ST. MARY'S UNIVERSITY

FACULTY OF INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE

Detection and Correction of Amharic Grammar Errors Using Deep Learning

By

Tewodros Fantaye Teka

Addis Ababa, Ethiopia

Jan 2025

Detection and Correction of Amharic Grammar Errors Using Deep Learning

BY TEWODROS FANTAYE

This thesis is submitted to the Department of Computer Science at St. Mary's University within the School of Research and Graduate Studies as part of the requirements to earn a Master's degree in Computer Science from the Faculty of Informatics.

Name: Tewodros Fantaye

Program: MSc. Computer Science

Phone No: +251911525964

Advisor: Alembante Mulu (PhD)

Addis Ababa, Ethiopia

Jan 2025

ACKNOWLEDGEMENT

I would like to begin by expressing my heartfelt appreciation to Almighty God and Saint Mary for their guidance and blessings throughout my life. Additionally, I want to sincerely thank my advisor, Dr. Alembante Mulu, for his invaluable insights, steadfast support, patience, understanding, and continuous encouragement during this research.

ABSTRACT

Effective communication in natural or human language relies heavily on grammatical accuracy. As a result, natural language processing (NLP) has emerged as a critical area of research, aiming to enhance computer's ability to comprehend and interact using human language. A sentence is considered grammatically correct when its word structure adheres to rules governing number, person, gender, tense, and other grammatical agreements. Numerous studies have explored various languages and grammatical frameworks to develop methods for verifying the grammatical accuracy of sentences.

The main aim of this study is to create and execute a system based on deep learning for identifying and rectifying grammatical mistakes in the Amharic language. The suggested method employs a Bidirectional Long Short-Term Memory (BiLSTM) Recurrent Neural Network (RNN), developed using Python 3.7, with Keras and TensorFlow serving as the backend. The evaluation of the BiLSTM model revealed an accuracy of 88.89%, along with a recall of 88.89%, precision of 89%, and an F1 score of 89%.

A significant challenge faced during this research was dealing with the complexity of Amharic words, which can have multiple meanings or reflect different levels of respect, thereby introducing ambiguity into the error detection and correction process. To enhance the effectiveness of detecting and correcting grammatical errors, it is crucial to include a comprehensive dataset of morphologically annotated sentences. Furthermore, future investigations should aim to refine the model and examine alternative methodologies to improve the system's overall performance and accuracy.

Keywords: NLP, Deep learning LSTM, BiLSTM, Amharic language, Amharic language error detection and correction.

TABLE OF CONTENTS

TABLES

FIGURES

ALGORITHIMS

ABBREVIATIONS/ ACRONYMS

BLSTM……. Bidirectional Long Short-Term Memory

BNC……. British National Corpus

AND……. Adjective Noun Disagreement

AVD……. Adverb Verb Disagreement

SVD……. Subject Verb Disagreement

OVD……. Object Verb Disagreement

INWS……...……. Incorrect Word Sequence

RAM……. Radom Access Memory

NLP …… Natural Language Processing

POS……. Part of Speech

SOV……. Subject Object Verb

CNN……. Convolutional Neural Network

RNN……. Recurrent Neural Network

LSTM……Long Short-Term Memory

CHAPTER 1: INTRODUCTION

1.1 Background

In recent years, the rapid evolution of technology has led to a notable increase in the use of electronic documents. Examples of these include word processing files, emails, webpages, and various other digital formats [1]. A language can be described as a structured system of rules or a set of symbols that are combined to convey and communicate information effectively [2]. From varies languages spoken in Ethiopia, Amharic is particularly significant as the official working language in a nation with a population exceeding 120 million people. While Ethiopia boasts a rich diversity of languages, Amharic is the most commonly spoken mother tongue for a large segment of the population and is also the most widely learned second language across the country. [3]

Natural language processing (NLP) is a field within artificial intelligence and computer science that aims to equip computers with the ability to understand, interpret, and interact with human language. A captivating feature of NLP is its potential to improve our comprehension of human language through computational techniques. This domain encompasses a range of theories and methods designed to tackle the challenges associated with facilitating natural communication between humans and machines [4].

In Russia and China Amharic language is started to give as one language subject in their education curriculum. In September 2023 Russia start to give Amharic and Swahili language for the children at list in four Moscow schools. And China was started teaching Amharic language at first degree level and published books for Amharic language teaching purpose. From famous university of Germany, Hamburg university start teaching Amharic language as one language subject years ago.

Amharic language is also not swallowed or covered by colonization language this because Ethiopia was not under colonization. Ancient Ethiopian kingdom write a letter for

European kings using Amharic language. Emperor Haile Selassie was making speech in United nation using Amharic language.

Like Arabic and Latin languages, computer application also developed for Amharic language and used in technology languages. So, we can simply write Amharic language in our mobile and computer.

So, after few years Amharic language will be well know and spoken all over the world and more research must be done on this language.

1.2 Motivation to Research

Ethiopia is a country characterized by linguistic variety, with many languages spoken throughout its different regions. Among these languages, Amharic is the most prevalent and influential. It is the official working language of the Federal Democratic Republic of Ethiopia and is also utilized as a working language in various regional administrations and city governments.

A significant number of documents in Amharic are generated and archived. These documents must be accessible. However, to create and retrieve such materials, there has to be a system for detecting and correcting errors in Amharic. Many people make grammatical mistakes in Amharic since it is not their native language. The lack of tools specifically designed for grammar detection and correction in Amharic often leads individuals to overlook proper grammatical structures while composing text [1].

Amharic is a language with complex morphology, which complicates the manual extraction of its linguistic features. This intricacy calls for the development of a system that can automatically learn these features through advanced deep learning methods. This challenge has inspired me to develop an Amharic grammar checker and corrector.

Amharic language is spread and introduced for the world, for the future it will be the famous language over the world. This is another reason that motivate me to develop Amharic language grammar checker and corrector.

1.3 Statement of problem

Amharic is one of the most complex and under-resourced languages in terms of morphology, which poses significant challenges for the development of efficient natural language processing (NLP) technologies. In Amharic, words—especially verbs—are inflected to indicate various grammatical functions at the same time, complicating grammar verification processes [3].

Frequent grammatical mistakes in Amharic include disagreements between subjects and verbs, adjectives and verbs, adverbs and verbs, as well as improper word arrangement. These difficulties underscore the necessity for sophisticated tools and systems to tackle the distinctive linguistic traits of the language [5].

For example: let us look Amharic sentence having agreement and word sequence problems.

A. Subject-verb disagreement => **ሰዉዬዉ** እኛ መስሪያ ቤት **ናት።።** this can be corrected as **ሰዉዬዉ** እኛ መስሪያ ቤት **ነዉ።።**
B. Object-verb disagreement => አልማዝ **መኪናዋን ሸጠዉ።።** this can be corrected as አልማዝ **መኪናዋን ሸጠችዉ።።**
C. Adjective-noun disagreement => የኢትዮጲያ መንገዶች ባለስልጣን ለግንባታ የሚያገለግሉ **አዳዲስ** የግንባታ **መሳሪያ** በግዢ ከዉጪ አስገባ።። and we can correct this as follows, የኢትዮጲያ መንገዶች ባለስልጣን ለግንባታ የሚያገለግሉ **አዳዲስ** የግንባታ **መሳሪያዎች** በግዢ ከዉጪ አስገባ።።
D. Adverb-verb disagreement => የተማሪዎች ምርቃት **በሚቅጥለዉ** ውር **እንደተካሄድ** የዩኒቨርሲቲዉ ዲን ገለጹ።። so we can correct this, የተማሪዎች ምርቃት **በሚቅጥለዉ** ውር **እንደሚካሄድ** የዩኒቨርሲቲዉ ዲን ገለጹ።።
E. Incorrect word order => ኢትዮጵያ የሰዉ ዘር መግኛ መሆኑዋን ቢቢሲ ባለፍዉ **ዘግቧል ሳምንት** ።። this can be ኢትዮጵያ የሰዉ ዘር መግኛ መሆኑዋን ቢቢሲ ባለፍዉ **ሳምንት ዘግቧል** ።።

According to the above example there are many errors that made in Amharic language, so we need to detect and correct those errors. Research has been conducted on Amharic grammar error detection[1]. However, this work is focus only on Amharic grammar error detection, but after error is detected it has to be corrected. To address the aforementioned

challenges and enhance the performance of previous work, this research will explore the impact of deep learning techniques on the detection and correction of grammatical errors in the Amharic language.

1.4 Question of The Research

This research seeks to address the following key questions:

- Which deep learning algorithms are most effective for detecting and correction grammatical errors in the Amharic language?
- What hyperparameter values yield the best performance for these algorithms?
- Which data set is most suitable for evaluating Amharic grammar accuracy?

1.5 Objectives

1.5.1 General Objective

The main aim of this study is to develop a system that identifies and corrects grammatical mistakes in the Amharic language using deep learning methods.

1.5.2 Specific Objective

In pursuit of the primary aim of this research, the following specific objectives have been established:

- To collect and prepare a suitable dataset from various sources for the model's training and testing.
- To examine and comprehend the grammatical structure and morphological characteristics of the Amharic language.
- To create a model that can autonomously identify and rectify grammatical errors in Amharic sentences.
- To evaluate and analyze the effectiveness of the proposed model.
- To perform a review of existing literature and related research in the area.

1.6 Limitations of the study

This research concentrates on creating a system aimed at identifying and rectifying grammatical errors in the Amharic language, without extending its focus to other languages. The system is designed to pinpoint and amend grammatical mistakes within Amharic sentences, primarily concentrating on agreement issues, such as disagreements between subjects and verbs, objects and verbs, adverbs and verbs, and adjectives and nouns, as well as those arising from incorrect word order. Furthermore, the study will tackle challenges associated with verbs that denote respect and terms that possess multiple meanings.

1.7 Significant of the Study

This research will enhance the development of various applications in natural language processing (NLP), which includes error correction for Amharic grammar, machine translation, answering questions, predicting words, retrieving information, and summarizing texts. Moreover, the system will aid users in correcting mistakes while using word processors. It will also support Amharic speakers, particularly those who are non-native, in producing official documents, emails, letters, and other written materials with improved precision.

1.8 Organization of the Study

The thesis is structured into several sections, with each one offering in-depth explanations and discussions of the related work. It encompasses the design and creation of a deep learning-based system for detecting and correcting grammatical errors in Amharic. The results and discussion section reveal the experimental results and test cases, providing an analysis of their significance. Lastly, the thesis concludes with a summary of the findings, recommendations, and suggestions for future research opportunities.

CHAPTER 2: LITERATURE REVIEWS

2.1 Over View

In this chapter, the basic concepts of an Amharic grammar checker and correction are explored, along with theoretical insights into the grammatical systems of the Amharic language. It begins with an introduction to the Amharic language, covering topics such as Amharic sentence structure, morphology, parts of speech, and common grammatical errors. In chapter there is also approach to grammar checker and corrector, like rule-based approach, statistical based approach, hybrid approach and also, we will see about deep learning. Finally, a detailed discussion of related works will be presented.

2.2 Amharic language

As a morphologically intricate language, Amharic is widely spoken in Ethiopia and is classified under the Semitic branch of the Afro-Asiatic language family. It shares the root-pattern morphological structure common to Semitic languages like Arabic[6].

In Ethiopia (ኢትዮጵያ), Amharic serves as the primary language for government operations. It is the official working language of the federal democratic republic of Ethiopia, as well as several regional states and city administrations. These include the Amhara, Benishangul -Gumuz, Southern Nation Nationalities and Peoples Region (SNNPR), Gambella, Addis Ababa city Administration and Dire Dawa city administration. [7].

2.2.1 Amharic Alphabet

The Amharic language utilizes its own distinct writing system called the Fidel, which originates from the ancient Ge’ez script. The Fidel consists of characters or letters that represent consonant-vowel combinations. The Amharic writing system includes 33 basic consonant shapes, each modified by one of seven vowels, and is written from left to right [1]. The Amharic alphabet is typically organized in a grid format, with consonants arranged vertically and vowels aligned horizontally. In the Amharic Fidel there are also 28 labialized

characters such as ሟ ፣ ሏ ፣ ሷ ፣ ሯ ፣ ቷ ፣ ዷ ፣ ቷ ፣ ኋ ፣ ዃ ፣ ኗ ፣ ኟ ፣ ዟ ፣ ዟ ፣ ጧ ፣ ጯ ፣ ፏ ፣ ፗ ፣ ጇ ፣ ኳ ፣ ዃ ፣ ኗ ፣ ኟ ፣ ቋ ፣ ጓ ፣ ፏ ፣ ሿ ፣ ፯ ፣ ፋ [8].

