Addis Ababa University

College of Natural Sciences

Development of Spell Checker for Guragina Language

Mengistu Gebre Bokane

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

March, 2024

Addis Ababa University

College of Natural Sciences

Mengistu Gebre Bokane

Advisor: *Yaregal Assabie (Ph.D.)*

This is to certify that the thesis prepared by *Mengistu Gebre Bokan,* titled *Development of Spell Checker Algorithm for Gurage Language,* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the examining committee:

	Name	**Signature**	**Date**
Advisor:	Yaregal Assabie (PhD)	_____	_____
Examiner:	Ayalew Belay (PhD)	_____	_____
Examiner:	Mesfin Kifle (PhD)	_____	_____

ABSTRACT

A spell checker is an essential tool in Natural Language Processing (NLP). Its purpose is to identify and correct spelling errors in text, providing suggestions for correct spellings in a specific language. Spelling errors can be categorized into two types: non-word errors and real-word errors. Non-word errors are misspelled words that have no meaning in the particular language, while real-word errors involve words that exist in the language but are used incorrectly in terms of semantics and syntax.

The research focused on non-word error detection as a strategic decision, given the complexity and limited resources available for the Gurage language, also known as Guragina. This language consists of over thirteen varieties and different orthographies, but there is a modern standard. Currently, there is no existing spell checker for any Guragina Language varieties or the standard. Addressing non-word errors first provides a solid foundation before tackling the more challenging task of real-word error detection and correction. This phased approach allows researchers to make meaningful progress on this under-resourced language, rather than attempting to solve the entire spell checking problem at once. The intention is to use the non-word spell checker as a starting point, then leverage that knowledge to progressively tackle real-word error handling.

This work introduce a non-word spell error checker for the standard Guragina Language. The system detects and corrects errors using Ratcliff algorithms for identification and distance calculator techniques for correction. The prototype of the system was developed using Python.

We evaluate the performance of the system using metrics such as accuracy of 98.27%, precession of 98.07%, recall of 97.75%, and F1 Score of 95.45%. Future work includes enhancing rule definitions by incorporating word classes, handling exceptions, adding supplementary spell checker functionalities, and expanding the system to encompass real-word errors.

Keywords: *Spell Checker; Gurage Language; Morphology, morphological analysis, linguistic resources, rule-based approach, under-resourced languages*

ACKNOWLEDGEMENT

First and foremost, I want to express my deep gratitude to the Almighty God and Virgin Mary for blessing me with good health, wisdom, strength, and grace, which enabled me to successfully complete and present this research study.

I would like to extend my heartfelt appreciation to my advisor, Dr. Yaregal Assabie, for his invaluable guidance, encouragement, and constructive feedback throughout the entire process. From the initial selection of my research topic to the final stages of the study, his unwavering support and motivation have played a vital role in my progress. I would also like to thank Dr. Fekede Minuta and Mr. Bahiru Lilaga for their invaluable linguistic support, as well as the Gurage Zone Cultural Tourism Office for providing the necessary materials for data collection.

I sincerely thank Mr. Ayechew Tefeta and the entire management team at my office for their encouragement and willingness to dedicate time to support my study. Their support has been crucial in helping me complete this research.

To my beloved wife, Blene (ብሌኔ), and my children, Eyoab and Aminen, I am deeply grateful for your unwavering support and encouragement throughout this journey, from the initial stages of topic selection to the presentation of my final defense.

I would also like to acknowledge and express my gratitude to my dear family, especially my mother Adanech Dereja, and my late father Gebre Bokane (even though you are no longer with me, this was your dream). Your love, encouragement, and support have been constant sources of strength throughout my academic life. Additionally, I extend my thanks to Mr. Mitku, all my siblings, and my friend Zelalem for their kind support.

In conclusion, I am sincerely grateful to all those who have contributed to my research study, whether through their guidance, encouragement, or support. Your presence in my life has made a significant difference, and I am truly indebted to each and every one of you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF FIGURES

ACRONYMS

Bi-LSTM	Bidirectional Long Short-term Memory
GL	Guragina Language
GLSC	Guragina Language Spell Checker
DCAUS	Direct Causative
DLT	Dictionary Lookup Technique
DCAUS	Indirect Causative
GWDA	Guragina Word Distance Algorithm
IPA	International Phonetic Alphabet
IPS	Impersonal
MA	Morphological Analysis
MT	Machine Translation
OCR	Optical Character Recognition
NLP	Natural Language Processing
QA	Question Answering
SC	Spell Checker
VC	Vowel Consonant

CHAPTER ONE: INTRODUCTION

1.1. Background of the Study

Natural Language Processing (NLP) is the computer-based approach to text analysis that is based on a set of theories as well as a set of technologies. The ultimate goal of NLP, as a branch of artificial intelligence, is to read, decode, understand and make sense of human languages in a meaningful way [1]. There are too many languages, most of them have different alphabets and a lot of rules. Their treatment of using a single linguistic tool is practically difficult. These are strong reasons to develop different tools for various languages including spellchecker.

In NLP, a spell checker (spelling checker) is one of the most important areas in the field, which focuses on detecting and correcting spelling errors in a text. Currently, many NLP applications use SC tools either independently or embedded within another system [2].

In different languages, spelling errors are mainly caused by insertion (insertion of the wrong character), deletion (omission of a letter), substitution (using one letter instead of the other), and transposition (miss positioning of letters). According to Kukich [3], spelling errors are generally split into two types: Typographic and Cognitive. Typographic errors occur when the correct spellings of the word are known, but the word is mistyped by mistake. These errors are mainly associated with the keyboard and therefore do not follow any language criteria.

The typographical errors fall into one of the four categories. A study by Damerau [4] shows that 80% of typographical errors come from one of the following four categories. Spelling error detection is the main thing in spell checking to detect the errors in written text. It is used to flag out misspelled words in a text. To suggest a set of possible corrections, a spellchecker has to identify the first misspelled words. The most important approaches that are used to develop an SC system are dictionary look-up and N-gram approaches [3], [7], [8].

Ethiopians speak around 80 different languages [9]. The majority of Ethiopian languages, including Gurage Language (GL), are under-resourced, as are the majority of global languages, in that they lack proper language processing tools and methodologies.

The language known as Gurage (GL) is referred to as Guragina and falls under the South-Ethiosemitic division of the Semitic branch within the Afroasiatic language family. It encompasses approximately thirteen different variations. The Guragina language is further divided into sub-groups, namely North Gurage, West Gurage, and East Gurage. The existence of these dialectal

variations and other related factors poses challenges when attempting to create a unified natural language processing (NLP) application, including spell checker (SC). However, despite the variations, a standard Guragina language and orthography have been recently established. This study primarily focuses on the development of a spell checking and correction system for the standardized Guragina language.

1.2. Motivation

Spell-checking is one of the most important applications, used to generate spelling error-free documents. Spell-checking techniques developed and implemented in languages [5] such as English, Arabic, Indonesian, Bangla, Indian, etc.

It is very common practice to see a lot of spelling errors in typed Guragina documents. Even fluent speakers and writers are discouraged to write for fear of being misspelled and/or do not know how to write using Guragina orography. In addition, most people who type Guragina make mistakes and do not even notice them. Even after the manual correction, it is still very common to see misspellings in printed or electronic documents that lead to misinterpretation of the meaning of a word. This has motivated us to develop an SC for the GL.

1.3. Statement of the Problem

A spell checker is considered one of the most important NLP applications used to enhance the success rate of NLP applications, like machine translation, search engines, social media, word segmentation, proofreading, information retrieval, and natural language understanding [10]. Especially, for the languages with low resources and lack of computer tools, like Guragina, SC plays a vital role.

The GL is one of the morphologically complex SOV and low-resource languages, with many varieties. Spell checker tool developed and implemented for one language cannot be applied to others directly [11]. To develop most NLP applications such as SCs for such languages, it may need to advise the language's expertise, understand the language morphology (word categories and their possible inflections, derivation, and compounding), and other its unique characteristics. For this reason, it is difficult to develop an SC system easily and needs research work for Gurgigna. On the other hand, most varieties of Guragina Language are morphologically similar and share

most of their linguistic features. To take advantage of this, we are interested in doing a pattern-based spell checker for the Standard Guragina Language.

Although SC is very important to the GL, to our knowledge, there is no study undergone about SC for Standard GL and any of its varieties. This work is very important not only as an SC but also will help to develop and preserve the language by encouraging and supporting the language users to write and communicate by using this language on different platforms.

Generally, the first and great sense that made us this research is that different studies have been performed on SC worldwide and some attempts of a few Ethiopian languages like Amharic, Afan Oromo, Tigrigna, etc., but not at all with Standard Guragina Language.

1.4. Objectives

1.4.1. General Objective

The main objective of this thesis is to develop a spell checking and correcting system for the Standard Guragina Language that leverages a morphology-based approach.

1.4.2. Specific Objectives

To accomplish the aforementioned general objectives, the following specific tasks have been undertaken:

- Review related research works to understand the morphological structure of Guragina words and to identify the state-of-the art in spell checking
- Collect a list of Guragina words used for testing.
- Develop an system that check and suggest corrections
- Develop the prototypes of the system
- Evaluate the performance of the system

1.5. Research Methodology

The research methodology employed in this study is the design science research methodology [12]. This methodology focuses on generating an artifact in the form of a model for a Guragina Language spellchecker. As a result, a carefully developed and tested artifact or prototype is produced, incorporating an algorithm to identify and correct spelling errors in Guragina.

Literature Review

Our research involved an extensive examination of diverse literature sources such as books, journals, articles, online resources, thesis, and dissertations. The purpose of this comprehensive review was to acquire relevant information for our study. Additionally, we actively sought input from linguistics experts to enhance our comprehension of the structure and characteristics specific to the Guragina Language. By collaborating with these experts, we were able to delve deeper into the linguistic aspects of the Guragina Language and explore established approaches for detecting spelling errors across various languages.

Data Collection

To carry out experiments on corpus-based SC collection of various forms of organized words is necessary. We collect and organize the text corpus for this research work from Welkite University, various medias, Gurage Zone Culture and Tourism Office.

Afterward, the data pre-processing was the next step in data collection. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data pre-processing is a proven method to resolve such issues. Data pre-processing prepare raw data for further processing. We collected a list of stem words of verb, nouns, adjectives, adverbs and various affixes through a combination of manual and semi-automated methods. This involved the systematic identification and extraction of affixes and derivations from individual words. By utilizing these affixes and deriving from a single word, we generated a wide range of derived words, including verbs, nouns, adverbs and adjectives. These derived words encompass various parts of speech commonly found in Guragina.

Prototype Development

We build a prototype to validate our work using the different tools. To develop the prototype of the system, appropriate supporting tools are used to facilitate the study. These include a morphological analyzer, the latest version of the Python programming language, and other free tools. Python is an interpreter, interactive, object-oriented programming language. Python combines remarkable power with very clear syntax. It has interfaces to many systems calls and libraries, as well as various window systems [13]

Evaluation

In this research, we use accuracy, recall, precision and F1- score as main evaluation metrics to measure the effectiveness of the selected approach for a Guragina Word Distance Algorithm (GWDA).

1.6. Scope and Limitations

This study specifically focuses on detecting and correcting spelling errors within the endorsed variants of the standard Guragina Language, among its various dialects. It is important to note that the Guragina writing system did not have consistent guidance on orthography until 2020, resulting in variations. In this study, we adopt a new standard and indorsed orthography by Gurage zone administration council from among these variations.

The spellchecker system identifies correctly spelled words as valid and flags incorrectly spelled words as invalid, while also attempting to suggest similar words as alternatives. However, it is crucial to acknowledge that the lack of a standardized text corpus and limited digital resources for the Guragina Language pose limitations to this study. The amount of prepared data for this study is also relatively minimal due to these constraints.

1.7. Application of the Results

The system, which can detect and correct spelling mistakes in a language, will help significantly the writing process on both offline and online applications. Users of the GL are can encourage writing and communicating in the language. This work will be very significant in various areas. Some of these include the following:

- In the learning and teaching processes of Guragina
- To generate a Guragina document with minimal or no spelling mistakes and save time for the writer.
- For the development of the language by encouraging writers with the language.
- For Language Standardization: Spell checkers play a role in language standardization efforts by promoting consistent spelling conventions. They help establish and enforce

commonly accepted spellings within a language, contributing to linguistic cohesion and clarity.

- In the development of other NLP applications like grammar checker, machine translation, OCR, information retrieval, text synthesis, text extraction, speech text synthesis, etc.
- For anyone who wants to communicate through the text of GL offline or online.

As a result, this work will represent a big step toward detecting spelling errors in the Guragina language, as well as a foundation for further NLP applications in the language. In addition to verifying the accuracy and professionalism of the document. Furthermore, it will save time and effort for a person to manually find misspellings throughout the document. Not only will this study's application benefit anyone who wants to process Gurgigna documents, but it will also provide a good opportunity for other researchers who want to develop other NLP applications using the language. In general, this work will aid language development through motivating writers, as well as teaching and learning activities.

1.8. Organization of the Thesis

The remaining part of this thesis is structured as follows. Chapter 2 covers the background, linguistic characteristics, morphology, and writing system of the Guragina Language. Additionally, this chapter goes through various types of spelling errors, and spell-checking techniques. Chapter 3 presents the relevant research on the use of spell checkers and associated topics. The design of the Guragina Language spell checking system is covered in the fourth chapter. The findings of the experimental spell-checking system will be shown in the fifth chapter. The last chapter, Chapter Six, will include a conclusion, a contribution, and future works.

CHAPTER TWO: LITERATURE REVIEW

This chapter gives a description of various spellchecker studies on both the local language and other languages using different techniques are discussed. It was used to obtain general information on the spellchecker, the GL writing system and the different approaches of SC techniques. In addition, presents a description of the state-of-the-art concepts related to our work. From Section 2.1 through Section 2.3 a step-by-step discussion of the sociolinguistics of Gurage region, the origin of Gurage and Guragina Language varieties, its writing system and its morphological characteristics is provided. The topics of general spelling errors and the various methods for error detection and correction are covered in Section 2.4. We provide a brief summary of this chapter in Section 2.5.

2.1. Socio-Linguistic Situation in the Gurage Region

The term "Gurage" refers to the population that the South Ethiopian-speaking community in the Central Ethiopia Regional state of Ethiopia, one of the most recent regions to be split off from the Southern Nation Nationality region. Geographically, Gurage is situated approximately 100 km far from south of Addis Ababa, which is more or less bordered by the Rift Valley lakes in the East, River Awash in the North and River Gibe in the West and Southwest in general [9].

The origin of Guragina varieties has been a subject of academic debate for decades among Semitists. The debate emerges from the enduring lack of clarity about the origin of the Semitic languages in general[15], [16].

Regarding where the Guragina variety came from, there are three hypotheses. The first one presumes that the Gurage people originated in modern-day Eritrea, in a region known as Akale Guzay under the rule of Amdetsion (1314-1344). Consequently, it is believed that the Guragina Language's forefathers were King Amdetsion's troops. Oral tradition has offered evidence for this notion. According to the second hypothesis, the ancestors of the Guragina speakers migrated from east of Ethiopia during the expansion of Ahmed Gragn (1524-1543), a Muslim warrior. The Islamist militia and army moved to the Gurage region during the expansion. The third theory holds that the Guragina variations have African roots and were spoken in southern Ethiopia prior to Amadetsion's rule and the expansion of Ahmed Grange. According to Menuta [14], the notion that

Ethiopian Semitic is the ancestor of all Semitic languages also raises the possibility that Guragina variants are Ethiopian in origin and were spoken there from the start. The origins of the Gurage people and the Guragina variations have mostly remained a mystery.

There were also different oral traditions regarding the origin of the language and the people of Gurage. Based upon [17], the term, Gurage is from Gura - somewhere in Eritrea. The oral tradition is that the suffix "ge" means "land" or "village," whereas the word "Gura" means "the people," which suggests that "Gurage" means the land or country of the Gura people. There are other words along those lines that seem to support that argument.; for example, Harar-ge ‘the land of the Harari people’, Abessh-ge ‘name of a district in Gurage Zone’, Arat-ge ‘a place in Kistane’ [14]. The author says that "Gura" is a common word among Gurage communities that means "left", the opposite of "right". Hetzron [18], opposes this oral tradition concerning the origin of the Gurage people and language. According to the author, the traditional argument is not supported by any linguistic evidence.

Socially, the Gurage communities socially organize themselves in different ways. Ethnic hierarchy is one means of organization which consists of Tib, Bet, Wefencha and Den[14]. Tib is the highest social hierarchy. It encompasses almost all Gurage clans. Bet is the second broad social hierarchy which also consists of several Gurage clans. Wefencha is the third social hierarchy which consists of sub-clans within the clan. Den is the fourth and the smallest social hierarchy which usually refers to members of a sub-clan [14].

Yejoka is a traditional justice System for high-level societal decisions to be made as well as a method for the administration of justice, by making meeting in the most common places and Yejoka. Its main responsibilities include defending the community from external aggression, enacting administrative and judiciary laws and supervising the implementations of the fundamental rules of low. This system is takes place under podocarpus tree or Yejoka by elder’s by using customary law called Kitcha.

With regard to the economic activities, the Gurage people are predominantly farmers [14], [19]. The main crop of the area is ‘Ensete’ from which they make different kinds of food, mainly a kind of bread called wusa. Wild bananas usually grow in the lowland and semi-highland areas of the Gurage Zone. The extreme highlanders, such as Muher do not plant wild bananas since they are not productive in such a cold area [19]. Besides wild bananas, the Gurage people produce different

kinds of cereals such as barley, wheat, maize, pepper, lentils, ‘teff’ and commercial crops such as coffee. Gurage people also rear animals such as cows, sheep, goats, horses, mules and donkeys. Above all, they are well-known as traders among the remaining Ethiopian communities.

Gurage people are highly mobile [14]. A large number of people live outside Guraage zone. The current establishment of Gurage is the result of a variety of social and historical reasons. Migration, the movement of the Gurage people from the Gurage region to other parts of Ethiopia, is called fannonet amongst the Gurage communities, also has its own influence on the current linguistic situation in the Gurage region.

