MEKELLE UNIVERSITY

ETHIOPIAN INSTITUTE OF TECHNOLOGY- MEKELLE (EIT-M)

SCHOOL OF COMPUTING

POSTGRADUATE PROGRAM IN COMPUTER SCIENCE

DEVELOPMENT OF A GEEZ GRAMMAR CHECKER USING A HYBRID APPROACH

BY: Aster Hagos Bisrat

ADVISOR: Guesh Dagnew (PHD)

CO-ADVISOR: Zewude T/haymanot (MSC)

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

Mekelle, Tigray, Ethiopia

July, 2025

DECLARATION

I, Aster Hagos Bisrat, hereby declare that this thesis titled "**DEVELOPMENT OF A GEEZ GRAMMAR CHECKER USING A HYBRID APPROACH**" is my original work and has not been submitted, either partially or in full, by any other individual for the award of a degree at any other university or institution.

Name of the candidate: Aster Hagos Bisrat

Signature: _____

Date: _____

APPROVAL FOR SUBMISSION

This thesis, entitled "**DEVELOPMENT OF A GEEZ GRAMMAR CHECKER USING A HYBRID APPROACH**" by Aster Hagos Bisrat, has been submitted for examination for the degree of Master of Science in Computer Science with my approval as advisor in the School of Computing, EIT-M and Mekelle University.

Name of Advisor: Guesh Dagnew (Ph.D.)

Signature: 

Date: 22-12-2025

CERTIFICATION

The undersigned certify that they have read and hereby recommend the Ethiopian Institute of Technology Mekelle (EIT-M), School of Computing, to accept the thesis submitted by Aster Hagos Bisrat titled " **DEVELOPMENT OF A GEEZ GRAMMAR CHECKER USING A HYBRID APPROACH**" in partial fulfillment of the requirements for the award of a Master's Degree in Computer Science.

Name of Advisor: Guesh Dagnew (Ph.D.)

Signature: 

Date: 22-12-2025

ETHIOPIAN INSTITUTE OF TECHNOLOGY- MEKELLE (EIT-M)

SCHOOL OF COMPUTING DEPARTMENT OF COMPUTER SCIENCE

DEVELOPMENT OF A GEEZ GRAMMAR CHECKER USING A HYBRID APPROACH

By Aster Hagos Bisrat

July, 2025

Name and Signature of Examining Board Members

1. Name of External Examiner: Tesfay Gidey(PhD)

Signature: _____

Date: 12/15/2025

2. Name of Internal Examiner: Fitsum Gebregziabher(PhD)

Signature: _____

Date: 12/19/2025

3. Name of the Chairperson: Teages Kalayu

Signature: _____

Date: 12/23/2025

Dedication

I dedicate this MSc thesis to my God, Saint Mary and my family, all people who have had their own share of contribution throughout my life. Love you!

Acknowledgment

First, I wish to express my gratitude to Almighty God and His Mother, Saint Mary, for providing me with life and the strength to see this thesis through. I am especially thankful to my advisor, Dr. Guesh Dagnewu, for his continual guidance over the course of the year. His patience, encouragement, to my work have been immensely rewarding to my career.

I would also like to thank the Department of Computer Science for providing me with the opportunity to conduct this work. I offer my sincere thanks to my co-advisor, Mr. Zewude Teklehaimanot, for his rich experience and contribution towards the successful completion of my research.

I wish to express my highest gratitude to Mr. Hawaz Beyene for his generosity, tolerance, and commitment. He constantly corrected my many mistakes and motivated me from the beginning to successful completion of this research.

I am also sincerely thankful to my colleague Mrs.Merahwait Tafere, for her support and collaboration during this process.

Finally, I would wish to dedicate my heartfelt thanks to my family for their coming love, support, and motivation throughout my study period.

List of Figures

List of Table

List of Algorithms

List of Acronyms and Abbreviations

AVA	Adverb Verb Agreement
IR	Information Retrieval
MS	Masculine
NN	Noun
OB	Objective
PL	Plural
POS	Part of Speech
PRP	Proper noun
SB	Subjective
SN	Singular
SOV	Subject-Object Verb
SP	Second Person
SVA	Subject Verb Agreement
TD	Thread Person
VRB	Verb
WSA	Word Sequence Agreement

Table of Contents

Abstract

Grammar checking is a fundamental task in Natural Language Processing (NLP), aimed at verifying the grammatical correctness of a given text. In the contemporary digital era, humans employ various platforms like websites and mobile apps to produce copious amounts of text. For the purpose of suitable communication, suitable rules of grammar should be used. but while proofreading huge chunks of texts manually, it becomes tedious, prone to errors, and not possible. Thus, there have been programs of automated grammar checkers for many languages like English, Amharic, and Afan Oromo. Despite its historical and liturgical importance, the Geez language currently lacks a dedicated grammar checking system. This thesis addresses this gap by proposing a grammar checker specifically designed for Geez. The proposed system employs a hybrid approach that combines rule-based methods with statistical techniques to identify grammatical errors related to subject-verb agreement, object-verb agreement, adverb-verb agreement, and word order. The system has five main modules namely Preprocessing, Bigram Formation, Rule based grammar checker, Statistical grammar checker, and Display Grammar Checking Result to process the input text and to build the bigram of the system. It is developed using Python programming language utilizing different IDEs such as Jupyter Notebook and PyCharm IDEs, and it employs a MySQL database for handling linguistic data. The database of the system consists of more than 18,000 Geez sentences as corpus. The rule-based part of the system checks Subject-Verb, Object-Verb, and Adverb-Verb agreement errors using 360 handwritten correct grammar rules. The statistical module detects WSA errors based on unique word sequences and a probabilistic language model. The system was evaluated through three separate experiments using a dataset of 400 manually prepared sentences, containing both grammatically correct and incorrect examples. The results demonstrate that the system achieved strong overall performance, with an average precision of 88.09%, recall of 89.35%, and F-measure of 88.68%. These results demonstrate how well and consistently the suggested grammar checker can detect grammatical mistakes in Geʿez text.

Keywords: hybrid approach, rule-based approach, statistical method, Geez language, grammar checker, and natural language processing.

CHAPTER ONE:INTRODUCTION

The use of language is essential for communication. Grammar accuracy is required to ensure meaning and clarity in communication. Despite the fact that the Geez language lacks contemporary computerized means, it still serves religious use and is an ancient and valuable Semitic language. While modern grammar checking systems have made significant strides for widely spoken languages, lesser-resourced languages like Geez remain underrepresented in computational linguistics. This research aims to address this gap by developing a grammar checker tailored specifically for the Geez language.

In this chapter, the key ideas and objectives for the development of a Geez grammar checker are laid out. It covers the history of Natural Language Processing (NLP) and how it is used in grammar checking, the unusual linguistic features of the Geez language, and the absence of a dedicated grammar checker. The chapter dictates the motivation, problem statement, research questions, and objectives, alongside the hybrid statistical and rule-based methodology, data collection using tools like the Holy Bible, and software such as Python and MySQL. The chapter also dictates measures of evaluation, scope in direction of syntactic and morphological errors, and limitations, such as hand corrections and limitation of availability of corpus.

1.1 Background

Language serves as a fundamental tool for human communication, enabling the exchange of ideas, knowledge, and culture. However, effective communication relies heavily on grammatical accuracy to convey intended meanings clearly and precisely. Errors in grammar can lead to misunderstandings, particularly in written texts, where contextual cues like tone or gestures are absent. In the digital age, the proliferation of text-based platforms such as social media, emails, academic writing, and online publishing has amplified the need for automated tools to ensure grammatical correctness. Natural Language Processing (NLP), a subfield of artificial intelligence and computational linguistics, has emerged as a key discipline in addressing these challenges by enabling computers to understand, interpret, and manipulate human language [1].

NLP applications, including grammar checkers, have evolved significantly since the 1950s, starting with early rule-based systems for machine translation and progressing to advanced

statistical and machine learning models in the 21st century. Grammar checking, in particular, involves analyzing text for syntactic, morphological, and sometimes semantic errors, often integrated into word processors like Microsoft Word or standalone tools like Grammarly. These systems have achieved high accuracy for high-resource languages such as English, where vast corpora and annotated datasets enable sophisticated models. For instance, rule-based grammar checkers for English and Swedish have demonstrated precision rates above 90% in detecting agreement errors and word order issues [1].

Despite these advancements, low-resource languages those with limited digital corpora, annotated data, or computational tools remain underrepresented in NLP research. Semitic languages, characterized by rich morphology (e.g., root-and-pattern word formation) and complex syntax (e.g., flexible word order like VSO or SOV), pose unique challenges for grammar checking due to their non-linear structures and inflectional variations . Languages like Arabic, a well-studied Semitic language, have benefited from dedicated grammar checkers that handle morphological complexity through hybrid rule-based and statistical approaches, achieving effective error detection for agreement and case mismatches. Similarly, for Ethiopian Semitic languages such as Amharic and Afan Oromo, grammar checkers have been developed using rule-based methods for syntactic errors and hybrid models incorporating n-gram probabilities, reporting precision and recall rates around 85-95%.

Geez (ግዕዝ), an ancient South Semitic language and the root of modern Ethiopian languages like Amharic and Tigrinya, exemplifies these challenges. As an alpha-syllabary (abugida) script with over 200 characters representing consonant-vowel combinations, Geez features intricate morphology, including gender, number, and case inflections, and flexible sentence structures (e.g., SOV, VSO). Historically used in the Aksumite Kingdom and still serving as the liturgical language of the Ethiopian Orthodox Tewahedo Church, Geez holds immense cultural and religious significance, with a vast literary corpus including the Bible and ancient manuscripts. However, its status as a low-resource language lacking large digitized corpora and modern NLP tools has hindered computational advancements. While preliminary work exists on Geez POS tagging, morphological analysis, and spell checking, no comprehensive grammar checker has been developed prior to this study, leaving a critical gap in automated linguistic support for Geez texts.

This research addresses this gap by proposing a hybrid grammar checker for Geez, combining rule-based methods for explicit syntactic rules with statistical techniques for probabilistic error detection. By leveraging a corpus derived from sources like the Geez Bible and employing tools such as Python for implementation and MySQL for data management, the system targets key errors in subject-verb, object-verb, adverb-verb agreements, and word sequences.

Geez POS tagger, morphological Analyzer, sentence parser, spell checker are some of the attempts that have been made in Geez language processing in the past years. However, to the best knowledge of the researcher there is no research conducted for Geez grammar checker. Therefore, this research is aimed at filling the gap on Geez grammar checker, which assists users in eliminating mistakes in their Geez writings.

1.2 **Motivation**

Electronic texts, such as books, blogs, emails, newspapers, and scholarly theses, have greatly improved communication among Geez speakers due to the extensive use of personal computers and mobile devices. A grammar checker is necessary for users to produce high quality, error-free documents without concentrating on grammatical errors because these texts are produced on a daily basis. Better data management and organization are made possible by using proper grammar in electronic documents. Many documents contain grammatical mistakes because there is not a specialized Geez grammar checker, which undermines their professionalism and clarity.

An automatic Geez grammar checker/corrector that would identify these errors and provide solutions would greatly improve the quality of electronic texts. By creating professional, understandable, and grammatically sound documents, this tool would enhance communication while preserving Geez's linguistic integrity. The researcher is motivated to develop a grammar checker for Geez that will ensure user inputs are grammatically correct in order to get around the challenges brought on by the absence of such a tool.

1.3 Statement of problem

Grammar checkers have been developed for various languages, such as English [2],[3] Swedish [4], Chinese [5], Amharic [6], and Afan-Oromo [7], to identify syntactical errors in text inputs. These tools are integral to numerous natural language processing (NLP) applications, including question answering, semantic analysis, and dialog systems. in order to verify that sentences are grammatically correct, grammar checkers are now frequently included in word processors as standalone programs or modules. For instance, the Amharic grammar checker employs a hybrid approach combining statistical and rule-based methods [8], while the Afan-Oromo grammar checker relies solely on a rule-based approach [7].

However, the grammatical structure of Geez differs significantly from languages like Amharic, Afan Oromo, English and others. When developing a grammar checker, these variations include unique morphological, syntactic, and phonological characteristics. To our knowledge, there has n't been any research or development done on a grammar checker for Geez. While Amharic and Geez share some linguistic features, they diverge in critical areas such as morphological processing and syntax. The success of the hybrid statistical and rule-based approach for Amharic [6] suggests that a similar methodology could be effective for Geez, provided the language's distinct characteristics are addressed. Consequently, there is a pressing need to develop a Geez grammar checker using a combined statistical and rule-based approach to ensure accurate detection and correction of grammatical errors, tailored to the specific linguistic properties of Geez.

1.4 Research Questions

1. How can a representative Geez corpus be constructed for training and evaluating a grammar checker?
2. What algorithm can be designed for effective grammar checking in Geez?
3. How can a prototype Geez Grammar Checker be developed based on the proposed algorithm?
4. How can the performance of the Geez Grammar Checker be evaluated using standard metrics?

1.5 General Objective

The general objective of this research work is to develop an effective Geez Grammar Checker using a hybrid of statistical and rule-based techniques by constructing a representative corpus, designing a suitable algorithm, building a functional prototype, and evaluating its performance using standard metrics.

1.5.1 Specific Objective

The specific objectives of this study are given as follows:

- To collect and construct a corpus for developing a training set to train the system and a test set to evaluate the performance of the developed prototype
- To design grammar checker algorithm for Geez text
- To develop a prototype of Geez Grammar Checker
- To validate the proposed system using standard performance evaluation methods

1.6 Methodology

The study uses structured methodology including collection and construction of a representative Geez text corpus, which is divided into training and testing datasets. A hybrid of rule-based and statistical grammar checker algorithms is designed to identify and correct grammatical errors in Geez text. Based on this algorithm, a prototype Geez Grammar Checker system is developed. Finally, the system's performance is evaluated using standard evaluation metrics to assess its accuracy and effectiveness.

1.6.1 Data collection

Geez documents were collected from multiple sources to build a representative and high-quality corpus. These sources included hard-copy books obtained through consultations with Geez language experts, ensuring the authenticity and linguistic accuracy of the materials. Among these sources, the Holy Bible was selected as the primary basis for the corpus. Its extensive use, rich vocabulary, and grammatical consistency make it a valuable resource for linguistic analysis and grammar checking.

1.6.2 Tools and Techniques

The system in this study was developed by the researcher using the Python programming language and the Anaconda framework. PyCharm enabled more structured coding activities, while Jupyter Notebook was used for interactive testing and prototyping. The linguistic information used for the grammar check was also processed and stored in a MySQL database.

Approaches

Research on grammar checkers has used various strategies that is, overall strategies or models used in order to detect and fix grammatical errors. The most commonly applied strategies are rule-based, statistical, and hybrid. The three strategies are elaborated below.

Statistical approach

A statistical approach does need a corpus to train the language model (LM) instead of handcrafted grammatical rules. Word sequences, which occur often in the corpus, can be considered correct in other texts to uncommon sequences might be error.

Rule-based approach

The rule-based approach is performed by constructing error detection rules manually for the given language. These rules are then used to find errors in the text that has already been analyzed. These rules often contain suggestions on how to correct the error found in the text.

Hybrid approach

As its name implies, this approach takes the combination of rule-based approach and statistical based approach by taking the advantages from both approach to improve the performance of the system. A grammar checker system based on a hybrid approach such as statistical and rule-based approaches give better results than the corresponding uncombined approaches.

1.6.3 Evaluation metrics

To evaluate the proposed grammar checker, standard metrics from Natural Language Processing such as Precision, Recall, and F measure were used. Precision measures how many flagged errors are correct, while Recall reflects how many actual errors the system detects. F measure balances these to give a single performance score. These well-established metrics effectively capture both

the accuracy and coverage of error detection, providing a reliable framework to assess the system's effectiveness in identifying grammatical errors in Geʿez texts.