	ä	u	i	a	e	ï	o
h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
h	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
m	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
s	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
r	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
s	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
sh	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
q	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
b	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
t	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
ch	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
h	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
n	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
ñ	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
a	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
k	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ

	ä	u	i	a	e	ï	o
h	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
w	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
a	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
zh	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
d	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
j	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
g	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
t'	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
ch'	ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
p'	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
s'	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
s'	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
f	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ

Figure 2.1 Ethiopia alphabets.

2.2.2 Amharic Sentences

The structure and word order in Amharic sentences differ from many other languages. An Amharic sentence can be structured as a statement, exclamation, question, or command. The subject, usually a noun phrase referring to a person or thing comes first followed by the predicate which consists of the verb and may include the object. The language follows a subject object verb (SOV) word order, placing the verb at the conclusion of the sentence[1]. This unique arrangement is a distinctive feature of Amharic grammar.

For example: - ሰለሞን ከበደን አሸነፈዉ።=> Solomon win Kebede.

እሷ ሰራተኛ ነች።=> she is a worker.

It has put before the objective of the sentence, when the given sentence adjective appears.

For example: - እሷ ጥሩ ሰራተኛ ነች። => she is a good worker.

The same word order is maintained I questions following the subject object verb structure. This means the subject comes first, followed by the object and the verb is placed at the end even in interrogative sentences.

2.2.3 Morphology of Amharic

Amharic is renowned for its morphological complexity, making it one of the most intricate languages in this regard. Nouns and adjectives in Amharic are inflected to indicate various grammatical features, including number, definiteness, gender, and case. Additionally, they are often combined with prepositions through affixation [6].

For instance, from the noun ልጅ(boy)፣ multiple words can be derived through inflection and affixation, such that:

ልጆች(boys)

ልጁ(the boy, this boy)

ልጄ(my boy)

ልጅሽ(your{feminine}boy)

ለልጅ(for boy)

ከልጅ(from boy) etc.

Similarly, the adjective ፈጣን(fast) can be transformed into various forms through inflection and affixation, such as:

The word for “fast” in Amharic takes different forms depending on its grammatical: ፈጣኑ (fast definite, masculine, singular), ፈጣኖች (fast, plural), and ፈጣኖቹ (fast, define, plural), among others. Amharic verb inflection and derivation are notably more complex than those of nouns and adjectives. A single verbal stem can produce multiple surface forms, and a single verbal root can give rise to several stems [3]. This intricate system demonstrates the richness and depth of Amharic’s morphological structure.

2.2.4 POS in Amharic

A word class for a given word is known as POS (part of speech). In Amharic language word class can be classified noun, Pronoun, verb, adjective, adverb, conjunction, preposition.

Amharic Noun

In Amharic nouns are primarily used to identify or label people, object or place. For example, name of people include አስቴር, ከበደ and ደረጀ, names of things include አልጋ, ጠረቤዛ and ወንበር, and names of places include ጎንደር, አርባምንጭ, and ናዝሬጥ. This demonstrates how nouns serve as essential markers for categorizing and identifying various entities in the language [9].

Table 2.1 Amharic Nouns(Adopted from[9])

	Number		**Gender**	
Word	Singular	plural	Feminine	Masculine
ፍየል	ፍየል	ፍየሎቹ	ፍየሏ	ፍየሉ
ዶሮ	ዶሮ	ዶሮቹ	ዶሪቷ	ዶሮዉ
ልጅ	ልጅ	ልጆቹ	ልጅቷ	ልጁ
መምህር	መምህር	መምህራን	መምህሯ	መምህሩ
ተማሪ	ተማሪ	ተማሪዎች	ተማሪዋ	ተማሪዉ

Amharic pronoun

Pronouns**,** which fall under the category of nouns, serve as substrates for nouns in sentences. They are grouped into personal, demonstrative, and interrogative prunus. In Amharic, personal pronouns are:

- 1st person: እኔ (I), እኛ(we)
- 2nd person: አንተ (you, masculine singular), አንቺ (you, feminine singular), እናንተ (you, plural)
- 3rd person: እሱ(he), እነሱ (they), አንቱ (you, formal) [5].

Table 2.2 Provides a summary of Amharic personal pronouns, as adapted from [5].

Person	**Number**		**Gender**	
	Singular	Plural	Masculine	Feminine
1st	እኔ	እኛ		
2nd	አንተ		M	
	አንቺ			F
		እናንተ		
3rd	እሱ		M	
	እሷ			F
		እነሱ		

Verb in Amharic

Amharic verbs are highly complex, typically composed of a stem along with up to four prefixes and four suffixes. In Amharic, verbs are conjugated to show person, gender, number, and tense, with the base form usually representing the third person masculine singular. A verb is a term that signifies an action, occurrence, or state of being, illustrating either an action in progress or the presence of a specific condition [10].

Amharic Adjective

An adjective provides additional details about a noun or pronoun, describing or qualifying it. Typically, it appears before the word it modifies. Adjectives help distinguish objects based on attributes such as shape, behavior, color, and more. Adjectives in Amharic change form to align with the gender, number, and case of the nouns they describe, adhering to inflection patterns akin to those used for nouns. [11].

Amharic Adverb

An adverb functions similarly to an adjective, but instead of modifying nouns, it modifies verbs. Adverbs can be categorized based on time, place, manner, and other circumstances [5].

For Example:

In the sentence "ልጁ በፍጥነት መጣ", ("the boy came quickly") the word "በፍጥነት", ("quickly") is an adverb that modifies the main verb "መጣ", ("came"). It provides additional information about how the boy came, specifying that is was done quickly.

Amharic Conjunction

In language, conjunctions act as connectors that join words, phrases, clauses, or sentences. These words, though limited in number, play a crucial role in combining verbs, nouns, and adjectives, helping to build cohesive and well-structured sentences.

The following are some Amharic conjunction እና፣ነገር ግን፣ስለሆነም፣ወይም፣ስለዚህ፣ ስለ፣ግን፣ እንደ[5]።

Example: ከበደ **እና** አበበ ነገ ይመጣሉ።

Amharic Preposition

Preposition are words that usually used before nouns and they are limited in number. Preposition are show the relation between noun and another part of a clause.

Some the following preposition are ለ፣እንደ፣ከ፣ወደ[6]።

Example: አስቴር ወደ ትምህርት ቤት ለመሄድ ተዘጋጀች።

2.2.5 Grammar Errors in Amharic

Agreement of Subject and Verb

In the example "አበበ ወደ ትምህርት ቤት ሄደ።" ("Abebe went to school"), the subject "አበበ" ("Abebe") is a third-person singular masculine noun, and the verb "ሄደ" ("went") matches

in gender and number, demonstrating subject-verb agreement. However, in the sentence "አስቴር መጣ።" ("Aster came"), the subject "አስቴር" ("Aster") is third-person singular feminine, while the verb "መጣ" ("came") is third-person singular masculine, leading to a subject-verb disagreement error.

In the sentence "አስቴር መጻፍ ገዛች።" ("Aster bought a book"), the subject and verb agree in number, gender, and person, making it grammatically correct. However, if the subject is changed to "ከበደ" ("Kebede"), the sentence becomes incorrect because the subject "ከበደ" ("Kebede") is masculine, while the verb "ገዛች" ("bought") is feminine, leading to a gender disagreement. Such inconsistencies are commonly referred to as subject-verb disagreement errors.

Noun and Adjective Agreement

Consider the example "ጥቋቁር በሮች መጡ።" ("The black Oxen come"). In this sentence, the adjective "ጥቋቁር" ("black") is plural, and the noun "በሮች" ("Oxen") is also plural, demonstrating agreement between the adjective and the noun.

However, in the sentence "ጥቁር በሮች መጡ።" ("The black Oxen come"), the adjective "ጥቁር" ("black") is singular, while the noun "በሮች" ("Oxen") is plural. This results in an adjective noun disagreement.

The sentence "በኢትዮጵያ አራት አዲስ ከተሞች ተመሰረተ።" ("In Ethiopia, four new cities were established.") is grammatically incorrect because of an adjective-noun disagreement. Here, the adjective "አዲስ" ("new") is singular, but the noun "ከተሞች" ("cities") is plural, resulting in a mismatch in number. When such inconsistencies occur, they are called adjective-noun disagreement errors.

Verb and Adverb Agreement

Another type of disagreement in Amharic grammar is the mismatch between adverbs and verbs. For example, in the sentence "ተማሪዉ ነገ ይመጣል" ("The student will come tomorrow"), the adverb "ነገ" ("tomorrow") and the verb "ይመጣል" ("will come") agree with each other, as both refer to a future action.

However, if we replace the adverb "ነገ" with "ትናንት" ("yesterday"), the sentence becomes "ተማሪው ትናንት ይመጣል" ("the student will come yesterday"). In this case, the adverb "ትናንት" ("yesterday") refers to a past action, while the verb" ይመጣል" ("will come") refers to a future action, creating a disagreement between the adverb and the verb.

Another example is the sentence: "የኢትዮጵያ እና የሱዳን ቤሄራዊ የእግር ኳስ ቡድን በመጪዉ ሳምንት ጨዋታ ተካሂዷል።" ("The national football teams of Ethiopia and Sudan played a match next week."). for this sentence, we can see that it is grammatically incorrect. The adverb "በመጪዉ" ("next") refers to a future time, while the verb "ተካሂዷል" ("played" indicates a past action. This creates disagreement between the adverb and the verb. When a sentence has such an inconsistency, it is referred to as an adverb verb disagreement.

Verb and Object Agreement

Another aspect of error detection in Amharic grammar involves ensuring agreement between the object and the verb in terms of number, person, and gender. For instance, in the sentence "አልማዝ ቤቱን ወሰደችዉ።" ("Almaz took the house"), the object "ቤቱን" ("the house") is third-person singular masculine, and the verb "ወሰደችዉ" ("took") is third-person singular feminine, showing agreement. However, replacing the verb with "ወሰደዉ" ("took") creates a mismatch, leading to an object-verb disagreement.

Sequence of Amharic Word

In Amharic, the standard sentence structure is Subject-Object-Verb (SOV). For example, the sentence "ነብሩን አንበሳዉ በላዉ" ("The lion ate the tiger") is incorrect because it violates the SOV order. The proper sequence should be "አንበሳዉ ነብሩን በላዉ" ("The lion ate the tiger").

Word sequence errors can result from problems such as adjective-noun disagreement, adverb-verb disagreement, subject-verb disagreement, or object-verb disagreement, all of which hinder the sentence's clarity and meaning.

Let us look some example:

1. በኢትዮጵያ በተለያዩ ከተምች የአካል ጉዳተኞች ተመለረተ ማህበረ።
2. የትምህርትቢታችን መምህር እሸቴ አዶ መጡ።
3. አመት ባለፈዉ ብዛት ያላችዉ ኮትሮባንዶች ይያዛሉ።
4. አባቴ ገዛ መኪናዉን።

The example we see above grammatically wrong because all sentence has incorrect word sequence. If the sentence has such kind of problem, we can say that incorrect sequence of word.

2.3 Approach to Grammar Checking and Correcting

Grammar consists of the rules that regulate the formation, structure, and understanding of sounds, words, sentences, and other components of a language. It is crucial for establishing the syntactic organization of well-formed sentences, akin to the guidelines that ensure clear and meaningful communication in natural languages.

Numerous research initiatives have been carried out to ensure the grammatical correctness of texts in various languages, employing diverse methodologies. Among the most widely utilized approaches are rule-based, statistical-based, and hybrid methods [12].

2.3.1 Approach of Rule Based

In the rule-based approach to grammar checking, error-detection rules are manually developed for a particular language. These rules are used to examine and pinpoint errors in texts that have been preprocessed, such as those annotated with part-of-speech tags. Additionally, the rules typically provide recommendations for fixing the identified errors. [13].

Implementing this technique requires the development of a large number of hand-crafted rules. A key limitation of this approach is that it is language specific, meaning the rules created for one language cannot be applied to others. Additionally, creating an exhaustive set of rules can be time consuming. One benefit of the rule-based approach is its adaptability; rules can be modified, new ones can be added, and outdated ones can be

removed as necessary. This approach has been applied in numerous studies for grammar checking in languages like English, Amharic [3], and Afaan Oromo. [13].