Nowadays, many Gurages are businessmen in the major cities of Ethiopia where Amharic is the dominant language; Amharic has been imported into the Gurage area in part by these traders. Guragina varieties are used solely as spoken languages (except Silt'e), and are restricted to domestic and commercial communication. Because of this and other several factor Gurage community were remaining decentralized; this weakened the people of Gurage and Guragina Language [14]. Recently, the communities of Gurage were institutionalized together, with a centralized administration. It being an opportunity to the Gurage community and disaggregated language and culture together [14].

2.2. Overview of Guragina Language

The linguistic variations discussed in this study are primarily spoken in the southwestern region of Ethiopia, located approximately 158 km from Addis Ababa. This area is traditionally referred to as the Gurage area, and the term "Gurage" is used to describe both the linguistic features of the region and the people residing there [14], [17]. However, it's important to note that not all communities in the Gurage region are Semitic, such as the Libido and K'abeena communities.

Ethiosemitic languages belong to the Semitic language family and are spoken in Ethiopia and Eritrea. They are broadly categorized into North and South Ethiosemitic. The South Ethiosemitic varieties are spoken in the central, eastern, and southwestern parts of Ethiopia. This study specifically focuses on the Guragina Languages, which are a group of languages spoken in the southwestern region of Ethiopia.

In sociolinguistics and stylistics, the term "variety" refers to any distinct linguistic system. The Gurage area is linguistically highly diverse, with different language varieties often further divided into dialects and sub-dialects. This area is situated within the Ethiopian linguistic area, which is known for its remarkable linguistic diversity [19]. Guragina belongs to the Semitic branch of the Afroasiatic language family and comprises approximately thirteen varieties [22].

2.2.1. Variety of Guragina Language

Distinguishing between dialects and languages and quantifying the similarities between different language varieties have been fundamental questions in dialectology. Typically, the measurement of linguistic distance between two or more language varieties is used to approach these questions [16]. In general, the greater the structural similarity between two languages (phonetic, morphological, lexical, or syntactic), the closer they are considered to be. When the structural similarities are significant, the varieties are often regarded as dialects of the same language [19]. However, the distinction between dialects and languages is more complex, and non-linguistic factors (such as social, cultural, political, and psychological aspects) often play a crucial role. This means that determining linguistic distance based solely on structural similarity may not be sufficient to determine whether two varieties should be classified as dialects of the same language or as separate languages [23].

Currently, the Gurage zone is administratively divided into thirteen districts: Ezha, Geto, Gumer, K'abeena, Libido, Mesqan, Kistane, Abeshge, Chaha, Endegagn, Inor, Muhir-Aklil, and Wolane. It's worth noting that the Gurage region is home to speakers of non-Semitic languages such as K'abeena and Libido [14].

According to Menuta [14], the Gurage zone is further subdivided into four agroclimatic zones: alpine (above 3200 meters), temperate (between 2300 and 3200 meters), subtropical (between 1500 and 2300 meters), and tropical (between 1100 and 1500 meters above sea level). The alpine climate zone surrounds Gurage Mountain, also known as Zebidar, which reaches an altitude of 3600 meters. This mountain acts as a geographical barrier, dividing the Gurage area into east and west and hindering inter-group contact between the people of Gurage East and Gurage West. Linguistically, this mountain has been used as a reference point, with the Eastern Guragina language referred to as "Oriental Gurage" and the Western Gurage language as "Occidental

Gurage" [28]. From a linguistic perspective, the sub-groups of Guragina, as described in Section 1.1, can be categorized as follows: North Gurage, which includes Kistane and Dobi; West Gurage, which encompasses Meskan, Cheha, Ezha, Gumer, Muher, Inor, Endegagn, Enner, and Geto; and East Gurage, which consists of Welane, Silte, and Zay. Due to the existence of these various dialects and other related factors, developing NLP application, including a spell checker is challenging.

A survey conducted by Menuta [14] shows that almost all Guragina speakers are bilinguals. Speakers of the Guragina varieties have a frequent contact among each other. Many of them have contact with Chaha, compared to other Guragina varieties. The majority of the participants believe that they understand Chaha better than other Guragina varieties. They also attribute positive value to Chaha.

Gurage Mountain was mentioned as the main geographical barrier to contact among the Guragina speakers. For instance, Mesqan and Kissane do not have a frequent contact with the speakers of Chaha, Muher and Inor because of this barrier. Lack of public transport and hostility among the Gurage clans are indicated as other obstacles [14]. In market places, Guragina is more frequently used than Amharic. In schools, religious places and administration, Amharic is the dominant language. According to this study, Amharic (63%) is more frequently used at home than Guragina (42%).

The reorganization of the Gurage ethnic group and the standardization of linguistic varieties continued to present challenges. The biggest challenge has been the ethnic diversity within the Gurages, and the disagreements between the Gurages ethnic groups and the political elites about language materials to be taken into account in the standardization process [20], [21].

In 2013, the Gurage Zone established a Language Board to promote and standardize Guragina, and introduced an official Guragina script, and endorsed by Gurage zone administration council in 2020. According to [24], the aim of improvement of this orthography is to make it simple, pattern based and also to standardize it.

In the process of Guragina Language standardization, in the endorsed language of Gurage, particularly Cheha plays a central role, along with incorporating certain vocabulary from other language varieties. As a result, our detailed analysis primarily centers on the morphology of Cheha, considering its significance in the standardization efforts.

2.2.2. Guragina Language Writing System

Guragina is written left-to-right, separated by white space, which uses Ethiopic script as a subgroup of Ethiopian Semitic language. For over half a century the Gurgege language has used the Amharic Fidel with some additional letters developed for sounds which are not found in Amharic. The Guragina orthography underwent seven separate modernization attempts, mostly in the palatalized and labialized graphemes, creating short "eras" in the orthography since 1950 [20]. These different writing systems were used to publish some fictional work in Guragina Language as well as a New Testament, called Geder Gurda, and the complete Bible called Meta'af Qidus. As the Guragina Language was not in past used in institutions such as media, education and administration bureaus, the orthography was not officially recognized. Because of this, Guragina variants are only used orally, with the exception of Silt'e, which has been written since 1982 and is teaching in primary schools since 1995 [20].

Like the original Ethiopic script, the current Guragina script is an alpha syllabary with symbols that mainly represent consonant-vowel sequences. On September 14, 2021, the Unicode Standard version 14.0 was formally adopted, giving the Modern Guragina Orthography set's letter additions official acceptance on a global scale [27]. We can read many books which are written in this standardized Guragina language and its modern orthography.

As presented in Appendix 1, the complete modern Guragina character set contains 31 base characters (consonant) vertically arranged in seven columns (vowels), which gives 217 characters, and the other 8 characters vertically listed in 5 columns, which gives 40 characters. This shows that, the Guragina syllabary, or "Fider", comprises total of 257 syllographs. Even though many of the letters or orthographies of Amharic and other Ge'ez scripts, there are some variances. For instance, Guragina's modern orthography gives new shapes to five velar syllables that had previously been visually identical under the writing practices of other Ethiosemitic languages (for example, ጋ *'gʷə'* vs ጐ *'gʷə'* …). Second, there is only one 'ha' (ሐ) in Guragina as opposed to five in Amharic. The usage of words like "ኣ in place of ኧ, ኣ for ኣ, and ሐ in place of ኸ "among others, as part of modern convention are another distinction. Not only that, but "and" also ኣ and *ዐ* has the same sound in Amharic but not in Guragina. Last but not least, there are several characters like ኽ *'kʲə'*, ሕ *'hʲə'*, ኻ *'gʲə'*, ቡ *'bʷə'*, ሑ *'hʷə'*, ሙ *'mʷə'*, ቈ *'kˠʷə'*, ፉ *'fʷə'*, ጕ *'gʷə'*, and ፕ *'pʷə'*, that are exclusively present in Guragina and not in Amharic. In Guragina,

these letters are rather common. A designed version of the Garagina keyboard has just been accessible for all popular operating systems [4]. The most recent Guragina script (seven orders of consonants and vowels) was shown in Appendix 1, which is adopted from [20].

2.3. Morphology

Morphology is about the structure of words. Word is the key element in morphology. All languages have words and in all languages some words, at least, have an internal structure, and consist of one or more morphemes.

Morphemes are the smallest meaningful unit of the structure of language whereas the biggest units are words. For instance, the word "cats" is made up of the root morpheme "cat" and the suffix morpheme "s," which indicating plural.

Guragina is a Semitic language, and Semitic languages generally have complicated morphologies. It shares linguistic roots with other Semitic languages like Hebrew, Arabic, Amharic, and Syrian.

2.3.1. Verb

A verb is the most crucial element of a sentence is the part that describes the subject of the clause, expresses an action (cut, halt, search), an occurrence (snow, happen), a change (include, shrink, dissolve, broaden), or a state of being (e.g., is, be, have). Clauses and sentences cannot be formed from a set of words without a verb.

The Guragina Language is also characterized by the typical root-and-pattern morphology common throughout all Semitic languages. The root-and-pattern morphology's fundamental idea is as follows. First, there is a root, called a radical, comprises of a set of consonants with a basic lexical meaning (core meaning of a word) and arranged in a particular order [7]. Next, there are various regular patterns, also known as templates, that control the quality and placement of vowels in relation to the radicals[10]. Radicals do not naturally appear as words; rather, they must be incorporated into a template to serve as a base.

It is necessary to be familiar with the verbal systems of the Semitic languages in order to understand how the root and pattern morphology evolved and persisted. The Semitic verb morphology is described based on the complex non-concatenative/root-and-pattern system [25].

Guragina is one of these languages, and it describes the simple verb stems using a non-concatenative/root-and-pattern morphological process [10]. According to [26], in many languages, including Guragina, the verb shows more morphological complexity than any other word class. Even it shows the most complex form in Guragina [19]. As shown in table 2.1, from Guragina root word ቸነ/ ʧənə we can drive different word for different person.

Table 2. 1 *Derivation of ʧənə for different person.*

Example	**Phonetic Representation**	**Suffix**	**Translation**
ቸነ-ሑም	*ʧənə-hʷɨm*	*hʷɨm*	I came
ቸነ-ኽም/ቸነ-ሐም	*ʧənə-hʲɨm /ʧənə-həm*	*hʲɨm/ həm*	You came
ቸነ-ችም/ቸነም	*ʧənə-ʧɨm / ʧənə-m*	*ʧɨm / m*	s/he came
ቸነ-ነም	*ʧənə -nəm*	*nəm*	we came
ቸነ-ማም/ቸነ-ቦም	*ʧənə-mam/ ʧənə- bom*	*mam / bom*	they came

The verbs are can be derived from the simple verbs by affixes. The other method is derivation of verb by interdigitating vowels into the C-pattern to create different verb stems, and the roots (root radical) are used to make the inflectional verb conjugations.

In Guragina, verbs can range in length from one radical to four radicals [22]. Similar to tri-consonantal root verbs, mono-radical verbs are also common, particularly in Guragina variants [10]. For instance, the sentential verbs : akʲə “He chew.”, epə “He worked”, ela “He wished”, amə “He gave.”,efə “He covered.” ʃə “He wanted" are realized as they have mono-radicals in their surface structure [14].

Derived Stems

The verbal process of derivation illustrates how root morphemes change in the verbs' meaning in relation to their lexical and semantic content. This morphological process involves change of forms, increase number of root-morphemes, and change of syntactic category and semantic usage [19]. In Guragina, the derivations of complex verb stems are the passive marker /tə-/, causative marker /a/t-/, and verbal noun /-ot or wə-/ which are exemplified in Table 2.2.

Table 2. 2 *Derivations of Guragina complex verb stems with tə-*

Root	Base	Translation	tə-stem	Translation
s-b-r	*səpər -*	break	*tə-səpər-*	was broken
b-l- a	*bəna-*	eat	*tə-bəna-*	was eaten
a-n-t'	*ant'-*	cut	*t-ant'ə-*	was cut

Furthermore, the bound morpheme /tə-/ is also used in reciprocal expressions. Here, the penultimate consonant of the triliteral radical is repeated and in between the first radical, and penultimate radical the low central vowel /a/ is inserted. Consequently, as exemplified in the in the Table 2.3, they have the template slot tə-CəCaCəC- in the canonical triradical root morphemes. For instance, the t-stem *təsəpər-* becomes *təsəpapər-o* "they broke each other".

Table 2. 3 *Reduplication of word with morpheme tə*

Passive	Translation	Reduplicative stem	Translation
tə-səpər-	was broken	*tə-səpapər-*	break one another
tə-ʃəgər-	was exchanged	*tə-ʃəgagər-*	exchange one another
tə-agəz-	was helped	*tə-gagəz-*	help one another

Causative Stem

In Guragina, there are two types of causatives markers, direct causative and indirect or a double causative. The intransitive stems are mainly where the direct causative is derived. It is derived with /a-/. The passive and indirect causatives are both represented by the indirect causative and it derived with /at-/. The examples in Table 2.4 demonstrate the derivation of causatives (direct and indirect), in perfective and imperfective aspects, and the jussive mood.

Table 2. 4 *Derivation of Causatives stem in Perfective, Imperfective and Jussive words*

Verb	Translation	Type	Perfective	Imperfective		
			Past	Present	Future	
wəndə	Get down	BASIC	*wənd-m*	*ji -wərəd*	*ji-wərd-te*	*jə-wərd*
		DCAUS	*a-wənd-m*	*a-wənd*	*ji-wərd-te*	*j-a-wərd*
		ICAUS	*at-wənd-m*	*j-at-wərd*	*j-at-wərd-te*	*j-at- wərd*
bəna	Eat	BASIC	*bə na-m*	*ji-bə ra*	*ji-bəra-te*	*jə -bra*
		DCAUS	*a-bəna-m*	*j-a-bəra*	*j-a-bəra-te*	*j-a-bra*
		ICAUS	*at-bəna-m*	*j-at-bəna*	*j-at-bəna-te*	*j-at -bəra*
səʧ'ə	Drink	BASIC	*səʧ'ə-m*	*ji-səʧ'*	*ji-səʧ'-te*	*jə-st'e*
		DCAUS	*a-səʧ'ə-m*	*j-a-səʧ'*	*j-a- səʧ'-te*	*j-a-stʧ'*
		ICAUS	*at-səʧ'ə-m*	*j-at-səʧ'*	*j-at- səʧ'-te*	*j-at-siʧ'*
sijə	Buy	BASIC	*sijə -m*	*ji-sijə*	*ji-sijə -te*	*jə -sə jə*
		DCAUS	*a- sijə-m*	*j-a-sijə*	*j-a-sijə-te*	*j-a-səjə*
		ICAUS	*at- sijə-m*	*j-at-sijə*	*j-at-sijə-te*	*j-at-səjə*
kəna	Climb	BASIC	*kə na-m*	*ji- kə ra*	*ji- kə ra-te*	*jə -kra*
		DCAUS	*a-kəna-m*	*j-a- kəra*	*j-a-kəra-te*	*j-a-kra*
		ICAUS	*at-kəna-m*	*j-at- kəra*	*j-at-kəna-te*	*j-at-kira*

Verbal Noun

In Guragina, nouns are derived from the jussive stems morphologically. The root-morphemes take the suffixes /-ot/ in order to change category from verb into nouns. In the canonical triradical root consonants, the root morpheme and suffixes are linearly appeared in newly derived nouns except the inserting of the epenthetic vowel to solve consonant constraint in the variety as can be seen from the data in Table 2.5.

Table 2. 5 *Guragina Verbal nouns*

Jussive stems	Translation	Infinitive	Translation
bɨhʲ -	cry	*bɨh-ot*	*Crying*
dɨrg-	Hit	*d ɨrg-ot*	*Hitting*
kʼʷatʼɨr -	tie	*kʼʷatʼr-ot*	*to tie / tying*

Verb Negation

When a verb is negated, a negation marker is added before the conjugated verb. In Guragina, their three verbal negative markers [10], [19], [27]. In order to do to accomplish this, the marker for the all-perfective is /*an-/*, for the Imperfective and Jussive it is bound morpheme /*a-/*, and for the prohibitive it is prefix */ɨn-/*.

Negated Perfective

The suffixes of the perfective are different for each person. As seen in Table 2.6, the perfective is negated by adding *an-* just before the base.

Table 2. 6 *The negation of stem bəkər "miss"*

Subject	Singular	Plular
1	*an-bəkər-hʷ*	*an- bəkər-nə*
2m	*an- bəkər-hə*	*an-bəkər-xu*
2f	*an- bəkər-hʸ*	*an-bəkər-xɨma*
3m	*an- bəkər-ə*	*an- bəkər-o*
3f	*an-bəkər-tʃ*	*an-bəkər-əma*
IPS	*an- bəkwər -i)*	

Additionally, the negation in this inflectional verb conjugation is produced by the morpheme /an-/, which is defined as being symmetric or linear[28]. We provide examples of this verbal negation in a Table 2.7 for all perfective verb types.

Table 2. 7 *Perfective negations*

Affirmative		**Negative**	
Guragina	**Meaning**	**Guragina**	**Meaning**
sənəqə	He stole.	*an- sənəqə*	He did not steal.
zəgədə	He remembered	*an-zəgədə*	He did not remember.
sefə	He sewed.	*an-sefə*	He did not sew.
mesə	He dug up.	*an-mesə*	He did not dig up.
zɨmaməda	he Mixed-up	*an- zɨmaməda*	He did not Mixed-up
ʒɨpapərə	He flipped	*an-ʒɨpapərə*	He did not flip
qʷəmə	He stood	*an- qʷəmə*	He did not stand
qʷəqʷərə	*He* squeezed	*an-qʷəqʷərə*	He did not squeeze.

This morpheme /an-/ is also combined with the subordinate morpheme /bə-/to create negation in conditional (if clauses). When the subordinator and negation morphemes are morphologically connected, the central mid vowel /ə/ is deleted, realizing the morpheme like from the word sətʃə "drank" we can construct bə-an-sətʃ "If he did not drink".

