphological Awareness,” *Review of Educational Research*, vol. 93, no. 4, pp. 528–558, Aug. 2023, doi: 10.3102/00346543221123816.

[3] G. O. Ganfure, “Comparative Analysis of Deep Learning Based Afaan Oromo Hate Speech Detection,” In Review, preprint, Jan. 2022. doi: 10.21203/rs.3.rs-1289454/v1.

[4] M. Maimaiti, A. Wumaier, K. Abiderexiti, and T. Yibulayin, “Bidirectional Long Short-Term Memory Network with a Conditional Random Field Layer for Uyghur Part-Of-Speech Tagging,” *Information*, vol. 8, no. 4, p. 157, Nov. 2017, doi: 10.3390/info8040157.

[5] by Steven Bird, Ewan Klein, and Edward Loper, *Natural Language Processing with Python*.

[6] “https://en.wikipedia.org/wiki/Oromo_language.”

[7] M. Degefa Mosa, “Morphological Analyzer for Afaan Oromoo Using Machine Learning.”

[8] K. Genna, “AFAAN OROMO MORPHOLOGICAL ANALYSIS: A HYBRID APPROACH.”

[9] W. Gobena, “Inflectional morphology in Mecha Oromo,” *J. Lang. Cult.*, vol. 8, no. 8, pp. 110–140, Aug. 2017, doi: 10.5897/JLC2016.0395.

[10] K. Cao and M. Rei, “A Joint Model for Word Embedding and Word Morphology.” arXiv, Jun. 08, 2016. Accessed: Nov. 05, 2023. [Online]. Available: http://arxiv.org/abs/1606.02601

[11] D. Trajanov *et al.*, “Review of Natural Language Processing in Pharmacology,” *Pharmacol Rev*, vol. 75, no. 4, pp. 714–738, Jul. 2023, doi: 10.1124/pharmrev.122.000715.

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

[13] E. Akyürek, E. Dayanık, and D. Yuret, “Morphological Analysis Using a Sequence Decoder,” *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 567–579, Nov. 2019, doi: 10.1162/tacl_a_00286.

[14] Á. Elekes, A. Englhardt, M. Schäler, and K. Böhm, “Resources to Examine the Quality of Word Embedding Models Trained on n-Gram Data,” in *Proceedings of the 22nd Conference on Computational Natural Language Learning*, Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 423–432. doi: 10.18653/v1/K18-1041.

[15] Y. Bengio, Ed., *Learning deep architectures for AI*. in Foundations and trends in machine learning, no. Vol. 2, No. 1, 2009. Boston, Mass.: Now, 2009.

[16] J. Brownlee, “Evaluate the Performance of Deep Learning Models in Keras.”

[17] G. Franchi, A. Fehri, and A. Yao, “Deep morphological networks,” *Pattern Recognition*, vol. 102, p. 107246, Jun. 2020, doi: 10.1016/j.patcog.2020.107246.

[18] A. W/Mariam, “Rule Based Afaan Oromoo Morphological Synthesis.”

[19] M. Gasser, “HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya,”

[20] A. Barkeessaa, “Fufiilee Heddummina Maqaa Barumsa Afaan Oromoo Keessatti: Maleenyaa Xinqooqa Hujootiin Rakkoo Addaan Baasuufi Furmaata Barbaaduu *”.

[21] A. Abeshu, “Analysis of Rule Based Approach for Afan Oromo Automatic Morphological Synthesizer,” *Sci. Technol. Arts Res. J.*, vol. 2, no. 4, p. 94, Jan. 2014, doi: 10.4314/star.v2i4.16.

[22] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching Word Vectors with Subword Information,” 2016, doi: 10.48550/ARXIV.1607.04606.

[23] G. Jurdzinski, “Word Embeddings for Morphologically Complex Languages,” *SI*, vol. 1/2016, 2017, doi: 10.4467/20838476SI.16.010.6191.

[24] “‘Sanyii Jechaa fi Caasaa Isaa (Word and its Structure).’ Finfinnee, Ethiopia,”.

[25] Mamo, “Sentence prediction system for Afan Oromo text.”

[26] M. Hasnain, S. Ryul Jeong, M. Fermi Pasha, and I. Ghani, “Performance Anomaly Detection in Web Services: an RNN-based Approach Using Dynamic Quality of Service Features,” *Computers, Materials & Continua*, vol. 64, no. 2, pp. 729–752, 2020, doi: 10.32604/cmc.2020.010394.