2.3.2 Statistical Based Approach

The statistical method for grammar checking relies on training a system using a corpus to determine proper language usage. A part-of-speech (POS) annotated corpus is utilized to produce a list of POS tag sequences, with some sequences appearing more frequently than others [13]. The system derives statistical insights from the corpus and compares the text to this data. Techniques like n-gram modeling are frequently applied to capture these statistical patterns.

One major limitation of this approach is that the system's effectiveness is highly reliant on the size and quality of the corpus. Larger and higher-quality corpora typically improve performance, whereas smaller or lower-quality corpora can lead to reduced accuracy.

The statistical approach often utilizes n-gram models to determine the likelihood of a text based on sequences of previous words or tokens. In this framework, a document is viewed as a sequence of n tokens, with probabilities assigned to these sequences. The n-gram probabilities are extracted from a training corpus, allowing the system to spot grammatically incorrect sentences. After analyzing the entire text, the system detects errors by assessing the probability of the text using n-gram analysis. A probability threshold is established, and sequences below this threshold are flagged as potential errors [14].

2.3.3 Hybrid Approach

By integrating rule-based and statistical techniques, a hybrid system aims to mitigate the weaknesses of each method when used on its own. By combining these two components effectively, the system leverages the strengths of both approaches. Error detection can be performed using statistical methods, rule based methods, or both. The primary role of the hybrid approach is to resolve any discrepancies or conflicts that arise between the outputs of the two methods [15].

2.3.4 Deep learning

Deep learning, a branch of machine learning, centers on algorithms modeled after the brain's architecture and operations, commonly called artificial neural networks.

2.3.4.1. Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a class of deep neural networks primarily designed for computer vision tasks. When fed with images or videos, CNNs enable AI systems to automatically extract features for tasks such as image classification, face recognition, and semantic segmentation.

Unlike fully connected layers, CNNs feature convolutional layers that extract basic features by performing convolution operations. Each layer applies nonlinear functions to weighted sums of localized subsets of outputs from the previous layer, facilitating weight reuse.

Through the use of various convolutional filters, CNNs can capture high-level representations of input data, making them highly effective for computer vision applications. Examples of CNN use cases include image classification and object detection [16].

2.3.4.2. Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) are a class of neural networks tailored for managing sequential data. They were developed to overcome challenges posed by time series data and sequential input formats.

In an RNN, the input for any step incorporates both the current input and information from previous steps. This results in connections between nodes that create a directed graph spanning a temporal sequence. Furthermore, each neuron in an RNN has an internal memory that holds information from past computations, facilitating efficient sequence processing.

A key limitation of traditional neural networks is their inability to capture sequential relationships as they treat inputs and outputs as independent. RNNs overcome this By

incorporating temporal dependencies, RNNs effectively address limitations and are highly suitable for tasks involving sequential data[17].

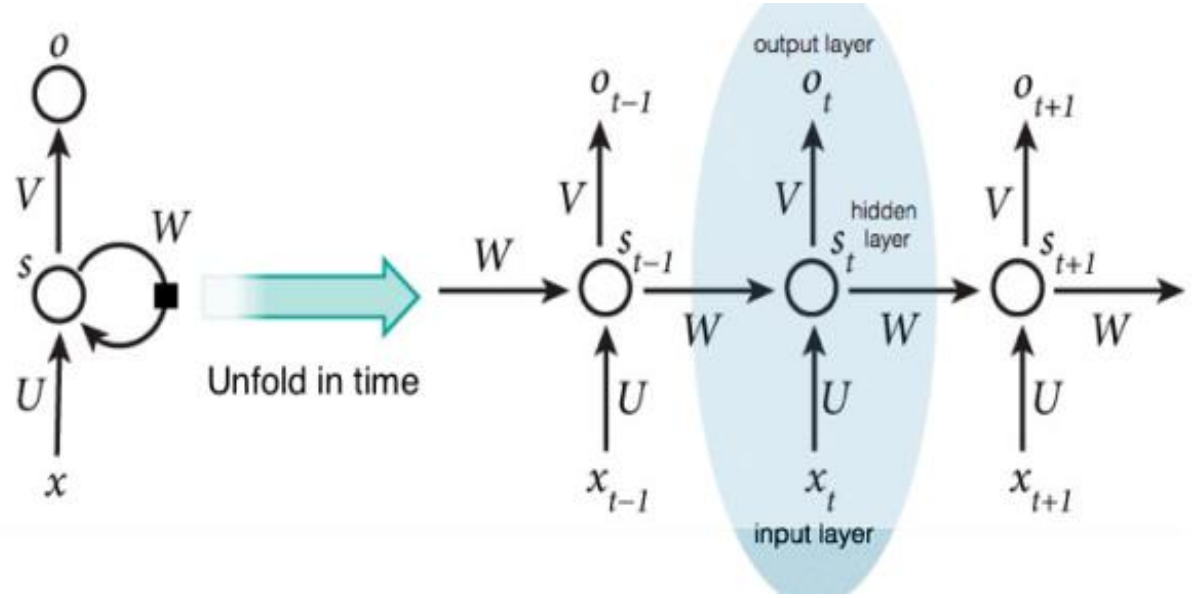

Figure 2.2: Recurrent Neural Network [17]

The diagram demonstrates how an RNN, when unfolded over time, resembles a feedforward neural network. The blue highlighted section represents the process similar to that of a feedforward neural network, consisting of an input layer (xt), a hidden layer (st), and an output layer (ot). The parameters U, V, and W denote the weights that the model learns during training.

The key different between a feedforward neural network and an RNN is the inclusion of an additional input, st-1, which is fed into the hidden layer st. If the blue-highlighted path represents the current time step (t), the previous time step (t-1) and the subsequent time step (t+1) are also shown. At time step t+1, the current hidden layer state (st) is passed along with the input (xt+1) to the next hidden layer (st+1).

Within the hidden layer, an activation function is applied to the sum of the previous hidden layer state (st-1) and the current input (xt) [17].

2.3.4.3. Long Short -Term Memory Network (LSTM)

LSTM (Long Short-Term Memory) network are designed to handle time intervals exceeding 1000 steps, even with noisy or complex input sequences, without losing their ability to process shorter time lags. This is achieved through a specialized architecture that maintains a constant error flow within internal states, facilitated by a unique component called the memory cell.

Three gates the forget gate, the input gate, and the output gate control the memory cell. and output gate. These gates use sigmoid functions to regulate how much information is allowed to pass through or be blocked. Sigmoid functions output values in the range [0,1], where 0 means no information passes through, and 1 means all information is allowed. Each gate has its own set of weights, which are optimized during training using gradient descent [16].

For a detailed explanation of the forward propagation process in LSTMs, refer to the diagram below.

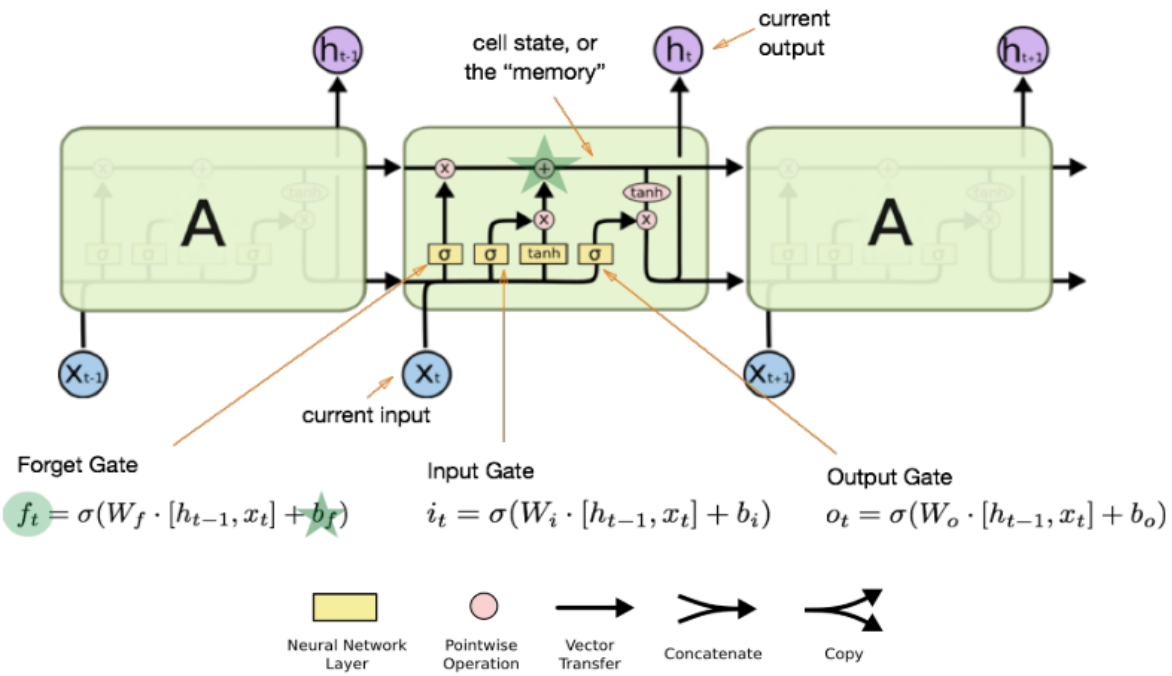

Figure 2.3 The hidden state of a LSTM[17]

2.3.4.4. Bidirectional Long Short-Term Memory Network (BLSTM)

Bidirectional Long Short-Term Memory (BLSTM) networks are based on bidirectional RNNs, which process sequence data in both forward and backward directions using two separate hidden layers. In BLSTMs, these hidden layers are connected to the same output layer. Studies have demonstrated that bidirectional networks perform substantially better than unidirectional networks across various tasks.

BLSTM is a more sophisticated variant of the Long Short-Term Memory (LSTM) network. While LSTMs and standard RNNs rely solely on past context for information, BLSTMs can access both past and future context. This dual-directional capability allows BLSTMs to address limitations inherent in traditional RNNs and LSTMs, making them more effective for tasks requiring comprehensive sequence understanding [18].

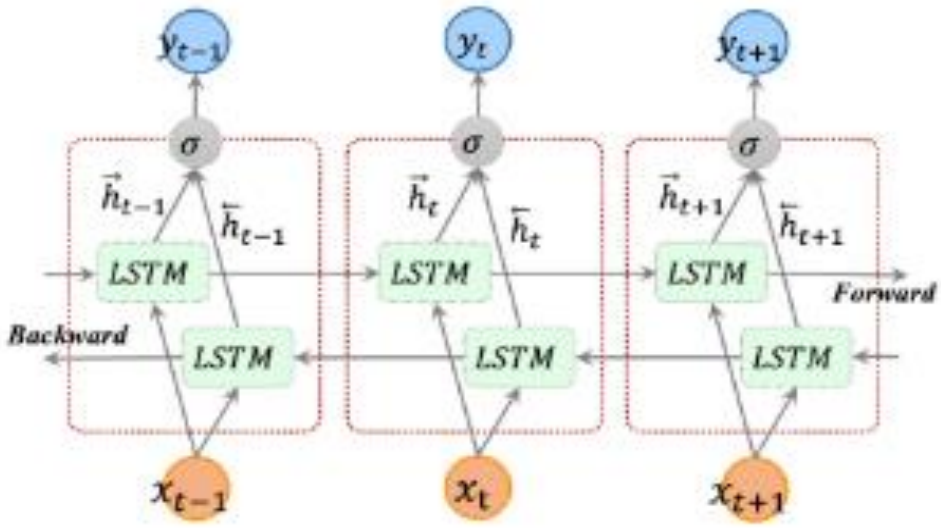

Figure 2.4 The Bi-LSTM RNN Model[18].

The diagram depicts the structure of an unfolded Bidirectional LSTM (BLSTM) layer, made up of two elements: a forward LSTM layer and a backward LSTM layer.

2.4 Related Work

Numerous studies have been conducted on grammar checkers for various languages, including English, Afaan Oromo, Tigrigna, and Amharic, using diverse methodologies. This section presents a review of previous studies and works related to grammar checking.

2.4.1 English Grammar Error

[12] conducted a study on a style and grammar checker for the English language, aiming to create a system that could evaluate both style and grammar in texts. The system analyzes input text, detects potential errors, and starts by dividing the text into individual words and tagging each word with its part of speech to support error identification.

The system relies on predefined grammar and style rules to identify errors. After tagging, the input text is compared against these rules, which define acceptable sequences of words, their parts of speech, and chunks. According to the researchers, measuring the system's precision and recall depends on having a corpus of unedited texts where all errors are labeled. At the time of the study, no such publicly available dataset existed.