Imperfective Negation

The negative marker morpheme /*a-/* is appended to all verb types in the imperfective aspectual verb conjugations in Guragina [10], [19], [27] , as seen in Table 2.8. It is clear from the table that, negative marker */a-/* is prefixed and attached with the subject marker morpheme / *jɨ-/*.

Table 2. 8 *Negated imperfective*

Affirmative		Negative	
Guragina	**Meaning**	**Guragina**	**Meaning**
ji- sətʃə	He drunk.	*a-ji-sətʃə*	He does not drink.
ji-kʷəm	He stands up.	*a-ji-kʷəm*	He does not stand.
ji- metʼir-	He differentiates	*a- ji- metʼir-*	He does not differentiate.
ji-banr	He demolishes	*a-ji-banr*	He does not demolish.
ji-qatʼr	He ties	*a-ji-qatʼir*	He does not tie.
ji-qʷənis	He reduces	*a-ji-qʷnis*	He does not reduce.
ji-qʷəlf	He locks	*a-ji-qʷəlf*	He does not lock.

Negated imperative and jussive

The other negative marker used by Guragina, are *at-*, *e*, *in.* In which, giving command or the imperative verbal negation marked by the morpheme */a-/*, the jussive verbal negation marked by the morpheme */e-/*, as shown in Table 2.9, and prohibitions, or the negative command marked by the morpheme /in-/, as shown in Table 2.10.

Table 2. 9 *Negated imperfective and jussive*

Affirmative		Negation	
Guragina	**Meaning**	**Guragina**	**Meaning**
sit'e-	drink!	*at- sit'e-*	Don't drink!
bira	You eat!	*at-bira*	Don't break!

Additionally, in many south Guragina the prohibition or the negative command is expressed by the morpheme /in/ interchangeably with /a/[19], [23] without significant semantic difference, just like to Amharic *atbila vs. indatbila*. The information in Table 2.10 offers an illustration of this.

Table 2. 10 *Negative Command with morpheme /in-/ and /at-/*

Negated with/ in-/			Negated with /at-/		
Guragina	**Transliteration**	**Translation**	**Guragina**	**Transliteratio**	**Translation**
እን-በና-ሐ	*in-bəna-hu*	You shouldn't eat	ኦ-ት-ብር-ኦ	*a-t-br-o (3P)*	You don't eat
እን-ትማር-ሐ	*in-timari-hə*	You shouldn't	ኦ-ት-ማር	*a-t-mar (3S)*	You don't learn
እን-ሰነፍ-ሐ	*in-sənəf-hə*	You shouldn't	ኦ-ት-ሰረፍ	*a-t-sənəf (3S)*	You don't scare

Guragina uses the morpheme /e-/ in jussive to morphologically inflect negation, as shown in the Table 2.11. The jussive aspectual verb conjugation prefixes this negative marker morpheme to all verb types[19], [29] .

Table 2. 11 *Negation with morpheme /e-/*

Affirmative			Negation		
Guragin	Transliteration	Translation	Guragina	Transliteration	Translation
ኦ-ሰብር	*ə-sibr*	Let him brake.	ኤ-ሰብር	*e-sibr*	Do not let him open.
ኦ-ሐር	*ə-hʷər*	Let him go.	ኤ-ሐር	*e- hʷər*	Do not let him go.
ኦ-ቶራ	*ə-tora*	Let him sit	ኤ-ኦ-ቶራ	*e-tora*	Do not let him sit.

Tense and mood

Tense

Tense is a verb-based approach for expressing the time of an action or state in relation to the moment of speaking, as well as occasionally its continuance or completion. The three basic tenses are present, past, and future.

In Guragina, tense is morphologically represented by the auxiliary verbs (*banə, enə, nərə*, and *tanə*) and the bound morphemes *-te* and *-ʃə*, and it is classified into three basic distinctions as :past present and future respectively.

Guragina makes a distinction between the three verb bases that make up the three fundamental TAM forms: Perfective, Imperfective, and Jussive (including Imperative). The imperfective and jussive verbal forms have future time readings when they are suffixed with the bound morphemes -te and -ʃə. The perfective and jussive verbal forms denote an action that occurred at the moment of speaking, which indicates the present. The auxiliary verb banə or -ba “was” can be added to the first two to indicate past tense, and the -te- and -ʃə suffixes can be added to the last two to indicate alternative future tenses.

Present

The imperfective stem of the primary verb is inflected in Guragina to reflect the present reading, which is not clearly indicated. Additionally, as shown in example the locative clauses and predicates use the existence/auxiliary verb *nəre*, "there is" to indicate present tense readings.

Example 3

E kʷa	*aʒəməna*	*nere*
አኳ	አገሮመነ	ነረ

There is a wedding today.

Huta	*bɨzə*	*fɨrank nere-n*
ሑታ	ብዘ	ፍራንክ ነረ-ን

He has a lot of money.

ɨja-m agazɨ-m	*shewa*	*Nane*
እያ-ም አጋዝ-ም	ሸዋ	ናነ

I'm traveling to Addis Ababa with Agaz.

Past

Guragina has two morphological descriptions for past tense. The independent auxiliary verbs *banə* ("there was") and *tanə* ("when it was") are used to express past readings, and the inflected perfective verbal forms are required to indicate past reading. For example, *zega banə* "He was poor, *zega tane* "When he was poor" *etc*. The adjuncts (adverbs) are attached for further temporal time reference; the perfective verbal forms do not carry a further constraint of verbal activity. *səpʷər-ə-n-m "he brokem", dənəgʷ-ə-wi-m, bəna-m, bəna-tʃɨ-m etc*. This auxiliary verb (*banə*) is realized as its full forms or it can be reduced into *ba* that is interpreted as "there was". i.e., *banə* and *ba* can be used interchangeably.

Future

In Guragina, the bound morphemes *-te* or-*ʃə* are concatenated to the aspectual imperfective verb forms for the readings of the definite and indefinite futures, respectively, to denote the futurity of an action. The negative imperative stem of the verbs is denoting the future negation morphologically and syntactically as: *e-sətʃ-ɨ-n*, which is translated as "He is not drinking," "He does not drink," or "He will not drink." However, these future markers are not attached to the negative imperfective verbal forms. All imperfective aspectual verb forms have the morpheme *-te* added, which is used to distinguish the predetermined future under external control. In order to

convey future readings in this variety, it must be concatenated into the imperfective verb conjugations of all verb kinds.

Mood

The morphological or grammatical method of modality expression is mood [27]. It relates to ideas like “possibility,” “necessity,” “permission,” and “obligation,” among others. In Guragina, verbs are grammaticalized for expressing the attitude of speakers. The modality of the verbs may have different realizations that are categorized into types such as agent-oriented, speaker-oriented, epistemic, and subordinating [23]. Thus, each of these types has their own sub types and there may be an intertwining of function among these types.

In Agent-oriented modality, there are number of sub-classes including obligation, necessity, ability and desire. In Guragina, the essence and moral duty for ones or others life is morphologically distinctive, and they are expressed by the auxiliary/existence verbs nərə-/banə- and the applicative (object malfactive) marker -b-. The morphological representation of moral obligation used by speakers of Guragina is shown in Table 2.12. The existence verb nərə- together with the malifactive marker /-b-/ are employed to define state of obligation. The malfactive marker is followed by the object pronominal suffixes in this morphological process.

Table 2. 12 *Different modality of obligation*

Subject	Obligation			
	Singular		Plural	
	Guragina	Transliteration	Guragina	Transliteration
1	ነረ-ብ-ኢ	*nərə-b-I*	ነረ-ብ-ንደ	*nərə-b-ndə*
2m	ነረ-ብ-ሐ	*nərə-b-hə*	ነረ-ብ-ሑ	*nərə-b-hu*
2f	ነረ-ብ-ሽ	*nərə-b-hʲ*	ነረ-ብ-ሕማ	*nərə-b-hɨma*
3m	ነረ-ው-ኦ	*nərə-w-ə*	ነረ-ብ-ኦ	*nərə-b-o*
3f	ነረ-ብ-ኣ	*nərə-b-a*	ነረ-ብ-ኦማ	*nərə-b-əma*

2.3.2. Noun

Nouns are words (not pronouns), according to [30], that can serve as the subject or object of a verb or the object of a preposition. Simple nouns the simple noun forms are formed by the root-and-pattern morphological system whereas derivative or morphologically generated nouns are referred

to as complex nouns. i.e., complex nouns are derived through affixes, and two or more lexemes are compounded to form a new semantically unitary concept.

Guragina nouns have both simple and complex forms [31]. They are inflected for number, gender, definiteness, and case. The discussion of each of these ideas is broken down below into their individual instances.

Inflection of Noun

Number

The plural nouns of Guragina marked by two different morphological inflections; internal vowel modification and substitution for kinship words [31], as demonstrated by Table 2.13.

Table 2. 13 *Different morphological inflections*

Singular			Plural		
Guragina	Transliteration	Translation	Guragina	Transliteration	Translation
አርች	*ərtʃ*	Boy	ደንቻ	*dəngʲa*	Boys
ምስ	*mɨss*	man	ገማ/ ገመያ	*gəma/gemja*	Men
አራም	*əram*	Cow	አሬ	*əre*	Cows
ገረድ	*gərəd*	Girl	ግሬድ	*gred*	Girls
ምሽት	*mɨʃt*	Wife	እሽታ	*ɨʃta*	Wives

The majority of plural numbers, however, are expressed indirectly on verbs. As shown in Table 2.14, they are also indicated by adding the third person pronoun (hɨno with all nouns and hɨnəma for human (Feminine) related nouns) that interacts with definiteness as demonstrated in example. Additionally, third person singular male *-huta* for all noun types and third person feminine *-hɨta* for third person singular are both acceptable.

Table 2. 14 *Examples of Guragina's Plural with third person pronoun*

Example			With Suffix				Suffix	
Guragina	Transliteration	Translation	Guragina	Transliteration	word-for-word translation	Translation	Guragina	Transliteration
እምር	*ɨmɨr*	Stone	እምር-ሕኖ	*ɨmɨr-hɨno*	stone-they	Stones	ሕኖ	*hɨno*

ወናድ	wənad	Mare	ወናድ-ሕኖ	wənad-hɨno	mare-they	Mares	ሕኖ	*hɨno*
ፌቅ	*Feq*	goat	ፌቅ-ሕኖ	*feq-hɨno*	goat- they	goats	ሕኖ	*hɨno*
ጀበን	*ʤəbən*	coffee pot	ጀበን-ሕኖ	*ʤəbən-hɨno*	coffee pot-they	coffee pots	ሕኖ	*hɨno*

In Guragina, feminine and masculine genders are indicated via internal modifications, and these nouns are mainly representing kinship terms and domestic animals [31]. Kinship word and domestic animals both lexically denote the masculine and feminine genders, as shown in example Table 2.15.

Table 2. 15 *Gender in Guragina*

Masculine			Feminine		
Guragina	Transliteration	Translation	Guragina	Transliteration	Translation
ምስ	*mɨss*	husband	ምሽት	*mɨʃt*	wife
ኦርች	*ərtʃ*	Boy	ገረድ	*gərəd*	girl
ኣባ	*aba*	Father	ኣዶት	*adot*	mother
ውር	*were*	Ox	ኣራም	*əram*	cow
ፈረዝ	*fərəz*	horse	ወናድ	*wənad*	mare

Additionally, to distinguish between animals of the feminine and masculine genders, the adjectives *arɨst* "female" and *təbat* "male" are employed [29]. Before the nouns, these words are combined like in Table 2.16.

Table 2. 16 *Guragina's Gender with "arɨst" and "təbat"*

Generic			Masculine			Feminine		
Guragina	Transliteration	Translation	Guragina	Transliteration	Translation	Guragina	Transliteration	Translation
ፌቅ	*feq*	Goat	ተባት ፌቅ	*təbat feq*	male goat	ኦርስት ፌቅ	*arɨst feq*	nanny goat
ግየ	*gɨjə*	Dog	ተባት ግየ	*təbat gɨje*	male dog	ኦርስት ግየ	*arɨst gɨje*	bitch

ቹር	*fʷur*	Rat	ተባት ቹር	*təbat fʷur*	male rat	ኣርስት ቹር	*arɨst fʷur*	female rat

Definiteness

In Guragina, referential restriction of nouns can be categorized into two i.e., definite and indefinite nouns. Indefinite nouns are referring to the generic word that has not been restricted its function as in *feq* goat". This word is not showing an individual goat but a generic one /any goat; it can be male, female and unknown [27].

In Guragina, definiteness of nouns is indirectly marked, and they are directly denoted by person pronouns of both feminine and masculine predominantly the third person as shown Table 2.17. The former is suffixed to the semantically feminine nouns (only people) and in animate nouns for diminutive case, and the latter is suffixed with all masculine nouns.

Table 2. 17 *Definiteness of nouns*

Generic			Generic with Suffix		
Guragina	**Transliteration**	**Translat**	**Guragina**	**Transliteration**	**Translation**
ገረድ	*gərəd*	girl	ገረድ-ሽታ	*gərəd-hʲita*	the girl
ኣርች	*ərtʃ*	boy	ኣርች-ሑታ	*ərtʃ-huta*	the boy
እማር	*ɨmar*	donkey	እማር-ሑታ	*ɨmar-huta*	the donkey

Therefore, in Guragina, definiteness is indirectly indicated by the third person masculine (huta) although the third person feminine singular is used for singular feminine nouns. The morpheme *huta* denotes definiteness in all animate and inanimate singular nouns. Table 2.18 shows the grammatical function of the morphemes h*ʲita* and *huta* in Guragina.

Table 2. 18 *Definiteness with hʲita and huta*

Grammatical		Translation	
Guragina	**Transliteration**		
ግሬድ-ሕነማ	*gred-hɨnəma*	girl.PL-DEF	the girls
ኣርች-ሑታ	*ərtʃ-huta*	boy-DEF	the boy
ደንኝ-ሕኖ	*dəngʲa-hɨno*	boy.PL-DEF	the boys

Therefore, definiteness in this variety is denoted by the third person singular and plural pronouns but the third person masculine singular is commonly used for all animate and inanimate nouns.

Case

The relationship among inflectional markers and independent nouns has grammatical and semantic notion. Thus, the syntactic relations of words are referred to as core case whereas the semantic relations are treated as peripheral case. The grammatical case encompasses subjective (nominative), objective (accusative), dative (indirect object) and genitive (possessive), and the semantic case contains locative, ablative, and vocative among other case markers. In Guragina, both the grammatical and semantic cases are discussed as follows.

Noun derivation

Nouns are also derived from the composition of two or more lexemes that creates a single semantic concept [31]. However, most of Gurarage nouns are resulted from derivational process by affixes such as -*nət*, -i, -*ot* /wə-, -*ənə*, -*na*, and -$k^{w}ə$. For instance, several adjectives and nouns in Guragina can be changed into abstract nouns by adding the bound suffix (morpheme) -*nət*, as listed in table 2.19. In this case if the last latter of the word (before adding -*nət*) is 4th column latter in Guragina syllograph, it changes to the 6th latter of its type, like "ጋዋ" to "ጋውነት" or Gawa change to Gaw-*nət*.

Table 2. 19 *Derivations of Noun*

Adjective or noun			**Abstract noun**		
Guragina	**Transliteration**	**Translation**	**Guragina**	**Transliteration**	**Translation**
ወሔ	*wəhe*	kind	**ወሔ-ነት**	*wəhe-nət*	Kindness
ሓሪ	*Hari*	Wise	**ሓሪ-ነት**	*hari-nət*	Wisdom
ጋዋ	*Gawa*	Fool	**ጋው-ነት**	*gaw-nət*	Foolish
ሙጥጥ	*mut'ɨt'*	Bad	**ሙጥጥ-ነት**	*mutʼɨtʼ-nət*	Badness
ጌፍ	*Gef*	Long	**ጌፍ-ነት**	*gef-nət*	Longines
ጀንጅር	*ʤenedʒer*	Fat	**ጀንጅር-ነት**	*ʤenedʒer-nət*	Fatness
ደንገነ	*dəngənə*	Rich	**ደንገነ-ነት**	*dəngən-nət*	Richness
ዜጋ	*Zega*	Poor	**ዜግ-ነት**	*zeg-nət*	Poverty
ቤሻ	*beʃa*	Friend	**ቤሻ-ነት**	*beʃ-nət*	Friendship
ባሪቅ	*barik'*	Old	**ባሪቅ-ነት**	*barikʼ-nət*	Old age
ወነሔ	*wənəhʷə*	neighbor	**ወነሔ-ነት**	*wənəhʷ-nət*	Neighborhood
ፈንገያ	*fəngʲa*	Thief	**ፈንገያ-ነት**	*fəngʲɨ-nət*	Theft
ትክ	*tɨkə*	Child	**ትክ-ነት**	*tɨkɨ-nət*	Childhood
ሰብ	*səb*	Man	**ሰብ-ነት**	*səbɨ-nət*	Personality

Compound Nouns

Nouns in Guragina are also morphologically created by combining two or more simple nouns, which are then regarded as a single unified semantic word. Two or more words that relate to a single item, situation, action, or event are combined. Guragina (Ethiopic scripts), transliteration, and the translation of each individual word and compound word are all shown in Table 2.20, respectively.