1.7 Scope

To find grammatical mistakes in Geez written texts, this research aims to develop a grammar checker application for the language. By combining statistical and rule-based methods, the system will be able to detect both syntactic and morphological errors, which is especially suitable for the distinct linguistic features of Geez. Word order and subject-verb agreement errors are the grammar checker's target error types. The rule-based component is also intended to identify grammatical errors that are likely to occur in simple sentences.

1.8 Limitation

Even though the study yielded positive results, there are some restrictions that limit the suggested grammar checker's overall usefulness. Grammar mistakes are identified by the system, but no automatic corrections are offered, so users must edit the text by hand. Also, the scarcity of annotated Geʿez corpora could cause rare or intricate grammatical constructions to be overlooked. The manual creation of the grammar rules based on expert input may have introduced bias and limited coverage to more commonly occurring structures. Conceptually, the system eliminates semantic and discourse-level analysis and concentrates only on grammar, simplifying some of Geʿez's more intricate linguistic features. Time and financial limitations also limited the scope of data collection and the depth of system development.

1.9 Thesis Organization

This thesis is organized into five chapters. Chapter 1 introduces background of the study, motivation of the study, statement of the problem, general and specific objectives, significance, scope and limitation of the study, along with the thesis organization. Chapter 2 reviews related work on grammar checking tools, natural language processing techniques, and relevant studies on Geez and language. Chapter 3 describes the research design, including data collection, and the hybrid approach combining rule-based and statistical methods. Chapter 4 presents the implementation details, experimental setup, evaluation, and discusses the findings of the Geez

grammar checker. Finally, Chapter 5 presents the conclusions, and suggests directions for future research.

CHAPTER TWO:LITERATURE REVIEW AND RELATED WORKS

2.1 Introduction

For providing an overview of the most significant phrases and terms discussed in the thesis, chapter one provides general issues that form the groundwork for the rest of the study. It touches on the Geez language, how to design grammar checkers, and how to measure such systems. The subtopic covering the Geez language tackles detailed descriptions of POS, morphology, grammatical structure, and common grammatical mistakes. The rule-based, statistical-based, and hybrid approaches are considered in the development section of grammar checkers. The final section presents the measures of evaluation used to evaluate grammar checkers.

2.2 Geez Language

Geez is an ancient south Semitic language of Ethiopia and Eritrea in the horn of Africa and became Aksum's civilization kingdom official language [2],[9]. Geez is still the liturgical language of EOTC since the early 4th century, Ethiopian Catholic Church and Beta Israel Jewish community of Ethiopia. Despite its speakers becoming less in number around the 13th century, it maintained it as the primary written language of Ethiopia up to the 20th century. Religious and secular writings are included in the literature list of Geez language [10],[11].

In Ge'ez script, a character represents a consonant and a vowel combination that makes the language alpha syllabary script or "Abugida" [12], [10],[13]. In abugida, a character represents one sound either its consonant or vowel. Geez has a free order of sentence structure but usually falls on SVO, VSO and SOV [10].

2.2.1 Geez POS

POS is an appropriate given class of word, in which each word of a written text has a unique word class. There are seven major parts of speeches in Geez language. Nouns, Pronouns, Adjectives, Verbs, Adverbs, Prepositions and Conjunctions [1].

Nouns

Noun refers anything that represent a thing, feeling, place, animal, person and idea. There are two ways to construct a noun: Nouns constructed by nature, and Nouns derived from verbs.

Noun constructed by nature includes for example: ዓራት/arat (bed), ድማህ/dmah (mother), ክሳድ/ksad (neck), ፈረስ/Feres (Horse). Nouns that are constructed or derived from verbs also used as formal nouns in clauses, phrases, and sentences [14].

Verb Noun: ሐለየ (thought) ሕሊና (think) ጥዕየ (Healthy) ጥዒና (health) ባረከ (blessed) ቡራኬ (Blessings) to make plural noun for Ge'ez sentence these alphabets or fidels used: አ፣ት፣ን፣ያን፣ያት፣ው፣ል. While 'አ' always added on the begging of the word, all the others come at the end of the word.

Table2. 1Making of plural nouns for Geez language by adding fields

Added fidels	**Original known**	**Inflicted to**
አ	ደብር	አድባር
አ.....ት	ገብር	አግብርት
ት	ንጉሥ	ነገሥታት
ን	ባዕድ	ባዕዳን
ያን	ዘማሪ	ዘማሪያን
ያት	ውዳሴ	ውዳሴያት

Pronouns

Pronouns shortly considered as a substitution to a noun or noun phrase. Without pronouns, it is mandatory to mention nouns and this in turn makes our speech and writing more cumbersome. Geez language has 10 pronouns [15]. The 10 pronouns of Geez language.

Table2. 2 Geez pronouns list [16]

ግእዝ	English
አነ	I
ንሕነ	We
አንተ	You
አንቲ	
አንትሙ	
አንትን	
ውእቱ	He
ይእቲ	She
ውእቶሙ	They
ውእቶን	

Geez pronouns used as pronouns, as verb to be and as demonstrative pronouns. They further divided into pronouns of gender, pronouns of number, personal pronouns and pronouns based on their task. These can summarize in table 2.3.

Table2. 3 Geez pronoun with their respective task [16]

Personal			Gender			Number		Based on their task		
First person	Second person	Third person	Male	Female	Both	Singular	Plural	Near indicator	Far indicator	Common
አነ	አንተ	ውእቱ	አንተ	አንቲ	አነ	አነ	ንሕነ	አንተ	ውእቱ	አነ
ንሕነ	አንቲ	ይእቲ	አንትሙ	አንትን	ንሕነ	አንተ	አንትሙ	አንቲ	ይእቲ	ንሕነ
	አንትሙ	ውእቶሙ	ውእቱ	ይእቲ		አንቲ	አንትን	አንትሙ	ውእቶሙ	

	አንትን	ውእቶን	ውእቶሙ	ውእቶን		ውእቱ	ውእቶሙ	አንትን	ውእቶን	
						ይእቲ	ውእቶን			

As ‘verb to be’, pronouns also expressed as a past tense

Table2. 4 Geez pronoun as verb to be [13]

ግእዝ	English
ውእቱ	Be, is, was
ይእቲ	Be, is, was/will be
ውእቶሙ	Are, were
ውእቶን	Are, were/will be
አንተ	Are, were
አንቲ	Are, were/will be
አንትሙ	Are, were/will be
አንትን	Are, were/will be
አነ	Am, was/will be
ንሕነ	Are, were/will be

Pronouns as demonstrative pronouns

Table2. 5 Geez pronouns as demonstrative pronouns[13]

ግእዝ	English
ዝ፣ዝንቱ	This
ዛ፣ዛቲ	This (Feminine)
እሉ፣እሉንቱ	These
እላ፣እላንቱ	These (Feminine)
ዝኩ፣ዝስኩ፣ውእቱ	That

እታክቲ፣አንትኩ፣ይእቲ	That (Feminine)
እሙንቱ፣እልክቱ፣ውእቶሙ	Those
እማንቱ፣እልክቶን፣እልኮን	Those (Feminine)

Adjectives

A word further describes, define and identify noun or pronoun. While nouns tell us about things nature, adjectives stand to tell us about their behavior or characteristics like type, color, property, shape, size [2] adjectives can be constructed by changing the verb into a word which his last alphabet is the third alphabet the arrangement like ፈጠረ→ ፈጣሪ or by changing the verb into a word which his last alphabet is the sixth alphabet like ተግሀ → ትጉህ or by adding a ‘መ’ alphabet or Fidel on the verb like ዘመረ (gave thanks) → መዘምር. Another way of forming an adjective is by adding the fidels ‘ዊ (ይ)’ or ‘ዊት’ like ገሊላ→ ገሊላዊ/ይ፣ኢትዮጲያ → ኢትዮጲያዊት. In Ge’ez, there are also demonstrative adjectives that used to express near or far things. For example, ዝ/ዝንቱ (This (male)) ፣ዛ/ዛቲ (This (female) and ዝክቱ/ዝኩ (That – (male)) ፣እንታክቲ (That – (female)). There are also adjectives to represent an amount of a thing like, ሕቅ → few፣ንስቲት/ሕዳጥ → bit፣ብዙኅ → a lot፣ንሕኑሕ → a lot፣ኵሉ → all. Other forms of constructing an adjective are adjectives of numbers like አሐዱ፣ክልዔቱ, interrogative adjectives like መኑ፣ምንት፣አይቴ, and adjectives to plurality for both men and women like ቀተለ (kill) → ቀተልት (Kellers).

Verbs

Geez sentences have a verb on their middle sentences[10]. Inflection are used to Ge’ez words with respect to person, gender and number. Verbs of the language may be either in a perfect (past form)

or imperfect form (present and future forms). The verbs of the Ge'ez language have Semitic nonlinear word formation with intercalation of roots with vocalic pattern. In Ge'ez language, there are eight root verbs, with different characteristics, that lead the time behavior and using their morphology style[10], [5]other similar verbs follow these root verbs.

Table2. 6 Geez language root verbs [17]

Verb Category	Meaning
ቀተለ /q'at'al'a/	He killed
ቀደሰ /qaddasa/	He consecrated
ገብረ/gebra /	He did
ተንበለ/tanbala/	He begged
ባረከ/baraka/	he blessed
ዴገነ/dagana/	He followed
ክህለ /khla/	He able
ኖለወ /nolawa/	He kept

Adverbs

Adverb is a word that is used to change, modify or qualify several types of words like verb. There are six types of adverbs in Geez language. These are Adverbs of time, frequency, place, manner, reason, and question. Adverbs of time tells the time when it is used on the sentence. For example, ጌሠም (tomorow), ትማልም (yesterday). Adverbs of frequency describes how many times an event happens, ኩለኄ (always), በበጊዜሁ (evreytime). Adverbs of place gives us an information on a place, ሀየ (hear), ጥቃ (near). Adverbs of manner describes how one thing takes place, እሙነ (Ofcourse), ክሡተ (clearly). Adverbs of reason presents the reason on the occurrence a thing, አምጣነ (As much as), በእንተ (In Care Of). Finally, adverbs of question stand to raise a question, እፎ (How), ምንት (What).

Prepositions

Prepositions lied on nouns or pronouns to connect the people, objects, time and locations of a sentence.

Table2. 7 some preposition of Geez language [13]

Geez	English
ዲበ፣ላዕለ፣መልዕልተ	On, above
መትሕተ፣ታሕተ	Under
ውስተ፣ውሰጤ፣ማእከለ	In/inside/ in the middle
ቅድመ፣ድኅረ	Before/ after
ኃበ፣መንገለ	To
ህየንተ፣በእንተ፣በይነ	About
እም፣እምነ	From

Conjunctions

Conjunctions are words to connect clauses or sentences or to coordinate words in the same clause.

Table2. 8 Some conjunction of Geez language [13]

Geez	English
ከመ	As … As
አምሳለ	
በዘ	
በእንተ	About
ህየንተ	
በይነ	
እንበይነ	
አመ	In…time
ሶበ	
ጊዜ	
አምጣነ	And/Due to

እስመ	
አኮኑ	

2.2.2 **Geez Morphology**

Morphology is the study of word formation. It tries to discover the rules that govern the formation of words from the smaller meaning bearing units, morphemes, in a language [8]. Morpheme is the building block from which a word is made. It could not be broken down further into meaningful part [8]. Ge'ez has not been studied linguistically, especially with respect to computational morphology. In this topic, the researcher describes the morphology of Geez word classes according to their derivation and inflection characteristics.

Geez nouns Derivation: Geez nouns can be derived from verbal roots by infixing vowels between consonants, stems by prefixing or suffixing bound morphemes, stem-like verbs by suffixing the bound morpheme, nouns by suffixing bound morphemes and compound words Dillmann [18], [19].

Geez Noun Inflection: Geez nouns Inflection: In Geez language, nouns can be inflected for gender, number, definiteness, and case. In Geez language, there are three genders: masculine, feminine, and common. The masculine has no special termination. However, the feminine has the termination ት (t) e.g., ዐመት (amet) /year, ቄላት (qwelat) /valley or sometimes it has no ending. There are two numbers: singular and plural. The plural formed in two ways [20]:

• **Strong plural**: formed by means of the termination ን (n) for the masculine, e. g. ነግድ(negd) traveler, p. ነግዳን (negdan) and ት (t) for the female, e.g., ዐመት(amet)/servant, and for plural ዐመታት(ametat)/ servants.

• **Broken plural**: formed by vocal modification and by prefixes and suffixes. E.g., ቀተል (qetel)/killed, ቀትል(qetl), ቅትል (qtl) became አቅታል (aqtal). In Geez nouns, there are four cases: nominative, vocative, genitive, and accusative. The nominative and genitive have no distinct termination. The vocative is the same as the nominative, or is the nominative with a prefixed or suffixed አ (a), e.g. አግብር (agbr) /servant. The accusative is differing from the nominative because unlike nominative, it is formed by vocal change. E.g., the nominative ሙስል(mesl) become as ምስለ

(msle)/image as accusative [21]. In general, Table 2.9 shows summarized form of noun Inflection by using single word Camel.

Table2. 9 Example of Morphological change for Geez noun[21]*: □ □ □*

No	Geez	Transliteration	Meaning
1	ግመል	Gmel	Camel
2	ግመሉ	Gmelu	him Camel /the Camel
3	ግመላ	Gmela	her Camel
4	አግማል	Agmal	Camels
5	ለአግማሊሁ	le-agmal-ihu	for him Camels
6	ለአግማሊከ	le-agmal-ike	for your Camels
7	ለአግማሊከኒ	le-agmal-ike-ni	also for your Camels
8	በአግማሊሆን	be-agmal-ihon	By their Camels
9	በአግማል	be-agmal	By Camels

10	አግማለ	agmal-e	The accusative case of Camels for him Camels
11	አግማሊሁ	agmal-ihu	The accusative case of Camels for him Camels
12	እምአግማለ	em-agmal-e	From Camels
13	ወለአግማሊከኒ	we-le-agmal-ike-ni	and also for your Camels
14	ለአግማሊሃ	le-agmal-iha	for her Camels
15	አግማሊየ	agmal-iye	My Camels
16	ኢአግማሊሆን	i-agmal-ihon	not their Camels

Geez Verb Derivation: Geez verbal stems (from which various forms of verbs are formed) can be derived from verbal roots by affixing vowels, repeating penultimate consonants and affixing vowels. Can be also derived from verbal stems by affixing morphemes.