To address this, the researchers created two corpora for evaluation. The first corpus contained texts with errors, while the second consisted of texts from the British National Corpus (BNC), which had very few errors. The system was tested on the BNC corpus, identifying 16 errors in 79,900 sentences. It was also evaluated on another corpus containing sentences with errors and compared to Microsoft Word 2000's grammar checker. The proposed system detected 42 errors, compared to Microsoft Word's 49 errors.

2.4.2. Tigrigna grammar Error

[14] conducted a study on Tigrigna grammar error detection, with a focus on identifying grammatically incorrect sentences. The system was designed using a hybrid approach that combined rule-based and statistical methods. A total of 114 rules were implemented to detect errors in noun-modifier agreement, adverb-verb agreement, object-verb agreement, and subject-verb agreement. Furthermore, n-gram probability was utilized to identify errors in word sequences.

The system is composed of five primary modules:

- **Tag Sequence Gathering**: Extracts and calculates the probability of unique tag sequences from a Tigrigna corpus.
- **Preprocessing**: Performs tokenization and tagging on the input text.
- **Rule-Based Grammar Checking**: Uses manually designed rules to identify errors in subject-verb, adverb-verb, object-verb, and noun-modifier agreement.
- **Statistical Grammar Checking**: Detects word sequence errors using a language model.
- **Grammar Error Filtering**: Integrates errors found by the rule-based and statistical approaches, specifying the error type and the words involved.

With an average precision of 87.9%, recall of 87.5%, and an F-measure of 87.6%, the system demonstrated strong performance. However, the researchers pointed out that its effectiveness was hindered by the use of a manually developed dictionary for tagging and the segmentation of words into sentences, which sometimes resulted in incorrect sentences being misclassified as correct.

2.4.3 Grammar checker of Swedish

The grammar error detection system Gransake, developed by [19] for the Swedish language, utilizes a hybrid approach combining rule-based and statistical methods. It was trained on a corpus that had been manually tagged.

The system works through these stages:

- **Tokenization**: Gransake divides the input text into sentences and words.
- **Tagging**: A tagging module assigns part-of-speech tags to each word.
- **Rule-Based Checking**: The tagged output is checked against manually designed rules to evaluate grammatical accuracy.
- **Spelling Checker**: Gransake additionally features a spelling checker and correction capability.

While Gransaka outperforms purely rule based system, it occasionally products false flags when words are incorrectly tagged.

2.4.4 Afaan Oromo Grammar Checker

[13] Conducted research on a grammar checker for the Afaan Oromo language, designed to identify grammatically incorrect texts. The study employed a rule-based approach, utilizing manually crafted rules.

The proposed system consists of five main modules:

- Tokenizer Module**:** This component breaks the input text or paragraph into sentences and then separates them into individual words.
- Pos Tagger Module: This module assigns part of speech tags to each word received from the tokenizer.
- Stemmer Module: This module identifies the root and affixes of each tagged word.
- Grammatical Relation Finder: This module analyzes the grammatical relationships between words.
- Suggestion Creation Module: This module generates suggestions for correcting sentences.

The system was evaluated based on precision and recall, achieving average performance scores of 88%, 89%, and 80%, respectively.

2.4.5 Arabic Grammar Error detection

[20] designed a grammar error detection system for the Arabic language using a deep learning approach, particularly a Recurrent Neural Network (RNN). The study created a web-based tool that is accessible through a browser.

The system operates as follow:

- The user inputs text into the web browser.
- The input is sent to a sever, which hosts the deep learning model to check for grammatical errors.
- The results are displayed back in the web browser.

However, the researchers did not provide specific performance metrics for the system. They noted that the corpus used posed challenges, as it did not include information about error types, limiting the system's ability to classify errors effectively.

2.4.6 Amharic Grammar Error detection

[3] performed the initial research on Amharic grammar error detection, combining rule-based and statistical-based approaches. The rule-based method was employed for simple sentences, where rules are more straightforward to create, while the statistical method was used for both simple and complex sentences, where rule creation is more complex.

The statistical (machine learning) approach utilized n-gram probability to detect errors, whereas the rule-based approach relied on handwritten rules.

The rule-based system included the following modules:

- Sentence Splitter: Breaks the input text into sentences and subsequently into individual words.
- Morphological Analyzer: Assigns linguistic features such as person number, gender, and other to each word.
- Grammar Relation Finder: Identifies grammatical relationships like subject verb and object verb agreements within sentences.
- Grammar Checker: Matches the output of the grammar relation finding with a language model to detect errors.

The system's performance was assessed using precision and recall metrics. The rule-based approach yielded a precision of 92.45% and a recall of 94.23%. For the statistical approach, the bigram model achieved 59.72% precision and 82.69% recall, while the trigram model achieved 67.14% precision and 0.38% recall.

The study also pointed out limitations in the rule-based approach for modeling complex Amharic sentences, as phrase structure grammar views sentences as linear word sequences, making it challenging to manually create comprehensive grammar rules for all sentence variations.

2.4.7 Research Gap

We have explored several studies on grammar checkers that adopt various techniques, including deep learning, hybrid, statistical based, and rule-based methods. From this review, we observed that the rule-based approach has limitation, particularly with compound, compound complex, and complex sentences, as it often produces false flags. The statistical based approach, on the other hand, requires a large annotated corpus and relies on manually extracted features.

Deep learning approaches, however, offer the benefit of automatically learning features, which eliminates the need for manual feature extraction. Hybrid approaches, which combine rule based and statistical methods, still face challenges in detecting errors effectively, especially as sentence length increases or for complex sentence structures.

Deep learning, particularly bidirectional recurrent neural network (RNNs), can address these issues by learning word sequences in both forward and backward directions. This capability allows the model to detect and correct errors in long or complex sentences, such as compound and compound complex sentences, making it a more robust solution for grammar error detection and correction.

CHAPTER 3: METHOD AND APPROACHES

3.1 Overview

In this chapter, the system architecture and components of the proposed Amharic grammar error detection and correction model are described in detail. Furthermore, the design of a deep learning-based grammar checker model, designed to identify grammatical errors in Amharic sentences, is presented. The model leverages a Bidirectional Long Short-Term Memory (BiLSTM) recurrent neural network for this purpose.

3.2 The Proposed System Architecture

The system architecture of the proposed deep learning-based grammar checker and corrector is shown in Figure 3.1. It includes several subcomponents: preprocessing, tokenization, morphological feature extraction, word embedding, a Bidirectional Long Short-Term Memory (BiLSTM) recurrent neural network, grammar result generation, and grammar correction. Each module's functionality is described in detail below.

1. Preprocessing Module:
    - Tokenization: Split the input text into sentences and further into individual words or tokens.
    - Morphology Based Tagging: Analyzes words morphologically and assigns appropriate tags.
    - Ta Splitting: Separates Amharic words from their morphological tags or annotations.
    - Sequence Padding: Ensures all sentences are of equal length by adding padding where necessary
2. Embedding Module:
    - Converts the padded sequences into dense vector representations, enabling the model to process the text numerically.
3. Bidirectional Long Short-Term Memory (BiLSTM) Module
    - By taking dense vector representations as input, the model learns word sequences in both forward and backward directions. This bidirectional

learning capability allows the model to effectively identify and correct grammatical errors.

The final goal of this research is achieved through the BiLSTM module, which performs Amharic grammar error detection and correction by analyzing the sequence of words in context.

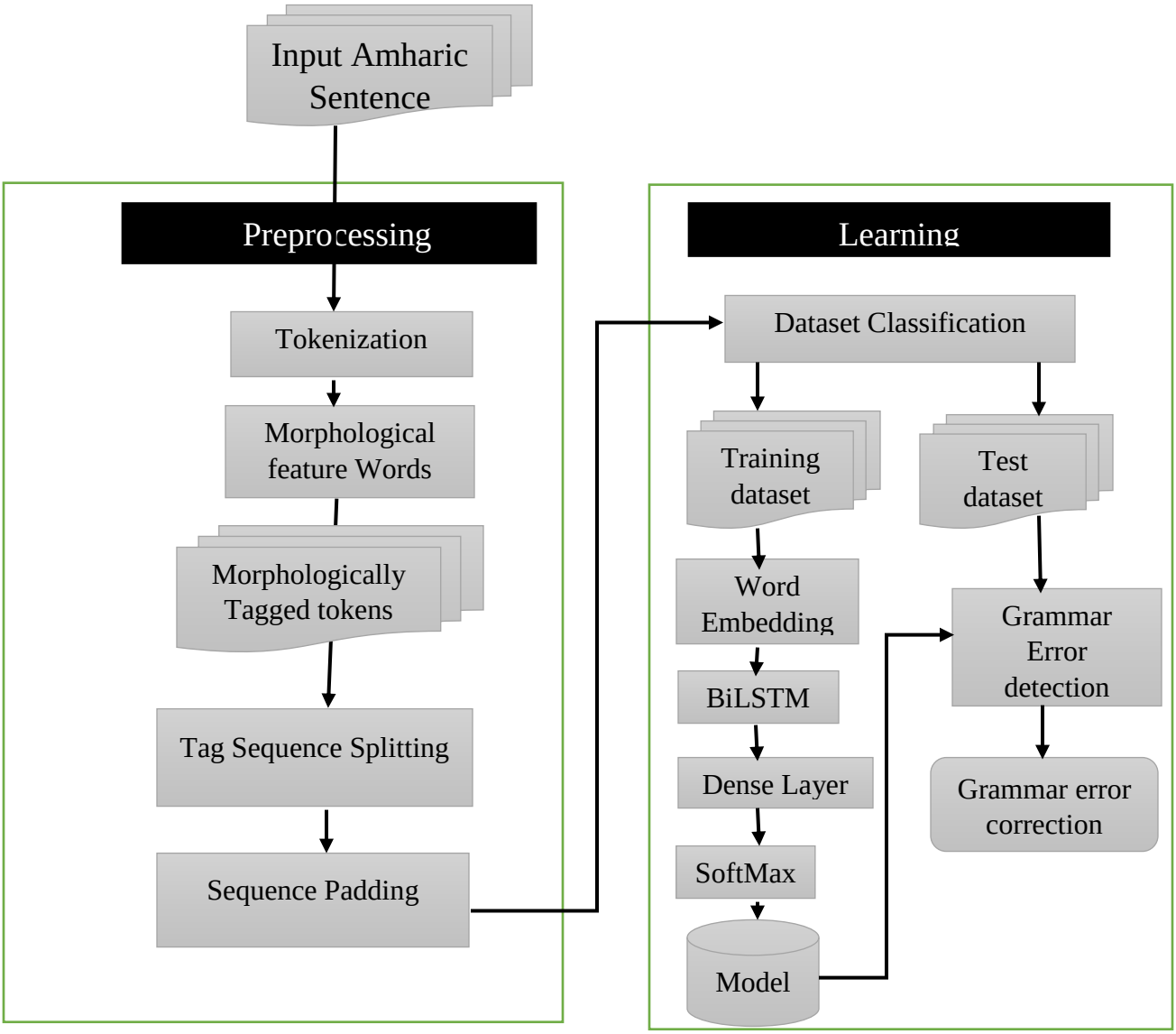

Figure 3.1 The Architecture for deep learning- based Amharic Grammar error detection and correction.

3.3 Preprocessing

The preprocessing module is the starting component of the proposed system, comprising four essential elements: tokenization, morphology-based part-of-speech tagging, tag splitting, and sequence padding.

3.3.1 Tokenization's

Tokenization is an essential step in natural language processing (NLP), involving the splitting of a sentence into individual tokens or words. For this study, we utilized the Keras tokenizer to split strings into sequences. Keras also provides a text to sequence function to convert words into numerical sequences.

The primary role of the tokenization component is to clean and split the input text. Cleaning involves removing unwanted characters, such as non-Amharic text or symbols irrelevant to the study. The input text is first split into sentences using punctuation marks like double colons (።), exclamation mark (!), and question mark (?). Each sentence is then further divided into individual words or tokens.

For example, consider the sentence: "አብርሃም ወደ አሜሪካ ሄደ።" the tokenization component split this into ["አብርሃም", "ወደ", "አሜሪካ", "ሄደ"]. If the sentence contains unwanted symbols, such as "አብርሃም ወደ @ አሜሪካ ሄደ።", the preprocessing module uses the Keras tokenizer to remove or handle such symbols.

This pseudocode for the tokenization process is as follows

Input:

- InputText: The text to be processed.

Process:

1. **Split Text into Sentences**:
    - Divide the InputText into individual sentences (SingleSentence) based on punctuation marks (e.g., ?).
2. **Tokenize Each Sentence**:
    - For each SingleSentence:
        - Split the sentence into individual words (Wi) using whitespace as the delimiter.