Table 2. 20 *Guragina Compound word*

	Single word		Compound word (noun)		
Guragina	**Transliteration**	**Direct translation of each word**	**Guragina**	**Transliteration**	**Translation**
ውር ኣጤበ	*wər at'ebə*	ox finger	**ውርኣጤበ**	*wərat'ebə*	Thumb
ውር ኩታራ	*wər kutara*	*ox hen*	**ውርኩታራ**	*wər kutara*	Roaster
ወሐመያ	*wəhɛ məja*	good path	**ወሐመያ**	*wəhɛməja*	Holiday
የቤት እም	*jəbet em*	House mother	**የቤትእም**	*Jəbetem*	Wife
ሓኪም ቤት	*hakim bet*	doctor house	**ሓኪምቤት**	*Hakimbet*	hospital or clinic
ጉነር ፍርጠት	*gunər fɨrt'ət*	head spilt	**ጉነርፍርጠት**	*gunərfɨrt'ət*	headache
ኣዳር ሑጅር	*ɨdar hudʒɨr*	night cloth	**ኣዳርሑጅር**	*ɨdarhudʒɨr*	Blanket

In addition to that, in this variety, several compound nouns are formed by the combination of ab "ኣብ" with a simple noun to create a semantically single word, such as ኣበቃጥ "husband", ኣበሩስ "family", ኣበጥብ "same clan", ኣበጐዳ "friend", etc…

Root-and-Pattern (Noun Formation)

Guragina uses a root-and-pattern construction for basic nouns, which is a feature of Semitic morphology exclusively. Here, as one can understand from the Table 2.21, nouns are formed by combining different vocalic and consonantal affixes into roots with two, three, or four consonants.

Table 2. 21 *Guragina's root-pattern nouns*

Template	Root		Noun		Translation
	Guragina	Transliteration	Guragina	Transliteration	
CCə	ብ-ኽ	*b-kʲ*	ብኽ	*bɨkʲə*	mourn
	ብ-ኽ	*s-kʲ*	ስኽ	*sɨkʲə*	run away
COC	ም-ት	*m-t*	ሞት	*Mot*	death
	ጥ-ም	*tʾ-m*	ጦም	*tʾom*	Fast
CəC	ድ-ም	*d-m*	ደም	*dəm*	blood
	ብ-ር	*b-r*	በር	*bər*	river
	ፍ-ስ	*f ʷ-s*	ፈስ	*Fʷəs*	Fart
CɨCC	ም-ክ-ር	*m-k-r*	ምክር	*mɨkr*	advice
	ድ-ም-ድ	*d-m-d*	ድምድ	*dɨmʷd*	pair
	ቅ-ን-ም	*k-n-m*	ቅኔመ	*kɨnemə*	insult
CəCəC	ስ-ን-ፍ	*s-n-f*	ሰነፍ	*sənəf*	laziness
	ድ-ር-ግ	*d-r-g*	ደረግ	*dərəg*	strength
	ሞ-ብ-ዝ	*g ʷ-b-m*	ጐበዝ	*gʷəbəz*	cleaver
CɨCaC	ን-ፍ-ስ	*n-f-s*	ንፋስ	*nɨfas*	wind
	ስ-ክ-ር	*s-k-r*	ስካር	*sɨkar*	intoxication
	ቅ-ም-ር	*q-m-r*	ቅማር	*qɨmar*	Louse
CɨCCəCa	ም-ዝ-ግ-ብ	*m-z-g-b*	ምዝገባ	*mɨzgəb*	registration
	ስ-ብ-ስ-ብ	*s-b-s-b*	ስብሰባ	*sɨbsəba*	meeting
	ም-ር-ም-ር	*m-r-m-r*	ምርመራ	*mɨrməra*	investigation
CɨCCɨC	ም-ስ-ክ-ር	*m-s-k-r*	ምስክር	*mɨskɨr*	witness
	ስ-ን-ጥ-ር	*s-n-tʾ-r*	ስንጥር	*sɨntʾɨr*	splinter
CəCCəC	ስ-ን-ብ-ር	*s-n-b-r*	ሰንበር	*sənbər*	Stripe of stomach

To determine the kind of words in the context of root consonants, the vowel's quality is crucial. For example, vocalic alteration in the stem template can change the word classes of the terms *bɨkʲə* "ብኽ" is noun and mean crying or sorrowful while *bəkʲə "ለኸ"* He cried. " mʷətə He died. Respectively. Here, the difference lies on the vowels that are inserted in the radical consonantal roots.

2.3.3. Adjective

Adjectives in Guragigna, despite being a few in number, share morphological and syntactic similarities with nouns and pronouns. They undergo morphological inflection for number, case, and definiteness, but not for gender. While singular number is not explicitly marked, plural numbers are expressed through morphological manifestations. Plural adjectives are formed by

reduplicating stems, for example, ብሻብሻ "biʃabiʃa" meaning "red ones" Alternatively, pronouns, particularly the third person masculine plural suffix "-xɨno," are commonly used to indirectly denote plural adjectives, such as ብሻሕኖ "biʃaxɨno" meaning "red ones" Additionally, adjectives can be derived morphologically from other basic parts of speech using affixes like “jə-,” "ɨ…jə," ‘-əma,” "-ənə," and "-w.

2.3.4. Pronouns

In their morphological forms, pronouns can be both simple and complex. They serve the same purpose as nouns or noun phrases. Specifically, Personal pronouns are used interchangeably with names of people, places, and things in Guragina, whether in spoken or written language. Unlike other Ethio-semitic languages like Amharic, the feminine second- and third-person plural is distinct. The fundamental personal pronouns included in Table 2.22 fall within this variety.

Table 2. 22 *Basic Guragina Personal Pronouns*

Person	Singular			Plural		
	Guragina	Transliteration	Translation	Guragina	Transliteration	Translation
1	እያ	*ɨja*	I	ይና	*ɨɨna*	we
2M	ኣሐ	*ahə*	You	ኣሑ	*ahu*	you
2F	ኣሽ	*Ahʲ*	You	ኣሕማ	*ahɨma*	you
3M	ሑት/ታ	*hut/a*	He	ሕኖ	*hɨno*	they
3F	ሽት/ታ	*hʲit/a*	She	ሕነማ	*hɨnəma*	they

On the other hand, the genitive case marker (ʔə-) is morphologically attached to the independent pronouns for denoting objective (accusative) pronoun. This is demonstrated in Table 2.23. These complex objective pronouns alter the way nouns behave morphologically. For example, the generic noun *kutara*, "hen," might be transformed with these genitives (possessions) pronouns as *ʔəhuta kutara* "a hen of him" or pronominal possessive markers, "*kutarata*, "his hen".

Table 2. 23 *Plural and Singular Objective (accusative) pronoun with ʔə*

Singular				Plural			
Guragina	Transliteration		Translation	Guragina	Transliteration		Translation
ይያ	*ʔə +ɨja*	*ʔɨja*	me	ይና	*ʔə +ɨɨna*	*ʔɨna*	us
ያሕ	*ʔə +ahə*	*ʔahə*	you	ያሑ	*ʔə+ ahu*	*ʔahu*	you
ያሽ	*ʔə+ ahʲ*	*ʔahʲ*	you	ያሕማ	*ʔə +ahɨma*	*ʔahɨma*	you
የሑት/ታ	*ʔə +huta*	*ʔəhut/ta*	him	የሕኖ	*ʔə +hɨno*	*ʔəhɨno*	them
የሽት/ታ	*ʔə +hʲita*	*ʔəhʲit/ta*	her	የሕነማ	*ʔə +hɨnəma*	*ʔəhɨnəma*	them

As one can understand from the Table 2.24, similar to the independent personal pronouns, Guragina also defines possessive pronominal suffixes for person, gender, and number.

Table 2. 24 *Pronominal suffixes of possession*

Person	Pronominal suffixes		Example		
	Guragina	Transliteration	*Guragina*	Transliteration	Translation
1PS	(አ)ና	*-(ə)na*	አባ-ና	*aba-na*	my father
1PL	(አ)ንዳ	*-(ə)ndə*	አባ-ንዳ	*aba-ndə*	our father
2PMS	ሐ	*-hə*	አባ-ሐ	*aba-hə*	your father
2PMPL	(አ)ኣሑ	*-(a)hu*	አባ-ኣሑ	*aba-hu*	your father
2FS	(አ)ኽ	*-(a)hʲ*	አባ-ኽ	*aba-hʲ*	your father
2FPL	(አ)ሕማ	*-(a)hɨma*	አባ-ሕማ	*aba-hɨma*	your father
3MS	(አ)ታ	*-(ə)ta*	አባ-ታ	*aba-ta*	his father
3MPL	ሕኖ	*-hɨno*	አባ-ሕኖ	*aba-hɨno*	their father
3FS	ኽታ	*-hʲɨta*	አባ-ኽታ	*aba-hʲɨta*	her father
3FPL	ሕነማ	*-hɨnəma*	አባ-ሕነማ	*aba-hɨnəma*	their father
1PS	(አ)ና	*-(ə)na*	አዶተ-ና	*adot-ənа*	my mother
1PL	(አ)ንዳ	*-(ə)ndə*	አዶተ-ንዳ	*adot-əndə*	our mother
2PMS	ሐ	*-hə*	አዶተ-ሐ	*adot-ahə*	your mother
2PMPL	(አ)ኣሑ	*-(a)hu*	አዶተ-ኣሑ	*adot-ahu*	your mother
2FS	(አ)ኽ	*-(a)hʲ*	አዶተ-ኽ	*adot-ahʲ*	your mother
2FPL	(አ)ሕማ	*-(a)hɨma*	አዶተ-ሕማ	*adot-ahɨma*	your mother
3MS	(አ)ታ	*-(ə)ta*	አዶተ-ታ	*adot-ətа*	his mother
3MPL	ሕኖ	*-hɨno*	አዶተ-ሕኖ	*adot-əhɨno*	their mother
3FS	ኽታ	*-hʲɨta*	አዶተ-ኽታ	*adot-əhʲɨta*	her mother
3FPL	ሕነማ	*-hɨnəma*	አዶተ-ሕነማ	*adot-əhɨnəma*	their mother

Guragina nouns are given the pronominal possessive affixes in order to display possession forms similar to English possessive adjectives. The vowel (a) and low central vowel (ə), as seen in the Table 2.30, are added between nouns and possessive markers. Conversely, if the vowel-final word

is joined with vowel-initial words, as in the lexeme's *aba* "father" with lexeme *na* gives *abana* "my father", *abanda* "our father" etc the first vowels of possessive maker is eliminated.

2.3.5. Number

A number is represented by a numeral, which can be a single symbol or a group of symbols. These lexical numerals are known as cardinal and ordinal numbers and are used to enumerate things, objects, etc. In Guragina does not mark singular numbers, however plural numbers are morphologically expressed.

Cardinal numbers are indicated by the arithmetic sets that identify numerical system of quantity. Thus, Guragina has decimal numerical system, that is, the class of numbers are expressed in counting system that they use unit of ten (10) as a base in the multi-digital numeral systems. it is similar to Amharic numbering system with a few pronunciations difference. From the Appendix 2, it is clear that, in most Guragina varieties, numbers are pronounced as follows: *at "አት", hwet, "ሑት" sost, "ሶስት", arbət "አርበት", amist "አምስት", sidist "ስድስት", səbat "ሰባት", simwit "ስምት", ʒətʾə "ዠጠ" and asir "አስር".* From eleven to nineteen we add *asirəm "አስረም"* as the prefix of each, such as *asirəm at "አስረም አት", asirəm hwet, "አስረም ሑት" and asirəm ʒətʾə "አስረም ዠጠ".*

As one can understand from the Appendix 3, the majority of Guragina varieties, including Guragina morphologically derive ordinal numbers from cardinal numbers by appending the suffix-*ənə*.

2.3.6. Adverbs

Adverbs are one of the grammatical groups of words that provide information about the where, when, how, how often, and in what way actions of verbs are carried out. Adverbs are less common in Guragina than other content terms, which are introduced by the relational morphemes such *tə-* and *bə-*. Adverbs are restricted with adverbs that are classified as time, frequency, manner, place and frequency according to their purposes. These adverbs and their uses in Guragina are covered in the section below.

Adverb of time

The role of the verbs in relation to past, present, and future time is explicitly constrained by these adverbs. In Guragina, adverbs of time are lexically expressed that are suffixed by particles like */-ra/* and */-ə/* to denote past and future expressions respectively [14]. This reality in Guragina is illustrated in Table 2.25.

Table 2. 25 *Guragina's Adverbs of time*

Present			Past			Future		
Guragina	Transliteration	Translation	*Guragina*	Transliteration	Translation	*Guragina*	Transliteration	Translation
አሓ	*əhʷa*	now	ትራማ	*tirama*	yesterday	ነገ	*nəgə*	Tomorrow
አኳ	*ʔəkʷa*	today	ሰስትራ	*səstəra*	three days before	ሰስተ	*səstə*	three days later
ገባት	*gəbat*	evening	ነበትራ	*nəbətəra*	four days before	ነባተ	*nəbatə*	four days later

The time adverbs tend to form compound nouns to describe function of the action or clause with nouns or words such as : *qirərə* "morning", *wana* "noon", *ədar* "night", kərə "day".

Adverb of frequency

Frequency adverb expresses how often an action is happened in relation to time. In Guragina, as can be realized in Table 2.26, adverbs of frequency are expressed by complete reduplication of an adverb of time, but the adverb *əhʷa* "now" is attached with the coordinating conjunction (-m) [32].

Table 2. 26 *Guragina's Adverbs of frequency*

Guragina	Transliteration	Literal translation	Translation
አኳ-ም-ነገ-ም	*əkʷa-m-nəge-m*	today and today	Everyday
አሓ-ም-አሓ-ም	*əhʷa-m-əhʷa-m*	now and now	Always /every time
ቅረረ ቅረረ	*qirərə qirərə*	morning morning	Every morning
የሕርም ሕርም	*jəhirm hirm*	year year	Every year
ዋና ዋና	*wana wana*	noon noon	Every noon
እንም ግዝየ / እንም ከረ	*inim gizjə or inim kərə*	a11 times or all day	Always /often
አታት ግዝየ	*atat gizjə*	One one time	Some times

Adverb of manner

In Guragina, an adverb of manner is described by the relational prefix *bə-*, and it is linearly attached to simple or complex nouns. Thus, as shown in Table 2.27, PPs morphosemantically denotes manner of the verbs and they are headed by the relational morpheme (bə-).

Table 2. 27 *Adverbs of manner*

Guragina	Transliteration	Translation
በ-ሩጫ	*bə-rutʃa*	*By running*
ተሸናሐ	*tə-hʲənahe*	*Carefully*
ቦሔነት	*Bo-wəhenət*	*In good manner*

Adverb of place

Adverbs of place are tending to show us where something is performed, placed, or its whereabouts. In Guragina, the function of an adverb of place is denoted by addpositions. Therefore, the prepositional morphemes in Table 2.28 represent the position or direction of object, person and thing that is talked about.

Table 2. 28 *Adverb of Place*

Guragina	Transliteration	Translation	Transliteration	Guragin	Translation
አግሬ	*əgɨre*	Bottom	*dən*	ደን	inside/under
ቅሪት	*qʷərit*	Top	*wəsɨtʼ*	ወስጥ	Bottom
ነን	*nən*	Up	*tət*	ተት	Low
አንቐ/አንቔ	*anqʲə / anqʲe*	behind/back	*jɨft*	ይፍት	in front of
መየ / መዬ	*məjə / məje*	around	*fwər*	ፎር	On

2.3.7. Conjunctions

Connectors syntactically attach the jumbling of any two phrases, clauses or sentences. In Guragina, two or more words are connected with the coordinator (-m) and disjunctive *wem*, like *ərtʃ-**m** gərəd-**m*** for **"**boy and girl, *mɨs wem mɨʃɨt* for "husband or wife" but the hypotactic concept is expressed by the relational morphemes *tə-* and *bə-* and *ʔə* like ***bə**-tʃən etc*...

2.4. Spelling Errors

Texts of languages can be generated from different sources either by humans as document typing or emailing software, or by machines such as optical character recognition (OCR) and machine translation (MT). In natural language processing, spelling error and spell checking is a very common task in those generated text. A spell checker (or spelling checker or spell check) is a computer program that checks for misspellings made by users who type in different applications. Currently, spelling error detection play a significant role in a variety of computer programs [2] [7], [33]. They could be used independently or embedded within other systems like word processors, optical character recognition, search engines, speech recognition, machine translation, browsers, and so on.

2.4.1. Types of Spelling Errors

Spell checker techniques are designed on the basis of spelling errors. Studies have been conducted to analyze the types and trends of spelling errors. According to kukich [3], spelling errors are generally categorized into typographic and cognitive errors based on the error patterns produced when typing words on a word processing application. The other researcher, Gupta [34] categorized as typographic, cognitive and phonetic errors

Typographic errors occur when the correct spellings of the word are known, but the word is mistyped by mistake. The majority of the time, they are caused by technological limitations of the input device (physical or virtual keyboard, or OCR system), like keyboard adjacencies, therefore do not follow any language criteria. For instance, when typing quickly, two close keys are frequently substituted. Although they may be dependent on local keyboard mapping or a localized OCR system[35].

Typographic errors can also be classified as a Single errors or Multi-errors [36]. A word is considered to have several errors if there are more than one. On the other hand, single errors are affecting just one letter. A study by Damerau [4] shows that 80% of typographical errors come from one of the following four categories.

Insertion: - inserting an extra character or single letter in the word.

Deletion: - missing or omission of a single letter a single character from the word.

Substitution: - replacement of a single letter by another single letter.

Transposition of letters: - mis ordering or swapping of two adjacent letters in the word.

Typographic errors can also be categorized into two based on the meaning they give: non-word and Real-word errors [5]. Non-word errors are spelling errors that do not give any meaning, while Real-word (semantic) errors are morphologically valid words that give no sense in context. Damerau [15] shows that 80% of the misspelled words in English are non-word errors. Table 2.29 shows some examples of typographical errors discussed so far.

Table 2. 29 *Typographic Errors*

Examples of different spelling errors					
Language	**spelling**	**Single letter**	**Single letter deletion**	**Single letter substitution**	**Transposition of two adjacent letters**
English	Correct	happy	happy	happy	happy
	Incorrect	hagppy	Hapy	hagpy	hpapy
Amharic	Correct	**አለን**	**አለን**	**አለን**	**አለን**
	Incorrect	**ደስ አለለን**	**ደስ አን**	**ደስ አበን**	**ደስ አንለ**
Guragina	Correct	**ሳረንደም**	**ሳረንደም**	**ሳረንደም**	**ሳረንደም**
	Incorrect	**ሳለረንደም**	**ሳረደም**	**ሳረብደም**	**ሳንረደም**

The other types of errors are **cognitive errors,** which are also called orthographic or consistent or real-word errors[35]. These mistakes occur due to misconceptions or when the correct spelling of words is unknown to the writer [3],[35]. In other words, the writer may not be aware of the proper writing style.