Geez Verb inflection: Geez verbs are marked for person, gender, number, case, tense/aspect and mood. The researcher use example of derivation and inflection of root verb □ □ □ (fqd) that is adapted from the work of Desta [22]

Table2. 10 Sample Geez verbs derived from [21]ፍቅድ

No	Word	Meaning	Gender	Person	Number	Time
1	ፈቀደ	feqede	He liked			
2	ፈቀደኒ	feqedeni	He liked			
3	ፈቀደነ	feqedene	He liked us			
4	ፈቀደከ	feqedeke	He liked you(2psm)			
5	ፈቀደክሙ	feqedekmu	He liked you(2ppm)			
6	ፈቀደኪ	feqedeki	He liked you(2psf)			

7	ፈቀደክን	feqedekn	He liked you(2ppf)			
8	ፈቀዶ	feqedo	He liked him			
9	ፈቀዶሙ	feqedomu	He liked them(3ppm)			
10	ፈቀዳ	Feqeda	He liked her			
11	ፈቀዶን	feqedon	He liked them(3ppf)			
12	አስተፋቀደ	Estefaqede	He caused others to like each other			
13	አስተፋቀዳ	astefaqda	He caused her to be liked with			
14	አፍቀደ	afqede	He caused somebody to be liked			
15	አፍቀደኒ	Afqedeni	He caused me to be liked			
16	አፍቀደነ	afqedne	He caused us to be liked			
17	አፍቀደከ	afqedeke	He caused you(2psm) to be liked			
18	አፍቀደክሙ	afqedekmu	He caused you(2ppm) to be liked			
19	አፍቀደኪ	afqedeki	He caused you(2psf) to be liked			
20	አፍቀደክን	afqedekn	He caused you(2ppf) to be liked			
21	አፍቀዶ	afqedo	He caused him to be liked			
22	አፍቀዶሙ	afqedomu	He caused them(3ppm) to be liked			
23	አስተፋቀዶሙ	astefaqedomu	He caused them(3ppm) to like each			
24	ተፈቅደ	tefeqde	He is liked by			
25	ተፋቀደ	tefaqede	He is liked with somebody			

26	አፍቀዳ	afqeda	He caused her to be liked			
27	አፍቀዶን	afqedon	He caused them(3ppf) to be liked			
28	አስተፋቀደኒ	astefaqedeni	He caused me to like with others			
29	አስተፋቀደነ	astefaqedene	He caused us to like with others			
30	አስተፋቀደከ	astefaqedeke	He caused you(2psm) to like with			
31	አስተፋቀደክሙ	astefaqedekmu	He caused you(2ppm) to like with			
32	አስተፋቀደኪ	astefaqedeki	He caused you(2psf) to like with			
33	አስተፋቀደክን	astefaqedekn	He caused you(2ppf) to like with			
34	አስተፋቀዶ	Astefaqedo	He caused him to like with others			
35	አስተፋቀዶን	astefaqedon	He caused them(3ppf) to like with			

Geez Adjectives Derivation: In Geez, like verbs and nouns, adjectives also have derivational and inflectional morphology. Geez adjectives can be derived from verbal roots by infixing vowels between consonants, nouns by suffixing bound morphemes, stems by suffixing bound morphemes and compound words of nouns and adjectives. Geez Adjectives can be marked for number by affixation of morphemes or repetition of consonants, definiteness by affixation of morphemes or vowels based on number, gender, and/or ending of the adjective, gender by affixation of the morpheme and object case by affixation of the morpheme [23].

2.2.3 **Geez Grammatical structure**

Grammar is commonly understood to be a language's set of rules, usually taught in relation to writing and proper usage[4]. It can be used to describe a system of guidelines designed to regulate specific facets of language use, such as the formation of meaningful words and longer sentence structures[24]. It is usually a limitation on the order in which words must be placed to form a sentence, phrase, or paragraph; these limitations are known as syntactic and semantic principles. A language's syntax explains how words can be paired together to create phrases, sentences, and paragraphs. A language's semantics deals with how words are changed in terms of gender and time number.

The grammar syntax and morphology differ in each language. Conversely, this means that, if there is a difference in syntax or morphology in two languages, these languages are not the same a language can be distinguished from another by its grammar; these languages need to be considered separately[24]. In this section, the researcher describes the phrases and sentences structure of Geez languages.

Geez Sentence

Geez language is free in word order of the sentence. It may take subject- verb-object (SVO), object-verb- subject (OVS) or verb-subject- object (VSO). Most of the time the order of word class depends on the type of verb that used. Like other Semitic languages, only Biblical Geez had two distinct sets of verbs form /aspects, called the imperfect and the perfect. The imperfect forms were used frequently for most purposes, while the perfect forms were used only occasionally for a few purposes. Importantly, the imperfect forms normally required VSO word order, the ordinary word order of the language is the perfect, a marked form, usually required a marked word order, SVO. Here are two examples from the book of the Gospel [11]:

• ወባረከ እግዚአብሔር ለአብርሃም በኵሉ /We Bareko Egziabher leAbriham Bekulu/ and the LORD had blessed Abraham in all things. Genesis 24:1

• እግዚአብሔር ባረከ ለአብርሃም ጥቀ /Egziabher Bareko leAbriham Tiqe/ the LORD blessed to Abraham greatly. Genesis 24:35

As in any other language, Geez has two chief members of the sentence namely subject and predicate[25], may be extended into larger groups of words. The Subject Every sentence, which is not imperfect, must contain a subject. Such subject is usually a substantive or a pronoun representing a substantive; but it may also be an adjective if it is invested with the force of a Substantive, or even an Adverb, when through the stimulus of speech, the adverb is raised to the position of a noun-substantive. An entire sentence even may take the place of subject, particularly a relative or a conditional sentence, example. የአክለኒ፡ዘረከብኩ፡ሞገሰ" it is enough for me that I have found favor", just as in other languages. The predicate of a sentence is usually a verb or an adjective (or participle). Certain adjectives, when used as predicates, are in all cases, or at least in certain cases, supplemented by a suffix. Those adjectives and participles also, which are formed by periphrasis with the relative pronoun.

The Subject

Every complete sentence must contain a subject. This subject is usually a noun or a pronoun representing a noun, but it may also be an adjective if it functions as a noun, or even an adverb, when the context of speech elevates it to the role of a noun. An entire sentence even may take the place of subject, particularly a relative or a conditional sentence, example. የአክለኒ፡ዘረከብኩ፡ሞገሰ"it is enough for me that I have found favor", just as in other languages.

2.2.4 Grammar Errors

When people are writing, they make mistakes, mostly the syntactic rules of a language, such as feature agreement, order or choice of constituents in a phrase or sentence, thus concerning a wider context than a single word. Those grammatically error writing are also happen in Geez language users. Grammar error may occur due to the subject's insufficient knowledge of the language rules or the written language norms that deviate from the already acquired (spoken) grammatical knowledge have to be learned and even can make grammar errors when writing on a computer due to rewriting or rearranging text [4], [26].

Common Geez Grammar Errors

As any other language the grammar of Geez happens, the words in a sentence can disagree according to person, gender, cases, tenses and number. Other agreement errors appear between the subject and the verb, which can involve long distance dependencies. In this section, the researcher take a look at the most frequent error types in Geez according to Geez textbooks. Frequent

agreement errors in Geez are located within the words in a text can disagree subject and verb, verb and object, adjective and noun, adverb and verb, noun and modifier, word sequence and others in gender, number and persons. Other agreement errors appear between the subject and the predicative, which can involve long distance dependencies.

Subject-Verb Agreement Errors: Is the most common errors with subject-verb agreement are to do with number, person and gender in Geez language. For instance, ነሰሐ፡እምነ፡ኀጢአተን።nshe emne hatiyaten/ He repented for their sin. In this example, the subject እሱ/esu/he which is taken from the object እምነ and the verb is ኀጢአተን/hatiyaten/ sin. The singular subject እሱ/esu/ is disagreeing with the plural verb ኀጢአተን, this make the grammar rule is incorrect. Hence, correct subject-verb agreement has same number that means singular subject takes singular verb. The correct grammar of the sentences is ነሰሐ፡እምነ፡ኀጢአቱ: /neseḥā:imine: ḫāt'ī'ātu / He repented for his sin.

Object-verb agreement error: it is the same as the subject-verb agreement error but in this case, the object and the verb of the sentence is expected to agree in number, gender, and person.

Noun-modifier agreement error: Geez nouns are modified by adjectives, determiners, quantifiers, and others. Adjectives in Geez inflect according to the gender, person and number of the head noun. For instance, in the ቀያህ ወ ል ድ /qeyah weld/ red boy, text the adjective ቀያህ/qeyah(red) is third person singular feminine; and The noun ወ ል ድ / wodi (Boy) is third person singular masculine. Both the Adjective and the noun agree in number and person. Whereas, disagree in gender maker. In order to correct the grammatical of this text either the adjective could be changed to male or the noun must be changed to fame gender. Therefore, the correct grammar is either; ቀይህወ ል ድ /qeyh weld or ቀያህወ ለ ት / q e y a h w e l e t . The quantifier in noun phrase disagrees with number agreement. For instance, ሰለስተፈረሥ/selsete feres/three horse in this phrase the quantifier ሰለስተ indicates many/plural whereas, the noun ፈረሥ is singular. The number-noun agreement rule in Geez is singular number+ singular noun, or plural number + plural noun, based on this rule the text is grammatically incorrect.

Adverb and Verb Agreement: Geez Adverb usually modifies the first verb that comes next to it in time, place, circumstance etc. The time adverbs describe the time at which an event takes place.

These adverbs may show a specific time at which a given action takes place or its duration. Geez verbs indicate the time at which action takes place in relation with the adverbs.

Word sequence agreement error: In Geez grammar rules all the verbs could be placed at the middle of a statement or at the end of a statement, unlike in English where verbs are placed in the middle of a statement ([Subject] [Verb] [Object]). The two commonly use cases are placing the verb at the middle ([Subject] [Verb] [Object]) as in the example (ጳዉሎስ ነምር ቀተለ) or placing it at the end of the sentence ([Subject] [Object] [Verb]) as in the example (ጳዉሎስ ቀተለ ነምር).

2.3 Approaches to the Development of Grammar Checker

2.3.1 Grammar Checker System

Natural language processing (NLP) can be defined as the automatic (or semi-automatic) processing of human language[27]. It is narrowly used, as it is a large and multidisciplinary field, which includes machine translation, question answering, grammar checker and others for processing human languages. It is often studied to solve the problems of language processing in contrast with part of speech, morphology and other components of the human languages. Based on the linguistic characteristics several NLP studies are performed such as the studies on grammar checker[27], [19]. The purposes of those studies on grammar checker are for examining incorrect texts against the syntax and semantic structure of any language. A grammar checker system is a computer-based formal system for describing the rules and syntax allowable in the language to detect a construction error. The goal of a grammar checker is to check a sentence or text for its grammatical correctness, which helps to prepare an important document. Now, a part of most word processing programs flags what they perceive as stylistic, grammatical, or mechanical problems in a document by highlighting or underlining them, and upon request comment on, explains, and sometimes suggest corrections for each problem[15]. A grammar checker system includes components of tokenization, POS, morphological analysis, and grammar checker [5]. In those systems, ways of checking the grammars of a given sentence is performed by several techniques. However, it includes the tasks such as identifying the words of a given sentence and using a program by capturing and analyzing the generalization of a given word and classifies those words into their POS or other grammatical categories[25], , [6], [7].

The tokenization component is used to break input texts from an input file into sentences [4]. The tokenized sentences are further tokenized into words, symbols, or other meaningful elements

called tokens. Those tokens are used for grammar checkers to identify the word order and agreement rules in the given text. For instances, the text ትማልምአሠነ:የቤቶ።/tmal aseneye be^t^o (Yesterday *decorated his house*.) is tokenized into tokens („ትማልም “, „አሠነየ“, „ቤቶ“) [8]. The morphological analyzer component concerns the structure of words. Words are assumed to be made up of morphemes, which are the minimal information carrying unit. Thus, the grammar checker takes morphologically analyzed word gives as input to detect the agreement errors in gender, person, number, time, and etc.

The part of speech tagger component assigns each word and punctuation mark with appropriate parts of speech tag[28]. This component is treated as a main component of grammar checker[15]. Since it gives large amount of information about a word and its neighbor such as word categories like Noun, Verb, Adjective, and others. This is useful in grammar checker system because of words are ambiguous having of more than one possible part of speech and the goal is to find the correct tag for the situation. The most common grammar checker system development approaches are rule-based, statistical, syntax and hybrid approaches [9].

The rule-based approach is performed by constructing error detection rules manually for the given language. These rules are then used to find errors in the text that has already been analyzed. These rules often contain suggestions on how to correct the error found in the text [10]. According to[28], it consists of sets of rules used to analyze the matching of a text with in specified pattern in which POS tagged is performed. Generally, rule based approach is used to detects an error of input texts in which the text is considered as having POS tagged by matching against a set of rules previously constructed based on the syntax and morphology grammatical knowledge of the specific language. The grammar checker uses these rules to identify errors in each input text by comparing them against the rules. If any rule matches, the text is considered grammatically correct; otherwise, it is considered grammatically incorrect.

The rule-based approach has some characteristics. It has a strict sense of well-formedness in mind, imposes linguistic constraints to satisfy well formedness, allows the use of heuristics, and relies on hand-constructed rules that are to be acquired from language specialists rather than automatically trained from data[29].

The advantage of this approach is that the grammar checker requires no need of training data, good quality based on language having small amount of electronic data and low coverage of

linguistic and it is easy to incorporate domain knowledge into the linguistic knowledge which provides highly accurate results[30].

However, this approach also has some disadvantages. As it created manually, to build several hundreds of rules it could be time consuming and requires a great knowledge of a specific language.

2.3.2 **Statistical Approach**

A statistical approach does need a tagged corpus to train the language model (LM) instead of handcrafted grammatical rules. Word sequences which occur often in the corpus can be considered correct in other texts; too, uncommon sequences might be error [31]. In statistical approach, all the required statistical information is extracted from a set of corpora and the text will be compared against the extracted information. The statistics extracted by language modeling techniques are commonly known as n-gram statistics [31]. The problem of this approach that is, performance of a system depends on the amount of the corpus. Specifically, the performance is higher if the amount of the corpus size is large, else low in performance. In general, a statistical approach uses N-gram models in which a language model of the probability of a text, conditioned on some number of previous documents. In this model, the document is viewed as a sequence of n texts and probabilities. The probability of a text in a language is a sub sequence of n-neighbored tokens in a sentence, where n defines the number of tokens. The N-Gram probabilities come from a training corpus that is used by the system to detect a sentence which is grammatically incorrect. After the entire text of a document, it is scanned for errors and the result of a scan is a list of all possible errors in the text that measure the probability of the text using n-gram analysis. The possible error examines against the simply setting the threshold high enough for an error should be sufficient to satisfy this requirement. However, a grammar checker with a threshold that is too high will miss a large number of legitimate errors[31].

The fundamental equation of calculating the probability can be written as: P (w/h) where, w is the input text or word, h is the training data or corpus, and P (w/h) is the function of calculating the existence probabilities of w in h. Then the word w is classified as grammatically correct where the probability result meets some threshold otherwise the word is verified as incorrect[4], [5].

2.3.3 Syntax-based Approach

In this approach, a sentence is completely parsed into a tree like structure to check the grammatical correctness. The sentence is said to be correct if the parsing succeeds and incorrect if parsing does not succeed. The advantage of syntax-based approach grammar checker approach is it is always complete, that means any incorrect sentence will be detected, no matter how unknown the error is. Unfortunately, the grammar checker will only be able to detect that there is some error, but it will not be able to suggest that what exactly the error is. There is also a major problem of syntax based approach; it requires a complete grammar that covers every type of text needed to be checked. There are many grammar checkers but there is no robust grammar checker or a parser available today. Usually, parsers give more than one than result even for correct sentences because they suffer from natural language ambiguity[8].

2.3.4 Hybrid Approach

As its name implies, this approach takes the combination of either rule-based approach and statistical based approach or syntax based and rule based by taking the advantages from both approaches to improve the performance of the system[6]. A grammar checker system based on a hybrid approach such as statistical and rule-based approaches gives better results than the corresponding uncombined approaches[25].