- Add each word (Wi) to the TokenizedSentence list.

Output:

- TokenizedSentence: A list of words extracted from the input text, organized by sentences.

Algorithm 3.1 Tokenization module

3.3.2 Morphology based post tagging

The next component in the proposed Amharic grammar error detection and correction system is morphology-based part speech tagging. After the input text is tokenized, this component processes the tokens by assigning corresponding morphological tags from a pre-annotated database. To determine whether a sentence is grammatically correct, each word must first be morphologically tagged. Morphological features such as number, gender, person, tense, and part of speech are essential for this process.

A morphologically tagged corpus is prepared to facilitate this tagging. The tagging process is performed word by word, and once all token in a sentence are tagged, they are combined to form a tagged sentence. Below is the algorithm outlining the steps for tagging morphological information.

Input:

- TokenizedSentence: A sentence broken into individual tokens (words).
- MorphologyBasedTaggedCorpus: A corpus containing words with their corresponding morphological tags.

Process:
For each word (Wi) in the TokenizedSentence:

1. Check if Wi exists in the MorphologyBasedTaggedCorpus.
    - If **found**, add Wi along with its corresponding tag (tagOf(Wi)) to the TaggedSentence.
    - If **not found**, add Wi to the TaggedSentence with its tag set to Null.

Output:

- TaggedSentence: A sentence where each word is paired with its morphological tag (or Null if the tag is unavailable).

Algorithm 3. 2 Morphology-based post tagging

Table 3.1 Morphology-based tag description

Tag Name	**Description**
N_p	Noun plural in number
N_s	Noun singular in number
N_sm3	Noun singular in number, masculine in gender
N_sf	Noun singular in number, feminine in gender
N_sml	Noun singular in number, masculine in gender, first-person.
N_sf2	Noun singular in number, masculine in gender and second person
V_GER_Ssm3	Gerundive verb, singular in number and masculine in gender
V_IMPF_Ssf3	Imperfective verb, singular in number, feminine in gender and third person
V_PERF_Ssm2	Perfective verb, singular in number, masculine in gender and second person
V_JUSS_p3	Jussive verb, plural in number, third-person
V_IMPF_Ssm3>_Osm3	Imperfective verb with the subject (singular, masculine and third person) and object (singular masculine third person)
V_PERF_Sp3_Osm3	Perfective verb with the subject (plural third person) and object (singular masculine third person)
V_GER_Ssm3	Gerundive verb with the subject (singular masculine third person)
VREL_PERF_Ssm3_Osm3	Perfective verb with a subject singular masculine third person and object singular masculine third person
ADJ_s	Adjective singular in number
ADJ_p	Adjective plural in number
ADJ_m	Adjective masculine in gender
ADJ_f	Adjective feminine in gender
ADJ sm3	Adjective singular in number masculine in gender third person
ADJ sf3	Adjective singular in number feminine in gender third person
ADV	Adverb
ADV PERF	Adverb perfective in tense
ADV IMPRF	Adverb in imperfective in tense
N Ssm3	Noun with subject singular masculine in number third person
PUNC	Punctuation marks excluding end punctuation
ENDPUNC	End Punctuation tells the end of the sentence

Example:

Consider the sentence: “እነሱ መጡ።”

In order to tag the above sentence, the tagger searches for the words እነሱ and መጡ in the morphologically tagged corpus. After tagging, the sentence is represented as:

“እነሱ PROP_p3 መጡ V_PERF_Sp3 ። PUNC”

- PROP_p3: indicates that እነሱ is third person plural pronoun.
- V_PERF_Sp3: indicates that መጡ is a perfective verb in the plural form, third person.
- PUNC: Represent the punctuation mark (።) at the end of the sentence.

This tagging process helps identify the grammatical structure and feature of each word in the sentence.

Example 2, this is more complex: - ሙሴ ሕዝቡን ሰበሰበና እናንተ በአምላክ የማትታመኑ ሰዎች ስሙ! አሮንና እኔ ከዚህ አለት ውሃ እንድናወጣላችሁ ትፈልጋላችሁ ? አላቸዉ ።

ሙሴ <N_sm3> ሕዝቡን<N_p> ሰበሰበና<V_IMPFA_p2> እናንተ<N_p2> በአምላክ<NP_sm3> የማትታመኑ<VP_IMPF_Sp2> ሰዎች<V_p> ስሙ<V_IMPF_Sp2> !<PUNC> አሮንና<NC_sm3> እኔ<PRON_s1> ከዚህ<PRONP> አለት<N> ውሃ<N> እንድናወጣላችሁ<VP_SP1_Op2> ትፈልጋላችሁ<V_IMPF_Sp2>? <PUNC> አላቸዉ<V_PERF_Ssm3_Op3> ።<ENDPUNC>

3.3.3 Tag Splitting

Following the morphological tagging of the input text, the next step in the proposed system is tag splitting. This component is responsible for separating the Amharic text from its associated morphological features. By isolating the linguistic elements from their tags, the system can more effectively analyze and process the text for further tasks, such as grammar error detection and correction. This step is crucial for handling the morphological complexity of Amharic and ensuring accurate down streaming processing.

For example, from the tagged sentence:

“እነሱ PROP_p3 መጡ V_PERF_Sp3 ። PUNC”,

The tag splitting component filters out the morphological features, leaving only the original Amharic text: “እነሱ መጡ።”.

This step ensure that the text is prepared for further processing while retaining its grammatical structure.

1. ሙሴ <N_sm3> ሕዝቡን<N_p> ሰበሰበና<V_IMPFA_p2> እናንተ<N_p2> በአምላክ<NP_sm3> የማትታመኑ<VP_IMPF_Sp2> ሰዎች<V_p> ስሙ<V_IMPF_Sp2> !<PUNC> አሮንና<NC_sm3> እኔ<PRON_s1> ከዚህ<PRONP> አለት<N> ዉሃ<N> እንድናወጣላችሁ<VP_SP1_Op2> ትፈልጋላችሁ<V_IMPF_Sp2>? <PUNC> አላቸዉ<V_PERF_Ssm3_Op3> ።<ENDPUNC>
2. ትልቅ <ADJ> ልጆች<N_p> ሄዱ<V_PERF_Sp3>።<ENDPUNC>

The above sentence after splitting component the output of morphologically sentence will be as follows.

1. N_sm3 N_p V_IMPFA_p2 N_p2 NP_sm3 VP_IMPF_Sp2 N_p V_IMPF_Sp2 PUNC NC_sm3 PRON_s1 PRON_s1 PRONP N N VP_Sp1_Op2 V_IMPF_Sp2 PUNC V_PERF_Ssm3_Op3 ENDPUNC.
2. ADJ_p N_p V_PERF_Sp3 ENDPUNC

After splitting the morphological feature from the words, the next step is sequence padding. The sentences in our dataset vary in length, so to ensure uniformity for processing, each sequence must be of equal length. In this study, we set the maximum sequence length to 100. If a sentence is shorter than this maximum length, it is padded with “0’ until it reaches the required length.

The Keras library provides a pad_sequences function to handle this task. Padding can be applied either at the beginning (pre-padding) or at the end (post-padding) of the sequence, depending on the requirements of the model.

E.g.1. PRON N NP ADV_IMPF NP CONJ NP N N N V_PERF_Ssm3 N V_PERF_Ssm3 NC_sm3 VP_PERF_Sp3_Osm3 NP_sf3 N_P N NP N VN_sm3 V_GER_Sp3 ENDPUNC,

E.g.2. ADJ_p N_p V_PERF_Sp3 ENDPUNC

After padding is done the sequence with maximum length 100 will be as follows: -

E.g.1. [46, 1, 2, 18, 2, 41, 2, 1, 1, 1, 7, 1, 7, 85, 51, 4, 1, 2, 1, 14, 12, 3]

E.g.2. [33, 4, 13, 3]

Maximum pad length = 100

E.g.1 [0 0 0 …. 14 12 3]

E.g.2 [0 0 0 …4,13,3]

Read TaggedSentence, TaggedCorpus

Set TaggSplit=" "

FOR each TaggedSentence in TaggedCorpus

Split TaggedSentence by ENDPUNC

End For

FOR each Tagglist in TaggedSentence

Split Tagglist from Tokens

ADD tagglist in TaggSplitCorpus

End for

Algorithm 3.3. Tag splitting Module

3.4 Word Embedding

Word embedding can be defined as representation of words or documents in dense vector representation. To represent words in dense representation or in word vectors we can use word embedding layer. The function of the word embedding component in this proposed

system is to represent the output of the sequence padding component into real dense representation and reducing the size input to low dimensional space.

We have to change those morphological features into dense vector representation after we padding the sequence of tags, this is because bidirectional long short-term memory network accepts input in vector representation only. In Kera's library, the embedding layer has 3 arguments, such as input dimension, output dimension and input length. We chose 100 input length because the maximum number of sequences in our dataset is near to hundred but not greater than this. The number of vocabulary size is total number of unique tags sequences. We have used 32 output dimension and 100 input length. So, after representing sequence intro dense vector representation, the output has input to BiLSTM module. So, embedding (vocabulary size, 32, 100) is input to the Bidirectional recurrent neural network.

3.5 Bidirectional LSTM

BiLSTM is the modified version of long short-term memory recurrent neural network. BiLSTM recurrent neural network checks the sequence of vectors in forward direction and backward direction. This type of algorithm is better for sequence checking because the error may be at the beginning or at the end of the sentence. And BiLSTM solves the problem of vanishing gradient problem that arises from recurrent neural network.

The proposed model has many layers such as input layer, embedding layer, BILSTM layer, dense layer, dropout, SoftMax. BILSTM are followed after representing the sequence of tags in vector representation. After changing the tag sequence into a sequence of real value vectors, the next component of the proposed system is BiLSTM model. BiLSTM accepts input from word embedding component, that means the output of word embedding component is the input to long short term recurrent neural network. As shown in figure 3.1 the output of the embedding layer is given to BiLSTM layer, which means (None, 100, and 100) is the output of the word embedding layer, which contains features vectors. In our model the shape of embedding layer is (none, 32, 100). So, the BILSTM layer changes this shape in to (None, 32), so we have used 32 neurons. The BILSM network learns the sequence feature vector in forward and backward direction this helps to predict correctly. Therefore, in my proposed model has both LSTM and BILSTM. During training additional

layer are important such as dropout layer in order to minimize overfitting and underfitting of a model. So, in this proposed model 0.5 dropout layer are used because during training, the gap between training accuracy and validation accuracy is high, and also the gap between training loss and validation loss is high, so in order to minimize the problem we add dropout layer to my model. The dense layer changes the input data to output data by adding bias and some activation function. So, the output of BILSTM layer is given to dense layer. The dense layer accepts input from BiLSTM layer with a dimension of 32 and produce six dimensions. Because I have six classes such as adjective-noun disagreement, adverb-verb disagreement, correct class, incorrect word sequence, subject-verb disagreement and object-verb disagreement. Therefore, the function of dense layer is minimizing the dimensionality of bidirectional long short-term memory recurrent neural network into exact number of classes. Finally, SoftMax classifier assigns to each number of class based on probability distribution. So, this classifier displays six different decimal points since I have six labels in our dataset, so, after SoftMax classifier the type of grammar error disagreement have displayed.

Suppose T1, T2, T3, …, TN, where N is number of morphologically tag sequences in the sentence. After changing the TN into integers values using dictionary mapping of tags into D1, D2, D3, …., DN where D is dictionary mapping of tags. Then the corresponding sequences of integer values have padded.