This particular type of errors is user- and language-specific because it is more dependent on using the rules of the language. An example of this types of error would be typing “hiar” instead of “hair”, “ante” instead of “ant”.

In texts written in the Guragina language, cognitive mistake is common. The primary cause is that the language's orthography wasn’t standardized until 2020, as indicated in Section 2.5. Even once the language is standardized in 2020, the majority of the orthography is unfamiliar to the writer. Examples of this kind of error in this language include typing “**ብርሕማ**” *berehema* “instead of

“ቡርሕማ” *bʷerehema,* **“ዘፒትሑ”** *zəpitɨhu,* instead of **“ዘፒትሕሙ”** *zəpitɨhʷ* **“እሐ”** *’ɨhə* instead of **“እሓ”** *ɨha,* **“ቆቆሳ”** *k’ok’osa,* instead of **“ቈቈሳ”** *k’ʷe k’ʷesa,* etc.

Phonetic Errors happen when a character or a word is replaced out with a phonetically equivalent sequence of character or homophone word, respectively. An example of this types of error would be typing “cheap” instead of “cheep”, “ant” instead of “aunt” for English and typing **ሐሐብ** “ *həhəbɨ* instead of “**ሐሐብ”** *hʷəhʷəbɨ* **,** “**ሐብት**” *həbɨtɨ* instead of “**ኽብት**” *kɨəbɨtɨ* “**ኩር**”*kʷərɨ* instead of “**ቈር**” *k’ʷərɨ*. Similar to cognitive error, these types of error also common in Guragina language.

2.4.2. Techniques of Spell Checker

Spelling error detection and correction are the two main steps of a spell checker system, which may be carried out jointly or sparely. Error correction step gives the proper suggestions for any misspelled words and ranks them according to which terms in the list have the highest likelihood of being used instead. Different approaches are used to develop SC for different languages based on the characteristics of the language [6].

Error Detection Techniques

Spelling error detection approaches is the main thing in spell checking to detect the errors in written text. It is used to flag out misspelled words in a text. To suggest a set of possible corrections, a spellchecker has to identify the first misspelled words. Different approaches are applied to detect spelling errors for different language based on their language characteristics. N-gram analysis and dictionary lookup are the two most well-known methods for detecting errors in spell checker system development[3], [7], [8].

N-gram Analysis

N-gram analysis is one of the popular methods for detecting spelling errors. N-grams are n letters subsequences of words or strings taken from a string with a length of whatever *n* is set to[37].N stands for one, two or three. One letter n-grams are referred to as unigrams or monograms; two-letter n-grams are referred to as bigrams; and three- letter n-grams are seen as trigrams [37].

The input word's n-gram sequences are compared to the subsequences that were previously saved, and N-gram frequencies are stored in an n-dimensional matrix. It is a method to find

misspelled words in text and used for non-word errors [38], [39]–[41]. The term is considered to be misspelled if one n-gram is missed.

In order to pre-compile an n-gram table, a dictionary or corpus of text is usually required. Instead of comparing each entire word in a text to a dictionary, just n-grams are controlled. The n-gram algorithm was developed as one of the benefits is that it allows strings that have differing prefixes to match and the algorithm is also tolerant of misspellings. Each string that is involved in the comparison process is split up into sets of adjacent n-grams.

The table stores n-gram's existence or frequencies, any n-grams in an input string that have nonexistence or low frequencies are classified as probable misspellings. It is said that a set of positional binary n- gram arrays are able to detect error more accurately. Because each element in the positional binary n-gram arrays matches the exact position within each word. However, this raises the storage space problem due to the large capacity of the positional arrays. Since most misspellings do not contain any impossible n-grams, so n-gram analysis techniques are not good at detecting human generated errors but good at detecting machine- generated errors like OCR [39].

A check is done by using an n-dimensional matrix where real n-gram frequencies are stored. In general, n-gram detection technique work by examining each n-gram is an input string and looking it up in a precompiled table of n-gram statistics to determine either its existence or its frequency of words or strings that are found to contain nonexistence or highly infrequent n-grams are identified as either misspelling. The major advantage of n-gram algorithm is that it requires no knowledge of the language that is used with, so it is language independent or it is a neutral string-matching algorithm.

Dictionary Look Up

A dictionary/Wordnet is a lexical source containing a list of correct words for a specific language [38], [40], [41]. Dictionary lookup approach is one of the methods of developing spell checkers, which use a dictionary to find words. It is looking at every word in the dictionary of the lexicon, a corpus, or a combination of lexicons and corpora [35]. The non-word errors can be easily detected by checking each word against a dictionary. If the word exists in the dictionary, it is considered a correct word and if the word doesn't exist in the dictionary, it is flagged out as a misspelled word.

Dictionaries are represented in many ways, each with their own characteristics like speed and storage requirements. Large dictionary might be a dictionary with most common word combined with a set of additional dictionaries for specific topics such as computer science or economy. Big dictionary also uses more space and may take longer time to search. storage space requirement and time-consuming searching

The drawbacks of this method are difficulties in keeping such a dictionary up to date, storage space requirement and complete enough to include all the words in a text. It is the fact that it will be impossible to store all words and even if all are stored, they should be updated from time to time. In addition to this, a known shortcoming of dictionary-based systems is handling so-called real-word errors [14]. This kind of error is difficult to identify using these methods because the misspelled word exists in the dictionary.

Moreover, searching for a word in a large knowledge base would not be efficient [6]. To clarify this, a large dictionary requires more space and may take a longer time to search for a specific word and a too-small dictionary consider many words as invalid words but the words are valid in that specific language. Simultaneously, system response time should be reduced. Dictionary lookup and construction techniques must be tailored according to the purpose of the dictionary. To solve the problem of inefficiency, a technique called hashing is used.

Hash tables are the most common used technique to gain fast access to a dictionary. A hash table, commonly referred to as a hash map, is a type of data structure that uses an associative array or dictionary. It is an abstract data type that associates keys with values. A hash table utilizes a hash function to create an index. When a word is hashed, the hashing address or key is looked up in a hashing table, and the resultant hash shows where the relevant value is kept. i.e., compute its hash address and obtain the word stored at that address in the hash table that has already been built. The fundamental benefit of hash tables is their random-access nature that eliminated the large number of comparisons needed to search the dictionary, which makes dictionary searches faster [42].

The main drawback of hash tables is the requirement for creative collision-avoidance hash algorithm. We compute each hash function for a word before storing it in the dictionary, then we set the vector elements in the dictionary that correspond to the computed values to true. We determine the hash values for a word and search the vector to see whether it is a part of the

dictionary. If all entries corresponding to the values are true, then the word belongs to the dictionary, otherwise it does not.

Error Correction Techniques

The second step of spell checking, next to detection, is spelling error correction. Spell correcting refers to the attempt to endow spell checkers with the ability to correct detected errors, i.e., to find the subset of dictionary or lexical entries that are similar to the misspelling in some way. Error correction consists of two steps; the generation of candidate corrections and the ranking of candidate corrections [3], [38]. To find one or more viable correction words, the candidate generation procedure commonly uses a precompiled table of permissible n-grams. To rank order the candidates, the ranking procedure commonly uses a lexical similarity measure between the misspelled text and the candidates or a probabilistic assessment of the likelihood of the repair. Most of the time, these two procedures are viewed as independent processes and carried out in order.

Spelling error correction categorized in different way based on the way of handling. First it can be classified as interactive and automatic.

In interactive, the spellchecker can suggest more than on correction for each misspelled word and the user decide to select for replacement. In case of automatic correction, the spellchecker has to decide on the one best correction and the error is automatically replaced with it.

In in another way it can also classified as isolated word error correction and context-based error correction. Isolated term correction can work by identifying spelling errors for each word in a text whereas context-sensitive correction checks if a word (spelled correctly or not) gives a meaning context-wise. The former one is mainly used for correcting real word errors. Real word errors are words which have correct meaning in the dictionary but contextual error in a given text.

Rule-based, n-gram-based, minimum edit distance, similarity keys, probabilistic, neural networks, and noisy channel approaches are a some of the error correction techniques now in use [35], [42].

Minimum Edit Distance

The spelling correction method that has received the greatest attention and currently most widely used to spelling correction is the minimum edit distance method. This technique involves converting one string into another at the lowest possible cost or with the fewest possible editing steps [35]. In order to transform, this refers to the fewest amount of editing operations that must be performed between two characters. With the use of this technique, several operations, including insertions, deletions, substitutions, and transpositions, can be carried out depending on various potential costs for each of these operations. The
shortest edit distance between a misspelled word in a text and a word in the dictionary is typicall y what one concerned about.

According to various scholars, including Kuchich [3], [49], the majority of spelling errors might be corrected by the addition, deletion, substitution, or transposition of one letter or two characters. The dictionary word is referred to as a plausible correction if a misspelling may be changed into one by reversing one of the error processes (i.e., insertion, deletion, substitution, and transposition).

This method is only applicable to typos in single or more useful for keyboard input errors that phonetic errors[42], [43]. As a result, there are only fewer possible comparisons. By comparing words with a length of four to six characters to a list of words that appear frequently, the Edit Distance algorithm can identify spelling errors. If the search term is not included in the list, it is then looked up in a dictionary that has terms arranged by alphabetical order, word length, and character occurrence.

Where a word in the dictionary is one character longer than the detected word, then the first character in the dictionary word that is different is discarded and the rest of the characters are shifted one-bit position left [35]. If the two words match, then the word in the dictionary is reported to be the correctly spelled word by a single insertion.

On the other hand, the first character in the detected word that differs from the matching character in the dictionary word is considered as incorrect if the word in the dictionary is one character shorter [35]. The remaining characters in the misspelled word are therefore shifted one bit to the left and that particular character in the detected word is removed. The dictionary term is reported

as the correctly spelled word by a single deletion if the new misspelled word and the entry in the dictionary match.

When the lengths of the word in the dictionary and the misspelled word are the same, but they differ by one-character position, then the dictionary entry is reported as a candidate correction as they differ by a single substitution.

Where the lengths of the word in the dictionary and the misspelled word are the same, but they differ in two adjacent positions, the characters are proposed to be swapped. If the two words are the same, there is a match by a single transposition [35].

Rule Based Techniques

Rule-based technique works by having a set of rules that capture common spelling and typographic errors and applying these rules to the misspelled word. In this technique, incorrectly spelt words are changed into the correct ones by using algorithms that try to express knowledge of typical spelling error patterns. The information is provided as rules [35]. They work by comparing the misspelled word to a set of criteria that identifies common spelling and typographical errors

These rules may include general morphological information, the lengths of the misspelled words, and more. For example, they may explain how to turn a verb into an adjective by adding the suffix "-ing." [35] . Every right word produced by this algorithm is accompanied by a correction suggestion. You can rank the ideas by summing the probabilities for the relevant rules, which are included in the rules. The concept of edit distance can be viewed as a particular application of a rule-based system with a limitation on the number of rules that can be employed. Edit distance can be considered as a subset of a rule-based technique with limitations on the possible rules [44].

Similarity Keys

In this technique, every word is mapping of every word into a key [35]. The key to the misspelled word is computed and words with similar key values to it will be generated as candidate suggestions for the misspelled word[35], [38]. Words are converted into similarity keys that reflect relationships between the characters in the words, such as positional similarity, material similarity, and ordinal similarity, using similarity key approaches [35].

***Positional Similarity*:** Its name relates to the degree to which the matching characters in two strings are located in the same place. It typically appears in an OCR text and literary comparison, and it is reportedly too limited to be utilized by itself for spelling correction.

Material Similarity: This describes how similar two strings are when the characters are in a different order. As a gauge of material resemblance, correlation coefficients between the two strings — the misspelled word and a word made up of the exact same letters but in a different order — have been utilized. The material similarity is thought to be insufficiently exact for the task of spelling correction because all anagrams have the same basic elements. For instance, the words "break" and "baker", "brush" and "shrub", etc. are anagrams of each other and have a lot in common.

Ordinal Similarity: Ordinal similarity, like position similarity, describes character similarity between two words when they are ordered similarly.

Each dictionary word is given a key, and only dictionary keys are compared to the key generated for the non-word. The Non-standard word for which the keys were calculated. As a recommendation, the word with the most comparable keys is chosen. This method is fast because it only processes words that have similar keys. With a good transformation algorithm, this method can handle keyboard errors.

N-gram Based Techniques

As we discussed in n-gram analysis for spell checking in Section 2.4.1, it is a subsequence of a sequence of letters or words with n = 1, 2, or 3 letters. N-grams can be used in two ways, either without a dictionary or together with a dictionary. It can be used either with or without a dictionary depends on the situation. N-grams have been employed in a variety of ways, including as trigrams, bigrams, and unigrams, in text recognition and spelling correction algorithms. N-grams are used to identify the positions in which the misspelled word error occurs without the use of a dictionary. If there is a special method for correcting a misspelled word so that it only contains accurate n-grams, that is what is used. This method's effectiveness is limited. N-grams are used to define the distance between words along with a dictionary, however the words are always checked against the dictionary.

Probabilistic Techniques

The probabilistic approach is based on statistical features of the language [38]. Initially, probabilistic approaches such as OCR were utilized for text recognition [35]. The two common methods are transition or Markov probabilities and confusion or error probabilities [35].

a) ***Transition or Markov probabilities***, like n-grams, determine the probability that a given letter will be followed by another given letter in a given language. These probabilities can be calculated by gathering n-gram frequency data from a big corpus.
b) ***Confusion or error probabilities*** give the chance of a particular character substituting another one in a misspelled word. When given a sentence to correct, the system decomposes each string into letter n-grams and searches the lexicon for word recommendations by comparing string n-grams to lexicon-entry n-grams. When we have access to a dictionary or index, transition probabilities are ineffective.

The main difference between the two probabilities is that transition probabilities depend on language, while confusion probabilities depend on source[35]. To explain, confusion probabilities vary depending on the source since different OCR devices employ different methodology and features, such as font types, and each device outputs a unique confusion probability distribution. Human error can also contribute to confusion probability.

Neural Networks

Neural networks are also an interesting and promising technique, but it seems like it has to mature a bit more before it can be used generally. Back-propagation networks are now used, with one output node for each word in the dictionary and an input node for every conceivable n-gram at every position of the word, where n is commonly one or two [44]. Only one of the outputs should be active at any one time, showing which dictionary terms the network recommends as a correction. This strategy works for tiny dictionaries (under 1000 words), but it does not scale well. The disadvantage of this approach is the time requirements are too big on traditional hardware, especially in the learning phase.

2.5. Summary

In this chapter, we highlighted the sociolinguistic aspects of Guragina Language varieties. There are more than 13 Guragina Language (also called Guragina) varieties. Linguistically, the sub-groups of Guragina are North Gurage (Kistane and Dobi), West Gurage (Meskan, Cheha, Ezha, Gumer, Muher, Inor, Endegagn, Enner, and Geto), and East Gurage (Welane, Silte, and Zay) are the major linguistic subgroups of Guragina. West Gudrage further classified into Central west Gurage (Cheha, Ezha, and Gumer) and Peripheral West Gurage (Inor, Endegagn, Enner, and Geto).

We also addressed the morphology of standardized Guragina in this chapter. It has a complex morphological structure as an Ethio-semitic language. Guragina verbs, nouns, adverb and adjectives can be simple or complex in structure. Affixes are used to create complex verb forms such as passive causative, verbal noun, and verbal reduplication stems. Guragina verbs are also negated by affixes, which correspond to the subject, object, and applicative affixes.

CHAPTER THREE: RELATED WORK

3.1. Introduction

Several studies have been conducted to explore various methods for spell checking as a challenge within natural language processing (NLP). The field of NLP has extensively researched spell-checking, resulting in the development of numerous spell-checkers for different languages worldwide. To provide a comprehensive understanding of spell detection and correction systems, this chapter presents a background study that examines key works in this area.

Section 3.2 focuses on the spell checkers for specific sub-groups within the Afro-Asiatic language family, particularly the Semitic languages like Amharic and Arabic. Section 3.3 covers the spell-checking approaches and strategies employed for Cushitic languages (Afan-Oromo) and Omotic languages (Kafi Noonoo) within the same language family. In Section 3.4, the discussion shifts to Indo-European languages such as English and Bengali, exploring the spell-checking techniques used for these languages. Section 3.5 digs into the spell checker developed for one of the languages in the Austronesian family, namely Malay. Finally, Section 3.6 provides a summary of the chapter, highlighting the key findings and contributions related to this field of study.

3.2. Spelling Checker for Semitic Language

3.1.1. Spelling Checkers for Amharic

Yacob [45] attempted to implement the Metaphor algorithm for the Amharic language. In this work, the author encoded letters with similar sounds into a single letter to have a single representation of words. In this paper, as much as possible the researcher tried to eliminate most errors and, in the author's, experiment used 116 canonically spelled words and 166 misspelled words. According to the author's report, the Amharic Metaphone algorithm's success was up to 90%. The primary strength of this study is well-planned stylographic redundancies, but its weakness is the usage of a limited corpus.

In another research, Melaku Tilahun and Tesfa Tegegne [11] develop a spell checker tool for Amharic texts and demonstrated it by integrating it with open office. In addition, the authors adopted word formation rules for the Amharic language, which can be integrated into the lexicon used by an Amharic spell checker. It is also a word-level spell checker, particularly a non-word error detector spell checker. That is, it does not consider real-word errors and white space. In this work,

the authors try to add some new features that are not addressed in previous works, such as internal inflected words and repeated words. In addition, the usage of Unicode data by the author is supposed to increase the performance of spell checking by avoiding transliteration. Finally, they try to measure the performance of the system by taking 5 experiments and calculating the recall and precision and got the overall performance of the system is 97.27%

In another study[46] , spelling mistakes in Amharic and English are corrected using a corpus-driven technique with noisy channel. The Damereau-Levenshten edit distance method is used to assess how closely produced words to a misspelled word are related to one another. The study focuses in spelling correction for non-word errors. Given that Amharic is morphologically rich, the authors argue that it would be time-consuming to compile a list of all language-dependent rules for spelling correction.