2.4 Grammar Checker Evaluation

An evaluation of system results by comparing errors and correct text against a ground truth is common in various NLP applications such as grammar checkers, question answering systems, and others. Many NLP systems are evaluated using three key performance metrics: precision, recall, and F-score. Precision refers to the proportion of correctly identified errors among all the errors detected by the system, while recall measures the proportion of actual errors that the system was able to detect. In other words, precision evaluates accuracy, and recall evaluates coverage. The F-score combines both precision and recall into a single metric by calculating their harmonic mean. For instance, in tasks like Named Entity Recognition (NER), each tag may contain multiple components such as "type" and "extent." Systems being tested generate hypotheses that are compared with reference tags, and each matching component is evaluated as correct, substituted (wrong), deleted (missed), or inserted (extra). Although alignment and correctness decisions are

crucial, this explanation focuses only on computing performance once alignment is already established.

To support this evaluation, the following symbols are defined;

N = total number of slots in the reference,

M = total number of slots in the hypothesis,

C = number of correct slots,

S = number of substitutions (incorrect slots),

D = number of deletions (missing slots),

I = number of insertions (spurious slots).

From these, we derive:

$$N = C + S + D \quad (1)$$

$$M = C + S + I \quad (2)$$

Precision and recall are then calculated as;

$$\text{Precision(p)} = \frac{C}{(C+S+I} \quad (3)$$

$$\text{Recall(R)} = \frac{C}{C+S+D} \quad (4)$$

Precision accounts for substitution and insertion errors, while recall accounts for substitution and deletion errors. Systems can sometimes manipulate these metrics for instance, returning every possible output gives perfect recall but poor precision. Conversely, returning only one correct result yields high precision but poor recall. Thus, to fairly evaluate system performance, the F-score balances both metrics and is calculated as;

$$\text{F-score} = \frac{(2*P*R)}{(P+R)} \quad (5)$$

2.5 Related Work

As a job of natural language processing (NLP), grammar checkers have been the subject of several research employing diverse methods for different languages. This chapter covers past grammar checker projects for non-Ethiopian languages including English, Swedish, Bangla, Nepali, and Punjabi, as well as Ethiopian languages like Amharic and Afaan-Oromo. This section also examines the many approaches to grammar checker development, including rule-based, statistical, and hybrid techniques.

2.5.1 Development of Grammar Checker for Ethiopian Languages

Aynadis Temesgen and Yarega Assabie [14] have conducted research, which proposed to identify and detect grammar errors for Amharic text sentences using two different approaches of grammar checker system development, which are statistical and rule-based approach. The rule based Amharic grammar checker aimed to identify errors in simple sentences using hand-written grammar rule and consists of three modules such as Sentence Splitter, Morphological Analyzer, and Grammar Relation finder. The functionality of those modules is the sentence splitter module that is used to split input text into sentences and the sentence into words. The morphological analyzer module is performed to assign linguistic meanings to each word. The grammatical relation finder module is performed to assign grammatical relation between words in a sentence, and the grammar rule checker module is used to find match grammar relation of patterns extracted in the grammatical relation finder module against the hand-written rules [32]. The Statistical Amharic Grammar Checker (STAMGRAM) is used to identify grammar errors in simple and complex Amharic sentences using N-gram probability result of Trigram and Bigram. The performance of both the rule-based Amharic grammar checker and STAMGRAM approaches is evaluated using self-made corpus that contains grammatical errors. Based on the results that the researchers got, they concluded the rule-based Amharic grammar checker has performed 92.45% of precision, and 94.23% of recall. The STAMGRAM performs a precision of 59.72%, and recall 82.69% in Bigram and a precision of 67.14%, and recall of 90.38% Trigram. The researcher stated that the system shows a false alarm due to incomplete grammatical rules and quality of the corpus and recommends that the combination of the rule-based and statistical approach by mixing the good qualities of the two approaches to detect high-level grammatical errors can increase the performance of a grammar checker system.

Debela Tesfaye [26] has conducted a rule-based grammar checker for Afaan-Oromo language to determine grammar error in Afaan-Oromo texts by finding match grammatical rules designed manually. The proposed system in this study has five modules, namely, Tokenization, POST, stemmer, grammar relation finder and suggestion creation modules. The performance of this system is tested against manually prepared real world Afaan-Oromo texts based on recall and precision, and the result is 80% and 88.89% respectively. The researcher stated that some correct texts are identified as incorrect, and incorrect texts as correct. Thus, false flags are due to compound, complex and compound complex sentences because most of the grammar rules are constructed for simple sentences.

2.5.2 Development of Grammar Checker for non-Ethiopian languages

Xing and Wang [23] adapt a hybrid of rule based and statistical approaches of grammar checker system to detect English error texts. The study was developed to find and correct error types of article or determiner, noun number, preposition, verb form and subject-verb agreement errors. In this study, the researches constructs manually rules and develop language model using N-gram based on the Google corpus to detect the grammatical errors in English. The system has five components such as Vform, SVA, ArtOrDet, Nn, and Prep. In the ArtOrDet module the system detects error regarding to article or determiner by select the best candidate from a confusion set of possible article choices {a, the, an, ø}. In the Nn module the system detect the noun errors in the text whether it is singular or plural. In the Prep module, the system corrects system corrects errors for the five most frequently prepositions which are in, for, to, of and on. In the Vform module the system determines and corrects the correct form of a verb in the English texts. In the SVA module the system detects subject-verb agreement errors. The performance of the system based on the precession, Recall, and F-Score is 0.3712, 0.2366, and 0.2890 respectively.

Alam and UzZaman [2] developed a statistical Grammar Checking System for Bangla and English language. The study is proposed by considering the n-gram based analysis of words and POS tags to decide whether the sentence is grammatically correct or not. The proposed grammar checker works first by assign tag for each word of a sentence, next uses n-gram analysis (LM) to determine the probability of the tag sequence. Then if the probability is above some threshold then the sentence is considered grammatically correct otherwise incorrect. The performance of

this system tested using manually and automatically tagged corpus. Using manually tagged English corpus, the performance of the grammar checker for English is 63% (detected 545 sentences as correct, out of 866 correct sentences). Using manually tagged Bangla corpus, the grammar checker's performance is 53.7% (detected 203 sentences out of 378 correct sentences). The performance of the grammar checker also tested using 34 automatically tagged sentences for Bangla. Then, the grammar checker produced result with lower performance, which is 38% correct result. The researchers described limitation in their approach, with possible solutions of future work the grammar checker can be a hybrid system combining both statistical and rule based approach to increase the performance [29].

Rule based grammar checker for Punjabi language system [27] identifies grammatical errors in Punjabi texts such as modifier and noun agreement, subject and verb agreement, noun and adjective, order of modifier of noun in a noun phrase, order of verb in a verb phrase and the like. To detect the errors the system passes through few steps or phases initially, pre-processing task is done on the input text, which is tokenization, morphological analysis, rule-based part of speech tagging, chucking and finally using the grammatical error checking rule, grammatical errors internal to the phrases and the sentences will be identified and correction suggested. The evaluation of the grammar checker shows precision of 76.79%, recall of 87.08%, and F-measure of 81.61%. The researchers stated that the system generated some false alarms for complex and compound sentences. To reduce these false alarms they indicated to work in the future by combining other approaches.

Domeij and Knutsson [28] investigated the achievement of carefully constructed grammatically error detection rules for Swedish texts combining of probabilistic and rule-based techniques. The error detection rules are optimized using statistics of part of speech bigrams and words in a way that each rule needs to be checked as seldom as possible. The researchers proposed components to detect the grammatically incorrect Swedish texts such as Tagging and lexicon, Error rules, Help rules and erroneously split compounds components. The authors are still working with optimizing the system and improving the error rules. The preliminary test of the system is conducted with the error rules show that a recall rate above 50 percent and a precision rate above 90 percent for agreement errors and erroneously split compounds. In addition the authors stated that, they compare with other related studies and they got better results than systems that use rule-based.

Bal and Shresth [29] describes the architectural and system design of the Nepali Grammar Checker, which is in due course of research and development. The development uses a rule-based approach with the Grammar Checker consisting of independent modules. These modules then in turn serve as a pipeline for the overall integrated system. The Grammar Checker aims to check the grammatical errors such as nominal and verbal agreement, parts of speech inflections, phrase and clause structure and the different categories of sentence patterns for Nepali. Out of the modules proposed in the architectural and system design of the Nepali Grammar Checker, the research and development of the first two modules, namely the tokenizer and the morphological analyzer has been partially completed. A prototype of the two modules is also ready and in the process of being tested and added complexity hand ling features. As mentioned earlier, the proposed architectural and system design of the Nepali Grammar Checker is subjective to changes as further findings of the research work come out in future.

CHAPTER THREE: DESIGN of GEEZ GRAMMAR CHECKER

3.1 Introduction

Through the use of statistical word sequence analysis and manually created grammatical rules, the researcher created a hybrid Geez grammar checker system that can identify grammatical errors in Geez sentences. In the sections below, the system architecture and a detailed description of the modules and components that make up the system were covered.

3.2 System Architecture

This defines the system architecture by outlining the functions of the key components and the specific techniques used to accomplish their tasks. The proposed Geez grammar checking architecture consists of a series of cooperating modules that work to analyze and validate the grammatical correctness of input text. The system is a mixture of rule-based and statistical methods and utilizes tools such as a Geez corpus, word tag dictionary, and grammar rule list. The architecture of the system, in brief, consists of the following elements. In Figure 3.1.

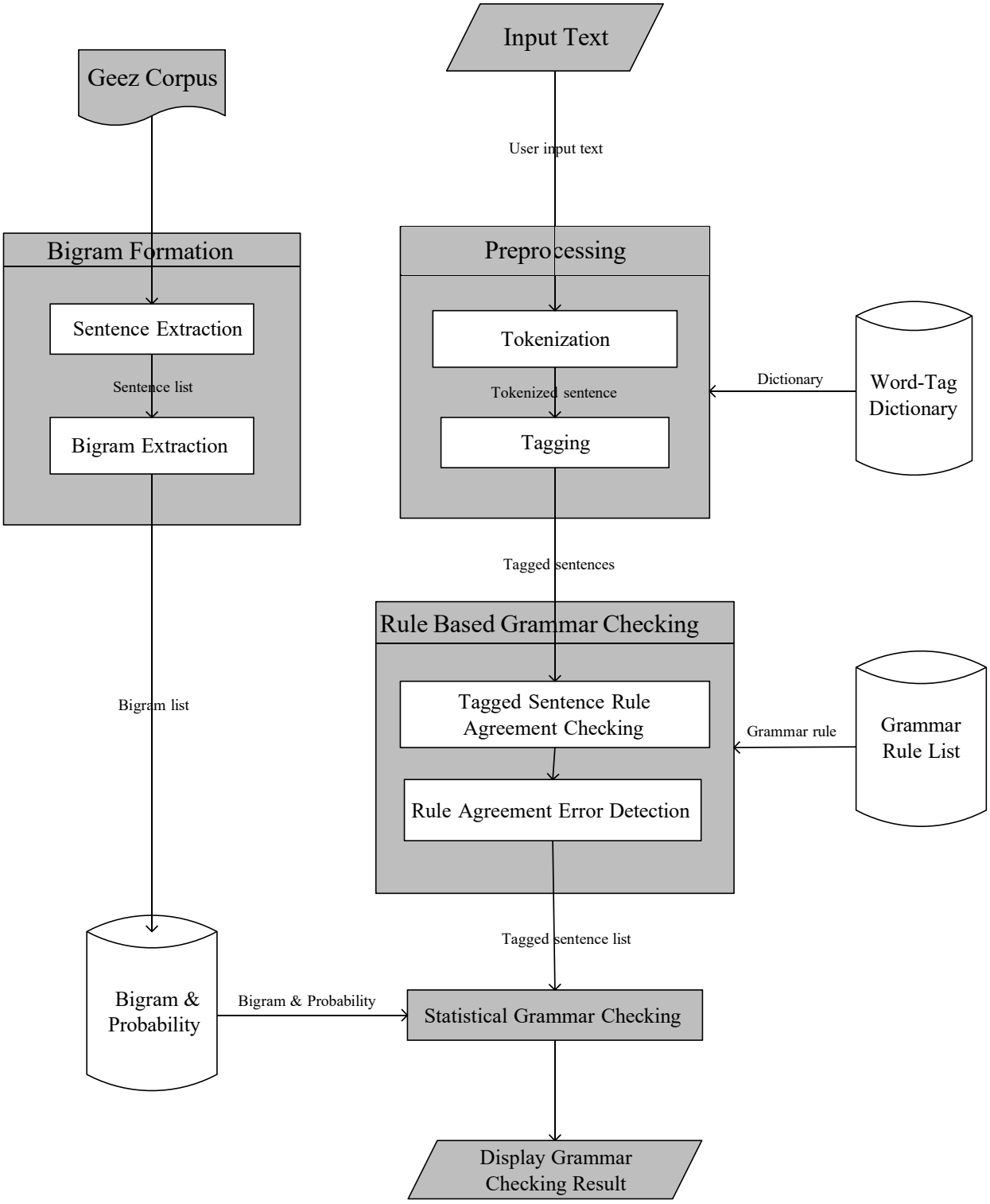


Figure 3. 1 Hybrid Geez Grammar Checker System Architecture (source own)

3.3 Bigram Formation

A statistical language model was developed using the bigram generation module, which was based on the typical word sequence in Geez. In order to do this, successive word pairs, or bigrams, were extracted and examined from the Geez corpus. As a result, the system was able to recognize typical word combination patterns in coherent sentences, which was helpful in identifying likely grammatical errors in the text that was downloaded. Since sentence extraction and bigram extraction are the two basic components used here.

3.3.1 Sentence Extraction

The bigram formation module's Sentence Extraction component was created to separate the input text into distinct sentences based only on the existence of predefined Geez sentence-ending symbols. The component used the following Unicode characters to detect sentence boundaries: the Ethiopian full stop (።), the Ethiopian question mark (፧), the Ethiopian semicolon (፤), the question mark (?) and the exclamation point (!). The Geez corpus, which was mainly made up of spiritual books and the Gospel, used these symbols as delimiters to indicate the end of each sentence. The relevant text passages were taken out and handled as full sentences as soon as these symbols were found. This simple technique guaranteed accurate sentence segmentation, which was a prerequisite for the system's later bigram extraction procedures.

To process the following Geez text, for example, the Sentence Extraction component used only the predefined sentence-ending symbols ። (U+1362), ፤ (U+1364), and the question mark (?) to determine sentence boundaries. These punctuation marks were the only way the extraction could distinguish between sentences. The input text: "ጸሎተ ሃይማኖት ነአምን በአሐዱ አምላክ እግዚአብሔር አብ አኃዜ ኵሉ። ገባሬ ሰማያት ወምድር ዘያስተርኢ ወዘኢያስተርኢ። ወነአምን በአሐዱ እግዚእ ኢየሱስ ክርስቶስ ወልደ አብ ዋሕድ ዘህልው ምስሌሁ እምቅድመ ይትፈጠር ዓለም። ብርሃን ዘእምብርሃን አምላክ ዘእምአምላክ ዘበአማን። ዘተወልደ ወአኮ ዘተገብረ ዘዕሩይ ምስለ አብ በመለኮቱ።" was segmented into the following sentences:

1. □□□ □□□□□□ □□□□ □□□□ □□□□ □□□□□□□□ □□ □□□ □□□
2. □□□ □□□□ □□□□ □□□□□□□ □□□□□□□□□□

3. □
□ □ □ □ □ □ □ □ □ □

4. □

5. □

The text that came before each sentence was saved as a complete sentence, and each sentence was extracted when the delimiter '□' was found. This procedure made sure that input texts were consistently segmented, which is necessary for precise bigram formation in the system's later phases.