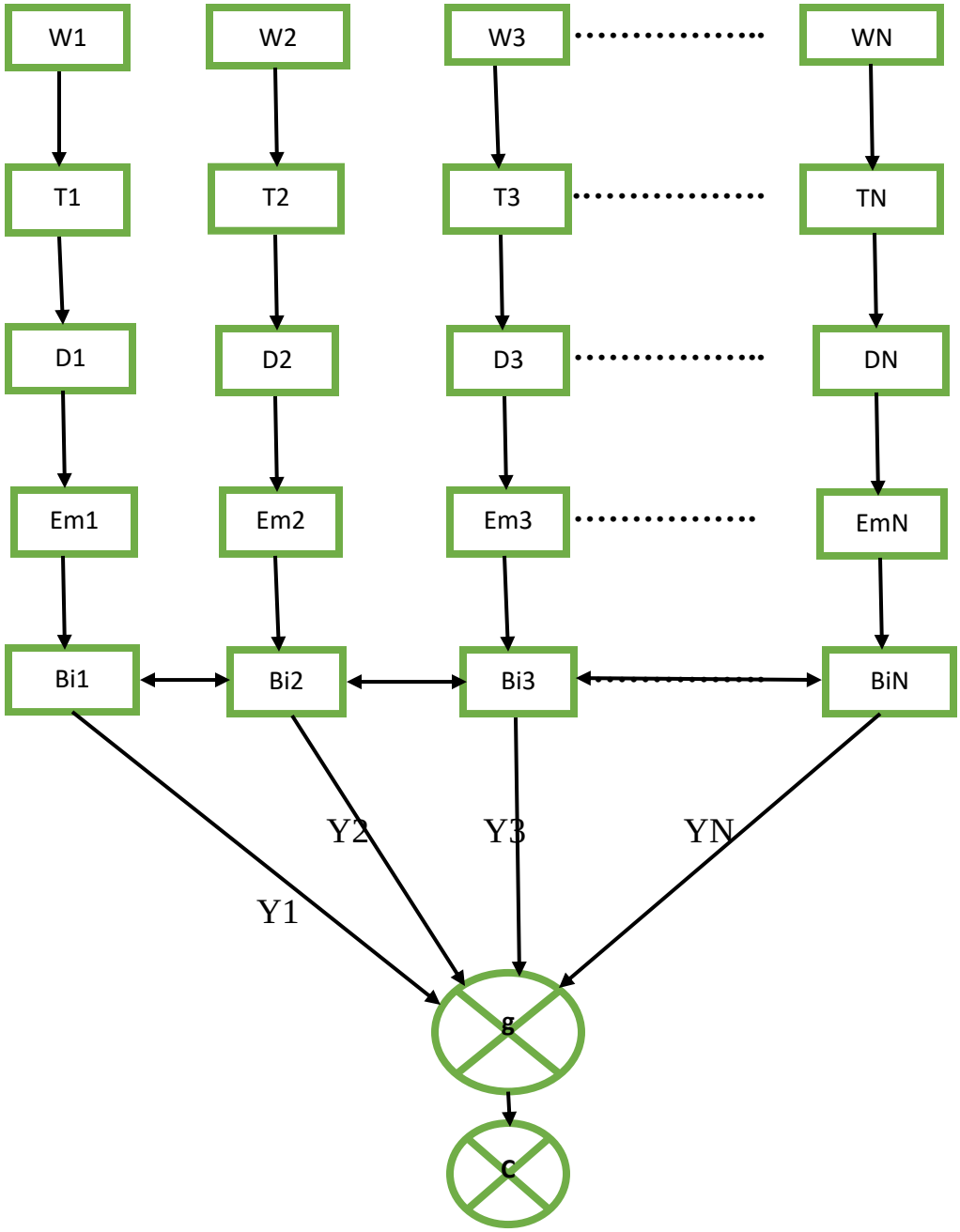

Figure 3.2 sample flow diagram for proposed system

The above figure worked as follows

T (W1, W2, W3, …., WN) =T1, T2, T3, …., TN.

D (T1, T2, T3, …., TN) =D1, D2, D3, …., DN.

E (D1, D2, D3, …., DN) = Em1, Em2, Em3, …., EmN.

B (Em1, Em2, Em3, …., EmN) =Bi1, Bi1, Bi3, …., BiN.

L (Bi1, Bi2, Bi3, …., BiN) = y1, y2, y3, …., yn

∑Y1+Y2+Y3, …. +YN=g

If g not correct ⟶ C

Where T1, T2, T3, …, TN refers to sequence of morphologically tagged tokens.

D1, D2, D3, …, Dn refers to dictionary mapping of sequences of integers.

Em1, Em2, Em3, …, EmN refers to embedding of sequences of integers.

Bi1, Bi2, Bi3, …, BiN refers to Bidirectional recurrent neural network.

Y1, Y2, Y3, …, YN refers to the output of each bidirectional recurrent neural network.

g represents the result of grammar checker and **c** refers the grammar corrector.

3.6 Demonstration

The overall model of my proposal Amharic grammar error detection and correction is worked as follows. Let us see the following sentence as an example.

1. አበበ a ትናንት ያገባል ።
2. ጫላ ነገ ያገባል ።
3. ያሬድ ምሳ በላ ።
4. ከበደ በሶ በላች ።
5. ኤፍሬም ጥቋቁር ላም አሉት ።
6. መላኩ ነጫጭ =በግ አሉት ።

As shown in the figure 3.1, the first component of the proposed model is tokenization. Under the tokenization component, the function of this component is cleaning non-Amharic sentences and tokenizing the sentence into words. For example, in the first sentence, there is an English letter ‘a’, therefore we have removed from Amharic sentence

under this component. After tokenizing the sentence into tokens, the next component is finding the morphological feature of the tokens. So, the morphological information of the above six sentences are as follows: -

1. አበበ N_Ssm3 ትላንት ADV_PERF ያገባል V_IMPRF_Ssm3 ። EDNPUNC
2. ጫላ N_Ssm3 ነገ ADV_IMPREF ያገባል V_IMPREF_Ssm3 ። EDNPUNC
3. ያሬድ N_Ssm3 ምሳ N በላ V_PERF_Ssm3 ። EDNPUNC
4. ከበደ N_Ssm3 በሶ N በላች V_PERF_Ssf3 ። EDNPUNC
5. ኤፍሬም N_Ssm3 ጥቋቁር ADJ_P ላም N አሉት V_PERF_Sp3። EDNPUNC
6. መላኩ N_Ssm3 ነጫጭ ADJ_P በግ N አሉት V_PERF_Sp3 ። EDNPUNC

After finding the morphological information each word in sentence, the next component is tag splitting, that is splitting the tag sequence from a morphologically tagged sentence.

Removing Amharic sentence from tag sequence is important to reduce or minimize the computational time of the system and also the model have learned training data easily, because many different Amharic words may have the same morphological information. Example, when we see the first word of all six sentences have a different name but all of the words have the same morphological information as shown below.

1. N_Ssm3 ADV_PERF V_IMPRF_Ssm3 EDNPUNC
2. N_Ssm3 ADV_IMPRF V_IMPRF_Ssm3 EDNPUNC
3. N_Ssm3 V_PERF_Ssm3 EDNPUNC
4. N_Ssm3 N V_PERF_Ssf3 EDNPUNC
5. N_Ssm3 ADJ_p N V_PERF_Sp3 EDNPUNC
6. N_Ssm3 ADJ_p N V_PERF_Sp3 EDNPUNC

So, the next component of the proposed Amharic grammar error detection and correction is sequence padding. From the example, we can see that all six sentences have different length. However, in deep learning the length of the data should be the same size and we must feed a matrix of normalized values. That means all the tokens in the sentence should have the same length. So, the result of the sentence after padding is as shown below.

```
[[ 5  6  3  2  0]
 [ 7  8  3  2  0]
 [ 9 10 11  2  0]
 [12 13 14  2  0]
 [15 16 17  4  2]
 [18 19 20  4  2]]
```

Figure 3.3 Padded Sequence

The result shows that the first four sentences have the same length and last two sentences have the same length. So, it needs to pad the first to sentence by replacing “0” to each sentence, because the largest sentence in this example has five tokens to, which is greater than the other four sentences so the value of padding have five (‘5’). So, it needs an additional one ‘0’ at the end of the begging of the sentence.

After padding the input data have given to the embedding layer. The function of this component is representing the data with dense vector representation. This is because in order to feed the data to deep learning model the input data should be normalized and it must be in the form of vector representation. The other main advantage of word embedding is representing making the large dimensional space to low and the model has learned efficiently.

In order to use deep learning models for naturel language processing, it needs to transform words into a numeric representation. In our case in order to represent words in vector representation, we use Keras embedding layer which little bit similar to word2vec, but the difference is word2vec is the unsupervised method in order to make similar words together in the embedding space. However, Keras embedding is a supervised method which learns depends on the data input. To find words into the context, we have train word2vec. For example, to tell if “ወሃ” is a likely word given the “ወሸወ ጠጣ።” So, in this sentence we expect word2vec. However, keras embedding is learned as a layer of LSTM, the LSTM is trained in order to predict whatever we want. For example, we train for grammar checking, or analysis, the difference is the data that we input[21]. So, embedding layer learn features for a specific problem. The other importance of embedding layer is to minimize high dimensional space to low dimensional space that means. Minimizing the input feature space to smaller one.

For example, if we give vocabulary ‘11’ which the number of unique words and the embedding length is ‘5’ and the size of the input length should be the same with the length of the largest sentence. So, in our case the input length is ‘11’. So, when we see the result, each token of a sentence has represented with five values. And also, words having the same name are represented with the same vector representation. All of the sentence having padding value ‘0’ also represented with the same vector. Look at the first four sentence, when we see the embedding value of the last words is the same which is represented to ‘0’.

```
import numpy as np
import re
import nltk
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding

amharic_texts = [
    "አበበ ትናንት av ያገባል ።",
    "ጫላ ነገ ያገባል # ።",
    "ያሬድ ምሳ በላ ።",
    "ከበደ በሶ በላች ።",
    "ኤፍሬም ጥቁቁር ላም አሉት ።",
    "መላኩ ነጫጭ በግ + አሉት ።"
]

def clean_text(text):
    text = re.sub(r'[^\w\s።]', '', text)  # Remove unwanted characters
    text = text.lower()  # Convert to lowercase
    return text
```

```
def tokenize_texts(texts):
    cleaned_texts = [clean_text(text) for text in texts]
    tokenizer = Tokenizer(oov_token="<OOV>")
    tokenizer.fit_on_texts(cleaned_texts)
    sequences = tokenizer.texts_to_sequences(cleaned_texts)
    return tokenizer, sequences

def pad_sequences_to_max_length(sequences, max_length):
    padded_sequences = pad_sequences(sequences, maxlen=max_length, padding='post')
    return padded_sequences

def create_embedding_model(vocab_size, max_length, embedding_dim):
    model = Sequential([
        Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=max_length)
    ])
    return model

def main():
    # Tokenize and clean texts
    tokenizer, sequences = tokenize_texts(amharic_texts)

    # Define parameters
    max_length = max(len(seq) for seq in sequences)
    vocab_size = len(tokenizer.word_index) + 1  # +1 for <OOV> token
    embedding_dim = 5



    padded_sequences = pad_sequences_to_max_length(sequences, max_length)


    model = create_embedding_model(vocab_size, max_length, embedding_dim=5)


    dense_vectors = model.predict(padded_sequences)

)
    print("Padded Sequences:\n", padded_sequences)
    print("Dense Vector Representations:\n", dense_vectors)

if __name__ == "__main__":
    main()
```

The output of the above code

```
[[[ 0.01415325 -0.04815719  0.02372627 -0.04935454 -0.02623599]
 [ 0.01559205 -0.0451448   0.00626413  0.04158482  0.02273277]
 [-0.02398608 -0.00586071  0.00877237 -0.00574011  0.02848277]
 [-0.01620913  0.0332143   0.00127568  0.01917002  0.01022679]
 [ 0.01386719  0.04074713 -0.02568331  0.01145451 -0.01837407]]

[[-0.00158125 -0.04247604  0.01056536  0.03166083 -0.0310398 ]
 [-0.00784756 -0.02851758  0.03964355 -0.01587114  0.04778036]
 [-0.02398608 -0.00586071  0.00877237 -0.00574011  0.02848277]
 [-0.01620913  0.0332143   0.00127568  0.01917002  0.01022679]
 [ 0.01386719  0.04074713 -0.02568331  0.01145451 -0.01837407]]

[[ 0.01583746 -0.04340458 -0.00517596 -0.01754392 -0.02193402]
 [-0.01942976  0.00321258  0.02101192  0.01363499  0.01123144]
 [-0.04481779 -0.04597786 -0.02260392 -0.02913864  0.01927132]
 [-0.01620913  0.0332143   0.00127568  0.01917002  0.01022679]
 [ 0.01386719  0.04074713 -0.02568331  0.01145451 -0.01837407]]

[[ 0.02561339 -0.03382635  0.04315476  0.0353005  -0.03004616]
 [-0.03751957 -0.00913785  0.03610896  0.00656302  0.02104156]
 [ 0.02763213 -0.04608851 -0.01936628  0.00938947  0.01568255]
 [-0.01620913  0.0332143   0.00127568  0.01917002  0.01022679]
 [ 0.01386719  0.04074713 -0.02568331  0.01145451 -0.01837407]]

[[ 0.02118354  0.0483419   0.01575344  0.04401766 -0.01316322]
 [ 0.00132259 -0.04689543  0.04251135  0.04321997  0.00771749]
 [-0.04111339  0.00071741  0.03232565  0.01758807  0.03521982]
 [ 0.01212383  0.01925235  0.04557897  0.00578109 -0.02400841]
 [-0.01620913  0.0332143   0.00127568  0.01917002  0.01022679]]

[[ 0.01711008  0.03903002  0.02117052  0.00416557  0.02807501]
 [ 0.035287    0.01557032 -0.00997595 -0.0181347  -0.03392502]
 [-0.02754194  0.01082083 -0.02832406  0.00733057 -0.04921005]
 [ 0.01212383  0.01925235  0.04557897  0.00578109 -0.02400841]
 [-0.01620913  0.0332143   0.00127568  0.01917002  0.01022679]]]
```

Figure 3.4 Dense vector representation of pad sequences

The last component of our model is BILSTM and the function of this component is learning the sequence input data to predict the exact class of the sentence. In this example, the input for BILSTM is the output of the embedding layer. Let us take the last sentence from the example above, the input diminution of BiLSTM have 5,5,5. Therefore, the bidirectional recurrent neural network has worked as follows below in the figure. Every work in the sentence is its unique vector representation. Each of the dense vectors is input to the forward and backwards direction after learning the sequence. We have a dense layer having six neurons and SoftMax classifier. Then the classifier classifies based on each probability distribution. So that the output of this sentence has “AND”, which stands for adjective - noun disagreement. Because the adjective of the sentence tells plural but the noun is singular. AVD stands for adverb verb disagreement, SVD stands for subject verb

disagreement, OVD stands for object verb disagreement, INWS stands for incorrect word sequence.