They created their own contemporary Amharic corpus (CACO), which was put together from publicly available Amharic newspapers, legal documents, journals, fiction, short stories, political books, the Amharic Bible, and children's books. For comparison, they utilized HaBiT, a big text corpus built from automatically crawled sites. Characters are converted into Latin-based characters after being extracted from paragraphs. The next steps involve replacing numerals with placeholders, splitting hyphenated words by deleting the hyphen and substituting a space character, and identifying and extracting unique phrases. Words are tokenized based on the two-dot separator character (:) and white space from the detected phrases. The remaining sentences that include words that only appear once across the whole corpus are then deleted. By presuming that the term that is only stated once is presumably spelt wrongly, this additional step is introduced.

The KenLM language-modeling toolset is used to train the language models. The error model was adapted from one developed by Norvig (2009) using 40.000 spelling errors. The most likely split possibilities based on the related word list and the CACO language model are used to separate terms with missing white spaces. Based on Amharic test data, the effectiveness of their method is assessed, and the findings are contrasted with those of the Aspell and Hunspell baseline systems.

89.4% accuracy, 80.6% recall, and 84.8% F1-score were the results of the evaluation for Amharic spelling error detection. Recall shows language coverage, F1 measure indicates the capacity to identify spelling errors, and precision flags all misspellings. Additionally, evaluation metrics for the HaBiT, Aspell, and Hunspell baseline systems were computed and compared. The findings

demonstrate that the suggested HaBiT technique improved spelling error detection (F1). However, when the term list from the HaBiT corpus was applied, the value of recall did not improve.

Additionally, the top five suggestions for the proposed system using CACO contain 77% of the correct spellings, as opposed to 34% for Hunspell, 62% for Aspell, and 75% for HaBiT. Suggestion lists were also assessed. The suggested approach excelled HaBiT, Aspell, and Hunspell in the top initial recommendations categories by 9%, 18%, and 35%, respectively. As long as they are typed using a QWERTY keyboard with direct mapping between keystrokes and characters, the authors' suggested approach may be applied to other written languages.

Using a hybrid methodology, Genet Assefa[47] developed an effective automated Amharic spell checker. The presented study employs the Metaphone and Edit Distance method. While the Metaphone technique is used to identify spelling errors, the edit distance approach is used to select the most likely correct word for the misspelled word. The author utilized a 125,000- dictionary word as well as 500 words for testing. According to the paper, when a phonetic algorithm and edit distance algorithm combination is used in the development of the spell checker, its error detection and correction suggestion skills are more than 95% successful. The technique taken in creating and implementing the spell checker is one of the system's fundamental flaws. The dictionary-based approach demands that each term be kept in a particular dictionary. Due to time and resource constraints, it is challenging to store all of the Amharic terms in the dictionary. Additionally, processing speed and time were not taken into account in the research.

In 2022, Maryamawit Shumetie [51] conducted research on a deep learning-based spell checker for the Amharic language. The author makes an effort to develop and implement a spell checker that takes into account the meaning of words both before and after the current word, which is meant to be corrected.

In the study, a deep learning dual-input model that can take into account the input word's context in both the right and left branches was presented. The experiment is conducted by setting up a baseline approach to the edit distance. The same number of data and training environments are used for the experiment. Since the loss function is based on Sparse Categorical Cross-entropy, accuracy and loss are used to evaluate the model. 116274 distinct words were collected from diverse sources for the system's training, testing, and assessment. Python was used to train the system on Google Collaboratory. It was put to the test by adding the incorrect words in a sentence.

It was able to make suggestions while identifying the incorrect word in the phrase that was provided. In the experiment, the suggested model outperformed the baseline model edit distance with a smaller loss value and higher accuracy. The precision of the edit distance was 0.68. The dual input encoder suggested model attained accuracy of 0.9349.

An optimization approach is used during the model training to reduce overcorrection and loss. Therefore, it can be concluded that the context-based dual-input model is more effective to the baseline technique for identifying and correcting spelling errors. For improved accuracy and decreased loss, the system may be improved in the future by using more deep learning algorithms and more data corpora. The author suggests that this technique may be employed in other languages as well.

3.1.2. Spelling Checkers for Arabic

Hamza et al. [52] present an Arabic spell checking approach based on morphological analysis and edit distance algorithms. They utilized a small corpus of 2784 misspelled words for testing. Finally, they tested evaluate and compared their system to the Levenshtein distance. They reported the results of the comparison by three error operations, such as insertion, deletion, and permutation, and discovered that their approach scores 85%, 81%, and 86% success for each error operation, respectively, while the other Levenshtein distance scores 44%, 81%, and 61% success for each corresponding error operation. The proposed system's average correction rate is 33.7% higher than the Levenshtein distance. The average correction time (0.10 ms) is significantly faster than the Levenshtein distance (0.19 ms).

Another study also carried out for Arabic spell checkers was by Shaalan et al. [49], who proposed automatic spell check for Arabic. They developed a tool capable of recognizing and suggesting corrections of misspelled input for common spelling errors. The system was composed of an Arabic morphological analyzer, lexicon, spell checker, and spell corrector. They limit their system to detect and correct spelling errors to isolated words. The tool developed tries to add missed characters, replace an incorrect character, remove the incorrect character and add a space to split words. The developed tool is helpful for automating the proofreading of the human-typed Arabic texts.

3.3. Spelling Checker for Afro-asiatic Language

Both Afan Oromo and Kafi Noonoo are languages that are part of the Afro-Asiatic language family, although they are categorized under separate branches within the family. Kafi Noonoo is Gongo sub group of the north Omotic family while Afan Oromo is Cushitic sub-group.

3.3.1. Spelling Checkers for Afaan Oromo

Gaddisa Olani & Dida Midekso [50] studied the design and implementation of a morphology-based spell checker for the Afaan Oromo language. The authors designed the spell checker using the dictionary lookup method together with the morphological analyzer. They used a total of 1811 words of test data and tested their system, and they reported using the following three matrices, lexical recall 88.62%, error recall 100% a, and precession 28.62%. The limitation of this work is the lexicon coverage of their knowledge bases is small.

3.3.2. Spelling Checkers for Kafi Noonoo

Fikru Tafesse and Yaregal Assabie [39] developed a morphological basis spell checking system for the Kafi Noonoo language. The suggested architecture of the spell checker in this study consists of four major components: tokenization, error detection, word suggestion, and error correction. To implement, dictionary lookup techniques and morphology-based approaches are utilized. The system prototype is created to test and evaluate the functionality and performance of the spell checker system. To test and assess the system, 2743 unique words were gathered from various sources. The evaluation metric was lexical recall, error recall, and precision, and the results were 95.91%, 100%, and 62.76%, respectively. The fundamental limitation of the study is the small vocabulary of root terms in the root dictionary. The intra-attention mechanism enhanced their model's overall performance, but it failed in several specific cases.

3.4. Spelling Checker for Indo-European Language

3.4.1. Spell Checker for Bengali Language

In 2017, Mandal and Hossain [53] developed a Bengali (Bangla) spell checker that could only handle typographic and phonetic mistakes. The authors both real-word and non-word errors in this work. To deal with these errors, they employ the confusion set approach. The disadvantage of this work is that it only used a small set without taking into account the morphology the language.

Hossain et al. l [37] did another study for a Bangla lexicon with about 1 million distinct terms. Based on this corpus and lexicon, the authors created a combined spell and grammar checker program that identifies unique spelling and grammatical errors while also providing suitable suggestions for both. The spell checker detects all forms of Bangla spelling errors with an accuracy rating of 97.21% using the Double Metaphone technique and Edit distance based on distributed lexicons and numerical suffix dataset. The shortcoming of this study is that split-word may not be detected as a misspelled word.

3.4.2. Spelling Checkers for English

Early research on spelling detection and correction is based on phonetic and string similarities like Metaphone and Damerau-Levenshtein edit distance algorithms [4]. The Levenshtein distance measures the similarity between two strings of characters. The Levenshtein distance and the Damerau-Levenshtein distance are similar, but the latter includes the transposition of two adjacent characters' type differences (which is a common typographical error). Candidate corrections are ranked from manually compiled lexicons with the help of these algorithms. GNU Aspell is a good example that follows this approach

GNU Aspell is a spell checker program developed by Atkinson [51]. It is a free and open-source spell checker designed to eventually replace Ispell. It can be used either as a library or as an independent spell checker. Its main feature is that it does a better job of suggesting possible replacements for a misspelled word than just about any other spell checker out there for the English language. Unlike Ispell, Aspell can easily check documents in UTF-8 without having to use a special dictionary. It also supports using multiple dictionaries at the same time. This spell checker also does not consider the morphological structure of the language and others.

The spell checker application developed by Bhaire et al. [52] uses an edit distance algorithm for spelling correction and a data structure tree for storing dictionaries. As a result, autosuggest is possible.

To improve its speed, the spell checker contains features such as multi-word suggestions. However, if the error is at the beginning of the word, the system does not provide any alternative spelling word from the list, and if the user-written word has too many errors, it may not offer any alternative spelling suggestion. The disadvantage of this system is that, because it is dictionary-based, it must keep all words (inflectional, derivational, and compound words) in the dictionary.

This consumes space and increases the likelihood of missing more common inflectional, derivational, and complex terms in the language.

Seth and Mieczyslaw developed a smart spell checker system called SSCS, which uses adaptive software architecture to correct users' errors. The system adapts to individual users by incorporating their feedback to modify its behavior. The architecture is designed like an adaptive controller, with a feedback mechanism for system adaptation. The spell checker utilizes five sources of knowledge and text files containing word lists as input. Its goal is to identify incorrect words, attempt to suggest replacements, and provide the user with options. The SSCS program demonstrates the Adaptive Software Architecture with nine Knowledge Sources (KSs) across three domains: Input, Error, and Evaluation. It uses dictionaries, including a user-defined dictionary, to detect incorrect terms. The system employs various knowledge sources such as character shifting, doubling, appending, removing, and switching to correct incorrect words. The use of multiple KSs improves system reliability and expands coverage of spelling problems. Compared to non-adaptive systems, their method achieved a significantly higher accuracy rate of selecting the correct corrected word in over 80% of cases. However, this system does not consider the morphology of the language, which is a limitation of the work.

3.5. Spelling Checker of Austronesian Language

3.5.1. Spell Checker for Malay

Basri et al. [49] developed an automatic spelling checker for Malay blogs, which aimed to detect and correct misspelled words without human input. The previous spell checker relied on a dictionary, but it could only correct words already present in the dictionary. To address this limitation, the researchers proposed a novel method consisting of five modules: tokenization, symbol elimination, removal of English words, stemming, and misspelled word identification. They tested their methodology using Malaysian-written Malay weblogs, specifically focusing on 11 out of 23 weblog categories. After processing the datasets, they obtained 3,046 unique words, and after removing English terms, they applied the spell checker to 2,675 words, successfully detecting and correcting 2,453 of them. However, their method missed 232 words (8.67% of all words). It's important to note that this work solely utilized a dictionary and did not account for the morphology of the language for inflectional, derived, or compounded words.

3.6. Summary

In this chapter, we conducted a thorough review of various spell checker studies conducted for different languages families, including Austronesian (Malay), Indo-European (English and Bengali), Afro-asiatic (Afaan Oromo and Kafi Noonoo) and Semitic (Arabic and Amharic). However, we couldn't find any specific studies related to spell checking in the Guragina Language. The absence of a comprehensive approach or methodology for generating root words. There is a lack of an effective and accurate spellchecker and suggester system for Gurage Language. Our main focus was to identify advanced techniques that are particularly suitable for handling the morphological complexity found in languages like Guragina. We examined the strategies, approaches, strengths, and limitations of each spell-checking system in relation to their respective research. The majority of spell checker systems rely on dictionary lookup and the Levenshtein edit distance algorithm as their foundation. Some publications have proposed morphology-based spell checks for languages like Amharic, Arabic, Kafi Noonoo, and Afaan Oromo. Through our analysis, we have determined that morphology-based techniques are considered state-of-the-art for addressing the challenges posed by morphologically complex languages. It is important to note that spell checkers designed for other languages cannot be directly applied to the Guragina Language due to its complex morphology, language variations, and other factors.

CHAPTER FOUR: DESIGN OF GURAGINA SPELL CHECKER

4.1. Introduction

In doing research, selecting the appropriate approach and design is essential. This leads to the use of a research paradigm that can lead to the desired outcome for the problem. A research methodology is chosen that incorporates procedures and techniques that are the best match for the research in order to achieve the goals and objectives of the study as well as to offer valid and trustworthy results. In order to address flaws that have been noticed, a new artifact must be designed, evaluated, and the findings must be presented. This research therefore perfectly fits the design science research technique, which is based on the approach described in [12] . In order to examine the issue and find a solution, many steps were taken as shown in Figure 4.1, which are covered in this chapter along with the research approach used.

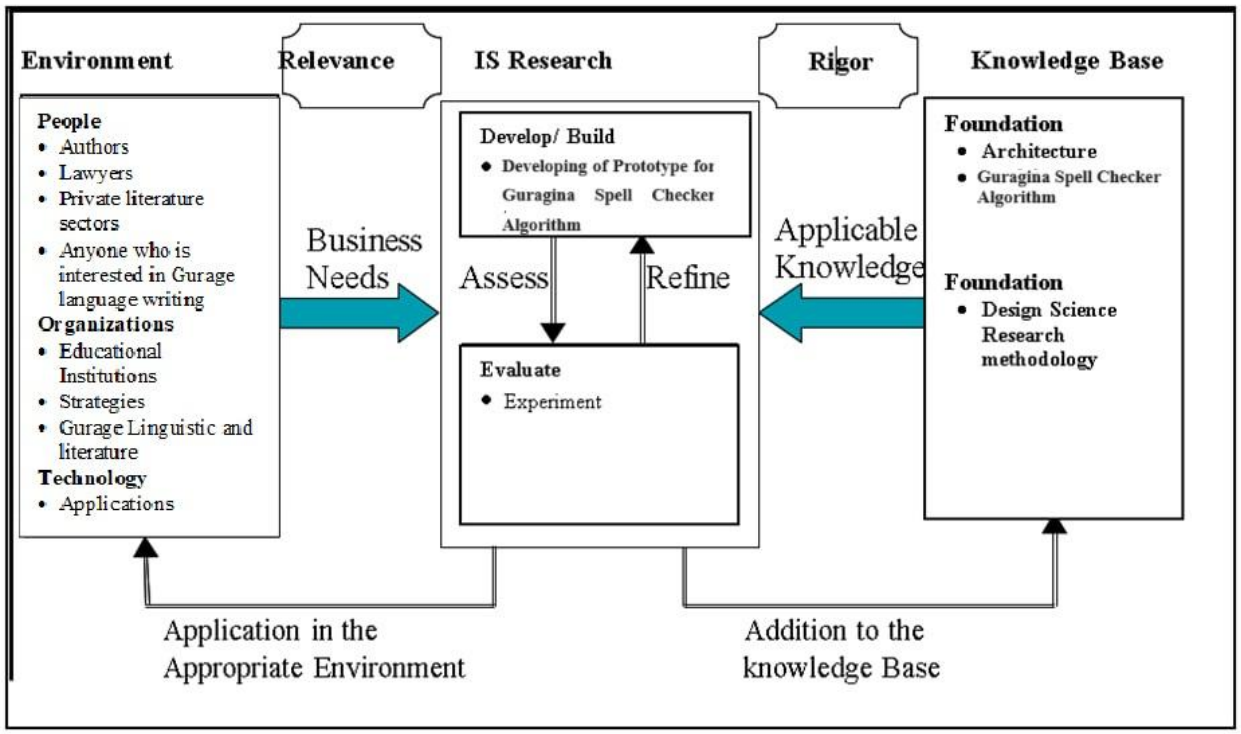

Figure 4. 1 *Design Science Research Process Model*

The spell checker algorithm's backend uses morphological characteristics and a stem and affix dictionary, which make up the other component of this design. The design and implementation of a spell checker for the Guragina Language were covered in this chapter.

4.2. General Architecture

Morphologically complex languages like Guragina are known for their extensive set of complex derivation rules, which poses challenges in defining and implementing all of these rules for tasks such as morphological generation and derivation. This can lead to performance issues when attempting to detect spelling errors by going through all the rules. To address this, the system utilizes an algorithm that resolves the performance concerns.

The proposed Guragina Language spell checker system has three major components to accomplish this. These are preprocessing, error detection, and correction suggestion components, as illustrated in the Figure 4.2.

The proposed Guragina Language spell checker system has three main components: preprocessing (including tokenization and vowel consonant (VC) mapping), error detection (involving stem extraction, and matching), and correction suggestion (with a Distance Calculator and Suggestion Ranking system). These components work together as illustrated in Figure 4.1 to accomplish the overall spell checking task.

The system uses two stored databases in the background. These are list of part of speech tagged stem word for different verb, noun, adverb, adjectives for matching and affixes are stored on another database. It also uses Guragina Stem Extraction, Guragina Word Distance Algorithm (GWDA) and Suggestion Ranking components. Fig 4.2 shows the detailed architecture of the system. The functionalities of each component are discussed in the following section.