3.3.2 Bigram Extraction

Bigram pairs of consecutive words were created from the segmented sentences in the Geez corpus by the Bigram Extraction component. This procedure provided the basis for building a statistical language model that represented common word patterns in Geez texts that were grammatically correct. Following the Sentence Extraction component's segmentation of the input text using predefined sentence-ending symbols, each tokenized sentence was broken down into individual words using word delimiter characters like white space (Unicode U+0020), the Ethiopian Comma (፣), the Ethiopian Colon (፥), and the Ethiopian Word space (፡), which is frequently used in Geez to denote word boundaries. Next, iterating through each sentence, the Bigram Extraction module extracted every pair of adjacent words in the order that they appeared.

For instance, consider the following sentence extracted from the above Geez corpus: "□ □" the following [illegible] were generated: [(□ □ □, □ □ □ □ □), (□ □ □ □ □, □ □ □ □), (□ □ □ □, □ □ □ □), (□ □ □ □, □ □ □ □), (□ □ □ □, □ □ □ □ □ □ □ □), (□ □ □ □ □ □ □ □, □ □), (□ □, □ □ □), (□ □ □, □ □)]. A statistical model of Geez language usage based on frequency was constructed by counting and storing these bigrams. Later, during grammar checking, this model was applied to determine whether the word sequences in a given sentence followed typical patterns found in the corpus. Possible grammatical errors may be indicated by uncommon or absent bigrams. Using this bigram-based method made it possible to capture structural regularities in Geez in a lightweight yet efficient manner. This is especially useful when

processing a low-resource language where deep linguistic tools might not be easily accessible.

3.4 Bigram and Probability

A relational database schema specifically designed for bigram statistics was created by the researcher to efficiently handle the storing and retrieval of bigrams and the probability values that go along with them. Each bigram created during the system's early processing stages had its probability data stored in a dedicated table. Based on the bigrams' frequency of occurrence within a given text corpus, their probability is computed. The formula specifically calculates the likelihood that one word will appear after another.

$$P(w_2 \mid w_1) = \frac{C(w_1, w_2)}{C(w_1)}$$

Where:

- C (w_1, w2) denotes the frequency count of the bigram (w_1, w_2) in the corpus
- C (w_1) is the total count of the word w_1 appearing as the first word in any bigram.

The conditional probability, P (w2|w1), indicates the likelihood that, given the observed data, the word w2 will come after w1. For a variety of natural language processing tasks, the model captures the contextual relationships between words by calculating these probabilities across all word pairs in the corpus. After being determined, these odds are kept in a specific database table for quick access and future use in predictive and language modeling applications. Persistent storage that can be updated incrementally and scalability to effectively manage large datasets are two benefits of storing bigram data in a database. Exploring and querying bigram statistics is made possible by the adaptive querying capabilities of SQL, and building more complex language models is made possible by integration with other language data, such as part-of-speech tags or higher order n-grams. The use of pre-computed bigram probabilities, which are calculated in advance, is the other advantage.

3.5 Preprocessing

The first module of the rule-based grammar checker, which gets the Geez text ready for grammatical analysis, is explained in this section. Tokenization and tagging are the two parts of the module. The input text is divided into distinct tokens at the sentence level in the tokenization component, enabling organized and controllable language processing. Each token is then given the

proper linguistic tags in the tagging component by utilizing a predefined tagger dictionary. This procedure guarantees precise annotation for use in subsequent grammar check phases.

3.5.1 **Tokenization**

The preprocessing module's segmentation of the user-inputted text into individual sentences is followed by tokenization of each sentence into smaller linguistic units, like words. Using punctuation marks like the Ethiopian full stop (።), Ethiopian question mark (፧), Ethiopian semicolon (፤), question mark (?) and exclamation mark (!) is how sentence segmentation is done. White space (Unicode U+0020), the Ethiopian Comma (፣), the Ethiopian Colon (፥), and the Ethiopian Word space (፡) are examples of word delimiter characters that are used to identify tokens within each sentence. After reading the Geez input text, the process divides it into distinct sentences. Following tokenization of each sentence into its component parts, sentence markers ([and]) are appended to indicate sentence boundaries. For instance, the text input. "□□□ □□□□□ □□□□ □□□□ □□□□ □□□□□□□ □□ □□□ □□□ □□□ □□□□ □□□□

□□□□□□ □□□□□□□□□ □□□□□ □□□□ □□□□ □□□□ □□□□□ □□□ □□ □□□

□□□□ □□□□ □□□□□ □□□□□ □□□□ □□□□ □□□□□□□ □□□□ □□□□□

□□□□□□ □□□□□ □□□ □□□□□ □□□□ □□□ □□ □□□□□□" is parsed by splitting it into tokens and adding sentence boundary markers. The resulting token list is:

['□□□','□□□□□','□□□□','□□□□','□□□□','□□□□□□□','□□',
'□□□','□□'], ['□□□','□□□□','□□□□','□□□□□□',
'□□□□□□□□'],
['□□□□□','□□□□','□□□□','□□□□','□□□□□','□□□','□□','□□□',
'□□□□','□□□□','□□□□□','□□□□□','□□□'],
['□□□□','□□□□□□□','□□□□','□□□□□□□','□□□□□'],
['□□□□□','□□□','□□□□□','□□□□','□□□','□□','□□□□□'].

The module's tokenization component, in general, divides the input Geez text into distinct units, providing the framework for precise part-of-speech tagging and efficient grammar checking in later system phases. The algorithm provided below outlines the main procedures in the tokenization section of the preprocessing module. First, sentences are segmented using standard

punctuation and Ethiopian script, and then word-level tokenization is carried out using predetermined delimiters.

```
FUNCTION InputTextTokenization(input text):
   // Step 1: Define sentence delimiters
   Sentence delimiters ← ['።', '፡', '፧', '?', '!']

   // Step 2: Split input text into sentences
   sentences ← Split input text using sentence delimiters

   // Step 3: Initialize an empty list to hold tokenized sentences
   tokenized sentences ← []

   // Step 4: Define word delimiters
   word delimiters ← [' ', '፨', '፥', '፦']
   // Step 5: For each sentence, perform tokenization
   FOR each sentence IN sentences DO
      tokens ← Split sentence using word delimiters
      Clean tokens by removing empty strings or whitespace
      Append tokens TO tokenized sentences
   END FOR
   RETURN tokenized sentences
END FUNCTION
```

Algorithm 3.1: Input text tokenization

3.5.2 **The tagging**

The preprocessing module's Tagging Component is in charge of giving each token in a sentence a morphology-based Part-of-Speech (POS) tag. The grammatical and morphological structure of the Geez language must be captured through this process. The component uses a tagger dictionary (Word-Tag dictionary) to assign each token a suitable POS tag after receiving a list of tokens from the previous tokenization component. Details about the grammatical category of each token are included in the tagger dictionary (e.g. G. noun, verb), as well as pertinent morphological

characteristics like gender (feminine or masculine), and number (singular or plural). "NTG" stands for "Not Tagged" and is used to indicate a token that is not in the dictionary. Sample tag types and descriptions for each are shown in Table 3.1.presents sample tag types along with their corresponding descriptions.

Table 3.1 sample tag type in the tagger dictionary

Tag	Description
NN_SN_SB	Singular Noun used as Subject
NN_FP_SN_SB_POS	First Person, Singular, and Possessive Noun used as Subject
PRPN_MS_SB	Masculine Proper Noun used as Subject
PRN_FP_SN_SB	First Person, Singular, Pronoun used as Subject
PRN_SP_SN_FM_OB	Second Person, Singular, Feminine, Pronoun used as Object
ADJ_FM_SN	Feminine, Singular, Adjective
VRB_SP_SN_FM_SB	Second Person, Singular, Feminine, Verb used as Subject
ADV_MNR	Adverb of Manner

The tokenized Geez sentence serves as an example of the tagging procedure ['□ □ □ □ □ □ □', '□ □ □', '□ □ □ □']. The tagging process would tag each token as indicated below, assuming that every token in the provided sentence was present in the tagger dictionary.

- □ □ □ □ □ □ □ -> PRPN_MS_SB: stands for Proper Noun, Masculine, used as Subject.
- □ □ □ -> ADV_MNR: stands for Adverb of Manner.
- □ □ □ □ -> VRB_INT_TP_SN_MS_SB: stands for an intransitive verb that requires a Third Person, Singular, and Masculine Subject.

Through this tagging process, the system is able to recognize the subject and comprehend the grammatical relationships in the sentence ("እግዚአብሔር") and its agreement with the verb ("ይመጽእ").As seen below:

["PRPN_MS_SB+ADV_MNR+VRB_INT_TP_SN_MS_SB"] The tagging component is a fundamental step in converting raw token sequences into linguistically meaningful representations appropriate for advanced grammatical evaluation. The final output of the tagging process would be a sequence of tags, delimited with a '+' character, which represents the tagged map of the original tokenized sentence.

3.6 Rule-based Grammar Checking

This module uses a predefined set of grammar rules to identify agreement mismatches in the input Geez sentences in order to detect grammatical errors. With the use of manually created rules that take into account the syntactic structure of the Geez language, it focuses on identifying Subject-Verb, Object-Verb, and Adverb-Verb agreement errors. The two parts that make up the module are Rule Agreement Error Detection and Tagged Sentence Rule Agreement Checking.

3.6.1 Tagged Sentence Rule Agreement Checking

Each word in the input sentence is given a part-of-speech (POS) tag. The system then evaluates the syntactic validity of the sentences by comparing the tag sequence to a list of predefined grammatical structures that are kept in a database table known as the Correct Sentence Rules Table. A crucial component of the rule-based grammar validation system is the Correct Sentence Rules Table, an ordered database of grammatical rules that define proper sentence structures. A string of part-of-speech (POS) tags, separated by a plus sign (+), represents each rule in the table. Syntactic structures are presented in this format in an easy-to-understand way. The table is composed of three main fields: rule_id, pattern, and description. For management and reference purposes, each rule has a unique identifier known as the rule_id. The pattern uses a tag sequence to store the actual grammar rule.

Consider the tokenized Geez sentence ['ሰሎሞን', ', 'ይመጽእ'], to illustrate how the rule-based grammar checking process is performed. Accordingly its mapped tag pattern would be

“PRPN_MS_SB+ VRB_INT_TP_SN_MS_SB”. The component checks this sentence pattern against a rule from the correct sentence rule table: “A singular masculine proper noun (PRPN_MS_SB) must be followed by a verb agreeing in number and gender (VRB_INT_TP_SN_MS_SB).” Here, '□ □ □ □' (PRPN_MS_SB) and '□□□□ (VRB_INT_TP_SN_MS_SB) satisfy the agreement rule, so no error is flagged. However, if the tokenized sentence were ['□ □ □ □', ', '□ □ □ □'] the pattern of its mapped tags would be "PRPN_MS_SB+ VRB_INT_TP_PL_MS_SB.". Because Geez sentences lack this pattern, the grammar checker would identify it as a sentence that violates Geez grammar because the sentence's masculine proper noun subject (PRPN_MS_SB) does not agree with a verb that needs a plural subject (VRB_INT_TP_PL_MS_SB).

This rule-based method is appropriate for grammar checkers and language learning tools because it is transparent and simple to customize. It is, however, inevitably constrained by the table's coverage of the rules; grammatically correct sentences that are not included in the rule set could be mistakenly marked as incorrect. Therefore, the accuracy of the system as a whole depends on the Correct Sentence Rules Table being complete. The steps in the rule-based grammar checker module's Tagged Sentence Rule Agreement Checking component are described in the algorithm below.

```
Algorithm TaggedSentenceRuleAgreementChecking
Input: tagged_sentence // A list of POS tags from the Preprocessing module
Output: error_flag (Boolean)
BEGIN
   // Step 1: Convert the tagged sentence into a pattern string
   tag_pattern ← Join tags in tagged_sentence with "+" separator
   // Step 2: Initialize error flag
   error_flag ← FALSE
   // Step 3: Check if the pattern exists in the Correct Sentence Rules Table
   exists ← QueryDatabase(
      "SELECT 1 FROM CorrectSentenceRulesTable WHERE pattern = ?", tag_pattern
   )
   // Step 4: Set error flag if pattern is not found
```

```
    IF exists IS FALSE THEN
        error_flag ← TRUE
    END IF
    // Step 5: Return result
    RETURN error_flag
END
```

Algorithm 3.2: Tagged Sentence Rule Agreement Checking

The algorithm determines whether the part-of-speech tag sequence in the input sentence complies with any legitimate grammatical patterns that are kept in the rules database. The error_flag is set to FALSE, signifying that the sentence is grammatically correct, if a match is discovered. The error_flag is set to TRUE if no match is found, indicating that the sentence is grammatically incorrect and needs to be sent to the following step for additional error diagnosis and correction.

3.6.2 **Rule Agreement Error Detection**

In order to verify that Geez sentences are grammatically correct, one of the most important modules is the Rule Agreement Error Detection component. The Tagged Sentence Rule Agreement Checking process specifically deals with sentences that have been marked as grammatically incorrect. Its main purpose is to determine the reason behind the grammatical error and to produce insightful error messages that can be utilized for additional research or fixing. The component first looks for words marked with NTG (Not Tagged Grammatically) in a sentence before flagging it. A list of all NTG-tagged words is compiled by the component and included in the generated error message if such tags are present. By highlighting the precise sentence components that were not able to be parsed or checked against accepted grammatical rules, this method provides instant insight into the error's origin. The component executes a structural function when there are no NTG-tagged words in the flagged sentence.

The Rule Agreement Error Detection component keeps track of all the error messages it generates in an orderly fashion. This structured data includes the original sentence, the type of grammatical error discovered, and any pertinent error descriptions. Other system components, such as error visualization, can process data more easily when the system is organized this way. Take a look at

the tokenized two Geez sentences below: ['ወትፍሥሕተ', 'ዘዚአየ', 'ውእቱ'], and ['ሰሎሞን', 'ይመጽኡ']. In the first sentence, assume that the word 'ወትፍሥሕተ' is tagged as NTG,

Stating that the tagger dictionary does not contain it. Therefore, it receives an error message from the Rule Agreement Error component, which is formatted as follows: "- The sentence ወትፍሥሕተ ዘዚአየ ውእቱ። contains untagged word(s): 'ወትፍሥሕተ'.". However, the second sentence lacks NTG elements and is structurally well-formed in terms of tagging. The verb ‘ይመጽኡ’ is in the plural form, but the subject ‘ሰሎሞን’ is in the singular. This discrepancy represents a numerical error in subject-verb agreement. The Rule Agreement Error component identifies the infraction and provides the precise error message, "Subject-verb number agreement error," after comparing the tagged structure to patterns in the Incorrect Sentence Rules Table. The steps in the Rule Agreement Error Detection part of the rule-based grammar checker module, which is in charge of determining the root cause of grammatical errors, are described in the algorithm below.

```
Algorithm RuleAgreementErrorDetection
Input: tagged_sentence, original_sentence  // tagged_sentence is a list of tagged words
Output: error_info  // A structured object containing the error message and related data

BEGIN
   // Step 1: Initialize variables
   error_message ← ""
   ntg_words ← EmptyList()
   error_info ← EmptyDictionary()
   // Step 2: Check for NTG-tagged words
   FOR EACH word IN tagged_sentence DO
      IF word.tag == "NTG" THEN
         Append word.text TO ntg_words
      END IF
   END FOR
```

```
    // Step 3: If NTG tags are found, report them as the error
    IF Length(ntg_words) > 0 THEN
      error_message ← "Sentence contains unrecognized words: " + Join(ntg_words, ", ")
     ELSE
      // Step 4: If no NTG tags, generate tag pattern
      tag_pattern ← Join all word.tag IN tagged_sentence WITH "+"

      // Step 5: Query Incorrect Sentence Rules Table
      rule_entry ← QueryDatabase(
         "SELECT description FROM IncorrectSentenceRulesTable WHERE pattern = ?", tag_pattern
      )
      // Step 6: Handle match or default error
      IF rule_entry IS NOT NULL THEN
         error_message ← rule_entry.description
      ELSE
         error_message ← "The grammatical rule of the sentence is not defined."
      END IF
    END IF

    // Step 7: Build structured error information
    error_info["original_sentence"] ← original_sentence
    error_info["error_message"] ← error_message

    // Step 8: Return error information
    RETURN error_info
END
```

Algorithm 3.3: Rule Agreement Error Detection

3.7 Statistical Grammar Checking

The Statistical Grammar Checking module uses a data-driven, probabilistic methodology to assess the grammatical correctness of Geez sentences. It makes use of patterns discovered through real-world language use, particularly the probability of word sequences based on statistical frequency, rather than depending only on manually created grammatical rules. This approach works especially well for spotting small or unusual mistakes that conventional rule-based systems might miss. A database table named Bigram and Probability, which houses a list of bigram pairs of consecutive words in Geez and their corresponding probabilities, forms the core of the module. These probabilities, which show how frequently one word follows another in actual usage, are taken from a sizable, previously examined corpus of Geez texts. The Preprocessing module provides a list of sentences to start the process. Every sentence is divided into a number of bigrams. The Bigram & Probability are then consulted by the system.