[27] G. Kessikbayeva and I. Cicekli, “Rule Based Morphological Analyzer of Kazakh Language,” in *Proceedings of the 2014 Joint Meeting of SIGMORPHON and SIGFSM*, Baltimore, Maryland: Association for Computational Linguistics, 2014, pp. 46–54. doi: 10.3115/v1/W14-2806.

[28] M. Abate and Y. Assabie, “Development of Amharic Morphological Analyzer Using Memory-Based Learning,” in *Advances in Natural Language Processing*, A. Przepiórkowski and M. Ogrodniczuk, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 1–13. doi: 10.1007/978-3-319-10888-9_1.

[29] T. Yeshambel , Mothe, Assabie Josiane, yaregal, “Morphologically Annotated Amharic Text Corpora.”

[30] X. TANG, “English Morphological Analysis with Machine-learned Rules.”

[31] D. Tesfaye, “Designing a Rule Based Stemmer for Afaan Oromo Text”.

[32] A. M. Rush, S. Chopra, and J. Weston, “A Neural Attention Model for Abstractive Sentence Summarization,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal: Association for Computational Linguistics, 2015, pp. 379–389. doi: 10.18653/v1/D15-1044.

[33] H. Bai, X. Zhan, H. Yan, L. Wen, Y. Yan, and X. Jia, “Research on Diesel Engine Fault Diagnosis Method Based on Stacked Sparse Autoencoder and Support Vector Machine,” *Electronics*, vol. 11, no. 14, p. 2249, Jul. 2022, doi: 10.3390/electronics11142249.

[34] B. Premjith and K. P. Soman, “Deep Learning Approach for the Morphological Synthesis in Malayalam and Tamil at the Character Level,” *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 20, no. 6, pp. 1–17, Nov. 2021, doi: 10.1145/3457976.

[35] J. P. C. Chiu and E. Nichols, “Named Entity Recognition with Bidirectional LSTM-CNNs,” 2015, doi: 10.48550/ARXIV.1511.08308.

[36] T. Jinbao, K. Weiwei, C. Yidan, T. Qiaoxin, S. Chenyuan, and L. Long, "Text Classification Method Based on BiGRU-Attention and CNN Hybrid Model," in *2021 4th International Conference on Artificial Intelligence and Pattern Recognition*, Xiamen China: ACM, Sep. 2021, pp. 614–622. doi: 10.1145/3488933.3488970.

[37] F. M. Shiri, T. Perumal, N. Mustapha, and R. Mohamed, "A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU," 2023, doi: 10.48550/ARXIV.2305.17473.

[38] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative Study of CNN and RNN for Natural Language Processing." arXiv, Feb. 07, 2017. Accessed: Nov. 09, 2023. [Online]. Available: http://arxiv.org/abs/1702.01923

[39] "Building a Convolutional Neural Network." [Online]. Available: https://www.theclickreader.com/building-a-convolutional-neural-network/

[40] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, Mar. 2020, doi: 10.1016/j.physd.2019.132306.

[41] A. Pektaş and T. Acarman, "Deep learning to detect botnet via network flow summaries," *Neural Comput & Applic*, vol. 31, no. 11, pp. 8021–8033, Nov. 2019, doi: 10.1007/s00521-018-3595-x.

[42] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction." arXiv, Nov. 23, 2019. Accessed: Nov. 05, 2023. [Online]. Available: http://arxiv.org/abs/1801.02143

[43] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017, doi: 10.1109/TNNLS.2016.2582924.

[44] J. Hu, Z. Yang, and H. Su, "Dynamic Prediction of Natural Gas Calorific Value Based on Deep Learning," *Energies*, vol. 16, no. 2, p. 799, Jan. 2023, doi: 10.3390/en16020799.

[45] T. Yemane, "Tigrinya Morphological Segmentation with Bidirectional Long Short-Term Memory Neural Networks and its Effect on English-Tigrinya Machine Translation."

[46] G. Abudouwaili, K. Abiderexiti, Y. Shen, and A. Wumaier, “Research on the Uyghur morphological segmentation model with an attention mechanism,” *Connection Science*, vol. 34, no. 1, pp. 2577–2596, Dec. 2022, doi: 10.1080/09540091.2022.2134843.

[47] Y.-H. Li, L. N. Harfiya, K. Purwandari, and Y.-D. Lin, “Real-Time Cuffless Continuous Blood Pressure Estimation Using Deep Learning Model,” *Sensors*, vol. 20, no. 19, p. 5606, Sep. 2020, doi: 10.3390/s20195606.