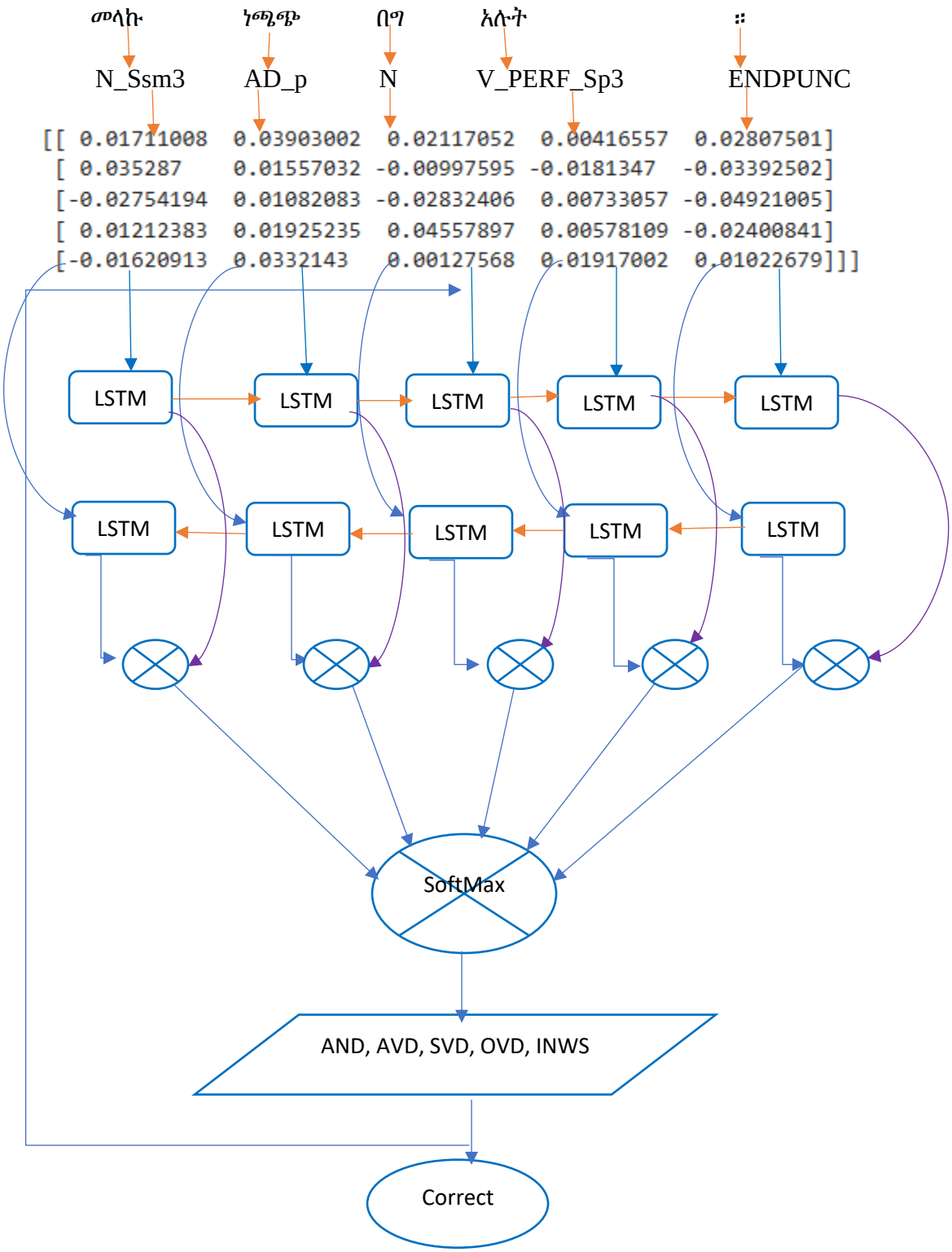

Figure 3.5 Steps to grammar checker and corrector using BILSTM

CHAPTER 4: RESULTS AND DISCUSSION

4.1 Overview

The target of this research is to develop a prototype of a deep learning based Amharic grammar error detection and correction. The prototype includes preprocessing, tokenization, morphology-based tagger, word embedding, and Long Short-Term Memory model. In this chapter we have discussed grammar checker and corrector evaluation metrics, corpus preparation for training data, and morphology-based tagger. We have also discussed experimental details and evaluation results of grammatical error checking and correcting.

4.2 Data Collection and preparation

The first activity of a deep learning-based grammar checker and corrector system is data collection and preparation. We have collected around 35,000 sentences from different source such as Newspapers, Books, Facebook, Politics book, Bible, Magazines, Short history, Fictions, History Books.

The dataset has contained both the correct and incorrect sentence. The incorrect sentence contains different types such as incorrect word sequence, subject –verb disagreement, object-verb disagreement, adverb-verb disagreement, and subject-verb disagreement.

I have prepared two corpus files, the first one is grammatically correct sentence and the second is grammatically incorrect sentence.

4.3 Development Environment

In this study, I have used certain supporting tools such as Keras, Tensorflow. Tensorflow is a powerful library that makes deep learning faster. Keras is user friendly and provides certain python libraries and other python libraries that used to enable developers to use optimized algorithms and implements popular machine learning techniques in classification. Some of the important python libraries used in this study are discussed below.

A) **Scikit-learn** is the main machine learning library that contains features of various classification, regression, and clustering. It also built on Numpy, SciPy, and Matplotlib provides tools for data analysis and data mining

B) **NumPy** is provided with mathematical function that can be used in many calculations and also with an n-dimensional array object in the dataset.

C) **Matplotlib** is the most important visualization libraries to analyze the data. It is a scientific plotting library usually to plot histograms, scatter graphs, lines, ROC curves, and other diagrams.

D) **Pandas** is used for data analysis it can take multi-dimensional arrays as input and produce charts/graphs. Pandas may take a table with columns of different datatypes. It may ingest data from various data files and databases.

4.4 Implementation

Experiments are done based on the prototype developed with Keras (TensorFlow as a backend) on Intel Core™ i7-8550U CPU, and 12 GB of RAM. The proposed model is trained for 100 epochs, a batch size of 64, and a starting or initial learning rate of 0.01. The data is partitioned into training and testing dataset such that 80 percent of the data is assigned for training the model and 20 percent of the data is assigned for testing.

Deep learning based Amharic grammar error detection and correction is implemented using bidirectional long short-term memory recurrent neural network. In this study, the prototype of the system used Tokenization of words, Tag sequence splitting, Sequence padding, word embedding, Amharic grammar error detector and corrector and several development tools.

4.5 Performance Evaluation and Testing

To check the performance of deep learning-based Amharic grammar error detection and correction evaluation and testing is important. So, I have used the following hyper parameters, such as epoch, batch size, learning rate, drop out, and Adam optimizer. I have evaluated the performance of the system using confusion metrics such as recall, precision and f1 measure (see table 4.1). I selected those evaluation metrics because the size of the

dataset is an imbalance; as a result, accuracy may not predict exactly, so confusion matrix is mandatory to evaluate the performance of the system.

True Positive – These are the correctly predicted positive as positive which means that the value of the actual class is positive and the value of the predicted class is also positive.

True Negatives – These are the correctly predicted as negative value which means that the value of the actual class negative and the value of the predicted class negative.

False Positives – It shows the actual class is negative but the classifier predicted as positive.

False Negative – The actual class is positive but the classifier predicts as negative. The other evaluation metrics are described.

Metric	**Formula**	**Description**
Accuracy	$\frac{TP+FN}{TP+FN+FP+TN}$	Overall performance of the model
Precision	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$	How the positive description is accurate
Recall	$\frac{TP}{TP+FN}$	Coverage of actual positive samples
F1-Score	$2 * \frac{Precision*Recall}{Precision+Recall}$	The harmonic means of precision and recall

Table 4.1 evaluation matrix

4.5.1 Test Result using Bidirectional Long Short-term Memory (BiLSTM)

In bidirectional long short-term memory network, we have used the following parameters such as epoch=100, batch size =64, Adam optimizer with learning rate 0.01 and dropout=0.5. We are splitting the dataset with 80% for training and 20% for validation and for testing as the performance implies the training accuracy around 91%, and the validation accuracy is 86% and the testing accuracy shows around 88.89%.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         (None, 100)               0
_________________________________________________________________
embedding_3 (Embedding)      (None, 100, 32)           2752
_________________________________________________________________
bidirectional_3 (Bidirection (None, 64)                16640
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 390
=================================================================
Total params: 19,782
Trainable params: 19,782
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 00098: val_acc did not improve from 0.86883
Epoch 99/100
3079/3079 [==============================] - 4s 1ms/step - loss: 0.2947 - acc: 0.8876 - va

Epoch 00099: val_acc did not improve from 0.86883
Epoch 100/100
3079/3079 [==============================] - 4s 1ms/step - loss: 0.2827 - acc: 0.8954 - va

Epoch 00100: val_acc did not improve from 0.86883
training Accuracy is 0.9187.
testing Accuracy is 0.8889.
F-measure  is 0.8904.
Recall is 0.8889.
Precision is 0.8983.
Keppa Score is 0.8983.
              precision    recall  f1-score   support