Consider the Geez sentence "ጳውሎስ ወጴጥሮስ መምህራን ውእቶን።", which is tokenized into ['ጳውሎስ', 'ወጴጥሮስ', 'መምህራን', 'ውእቶን']. From this, the following bigrams are formed: ('ጳውሎስ', 'ወጴጥሮስ'), ('ወጴጥሮስ', 'መምህራን'), and ('መምህራን', 'ውእቶን'). Assume that each bigram is assigned a probability value retrieved from a pre-computed bigram probability table: P ('ጳውሎስ', 'ወጴጥሮስ') = 0.78, P ('ወጴጥሮስ', 'መምህራን') = 0.56, and P ('መምህራን', 'ውእቶን') = 0.00 and the threshold value is set to0.05. Since the last bigram's probability (0.00) falls below this threshold, the sentence is flagged as potentially grammatically incorrect. The source of the issue lies in the misuse of the verb-to-be form 'ውእቶን', which is appropriate for feminine plural nouns, following the masculine plural noun 'መምህራን' ('male teachers'). The correct verb form should be 'ውእቶሙ', making the correct bigram ('መምህራን', 'ውእቶሙ').this example demonstrates how the statistical method can identify grammatically unlikely sequences and reveal agreement errors in Geez sentence structure. The grammatical errors that rule-based grammar checkers frequently overlook can be found with this statistical technique. A complex language like Geez may have too many linguistic structures for rule-based systems, which rely on predefined rules. Subtle mistakes like odd word order or missing elements can be found using the statistical method, which analyzes real word usage and assigns probabilities to word pairs. It enhances system performance overall, particularly in areas where the rule-based component is under-covered, and makes the grammar

checker more thorough and dependable for Geez texts. This algorithm demonstrates the main process of the Statistical Grammar Checker module, which uses bigram probabilities to assess the likelihood of word sequences in order to identify grammatical errors.

```
Algorithm StatisticalGrammarCheck
Input: sentence_list (list of Geez sentences from Preprocessing module)
      threshold (float: predefined minimum acceptable probability)
Output: flagged_sentences (list of tuples: sentence, list of low-probability bigrams)
Begin
   Initialize flagged_sentences as an empty list

   For each sentence in sentence_list do
      tokens ← Tokenize(sentence)  // Split sentence into word tokens
      bigrams ← GenerateBigrams(tokens)  // Create pairs: (w1, w2), (w2, w3), ...
      low_prob_bigrams ← empty list
      For each bigram in bigrams do
         probability ← QueryBigramProbabilityFromDB(bigram)
         If probability is NULL then
            probability ← 0.0  // If not found in DB, assume zero
         If probability < threshold then
            Add bigram to low_prob_bigrams
      If low_prob_bigrams is not empty then
         Add (sentence, low_prob_bigrams) to flagged_sentences
   Return flagged_sentences
End
```

Algorithm 3.4: Statistical Grammar Checker Algorithm

3.8 **Display Grammar Checking Result**

The last step in the Geez grammar checker system is the Display Grammar Checking Result module, which is in charge of clearly and understandably displaying the grammar analysis's findings. The module assesses whether any grammatical errors were found in the user's input text

once each grammar checking session is over. "There is no grammar error in the input text," for example, is a simple confirmation message that the system shows if no problems detected. Users can quickly find and comprehend the grammatical errors if they are detected because the module displays each one in the text's order of occurrence. This module shows the outcomes of the statistical and rule-based grammar checkers. The system indicates that there is a grammar problem in the corresponding sentence when it detects rule-based errors like subject-verb number disagreement or adverb-verb misplacement. errors that have been statistically identified.

CHAPTER FOUR :EXPERMENTAL SETUP and RESULTS

4.1 Introduction

In this chapter, we will discuss the corpus preparation for training data and the tagger dictionary, the development of grammar rules, the implementation of the prototype, and the evaluation of the system**.** It also discusses the experimental setup and evaluation process for grammatical error checking. Finally, the system**'s** performance results are analyzed and presented.

4.2 System Implementation Tools

Throughout the system implementation phase, a range of development environments and tools were used to successfully accomplish the goals of this study. These tools were chosen based on how well they handled various facets of the system's development, design, and data management. Python was used to implement the system's core because of its many libraries, flexibility, and robust support for data processing and system integration. PyCharm and Jupyter Notebook, two integrated development environments (IDEs), were used to promote effective code development and experimentation. While Jupyter Notebook, which runs on the Anaconda platform, was used for interactive coding, data analysis, and visualization tasks, PyCharm offered a stable environment for structured software development, debugging, and code management. The XAMPP Control Panel was used to deploy a MySQL database for the system's backend.
Lastly, the system architecture was designed and documented using Microsoft Visio. Visio's ability to clearly visualize system components, data flow, and overall structural design was essential for development planning and system framework communication.

4.3 Backend System Structure

The hybrid Geez Grammar Checker System backend architecture is designed to support the core functions of part-of-speech tagging, syntactic rule validation, and statistical language modeling. The MySQL relational database is the central part of this architecture, and is a structured repository of all linguistic resources and computational data needed by the system. It is organized into three major categories of tables, each corresponding to a key component of the grammar checking process: the tagger dictionary, grammar rule sets, and bigram probabilities.

4.3.1 Tagger Dictionary Structure

The Word-Tag dictionary, also known as the tagger dictionary, is an essential linguistic tool that associates Geez words with the appropriate part-of-speech (POS) tags. This part makes it possible for the POS tagger in the system to categorize each word according to the proper grammar rules when analyzing sentences.

The structure of the tagger dictionary table is as follows:

- **id**: A unique identifier assigned to each word entry.
- **word**: The Geez word being analyzed.
- **tag**: The POS tag corresponding to the word.

4.3.2 Grammar Rule Table Structure

The grammar-checking functionality of the system is built upon a comprehensive rule base consisting of both correct and incorrect sentence patterns. These rules are housed in two separate but structurally identical tables: Correct Sentence Rules Table, and Incorrect Sentence Rules Table.

Each rule is represented as a sequence of POS tags, encoding a syntactic structure found in either valid or error-prone Geez sentences.

The common structure of both tables is defined as:

- **rule_id**: A unique identifier for each grammar rule.
- **pattern**: A sequence of POS tags that represent a grammatical structure.
- **description**: A textual explanation of the rule. For correct rules, this describes the specific grammatical agreement being followed. For incorrect rules, it details the nature of the syntactic violation.
- **pattern_length**: The number of POS tags in the rule pattern.

The system can identify mistakes and give explanations for grammatically incorrect sentences by using these tables.

4.3.3 Bigram Probability Table Structure

The statistical part of the grammar checker, which evaluates sentence plausibility based on word co-occurrence patterns, is supported by the bigram-probability table. This method improves the system's capacity to identify mistakes that rule-based approaches might miss.

The table has the following structure:

- **id**: A primary key to uniquely identify each bigram entry.
- **word1**: The first word in the bigram.
- **word2**: The second word in the bigram.
- **probability**: The conditional probability, P(word2|word1), indicating how likely word2 follows word1 in natural Geez text.

By integrating this statistical data, the system is capable of identifying low-probability sequences that may signal grammatical anomalies.

4.4 Backend Data Preparation

The quality and completeness of the backend linguistic data that the Hybrid Geez Grammar Checker System uses have a significant impact on its efficacy and accuracy. This section describes the methodical procedure used to compile and arrange important datasets, such as the bigram probabilities, grammar rules, and tagger dictionary, required to support the system's statistical and rule-based components.

4.4.1 Dictionary Preparation

The grammar checker's part-of-speech tagging system is built on top of the tagger dictionary. An authoritative Geez dictionary book [31] [32] and other spiritual texts were among the many reliable sources that were carefully gathered to create this resource. In order to ensure linguistic accuracy, Geez language experts were instrumental in assigning the proper part-of-speech (POS) tag to each word. Initially, raw lexical data was compiled from print and digital sources, such as spiritual texts and Geez dictionaries, in order to prepare the tagged dictionary. After that, MS Excel preprocessing was used to produce a clear and succinct word list by methodically identifying and eliminating redundant and duplicate entries using spreadsheet features. Lastly, a custom Python

script that the researcher had created was used to carry out an automated processing step. The data was further cleaned by this script, which also confirmed its consistency.

The system's dictionary now has over 10,000 distinct Geez words, each associated with a POS, as a consequence of this thorough process. This resource is an essential input for the system's tagging and grammatical analysis functions. Figures 4.1 illustrates some of the entries in the tagger dictionary table saved in the database.

←T→			id	gzword	tag
Edit	Copy	Delete	101	ንሕነ	PRN_FP_PL_SB
Edit	Copy	Delete	102	ሠረገላ	NN_SN_SB
Edit	Copy	Delete	103	ሰብአ	NN_SN_OB
Edit	Copy	Delete	104	አስቴር	PRPN_FM_SB
Edit	Copy	Delete	105	ቅዱስ	ADJ_MS_SN
Edit	Copy	Delete	106	መጠወ	VRB_TD_SN_MS_SB
Edit	Copy	Delete	107	ክሡተ	ADV_MNR
Edit	Copy	Delete	108	ማርያም	PRPN_FM_SB
Edit	Copy	Delete	109	ማቴዎስ	PRPN_MS_SB
Edit	Copy	Delete	110	ሚመት	VRB_TD_SN_FM_SB

Figures 4.1: Tagger Dictionary sample data

4.4.2 **Grammar Rule Preparation**

The system's grammar-checking module uses a carefully selected set of syntactic rules that are mostly taken from a standard Geez grammar reference book [20]. The two categories of these rules are Incorrect Sentence Rules, which highlight common grammatical errors, and Correct Sentence Rules, which represent syntactically valid constructions in Geez. Initially, both rule sets were manually assembled using Microsoft Excel spreadsheets. By using a manual method, the researcher was able to examine and categorize rules according to grammatical agreement structures. Three different types of agreement errors were identified from the incorrect rules: SVA,

OVA, and AVA. A Python script was then created to guarantee structural integrity while automating the transfer of these rules into the database. Every rule whether right or wrong was encoded as a tag sequence pattern with related metadata. There are currently more than 360 rules for correct sentences and 150 rules for incorrect sentences in the database. Figures 4.2 and 4.3 illustrate examples of entries from the Correct Sentence Rules Table and Incorrect Sentence Rules Table, respectively.

rule_id	pattern	description	pattern_length
51	PRPN_MS_SB+VRB_INT_TD_SN_MS_SB	SVA	30
52	PRPN_FM_SB+VRB_INT_TD_SN_FM_SB	SVA	30
53	PRN_FP_SN_SB+VRB_INT_FP_SN_SB	SVA	29
54	PRN_SP_SN_FM_SB+VRB_INT_SP_SN_FM_SB	SVA	35
55	PRPN_MS_SB+VRB_TD_SN_MS_SB+PRN_FP_SN_OB	SVA	39
56	PRN_FP_PL_SB+VRB_FP_PL_SB+PRN_FP_SN_OB	SVA	38
57	PRPN_MS_SB+VRB_TD_SN_MS_SB_TD_PL_FM_OB+PRN_TD_PL_F...	OVA	54
58	PRN_TD_SN_MS_SB+VRB_TD_SN_MS_SB_TD_PL_MS_OB+PRN_TD...	OVA	59

Figures 4.2: Correct Sentence Rules Table sample data

rule_id	pattern	description	pattern_length
51	PRPN_MS_SB+VRB_INT_TD_SN_FM_SB	Subject Verb Agreement error in gender.	30
52	PRPN_FM_SB+VRB_INT_TD_PL_FM_SB	Subject Verb Agreement error in number.	30
53	PRN_SP_SN_SB+VRB_INT_FP_SN_SB	Subject Verb Agreement error in person.	29
54	PRN_SP_SN_FM_SB+VRB_INT_TD_SN_FM_SB	Subject Verb Agreement error in person.	35
55	PRPN_MS_SB+VRB_TD_SN_FM_SB+PRN_FP_SN_OB	Subject Verb Agreement error in gender.	39
56	PRN_FP_SN_SB+VRB_FP_PL_SB+PRN_FP_SN_OB	Subject Verb Agreement error in number.	38
57	PRPN_MS_SB+VRB_TD_SN_MS_SB_TD_PL_FM_OB+PRN_TD_SN_F..	Object Verb Agreement error in number.	54
58	PRN_TD_SN_MS_SB+VRB_TD_SN_MS_SB_TD_PL_MS_OB+PRN_TD.	Object Verb Agreement error in gender.	59

Figures 4.3: Incorrect Sentence Rules Table sample data

4.4.3 **Corpus Preparation**

To train and test the language model, a Geez language corpus was created to support the grammar checker's statistical analysis feature. This corpus includes over 18,000 sentences drawn from reputable spiritual writings and canonical texts like the Geez Bible. In order to facilitate manual inspection and preliminary preprocessing, the sentences were compiled first, then transcribed from reliable Geez sources and arranged in Microsoft Excel. The researcher's custom Python program was then used to perform an automated bigram extraction phase. Using the Excel files, this program reads the sentences, tokenizes them, extracts word bigrams from each sentence, and determines the conditional probabilities for each distinct bigram. By taking these actions, the statistical data that was produced was guaranteed to be complete and prepared for entry into the database of the system. More than 202,000 u make up the final dataset.

Figure 4.4 presents a snapshot of the bigram data as stored in the system's database.