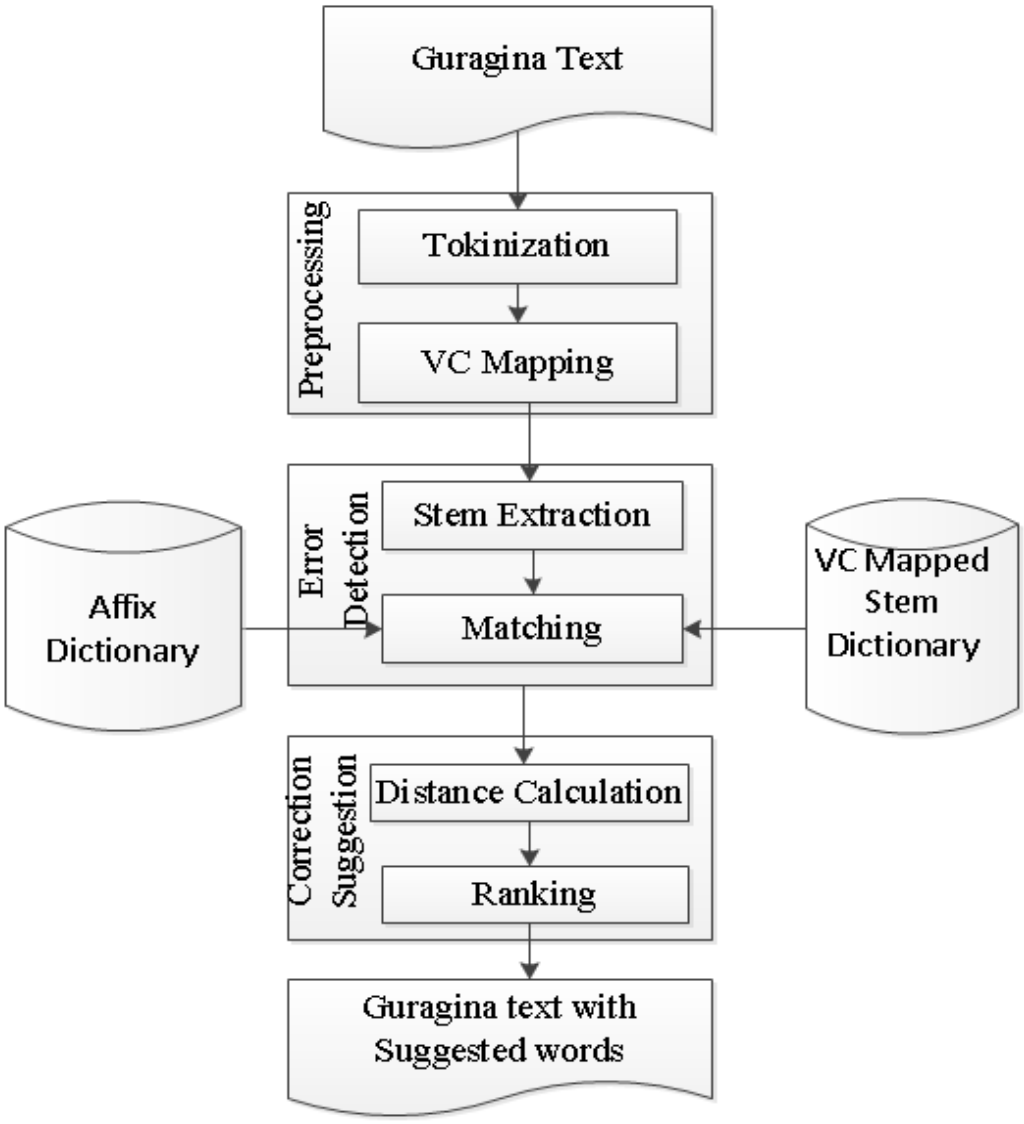

Figure 4. 2 *General Architecture*

4.3. Preprocessing

Preprocessing is an essential step in natural language processing (NLP) that involves transforming raw text data into a format suitable for further analysis or modeling. It typically includes several tasks aimed at cleaning, normalizing, and transforming the text. In In the Guragina language spell checker system, the preprocessing component primarily consists of tokenization and VC-mapping.

4.3.1. Tokenization

Tokenizer is responsible for breaking down a text or input into individual tokens or units, such as words, phrases, or characters. It segments the input text into meaningful units that can be processed further by other components. Tokenization is a fundamental step in many natural languages processing tasks, including spell checking. The tokenization algorithm for the terms is presented in Algorithm 4.1.

Input： PARAGRAPH

Output: WORDS_WITHOUT_PUNCT*: A list of tokenized words.*

```
1: Input a paragraph of text (PARAGRAPH).
2: Split PARAGRAPH into individual words using whitespace as the
     delimiter. Store the words in a list (WORDS).
3: Initialize an empty list to store the tokenized words without
     punctuation (WORDS_WITHOUT_PUNCT).
4: For each word in WORDS:
     a: Remove punctuation characters from the word.
     b: Append the modified word to WORDS_WITHOUT_PUNCT.
5: Output WORDS (tokenized words) and WORDS_WITHOUT_PUNCT (tokenized
     words without punctuation).
```

Algorithm 4. 1 *Tokenizer algorithm used by the spell checker*

In our system one of the preprocessing tasks is tokenization. Tokenization is a technique for dividing a block of text into separate tokens. Therefore, these tokens might be sentences, phrases, paragraphs, punctuation, or single words *(Lines 1)*. Our method is designed to assess a word's accuracy for the Guragina Language at the word level, hence in our instance for spell check, we take tokenization at the word level into consideration. Boundary delimiters like white space and punctuation are used to separate the text. Punctuation marks that appear at the end of a word are ignored and treated as separate tokens.

Words are separated from one another by white space. If a space is present, the word that follows it becomes a token. Therefore, in our text processing example, we consider blank space for tokenization *(Lines 2 -5).*

4.3.2. Vowel Consonant mapping

This program separates vowels and consonants in a word by iterating through each character and classifying it as either a vowel or a consonant. A vowel is a speech sound produced with an open vocal tract, typically represented by the letters 'አ''*ə'*,'ኡ''*u'*, 'ኢ''*i'*, 'ኣ'*'a'*, 'ኤ'*'e'*, 'እ''i', and 'ኦ'*'* o' in Guragina Language. Consonants, on the other hand, are speech sounds produced with a constriction or closure in the vocal tract, including letters such as 'ሐ'*'hɨ'*, 'ኽ' *'hʲɨ'*, 'ሑ'*'hʷɨ'*, 'ል'*'lɨ'*, 'ም'*'mɨ'*, 'ሙ'*'mʷɨ'*, 'ር' *'rɨ'*, 'ስ'*'sɨ'*, and so on. A vowel-consonant mapper takes a word as input and separates the vowels and consonants into different groups or lists. The purpose of a vowel-consonant mapper is to analyze the phonetic structure of words and to differentiate stem word and affix of each word. For example, the verb '*ቸነገም*' *'ʧ ənənəm'*, which means we came, map (convert) to '*ቸአንአንአም*' *'ʧ ənənəm'*.

4.4. Error Detection

The Error Detector component identifies errors in the input text. It analyzes the tokens or units provided by the Tokenizer, VC mapper and flags misspelled words. The Error Detector often relies on stem extraction, affix rule and matching to detect these errors.

4.4.1. Guragina Stem Extraction

Developing a stemmer for the Gurage language, a Semitic language with rich and complex morphology, is an important contribution, as stemming is a crucial step in various natural language processing (NLP) applications such as information retrieval, machine translation, and text classification.

For the Guragina language, one approach to represent the word structure could be through character embeddings. Character embeddings capture the sub-word-level information, which can be particularly useful for morphologically complex languages like Gurage.

The proposed stemmer has two key functionalities:

1. **Generating the derivation pattern:** This component is responsible for identifying the probable stem word by analyzing the word structure and extracting relevant prefixes, suffixes, and hyphens.

2. **Reversing the pattern:** This functionality allows the stemmer to reconstruct the original word from the identified stem and affix information.

The stemming algorithm, as outlined in Algorithm 4.2, is designed to process a word by removing specific prefixes, suffixes, and hyphens to extract the stem. This step is crucial for downstream NLP tasks, as it helps to normalize word forms and improve the performance of various applications. This step is crucial for subsequent NLP tasks, as it helps to normalize word forms and improve the performance of various applications.

Input: WORD

Output: STEM WORD

```
1: Read the word (WORD) to be processed.

2:   Iterate over each prefix in the list of Guragina prefixes
     (GURAGINA_PREFIXES):

     3: If the word starts with the current prefix:

          4: Remove the prefix from the word by slicing it from  the beginning.

          5: Break out of the loop.

6:   Iterate over each suffix in the list of Guragina suffixes
     (GURAGINA_SUFFIXES):

     7: If the word ends with the current suffix:

          8: Remove the suffix from the word by slicing it from the end.

          9: Break out of the loop.

10: Remove any remaining hyphens "-" from the word by replacing them with an
     empty string.

11: Return the processed word.
```

Algorithm 4. 2 *Stem extractor Algorithm.*

4.4.2. Matching

The spell checker follows a workflow that involves several steps. First, it accepts a block of text or single word, splitting it into words and remove additional punctuations. Then, it maps it to its vowel-consonant pattern.

A matching component is crucial for a spell checker, as it verifies the validity of a word by searching a dictionary of correctly spelled stem words in VC format. It accepts tokenized words from VC mapping or stems extraction and maps them accordingly, as shown in Algorithm 4.3.

The process starts by searching for the word in the stem database. If found, it is marked as correctly spelled (Lines 2). If not, the spell checker searches the affix database for matching affixes (Lines 4). If matches are found, the word is derived to its stem form and checked in the stem dictionary. If the stem word is found, the original word is marked as valid (Lines 4).

In cases where no matching affixes are found, the spell checker checks if the word is found in the stem dictionary. If it is found, the word is marked as correctly spelled. If it is not found, the spell checker flags it as incorrectly spelled and marks the position of the error (Lines 5).

Input: *a word (WORD) to be spell-checked*
Output: *found or not found in dictionary*

```
1: Convert WORD to its vowel-consonant (VC) representation
2: Search for WORD in the dictionary
3: If WORD is found:
   - Set result = "found in dictionary"
   - Mark WORD as correctly spelled
4: Else:
   - Search for matching affixes
   - If affixes match:
     - Drive WORD to its stem form
     - Search the generated stem in the dictionary
     - If the stem is found:
        - Set result = "found in dictionary"
      - Else:
       - Set result = "not found in dictionary"
       - Mark the error position
5: Return result
```

Algorithm 4. 3 *General Matching Algorithm*

If none of the derivation patterns from the matched rules match the word, the spell checker marks it as incorrectly spelled and notes the position of the error. This information is later used to generate corrections (Lines 15-16). If the affixes do not match or are not close to each other, the spell checker marks the error position and labels the word as invalid. If the result is an incorrectly spelled word, the spell checker replaces the characters at the error position with the probable correct characters and generates suggestions. It also searches for closer stem words and performs the same replacement and suggestion generation process (Lines 17-22). This workflow outlines the steps taken by the spell checker to determine the correctness of a given word and provide suggestions or corrections if needed.

4.5. Correction Suggestion

The Correction Suggester component provides suggestions or recommendations to correct the identified errors or improve the input text. It leverages the information from the Error Detector and the results of the Distance Calculator to generate a set of potential corrections or improvements by adopting the Ratcliff algorithm.

The Ratcliff algorithm, also known as the Longest Common Subsequence (LCS) algorithm, is a sequence matching algorithm that can be represented mathematically.

Given two sequences, such as words or strings, the algorithm finds the longest common subsequence (LCS) between them. A subsequence is a sequence derived from the original by deleting some or no elements without changing the order of the remaining elements. The algorithm calculates the length of the LCS between the two sequences. The longer the LCS, the more similar the sequences are considered to be. Based on the length of the LCS, the algorithm computes a similarity ratio, which is a value between 0 and 1. This ratio represents the similarity between the two sequences, where 1 indicates an exact match and 0 indicates no similarity. The algorithm sorts the sequences based on their similarity ratios, with higher ratios indicating closer matches. Let's break down the algorithm mathematically step by step:

Given two sequences, A of length m and B of length n, where $A = [a_1, a_2, ..., a_m]$ and $B = [b_1, b_2, ..., b_n]$, the Ratcliff algorithm can be defined as follows:

Compute the Longest Common Subsequence (LCS) matrix, C, of dimensions (m+1) × (n+1), initialized with zeros.

Iterate through each element ai in sequence A and each element bj in sequence B:

If ai is equal to bj, set C[i+1, j+1] = C[i, j] + 1.

Otherwise, set C[i+1, j+1] = max(C[i, j+1], C[i+1, j]).

The length of the LCS between sequences A and B is given by C[m+1, n+1].

Calculate the similarity ratio between A and B as the ratio of the LCS length to the maximum length of either sequence, max (m, n): similarity ratio = LCS_length / max (m, n).

Sort the sequences based on their similarity ratios, with higher ratios indicating closer matches.

This algorithm finds the longest common subsequence between two sequences and computes a similarity ratio based on the length of the LCS. The higher the similarity ratio, the more similar the sequences are considered to be.

4.6. Distance Calculator

The Distance Calculator computes the similarity or dissimilarity between different units or tokens in the text. It quantifies the distance, or measure of dissimilarity, between two units, often using metrics like edit distance, cosine similarity, or Levenshtein distance. In this study we designed distance calculator called Guragina Word Distance Algorithm (GWDA) used for Guragina Language, as shown in the Algorithm 4.5.

This component has the task of determining the distance between two words in the Guragina language. It utilizes a specially designed algorithm that provides to Guragina characters. A preliminary experiment was conducted to compare different distance calculation algorithms, and it was found that none of them yielded the desired result in terms of accurately measuring the difference between Guragina words.

Input: *two word*

Output: *distance of the word*

1. Initialization:

- *Read the two words, word1 and word2.*
- *Initialize two variables: comparisons to 0 (to count the number of character comparisons) and distance to 0 (to accumulate the difference score).*

2. Block Identification:

- *Identify the longest matching block of characters between the two words.*
- *Identify the remaining unmatching blocks of characters.*

3. Character Comparison:

- *Iterate through each unmatching block:*
  - *For each character in the block:*
    - *Compare the corresponding characters from word1 and word2.*
    - *Increment comparisons by 1.*
    - *Calculate a partial difference score based on the character families and orders:*
      - *If both characters belong to the same family and have the same order, add 1 to distance.*
      - *If both characters belong to the same family but have different orders, add 0.7 to distance.*
      - *If both characters belong to different families but have the same order, add 0.5 to distance.*
      - *If both characters belong to different families and have different orders, add 0.3 to distance.*
      - *If one of the characters is an empty string (''), add 0.1 to distance.*

4. Matching Block Contribution:

- *Iterate through the matching block:*
  - *Add the length of the matching block to distance.*

5. Difference Calculation:

- *Calculate the final difference score by dividing distance by comparisons.*
- *Return the difference score.*

Algorithm 4. 4 *Guragina Word Distance Algorithm (GWDA) used by the spell checker*

The distance calculator employs the algorithm presented in Algorithm 4.2. The algorithm begins by identifying matching blocks of characters between the two words. It also detects unmatched blocks of text in both words. Taking into account the matched blocks, it compares the combinations of characters from the unmatched blocks in the corresponding positions and calculates the average distance. Characters that belong to the same row of the Guragina alphabet are assigned a distance value of 0.7. This is based on the assumption that if a user mistakenly types a character within the same row, they have likely pressed the correct key for the character's family and only made an error with the vowel. For example, in Abyssinica SIL font, entering the character ሐ (hä) requires pressing the 'H' key followed by 'E', while entering ሑ (hu) requires 'H' followed by 'U'. Therefore, if the user types 'hu' instead of 'hä', the difference between the two is very small, resulting in a higher distance value. The distance calculator is designed so that a distance of one indicates that the compared words are exactly the same. Consequently, the value decreases towards zero as the difference between the words becomes larger (Fig 4.3). Thus, in this algorithm, a higher distance value signifies that the words are closer in similarity.

A distance of 0.5 is assigned to characters that belong to the same column in the Guragina alphabet, as it is considered more likely that the user made a mistake with the family rather than the order. Characters without a matching family and order are given a distance of 0.3, as this substitution is deemed less probable. Comparing a character to a missing character results in a distance of 0.1, while identical characters are assigned a distance of 1. These distance assignments are supported by the findings discussed in Section 5.2.1 and are depicted in Figure 4.4.

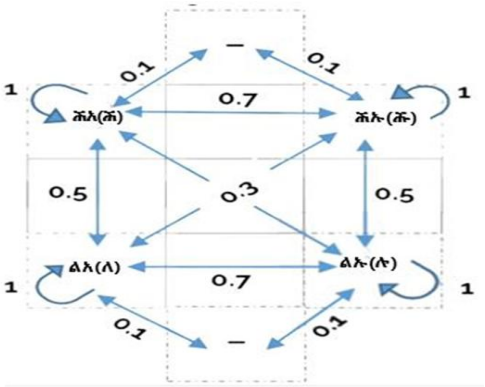

Figure 4. 3 *Examples of distance calculation's values*

To determine the weights assigned to Guragina characters based on their family and order, an analysis is performed using two words, namely "ቀፓም" and "ቀፐር". These words are chosen because they contain pairs of characters that share the same identity, characters from the same family, and characters in the same order.

Initially, the analysis relies on the following assumptions:

1. The distance between two words ranges from 0 to 1, where a value of 1 indicates that the words are exactly the same.
2. Fig. 4.5 illustrates five conditions, denoted as a, b, c, d, and e, each assigned a value between 0 and 1.