[48] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.” arXiv, Dec. 11, 2014. Accessed: Nov. 05, 2023. [Online]. Available: http://arxiv.org/abs/1412.3555

[49] A. Vaswani *et al.*, “Attention Is All You Need.” arXiv, Aug. 01, 2023. doi: 10.48550/arXiv.1706.03762.

[50] D. Bank, N. Koenigstein, and R. Giryes, “Autoencoders,” 2020, doi: 10.48550/ARXIV.2003.05991.

[51] Y. Song, S. Hyun, and Y.-G. Cheong, “Analysis of Autoencoders for Network Intrusion Detection,” *Sensors*, vol. 21, no. 13, p. 4294, Jun. 2021, doi: 10.3390/s21134294.

[52] Y. Song, S. Hyun, and Y.-G. Cheong, “Analysis of Autoencoders for Network Intrusion Detection,” *Sensors*, vol. 21, no. 13, p. 4294, Jun. 2021, doi: 10.3390/s21134294.

[53] J. Abate, V. Khedkar, and S. K. Tidke, “Integrated Model to Develop Grammar Checker for Afaan Oromo using Morphological Analysis: A Rule-based Approach,” *IJACSA*, vol. 12, no. 5, 2021, doi: 10.14569/IJACSA.2021.0120526.

[54] I. Bedane, “The Origin of Afaan Oromo: Mother Language”.

[55] G. Gutema Eggi, “Afaan Oromo Text Retrieval System,”

[56] W. TESEMA, “Investigating Afan Oromo Language Structure and Developing Effective File Editing Tool as Plug-in into Ms Word to Support Text Entry and Input Methods WORKINEH TESEMA”.

[57] W. Tegegne, “The Development of Written Afan Oromo and the Appropriateness of Qubee, Latin Script, for Afan Oromo Writing.”

[58] P. B, S. K.P., and M. A. Kumar, “A deep learning approach for Malayalam morphological analysis at character level,” *Procedia Computer Science*, vol. 132, pp. 47–54, 2018, doi: 10.1016/j.procs.2018.05.058.

[59] J. Izutsu and K. Komiya, “Morphological Analysis of Japanese Hiragana Sentences using the BI-LSTM CRF Model.” arXiv, Jan. 10, 2022. doi: 10.48550/arXiv.2201.03366.

[60] A. Kayabaş, A. E. Topcu, and Ö. Kiliç, “A novel hybrid algorithm for morphological analysis: artificial Neural-Net-XMOR,” *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 30, no. 5, pp. 1726–1740, Jul. 2022, doi: 10.55730/1300-0632.3901.

[61] S. Jha, A. Sudhakar, and A. K. Singh, “Multi Task Deep Morphological Analyzer: Context Aware Joint Morphological Tagging and Lemma Prediction,” 2018, doi: 10.48550/ARXIV.1811.08619.

[62] S. Shekhar, D. K. Sharma, and M. M. Sufyan Beg, “An Effective Bi-LSTM Word Embedding System for Analysis and Identification of Language in Code-Mixed social Media Text in English and Roman Hindi,” *CyS*, vol. 24, no. 4, Dec. 2020, doi: 10.13053/cys-24-4-3151.

[63] J. K. Abhishek Jindal, “FastText.”

[64] X. Rong, “word2vec Parameter Learning Explained.” arXiv, Jun. 05, 2016. Accessed: Nov. 05, 2023. [Online]. Available: http://arxiv.org/abs/1411.2738

[65] M. Tsutsumi, N. Saito, D. Koyabu, and C. Furusawa, “A deep learning approach for morphological feature extraction based on variational auto-encoder: an application to mandible shape,” *npj Syst Biol Appl*, vol. 9, no. 1, p. 30, Jul. 2023, doi: 10.1038/s41540-023-00293-6.

[66] V. P. K., P. B., C. C. V., S. K.P., and P. Poornachandran, “Deep learning based Character-level approach for Morphological Inflection Generation,” in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, Madurai, India: IEEE, May 2019, pp. 1423–1427. doi: 10.1109/ICCS45141.2019.9065571.

[67] *Kitaaba Afaan Oromoo*. [Online]. Available: https://t.me/Freeorobook

Appendix

Table 23: Data annotators with the amount of annotated words

Annotators	Contact	Given words	annotated words	Total annotated words
Barsiisaa Soorii(Msc)	0911048244	9,310	15,436	43549
Ababa Mamo(Msc)	0962486154	9,310	18,333	
Yadeta Yadesa(MSc)	0921410993	9,311	11580	