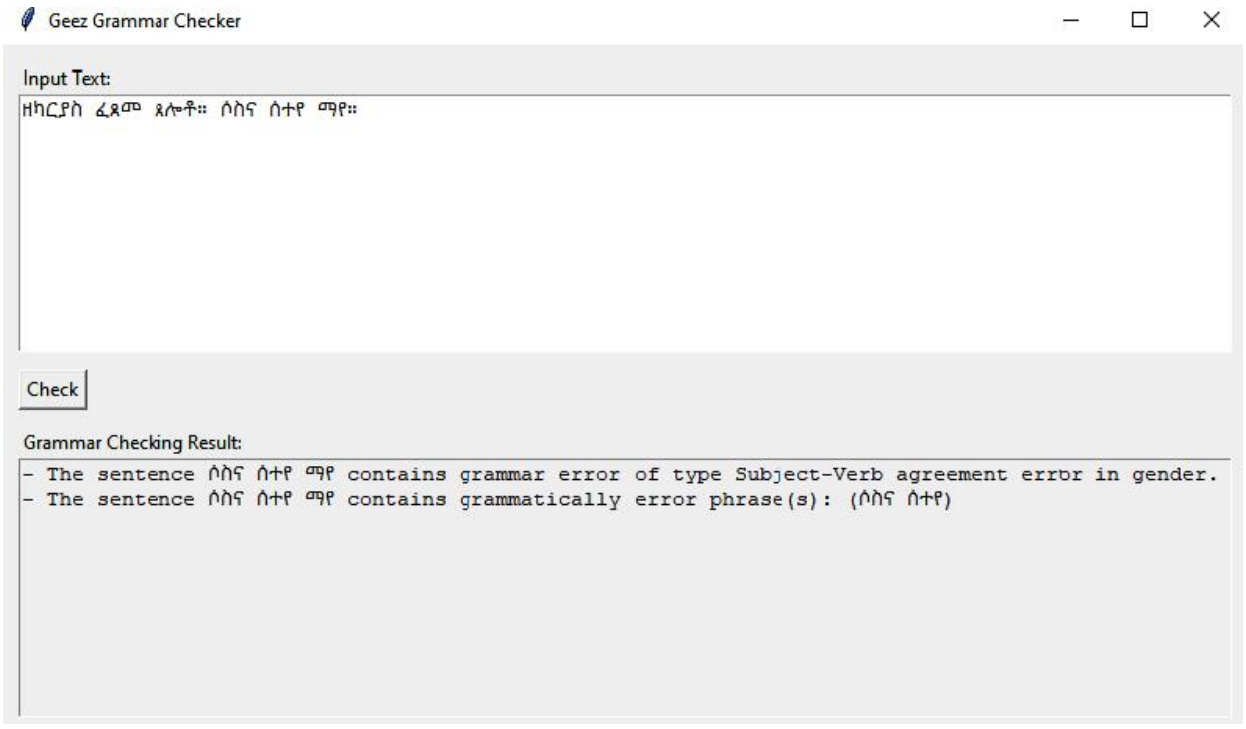
id	gzword1	gzword2	probability
1001	ሖረ	ኀበ	1
1002	ኀበ	ደብረ	0.36
1003	ብእሲ	ሖረ	0.75
1004	አነ	ውእቱ	0.67
1005	ሠናይ	በዓል	0.4
1006	አመ	ተሰቅለ	0.5
1007	ምስለ	አብ	0.64
1008	በሰላም	በፍቅር	0.95
1009	ቅድስት	ድንግል	0.8
1010	ድንግል	ማርያም	0.85

Figures 4.4: Bigram-Probability Table sample data

4.5 Application Development

We used the Python programming language to implement the hybrid Geez Grammar Checker System as described in the previous chapters. The PyCharm Integrated Development Environment (IDE), which offered the required tools for effective code writing, debugging, and project management, was used for the development. A Windows-based desktop application with a straightforward Graphical User Interface (GUI) to promote user interaction is the end product of this implementation. To guarantee usability for a wide spectrum of users, including those without technical expertise, the interface was purposefully kept simple. Users can type or paste Geez text they want to evaluate in the text input area. The user starts the grammar check process by clicking the "Check" button after entering the text. When activated, the system uses both statistical and rule-based analysis mechanisms created as algorithms to process the input text.

A sample interaction demonstrating the system's functionality and user interface is shown in Figure 4.5, illustrating how a user inputs Geez text and receives grammar analysis in real-time. The application thus serves as a practical and user-friendly tool for promoting proper Geez grammar usage through automated assistance.



Figures 4.5: Sample interaction with the Geez Grammar Checker application.

4.6 Evaluation

The performance evaluation of the hybrid Geez Grammar Checker System is conducted based on experimental results related to agreement and word sequence grammar errors. The system's effectiveness is assessed by manually counting correctly flagged errors and measuring correctness, completeness, and accuracy using recall, precision, and F-score computations. To evaluate the system, 400 grammatically correct and incorrect sentences are used, categorized into four grammatical error types, as shown in Table 4.1.

Table 4.1: Test Data

Grammatical error type	Numbers of flagged correct grammar error	Numbers of wrong flagged grammar errors	Total
SVA	70	10	80
OVA	80	15	95
AVA	80	5	85
WSA	128	12	140
Total	358	42	400

4.6.1 Evaluation Metrics

To evaluate the proposed grammar checker, standard metrics from Natural Language Processing such as Precision, Recall, and F-measure were used. Precision measures how many flagged errors are correct, while Recall reflects how many actual errors the system detects. F-measure balances these to give a single performance score. These well-established metrics effectively capture both the accuracy and coverage of error detection, providing a reliable framework to assess the system’s effectiveness in identifying grammatical errors in Geʿez texts.

As detailed in the methodology, these align with information retrieval paradigms in NLP, where precision mitigates over-flagging (critical for user adoption in grammar tools), recall ensures thoroughness (essential for low-resource languages with sparse data), and F-measure offers a

holistic view, often used in GEC surveys to benchmark against state-of-the-art systems. Formulas are as provided earlier.

FCE (Correctly Flagged Errors): As above, the number of correctly identified errors (true positives).

NFE (Not Flagged Errors): The number of actual grammatical errors that the system fails to detect (false negatives).

FWE (Falsely Flagged Errors): The number of instances where the system incorrectly flags a grammatically correct sentence or word sequence as erroneous (false positives).

$$= \frac{\quad}{+} \qquad (5.1)$$

Precision = $\frac{90}{(90+10)} = 0.90 \quad 90\%$

$$= \frac{\quad}{+} \qquad (5.2)$$

Recall = $\frac{85}{(85+15)} = 0.85 \quad 85\%$

$$\mathrm{F} - measure = \frac{2 * Precision * Recall}{+} \qquad (5.3)$$

F-measure = $\frac{(2*0.90*0.85)}{(0.90+0.85)} \approx 0.874 \; or \; 87.4\%$

4.6.2 **Test Result**

The number of correctly flagged grammar agreement errors from 358 grammatically correct Geez sentences and 42 incorrect ones was used to manually record the system's test results. System testing was carried out over three days in order to accomplish this, and the results of each day were designated as Exp1, Exp2, and Exp3, which stand for day 1, day 2, and day 3, respectively. 133

sentences were distributed and tested daily for each experiment. The results of correctly flagged grammar agreement errors are presented in Table 4.2 and Figure 4.3.

Table4.2 correctly flagged result in each experiment

Agreement Error Type	**Exp 1(%)**	**Exp 2(%)**	**Exp 3(%)**	**Average (%)**
SVA	80	74	80	78.0
OVA	83	81	79	81.0
AVA	73	78	77	76.0
WSA	82	76	78	78.7
Average	**80**	**77**	**78**	**78.4**

The Average experiment result of correctly flagged grammar error type is shown Figure 4.5

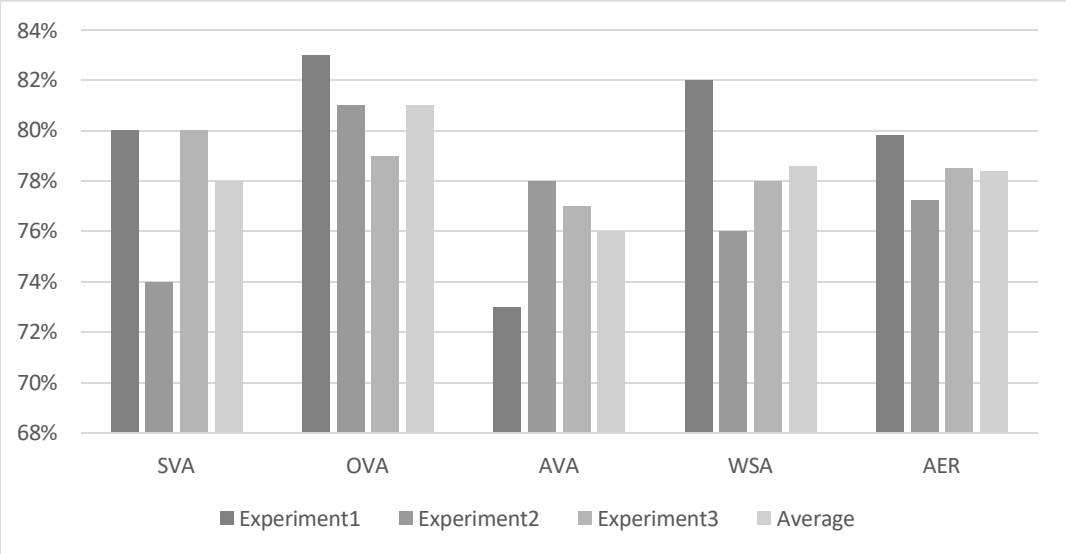


Figure 4.6. Correctly flagged grammar error

Using the precision, recall, and F-score metrics derived from the average experiment results of the grammar error types, the researcher evaluated our system's performance. The performance of the Geez Grammar checker system is detailed in Table 4.2. Table 4. 3The hybrid Geez grammar checker system's average performance.

Table 4.3 Average Performance of hybrid Geez Grammar Checker system

Grammar Error Type	**No of input Grammar Error**	**Correct Flagged Errors**	**Wrong Flagged Errors**	**Not Flagged Errors**	**Precision**	**Recall**	**F-Measure**
SVA	80	70	10	10	87.50	87.50	87.50
OVA	95	80	15	15	84.21	84.21	84.21
AVA	85	80	13	5	86.02	94.12	89.89
WSA	140	128	8	12	94.12	91.43	92.75
Total	400	358	46	42	88.61	89.50	89.05
Average					87.96	89.32	88.59

The average evaluation result of the Hybrid Geez Grammar Checker System is shown in Figure 4.3 based on recall, precision and f-measure evaluation metrics.

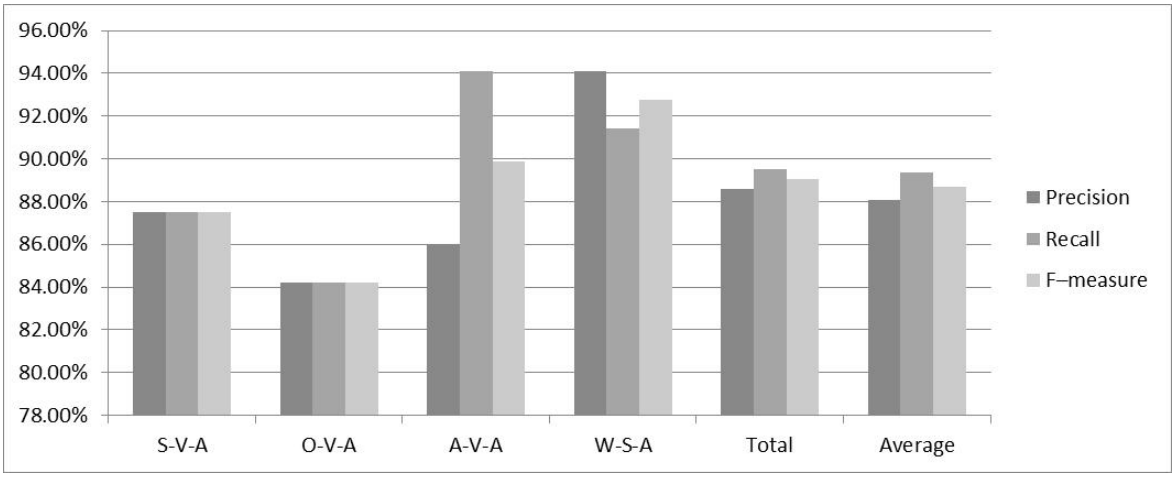


Figure 4.7 Average performance of Hybrid Geez Grammar Checker System

4.7 Discussion of the result

The performance evaluation of the hybrid Geez grammar checker was conducted manually using a carefully prepared dataset of 400 sentences. These sentences included both grammatically correct and incorrect constructions, and they were selected to reflect diverse grammatical agreement scenarios commonly found in Geez. The experiment was carried out in four separate phases, each targeting specific types of agreement errors: subject-verb agreement (SVA), object-verb agreement (OVA), adjective-verb agreement (AVA), and word sequence agreement (WSA). For each experiment, the system's outputs were compared to manually annotated ground truths to ensure accuracy and consistency in the evaluation process. These results are clearly presented in Table 4.2.

The system achieved a strong performance across all experiments, with an average of 87.96% precision, 89.32% recall, and 88.59% F-measure. These results indicate that the system was highly effective in accurately identifying and classifying grammatical agreement errors. Precision reflects the system's ability to correctly identify errors without false positives, while recall shows its success in capturing most of the existing errors. The F-measure, a balanced metric combining both, further confirms the system’s reliability and robustness in practical usage. These performance indicators demonstrate the strength of the hybrid approach that combines rule-based and statistical methods to handle the complexities of Geez grammar.

Overall, the experiment validates the system's capability to support grammar checking in the Geez language with high accuracy. The visual representation of the results in Figure 4.3 illustrates the system’s consistent performance across different sentence types and agreement error categories. This experimental success highlights the potential of the developed system as a foundational tool for Geez language processing and further strengthens its applicability in academic, educational, and computational linguistics domains. The outcomes also show case the significance of integrating linguistic rules with statistical models to achieve a reliable grammar checking solution for morphologically rich languages like Geez.

CHAPTER FIVE:CONCLUSION AND FUTURE WORK

5.1 Conclusion

This study presented the development of a hybrid grammar checker system for the Geez language, an important step in applying Natural Language Processing (NLP) to under-resourced languages. The system was created to reduce the time and effort involved in identifying grammatical errors manually in written Geez texts. While grammar checkers have been implemented for languages such as English, Amharic, and Chinese, to the best of the researcher's knowledge, no prior attempt has been made to develop a grammar checking system specifically for Geez. The goal of this research was to design a system that could identify grammatical agreement errors unique to Geez, contributing to both technological advancement and language preservation.

The system combines rule-based and statistical approaches to ensure accurate error detection. The rule-based component is designed to capture four major types of grammar agreement errors: Subject-Verb Agreement (SVA), Object-Verb Agreement (OVA), Adjective-Verb Agreement (AVA), and Word Sequence Agreement (WSA). A total of 420 rules were manually developed for this purpose. The statistical component complements this by identifying word sequence errors using bigram probabilities from a 10,000 unique Geez words that are labeled with each of their respective POS. To evaluate system performance, a dataset of 400 manually labeled sentences (both correct and incorrect) was used.

The evaluation results show that the system is effective in identifying grammar errors, particularly in independent sentences. It achieved a precision of 88.9%, a recall of 89.35%, and an F-measure of 88.68%. However, the system faced challenges in accurately processing dependent or complex sentences, especially those containing words not found in the tagging dictionary. Despite these limitations, the overall performance demonstrates that the proposed hybrid model is a promising tool for improving written Geez texts and can serve as a foundation for further development and linguistic tools for the language.

5.2 Contribution of the Work

This thesis targets the historically significant and under-resourced Geez language, making several important contributions to the field of Natural Language Processing (NLP). One of the main

contributions is the creation and deployment of a hybrid grammar checker system for Geez that combines statistical and rule-based methods. Although these methods have been used for many contemporary languages, this study is the first to apply them to Geez, showing how flexible hybrid grammar-checking techniques can be for ancient and morphologically complex languages. Additionally, the study presents a thorough set of manually created grammatical rules that specifically pinpoint four categories of grammar agreement errors that are prevalent in Geez: word sequence agreement (WSA), subject-verb agreement (SVA), object-verb agreement (OVA), and adjective-verb agreement (AVA). These 420 rule sets provide a fundamental framework and were created with the assistance of Geez language specialists.