           0       0.99      0.85      0.91       163
           1       1.00      0.98      0.99       147
           2       0.74      0.90      0.81       219
           3       0.96      0.97      0.96       138
           4       0.94      0.90      0.92       144
           5       0.83      0.74      0.78       152

    accuracy                           0.89       963
   macro avg       0.91      0.89      0.90       963
weighted avg       0.90      0.89      0.89       963
```

Figure 4.1 The performance of BiLSTM with epoch 100 and 80/20 splitting ratio

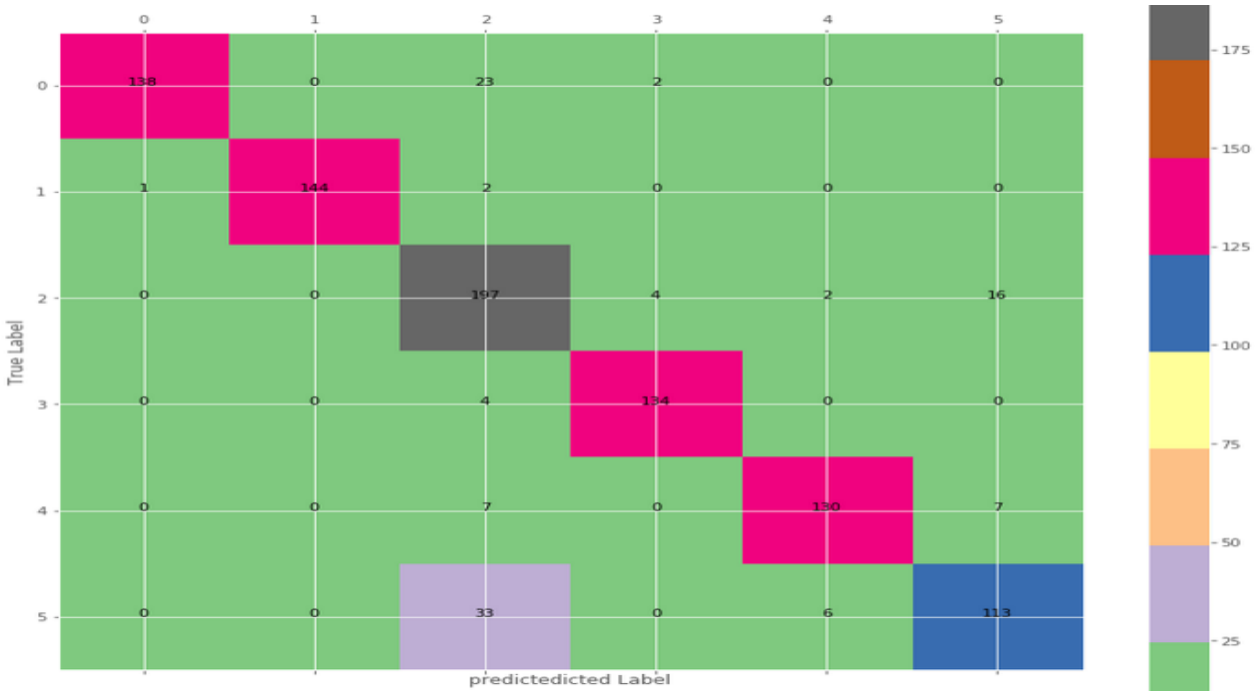

Figure 4.2 Confusion matrix for BiLSTM with 80/20 ratio

Figure 4.2 shows the confusion matrix and loss value. The matrix shows how many of the data is predicted actual class and incorrectly classified. When we are splitting the dataset with 80/20 ratio and validated with 20% and applying on bidirectional long short-term memory. The performance of the proposed system is 88.89% accuracy, 89% f1, 88.89% recall and 89% precision.

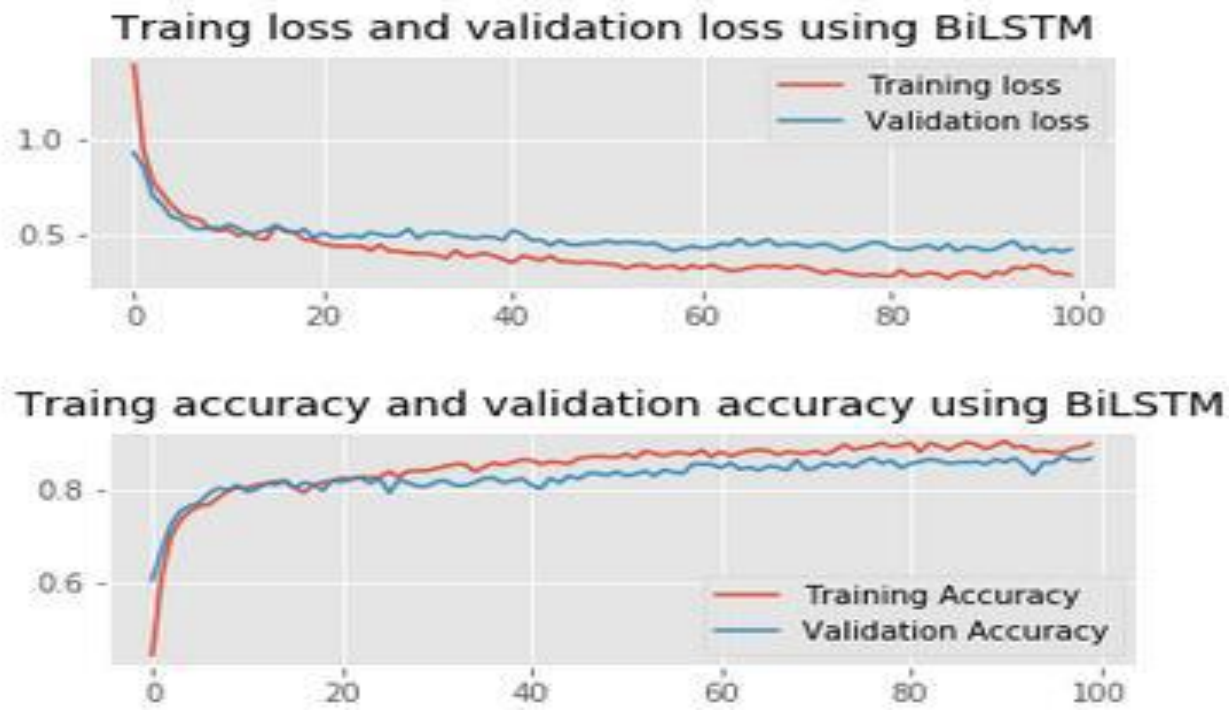

Figure 4.3 Accuracy and loss for training and validation with BiLSTM 80/20 ratio

The above figure shows about the performance of bidirectional long short-term memory with 80% of the data set is allocated for training and 20% the data is assigned for testing. Then training loss and validation loss of the model is 0.28 and 0.41 respectively.

4.6 Discussion of the result

I have developed Amharic grammar error detection and correction using deep learning that uses bidirectional long short-term memory. I have detected Amharic sentence errors when adjective and noun disagree with number and gender, adverb and verb disagree intense, subject and verb (disagree in number, gender and person), object and verb disagree in number, gender and person), and incorrect word sequences (adjective-noun disorder, adverb-verb disorder, object-verb disorder and, subject-object disorder).

I have evaluated the model by splitting the dataset with 80/20 ratio for training and testing respectively. In order to train, validate and test the model different hyper parameters are

used such as epoch=100, batch size =64 and dropout=0.5, and Adam optimizer with learning rate 0.01.

The experiment shows that BiLSTM performed 92% training accuracy and 88.89% testing accuracy with 80/20 splitting ratio.

The confusion matrix shows that the model correctly predicts most of the data, but some sentences are misclassified. This is often because a single sentence may contain multiple disagreement issues. For example, in the sentence "ከበደ ነገ አገባች ።": The subject "ከበደ" is singular, masculine, and third person, while the verb "አገባች" is singular, feminine, third person, and in the perfective tense. This results in a **subject-verb disagreement** due to the gender mismatch (masculine subject vs. feminine verb).

The adverb "ነገ" refers to the future tense, but the verb "አገባች" is in the perfective (past) tense, leading to an **adverb-verb disagreement**.

This sentence has two disagreement problems: **adverb-verb disagreement** and **subject-verb disagreement**. In such cases, the model may classify the sentence into both error categories. The BiLSTM network examines tag sequences in both forward and backward directions to identify such errors.

When compared to existing systems, the proposed deep learning-based model performs better. For instance: Aynadis and Yaregal (2013) achieved 92% precision and 94% recall for simple sentences using a rule-based approach, while their statistical-based approach achieved 67% precision and 90% recall. For complex sentences, the statistical-based approach detected 63.76% of errors.

In their 2019 study, Aynadis and Yaregal reported 68.18% accuracy for subject-verb agreement, 20% for adverb-verb agreement, and 81.25% for object-verb agreement. These systems performed well for simple sentences but struggled with compound, complex, and compound-complex sentences.

In contrast, the proposed model achieves **89% recall, 89% precision, and 89% accuracy** for all Amharic sentence types, including simple, compound, complex, and

compound-complex sentences, showcasing its superior performance across a broader range of sentence structures.

In the previous research's features are generated manually which is difficult to list down all rules because Amharic is morphologically rich and complex language. But the proposed system can generate and learn features automatically.

CHAPTER 5: CONCLUSION & FUTURE WORK

5.1 Conclusion

The goal of this research is to design & develop a system for finding and fixing grammar errors in Amharic using deep learning. There are several parts to this grammar checking module. First, we have the preprocessing module. This part cleans up the input sentences. It also breaks down sentences into tokens and tags them.

Next is tag splitting. This part takes the tags from the sentences. After that, those tags are changed into word vectors using word embedding. The last part is the BiLSTM component. It uses data from the word embedding section to check if the sentence is correct in grammar. If it's not correct, it will suggest a fix.

We trained & tested this new system with sentences we prepared. We gathered two types of text: one with correct grammar and another with errors. The system was built using Python 3.7 and Keras with TensorFlow as the backend.

In the end, we checked how well the deep learning system did for finding & correcting grammar errors. We used a confusion matrix for this evaluation. The results showed that this model achieved 88.89% accuracy, an F1 measure of 89%, and recall of 88.89%. The precision also stood at 89%.

5.2 Contribution of the study

Here's what this study brings to the table:

- I have created a corpus with over 35,000 words. That's a lot of data!
- The study suggests a deep learning method for checking grammar errors in Amharic.
- It also adds to the tagger model for the Amharic language.

5.3 Future work

We have done a lot of work to create a model using deep learning for finding and fixing grammar mistakes in Amharic. The model we made, along with the data we collected, can also help with other tasks in natural language processing (NLP), like translating languages. But to make the system even better, we need to put in more effort. Here are some things I noticed that we think should be done:

1. **Expand the Scope**: This study focused solely on Amharic grammar error detection and correction. We suggest extending the work to include both grammar and spelling error detection and correction.
2. **Building a Comprehensive Corpus**: a large Amharic corpus that includes morphological features, an automatic spelling checker, and a morphological analyzer would significantly improve results.
3. **Explore Other Deep Learning Models**: while this study utilized Bidirectional Long Short erm Memory (BiLSTM), we recommend experimenting with other deep learning architectures, such as encoder-decoder models, Transform neural networks, and standard LSTMs.
4. **Extend Other Local Languages**: The proposed approach should be adapted and applied to other local languages to develop grammar checkers for a wider range of languages.

These steps would contribute to more robust and versatile NLP tools for Amharic and other languages.

Reference

1. ALEMU, Y.D., *DEEP LEARNING BASED AMHARIC GRAMMAR ERROR DETECTION.* 2023.
2. Khurana, D., et al., *Natural language processing: state of the art, current trends and challenges.* Multimed Tools Appl, 2023. **82**(3): p. 3713-3744.
3. Yaregal, A., *Development of Amharic Grammar Checker Using Morphological Features of Words and N-Gram Based Probabilistic Methods.* International Conference, 2013.
4. Nadkarni, P.M., L. Ohno-Machado, and W.W. Chapman, *Natural language processing: an introduction.* J Am Med Inform Assoc, 2011. **18**(5): p. 544-51.
5. Tessema, T.T., *Word Sequence Prediction for Amharic Language.* 2014.
6. Menzel, M.Y.T.a.W., *Amharic Part-of-Speech Tagger for Factored Language.pdf.* International Conference 2009.
7. GOBENA, M.K., *IMPLEMENTING AN OPEN SOURCE AMHARIC RESOURCE.pdf.* 2010.
8. ANDUALEM, G., *Girmaw Andualem Thesis Document.pdf.* 2022.
9. Gasser, M., *A-Dependency-Grammar-for-Amharic.pdf.* 2010.
10. Wondwossen Mulugeta, a.M.G., *Learning Morphological Rules for Amharic Verbs.pdf.* 2012.
11. Willett, N.A.a.P., *Stemming of Amharic Words for Information Retrieval.pdf.* 2002.
12. Naber, D., *A Rule-Based Style and Grammar Checker.* 2003.
13. Tesfaye, D., *Paper 23-A rule-based Afan Oromo Grammar Checker* 2011.
14. Gebrekiros, A., *Development of Tigrigna grammar checker using hybrid approach.* 2018.
15. Gebreamlak, A., *DEPENDENCY BASED AMHARIC GRAMMAR CHECKER.* 2019.
16. S. C. Kremer and J. F. Kolen, e., *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies>.* 2001.
17. Mikami, A., *Long Short-Term Memory Recurrent Neural Network Architectures for Generating Music and Japanese Lyrics.* 2016.
18. Zhiyong Cui, R.K., Ziyuan Pu, Yinhai Wang, *Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction.* 2018.
19. Rickard Domeij, O.K., Johan Carlberger, Viggo Kann and K. Nada, Stockholm, *Granska an efficient hybrid system for Swedish grammar checking.pdf.* 1999.
20. Al-Khalifa, M.a., *A Proposed Arabic Grammatical Error Detection Tool Based on Deep Learning.pdf.* 2018.
21. Jason, B. *Machine Learning:Brownlee, J. (2019). What is Deep Learning? Machine Learning Mastery.[Online]Available:https://machinelearningmastery.com/what-is-deep-learning/,Augest 16,2016.* 2019.