5.3 Future Work

The researcher have tried to cover as many grammar error types as the researcher could to solve an error within a sentence in any input of Geez text. However, there are many types of grammar error that probably occur in a sentence are not covered. Hence, the researcher can recommend working on the following research works:

- Lexical Resource Improvement: The developed system currently contains more than 10,000 distinct tokens with part-of-speech (POS) and morphology tags. Performance can be increased by improving the current resource and selecting other modules such as an automatic spell checker, a high-performance POS tagger, and a specialized morphological analyzer to deliver improved word-level feature extraction and interpretation.
- Contextual Grammar Correction: This study focuses on error correction within independent sentences indicated by end punctuation. Future research may expand the grammar checker to act at the paragraph level, making it able to manage inter-sentential relations and contextual agreement among sentences.
- Handling of compound and complex sentences: Developing rule-based operations to decompose compound and complex sentences into their simpler counterparts via sentence boundary detection and structural analysis would enable the system to more accurately recognize and correct mistakes in syntactically more complex inputs.

References

[1] A. G. Gebreegzaabher, “College of Natural sciences SCIENCE IN PARTIAL FULFILLMENT FOR THE DEGREE OF College of Natural sciences,” no. June, 2018.

[2] M. J. Alam, N. Uzzaman, and M. Khan, “N-gram based Statistical Grammar Checker for Bangla and English,” *Ninth Int. Conf. Comput. Inf. Technol. (ICCIT 2006)*, pp. 3–6, 2006.

[3] D. Naber, P. F. Kummert, T. Fakultät, and A. Witt, “A Rule-Based Style and Grammar Checker,” 2003.

[4] A. Arppe, “Developing a grammar checker for Swedish,” *12th Nord. Conf. Comput. Linguist.*, pp. 13–27, 2000, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.95.5017&rep=rep1&type=pdf

[5] S. M. Cheng, C. H. Yu, and H. H. Chen, “Chinese word ordering errors detection and correction for non-native Chinese language learners,” *COLING 2014 - 25th Int. Conf. Comput. Linguist. Proc. COLING 2014 Tech. Pap.*, pp. 279–289, 2014.

[6] A. T. Gebru, “Design and Development of Amharic Grammar Checker,” no. March, 2013, [Online]. Available: http://etd.aau.edu.et/bitstream/handle/123456789/917/Aynadis Temesgen.pdf?isAllowed=y&sequence=1

[7] A. A. SHELEMO, “AFAAN OROMO GRAMMAR CHECKER USING HYBRID APPROACH,” *Nucl. Phys.*, vol. 13, no. 1, pp. 104–116, 2023.

[8] A. Temesgen and Y. Assabie, “Development of Amharic grammar checker using morphological features of words and N-gram based probabilistic methods,” *Proc. 13th Int. Conf. Parsing Technol. IWPT 2013*, pp. 106–112, 2013.

[9] ethiopian ortodox Church, “geez book.pdf,” *Tensae publiching house, Addis Abeba*.

[10] N. Y. Lin, K. M. Soe, and N. L. Thein, “Developing a chunk-based grammar checker for translated English sentences,” *PACLIC 25 - Proc. 25th Pacific Asia Conf. Lang. Inf. Comput.*, pp. 245–254, 2011.

[11] desta tekle Weld, “geez grammar.pdf,” *addis abeba*.

[12] M. Mozgovoy, “Dependency-based rules for grammar checking with LanguageTool,” *2011 Fed. Conf. Comput. Sci. Inf. Syst. FedCSIS 2011*, pp. 209–212, 2011.

[13] “□ □ □ □ □ □ □ with English examples (2).pdf.”

[14] C. Sciences, “College of Natural and Computational Sciences School of Information

Science By :," p. 95, 2018.

[15] L. Jindal and V. Rana, "Grammar Checking Using Pattern Matching Approach," vol. 6, no. 3, pp. 142–147, 2018.

[16] J. Fellman and T. O. Lambdin, *Introduction to Classical Ethiopic (Ge'ez)*, vol. 101, no. 4. 1981. doi: 10.2307/601267.

[17] B. Desta, "Design and Implementation of Automatic Morphological Analyzer for Ge ' ez Verbs," *Des. Implement. Autom. Morphol. Anal. Ge ' ez Verbs*, pp. 1–144, 2010, [Online]. Available: aau.etd.edu.et

[18] E. Grammar, *ETHIOPIC GRAMMAR Second Edition Ancient Language Resources*.

[19] J. Lee, "Automatic Correction of Grammatical Errors in Non-native English Text by John Sie Yuen Lee," *English*, no. 2002, 2009.

[20] "geez dictionary," pp. 1–5.

[21] M. K. Abera and Y. ASSABIE, "Development of Part of Speech Tagger for Ge'ez Language," no. October, 2017, [Online]. Available: http://etd.aau.edu.et/handle/123456789/18347

[22] Y. Keleta, K. Yamamoto, and A. Marasinghe, "Tigrinya Part-of-Speech Tagging with Morphological Patterns and the New Nagaoka Tigrinya Corpus," *Int. J. Comput. Appl.*, vol. 146, no. 14, pp. 33–41, 2016, doi: 10.5120/ijca2016910943.

[23] H. A. Nega, "SCHOOL OF GRADUATE STUDY Morpheme Based Bi-Directional Machine Translation The Case of Ge ' ez to Tigrigna A Thesis Submitted in Partial Fulfillment of the Requirement for the Degree of Master of Science in Computer Science," no. January, 2023.

[24] K. F. Shaalan, "Arabic GramCheck: A grammar checker for Arabic," *Softw. - Pract. Exp.*, vol. 35, no. 7, pp. 643–665, 2005, doi: 10.1002/spe.653.

[25] J. Bigert, *Automatic and Unsupervised Methods in Natural Language Processing*. 2005.

[26] □. □ □ □ □ □ □ □ □ □, "□ □ □ □ □ □ □," p. 52, 2015.

[27] S. Katz Bourns and J. Watzinger-Tharp, "Conceptions of L2 Grammar: Theoretical Approaches and their Application in the L2 Classroom," vol. C, 2008.

[28] N. S. Bhirud, R. . Bhavsar, and B. . Pawar, "Grammar Checkers for Natural Languages : A Review," *Int. J. Nat. Lang. Comput.*, vol. 6, no. 4, pp. 1–13, 2017, doi: 10.5121/ijnlc.2017.6401.

[29] J. Xing, L. Wang, D. F. Wong, L. S. Chao, and X. Zeng, “UM-Checker: A hybrid system for English grammatical error correction,” *CoNLL 2013 - 17th Conf. Comput. Nat. Lang. Learn. Proc. Shar. Task*, pp. 34–42, 2013.

[30] H. Beyene, “School of Graduate Studies Design and Development of Tigrigna Search School of Graduate Studies Design and Development of Tigrigna Search,” 2013.

[31] J. Carlberger, R. Domeij, V. Kann, and O. Knutsson, “A Swedish grammar checker,” *Citeseer*, 2000, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.5260%5Cnpapers2://publication/uuid/F7978AFE-423D-4BD0-8322-9646E5893BC9

[32] A. Gebreamlak, “COLLEGE OF NATURAL SCIENCES DEPENDENCY BASED AMHARIC GRAMMAR Abraham Gebreamlak Gebremariam A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science Addis Ababa , Ethiopia College of Natural Sciences,” no. September, 2019.

Appendix A: Geez Script writing alphabet

ሀ	HA	ሁ	HU	ሂ	HI	ሃ	HA	ሄ	HE	ህ	H	ሆ	HO
ለ	LE	ሉ	LU	ሊ	LI	ላ	LA	ሌ	LE	ል	L	ሎ	LO
ሐ	HA	ሑ	HU	ሒ	HI	ሓ	HA	ሔ	HE	ሕ	H	ሖ	HO
መ	ME	ሙ	MU	ሚ	MI	ማ	MA	ሜ	ME	ም	M	ሞ	MO
ሠ	SE	ሡ	SU	ሢ	SI	ሣ	SA	ሤ	SE	ሥ	S	ሦ	SO
ረ	RE	ሩ	RU	ሪ	RI	ራ	RA	ሬ	RE	ር	R	ሮ	RO
ሰ	SE	ሱ	SU	ሲ	SI	ሳ	SA	ሴ	SE	ስ	S	ሶ	SO
ሸ	SHE	ሹ	SHU	ሺ	SHI	ሻ	SHA	ሼ	SHE	ሽ	SH	ሾ	SHO
ቀ	KE	ቁ	KU	ቂ	KI	ቃ	KA	ቄ	KE	ቅ	K	ቆ	KO
በ	BE	ቡ	BU	ቢ	BI	ባ	BA	ቤ	BE	ብ	B	ቦ	BO
ተ	TE	ቱ	TU	ቲ	TI	ታ	TA	ቴ	TE	ት	T	ቶ	TO
ቸ	CHE	ቹ	CHU	ቺ	CHI	ቻ	CHA	ቼ	CHE	ች	CH	ቾ	CHO
ኀ	HA	ኁ	HU	ኂ	HI	ኃ	HA	ኄ	HE	ኅ	H	ኆ	HO
ነ	NE	ኑ	NU	ኒ	NI	ና	NA	ኔ	NE	ን	N	ኖ	NO
ኘ	GNE	ኙ	GNU	ኚ	GNI	ኛ	GNA	ኜ	GNE	ኝ	GN	ኞ	GNO
አ	A	ኡ	U	ኢ	I	ኣ	A	ኤ	E	እ	I	ኦ	O
ከ	KE	ኩ	KU	ኪ	KI	ካ	KA	ኬ	KE	ክ	K	ኮ	KO
ኸ	HE	ኹ	HU	ኺ	HI	ኻ	HA	ኼ	HE	ኽ	H	ኾ	HO
ወ	WE	ዉ	WU	ዊ	WI	ዋ	WA	ዌ	WE	ው	W	ዎ	WO
ዐ	A	ዑ	U	ዒ	I	ዓ	A	ዔ	E	ዕ	I	ዖ	O
ዘ	ZE	ዙ	ZU	ዚ	ZI	ዛ	ZA	ዜ	ZE	ዝ	Z	ዞ	ZO
ዠ	ZHE	ዡ	ZHU	ዢ	ZHI	ዣ	ZHA	ዤ	ZHE	ዥ	ZH	ዦ	ZHO
የ	YE	ዩ	YU	ዪ	YI	ያ	YA	ዬ	YE	ይ	Y	ዮ	YO
ደ	DE	ዱ	DU	ዲ	DI	ዳ	DA	ዴ	DE	ድ	D	ዶ	DO
ጀ	JE	ጁ	JU	ጂ	JI	ጃ	JA	ጄ	GE	ጅ	J	ጆ	JO
ገ	GE	ጉ	GU	ጊ	GI	ጋ	GA	ጌ	TE	ግ	G	ጎ	GO
ጠ	TE	ጡ	TU	ጢ	TI	ጣ	TA	ጤ	CHE	ጥ	T	ጦ	TO
ጨ	CHE	ጩ	CHU	ጪ	CHI	ጫ	CHA	ጬ	PE	ጭ	CH	ጮ	CHO
ጰ	PE	ጱ	PU	ጲ	PI	ጳ	PA	ጴ	TSE	ጵ	P	ጶ	PO
ጸ	TSE	ጹ	TSU	ጺ	TSI	ጻ	TSA	ጼ	TSE	ጽ	TS	ጾ	TSO
ፀ	TSE	ፁ	TSU	ፂ	TSI	ፃ	TSA	ፄ	TSE	ፅ	TS	ፆ	TSO
ፈ	FE	ፉ	FU	ፊ	FI	ፋ	FA	ፌ	FE	ፍ	F	ፎ	FO
ፐ	PE	ፑ	PU	ፒ	PI	ፓ	PA	ፔ	PE	ፕ	P	ፖ	PO

Appendix B: Importing Python libraries for Geez Grammar Checker

Import my Sql .connector for Run SQL queries from Python, retrieve or update data in MySQL.

import Json a standard format for data exchange **use for** Convert between Python objects and JSON strings.

import nltk a powerful library for processing and analyzing human language (text) use for Tokenization, stemming, tagging, parsing,

from nltk.util import bigrams a function from NLTK that returns pairs of consecutive words from a list Used in NLP to study word relationships, build language models.

import re use for Find, match, replace, or split text using complex rules.

import itertools use for Generate permutations, combinations, infinite sequences, etc.

Appendix C: Sample Bigram Formation

```
import pandas as pd
import re

# Load Excel file (no header)
df = pd.read_excel("C:\\Users\\lab7\\Desktop\\fip\\sample corpus.xlsx", header=None)  # Replace with your file

# Assuming text is in the first column (index 0)
text_column_index = 0

# Unicode delimiters
sentence_end_chars = [chr(0x1362), chr(0x1364), chr(0x003F)]  # ።, ፤, and ?
sentence_splitter = '|'.join(re.escape(ch) for ch in sentence_end_chars)
word_delimiters = r'[\u1360-\u1368\u0020]+'  # 0x1360-0x1368 and space

# Store bigrams (as 2-column lists)
all_bigrams = []

# Process each row
for index, row in df.iterrows():
    text = str(row[text_column_index])

    # Split into sentences
    sentences = re.split(sentence_splitter, text)

    for sentence in sentences:
        sentence = sentence.strip()
        if not sentence:
            continue
            # Tokenize sentence into words
        words = re.split(word_delimiters, sentence)
        words = [word for word in words if word]
```

```
        # Generate bigrams and store/display them
        for i in range(len(words) - 1):
            w1 = words[i]
            w2 = words[i + 1]
            all_bigrams.append([w1, w2])
            print(f"{w1} {w2}")

# Create DataFrame and export to Excel
bigrams_df = pd.DataFrame(all_bigrams, columns=["Word 1", "Word 2"])
bigrams_df.to_excel("bigrams_output.xlsx", index=False)

print("\nBigrams exported to bigrams_output.xlsx")
```

Appendix D: Sample Bigram Count

```
import pandas as pd
from collections import Counter

# Set file paths
input_file = "C:\\Users\\lab7\\Desktop\\fip\\bigram_input.xlsx"     # Replace with your actual file
output_file = "bigram_count.xlsx"

# Read the Excel file (expects bigrams in the first and second columns)
df = pd.read_excel(input_file)

# Check if the file has at least two columns
if df.shape[1] < 2:
    raise ValueError("Excel file must have at least two columns for bigrams.")

# Form bigrams from the first two columns
word1 = df.iloc[:, 0].astype(str)
word2 = df.iloc[:, 1].astype(str)
bigrams = list(zip(word1, word2))

# Count the frequency of each bigram
bigram_counts = Counter(bigrams)

# Create a new DataFrame from the bigram counts
result_df = pd.DataFrame(
    [(w1, w2, count) for (w1, w2), count in bigram_counts.items()],
    columns=["Word1", "Word2", "Count"]
)
# Sort by count in descending order
result_df = result_df.sort_values(by="Count", ascending=False)
# Export to a new Excel file
result_df.to_excel(output_file, index=False)
print(f"Bigram counts saved to '{output_file}'")
```

Appendix E: Sample Split

```
# Step 2: Split text into sentences
def get_sentences(text):
    sntncList = [sentence.strip() for sentence in re.split(sentence_splitter, text) if sentence.strip()]
    return sntncList
```

Appendix F: Sample Tokenize

```
# Step 3: Tokenize sentence into words
def get_words(sentence):
    words = [word for word in re.split(wordDelimiter, sentence) if word]
    return words
```

Appendix G: Sample Handling untagged words

```
# Step 5: Handle untagged words and record errors
def handle_untagged_words(wordTagJObject, curSntnc):
    for word, tags in wordTagJObject.items():
        if "NTG" in tags:
            key = f"- The sentence {curSntnc} contains untagged word(s):"
            value = ', '.join([word for word, tags in wordTagJObject.items() if "NTG" in tags])
            grmrErrorDct[key] = value
```