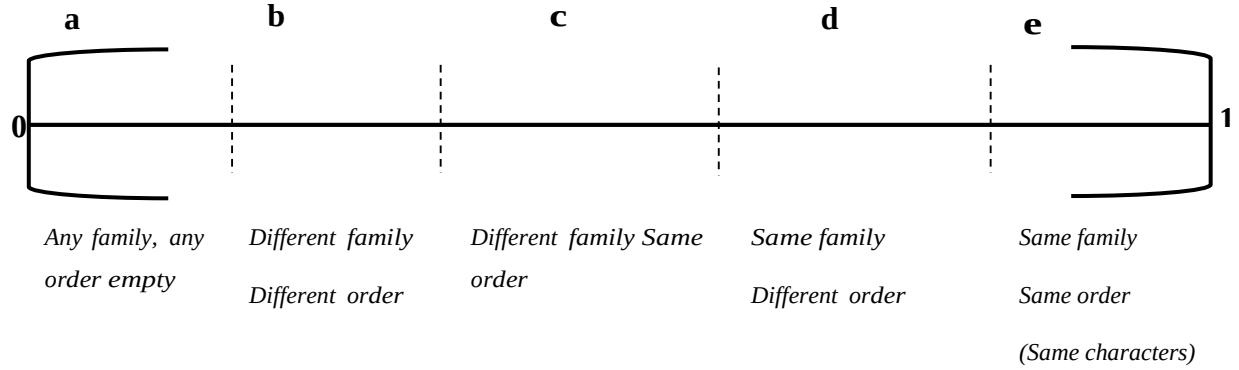

Figure 4. 4 *Distance between characters by family and order*

- Represents any family, any order, and is assigned a value of 0.
- Represents different families, and is assigned a value greater than 0 and less than 1.
- Represents different orders within the same family, and is assigned a value greater than the average.
- Represents different families and different orders, and is assigned a value less than the average.
- Represents the same family and the same order (same characters), and is assigned a value of 1.

3. The distance between characters of different families and orders is considered to be in the middle, resulting in a distance value of 0.5.
4. The distance between characters of the same family but different orders are greater than the average value, while the distance between characters of different families and different orders

is less than the average value. This is because the probability of making the first mistake (same family, different order) is considered to be higher than the second mistake (different family, different order).

By incorporating these assumptions, the weights between Guragina characters based on their family and order can be determined.

4.6.1. Suggestion Ranking

This component first calculates the distance between each suggestion and the candidate word using the Guragina word distance algorithm (GWDA) function. As shown in the Algorithm 4.5, It then filters out the suggestions that are too far away based on the distance_threshold parameter, and sorts the remaining valid suggestions by their distance from the candidate word. Finally, it returns the top max_suggestions suggestions.

Input:

- candidate_word (str): The word to be checked for spelling errors.
- suggestions (list): A list of suggested words.
- distance_threshold (int, optional): The maximum allowable distance between the candidate word and a suggestion. Defaults to 2.
- max_suggestions (int, optional): The maximum number of suggestions to return. Defaults to 5.

Output:

- A list of tuples, where each tuple contains a suggested word and its distance from the candidate word, sorted by distance in ascending order.

1. Initialize an empty list called `distances` to store the distance between each suggestion and the candidate word.

2. For each word in the `suggestions` list:

a. Calculate the distance between the `candidate_word` and the current word using the `gwda` function.

b. Add a tuple containing the current word and its distance to the `distances` list.

3. Initialize an empty list called `valid_suggestions` to store the suggestions that are within the `distance_threshold`.

4. For each tuple in the `distances` list:

a. If the distance in the tuple is less than or equal to the `distance_threshold`, add the word from the tuple to the `valid_suggestions` list.

5. Sort the `valid_suggestions` list by the distance value in ascending order.

6. Return the top `max_suggestions` suggestions from the `valid_suggestions` list.

Algorithm 4. 5 *Suggestion Ranker Algorithm*

Together, these components form an architecture that aims to process and improve the input text by tokenizing it, detecting errors, calculating distances, and suggesting corrections.

In summary, the algorithm and architecture provide a solution for efficiently handling the complexities of morphologically complex languages, specifically in the context of spell checking and word suggestion generation.

CHAPTER FIVE: EXPERIMENT

5.1. Introduction

The primary objective of the assessment is to assess the performance of the proposed algorithm. This chapter provides an overview of the data collected to develop the GLSC prototype and test its effectiveness. It also discusses the tools employed in building the algorithm. Additionally, we present the prototype that was developed, along with the evaluation and test results obtained from the experiment. Lastly, we present our findings and conclusions based on the assessment.

5.2. Data Preparation

In contrast to other Ethiopian languages like Amharic, Afaan Oromo, and Tigrigna, the Guragina Language has extremely limited resources and is challenging to acquire in digital form. It does not have publicly available annotated corpus text for spell checker and any other NLP applications.

Source data

To build a dictionary lexicon and assess the spell checker system, this study primarily relies on a lexicon and textual data from the Gurage zone cultural and tourism office. Additional secondary data sources include books, children's storybooks, and the New Testament of the Bible.

Data description

The collected data forms a comprehensive linguistic resource that captures the rich vocabulary of the Guragina language. This dataset contains 656 words, including 256 unique adjectives, 881 nouns, and 1,300 verbs. It also includes detailed morphological information, such as 18 adjective prefixes, 18 adjective suffixes, 4 noun prefixes, 70 noun suffixes, 18 verb prefixes, and 340 verb suffixes. This extensive data allows for systematic analysis of word formation patterns in the Guragina language.

The researchers worked with linguistic experts to understand the structure and characteristics of the Guragina language and to prepare the dictionary lexicon and affix dictionaries with rules. adverb and their affix.

Table 5. 1 *Composition of the lexica based on the category of the stem words and affixes*

Category	Verb	Noun	Adjective	Verb prefix	Verb suffix	Noun prefix	Noun suffix	Adjective prefix	Adjective suffix
Count	1,300	881	256	18	340	4	70	18	18

5.3. Implementation

Using the free version of a Google Collaboratory, the Guragina spell checker prototype was created and tested on Python. Python was chosen as the programming language for combination of an open-source nature; object-oriented architecture, straightforward syntax, rich NLP libraries, cross-platform compatibility, simplicity, efficiency, and beginner-friendliness make it a suitable choice for developing the Guragina spell checker prototype. Microsoft Office 2019 for documentation and Sublime Text are two other software tools utilized in the development of GLSC. Sublime Text is a popular text editor that is widely used by developers and researchers. It provides a range of features and functionalities that make it a valuable tool for writing and editing code, including Python scripts, as well as preparing and manipulating data in various formats. The Windows 11 operating system was used, providing improved speed, security, and productivity enhancements for a hybrid work environment.

Google Colab was employed for its extensive resources and faster results. It allows writing and executing Python code in any browser without configuration, providing access to free GPUs and easy sharing of work. Researchers can import datasets and evaluate models using Colab notebooks, utilizing the power of Google's hardware, including GPUs and TPUs, regardless of the researcher's own machine's capabilities. The Guragina Keyman keyboard and Abyssinica SIL font were used to type and prepare necessary documents in Guragina.

5.4. Prototype of the System

The Python programming language is used to implement the Guragina spell-checking system, which looks for misspelled words and suggests a list of proper ones in their place. The created prototype has the capabilities of identifying incorrect words, offering suggestions, and, depending on the calculated distance, replacing incorrect words with right ones from the proposed list.

When a user types a word and clicks the spell check button, the spell checker verifies the word's accuracy; if the word is misspelled, it is highlighted in red. After that, a selection of suitable alternative terms for the misspelled word is shown, and the user may select one of these words to take its place. The misspelled word is then instantly changed with any of the proposed words the user has chosen from the created list by clicking the replace button.

The user then selects any of the suggested words from the generated list and clicks the replace button, then the misspelled word is immediately replaced with the word. Figure 5.3 shows the prototype that displayed after the user makes a replacement for a misspelled word by choosing from the suggestion list generated by the prototype. As shown in Figure 5.2, there are buttons used for different purpose as explained in the following

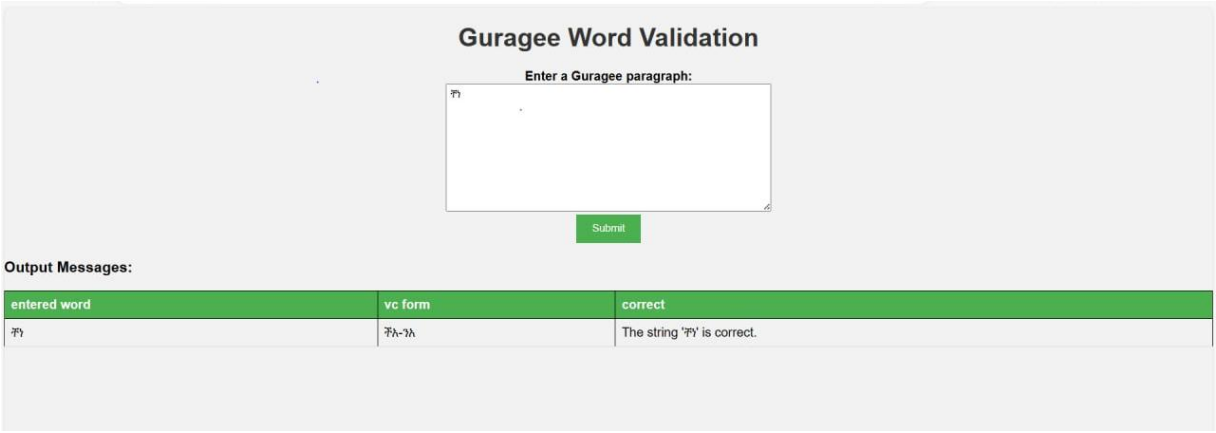

Figure 5. 1 *Prototype of the system*

5.5. Evaluation of the Ratcliff / Obershelp Algorithm

The main idea behind the Ratcliff algorithm is to find the longest common subsequence (LCS) between two sequences and use it to calculate the similarity or distance measure. The algorithm works by iteratively finding the LCS and recursively applying the same process to the remaining

portions of the sequences. It has a time complexity of O(N*M), where N and M are the lengths of the input sequences. This makes it suitable for comparing long sequences efficiently. Additionally, the algorithm can handle sequences with gaps, which is useful in scenarios where sequences may have insertions or deletions.

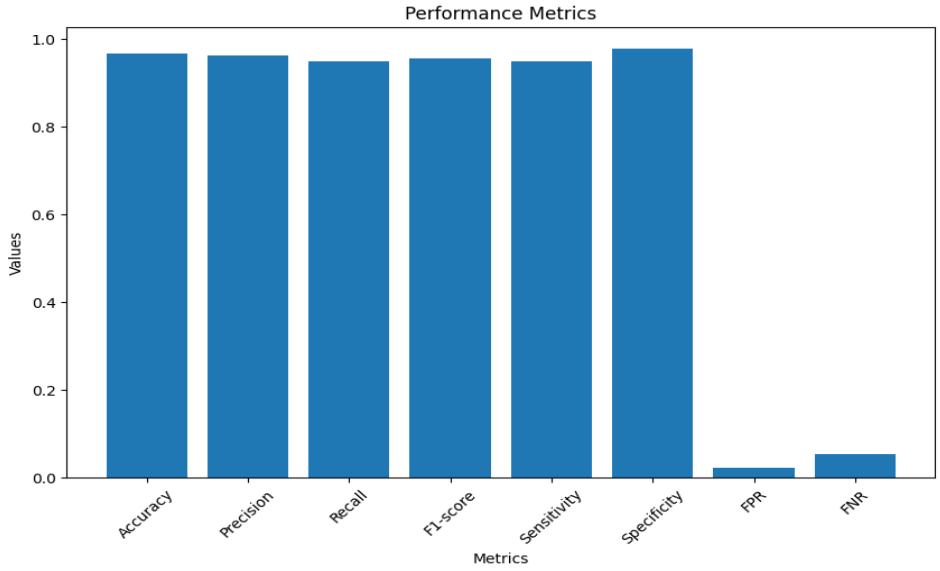

Figure 5. 2 *Evaluation distance calculation algorithm using histogram*

The results of this study indicate that the Ratcliff algorithm has achieved the following.

Table 5.2 *Evaluation distance calculation algorithm*

Metrics	result
Accuracy	98.27%
Precision	98.07%
Recall	97.75%
F1-Score	95.45%
Sensitivity	97.275%
Specificity	98.86%
FPR	11.36%

5.6. Performance Results of Distance Calculator

This experiment involved a comparison of eight distance similarity measure algorithms applied to the verb ሰፈ (*Sefə*) and its inflections. The algorithms evaluated in this experiment include cosine similarity, Euclidean distance, Manhattan distance, Minkowski distance, Jaccard similarity, Python's sequence matcher, Levenshtein distance, and Ratcliff/Obershelp. Table 5.2 presents the calculated distances for each word pair obtained by each distance calculation algorithm.

Table 5. 2 *Performance Results of Distance Calculators*

ሰፈ vs					
Distance	ሰፈ	ላትሰፈች	ቲዩትሰፎ	አትሰፍ	ታትሰፈዋ
levenstein	0	2	4	3	3
Sequence matcher	0	1.73205	2.23607	2	1.73205
Manhattan	0	322	157	319	274
Minkowski	0	238.446	112.3442	237.034	220.04
Guragina Word Distance_ _ Algorithm (GWDA)	1	0.46	0.4	0.475	0.46

The results obtained from Table 5.2 indicate that the Euclidean distance, Manhattan distance, Minkowski distance, and Levenshtein distance algorithms yield a distance of zero when comparing identical words. On the other hand, a distance value of one implies similarity for the cosine similarity, Jaccard similarity, and Python's Sequence Matcher algorithms.

It can be observed that the first five algorithms produce identical distances for the derivations or inflections of the verb ሰፈ (*Sefə*). Although these derivations are closer to the stem word, the degree of closeness is not uniform. This suggests that the Euclidean distance, Manhattan distance, Minkowski distance, cosine similarity, and Jaccard similarity are not suitable for word-level comparison in Amharic. Instead, they are more appropriate for comparing documents written in English or similar languages.

Python's Sequence Matcher algorithm yields better results compared to the previous algorithms. However, since it employs an exact matching technique, it fails to identify the similarity between

words like አትሴፈች and ታትሴፈዋ, despite their close relationship to the stem verb. On the other hand, the Levenshtein distance algorithm is commonly used for string comparisons in language processing, counting the number of operations required to transform one word into another. However, when applied to Guragina, it does not effectively capture the closeness or relationship between characters within the same family and in the same order. The distances obtained for 'ትዩትሴፎ', 'አትሴፈች' and 'ታትሴፈዋ' are significantly higher than expected. This indicates that Sequence Matcher and Levenshtein distance are not fully suitable for Guragina string comparison. Based on these findings, none of the algorithms evaluated are suitable for Guragina. Therefore, a new distance calculation algorithm is developed and implemented. The verb ሴፈ and its inflections' distances calculated using the distance calculator are shown in Table 5.2 demonstrates the distances computed for the same word pairs using the new distance comparison algorithm, highlighting its ability to identify the closeness between characters within the same family and in the same order.

5.7. Comparison of the Proposed System with Existing Ones

The main aim of this test is to evaluate the number of correctly accepted inflected and derivate words and correctly flagged out misspelled words by the spell checker system of Guragina Language. We also evaluate the performance of the spell checker algrithm by using evaluation metrics which are represented in Table 5.1

The system shows an average precision and recall of **96%** and **99%** respectively. Table 5.3 presents a comparison of the evaluation results of the current experiment with the results from a few other similar previous works focused on Semitic languages. From the figures shown on the table, Predictive Accuracy of **95%** shows that the system has highest possibility of handling any candidate word accurately, whether the word is valid or invalid. **99%** Recall shows that the system has the highest completeness to label correctly spelled words as valid. Lexical and Error Recall values of **96%** and **89%** designate the inclusiveness of the lexicon of the algorithm [39]. It also shows the efficacy of morphological analysis performed by the spell checker [39].

Table 5. 3 *Comparison* of *Evaluation results with previous similar work*

Research Work	Accuracy	Precession	Recall	F1 Score
[46]	89.4%	-	80.6%	84.4%
[50]	88.62%	28.62%	**100%**	-
[52]	44%	81%	-	-
[55]	87%	70%	89%	78%
This Work	**98.27%**	**98.07%**	97.75%	**95.45%**

5.8. Discussion

Based on the experimental results presented in Table 5.3, we achieved accuracy of 98.27% precession of 98.07%, recall of 97.75%, and F1 Score of 95.45%. These findings indicate that our system performs well with certain limitations.

One of the identified reasons is the limited lexical coverage of the dictionary used. It does not encompass all the words in the language, including personal names, place names, and scientific terms. Consequently, some correct words are flagged as invalid due to their absence from the dictionary. To address this issue, enhancing the stem dictionary by including the stem forms of these words would improve the system's performance.

The second reason is the mismatching of affixing rules. There is a possibility of applying an affix rule to a word of a different affix class, resulting in the recognition of meaningless words as correct. For instance, the prefix አ- works correctly with words like አ + ጣፈ = አጣፈ. However, when applied to the stem word አ+ አመ, it produces a word that does not exist in the language (አአመ). Similarly, the suffix -ኦት functions properly with words like ወረ +ኦት = ወሮት. However, when applied to the stem word ከና + ኦት, it generates a word that is not part of the language (ከኖት). Addressing these affixing rule mismatches would contribute to the system's accuracy.

Furthermore, the complexity of Guragina morphology should be taken into consideration. Currently, there is no developed morphological analyzer or generator specifically designed for Guragina. Developing a comprehensive morphological analyzer and generator would enhance the performance of the spell checker system.

In summary, improving the lexicon coverage of the stem dictionary, ensuring accurate classification of each stem word with the appropriate affix class, and implementing a full-fledged set of affix rules for the language would significantly enhance the system's performance.

CHAPTER SIX: CONCLUSION AND FUTURE WORK

6.1. Conclusion

As a community, writing using by their own language is very important for individuals, government and non-governmental organization. Specially, the languages with very low resource and not well standard, like Guragina the use of writing and spelling checker is more than facilitating document preparation. The spell checker support for the language development and standardizing effort by encouraging the user to write by using the languages alphabet.

To achieve this, this study has been done to design a model, implement and develop a prototype for Guragina Language error spelling checker. It involved study morphology, word derivation, and spelling errors that can occur in Guragina text writing and development of development spell checker. In addition, we adopted word formation rules for Guragina Language which can be integrated to the lexicon used by Guragina spell checker.

In the field of natural language processing, there are several research areas for different languages of the world. Among these areas, spell checking is one of the research areas where NLP is applied. Spell checking is about detecting misspelled words in a document and correcting misspelled words using possible suggestions provided during spell checking. Spell checker performs two main tasks, namely error detection and error correction.

Various techniques have been proposed to develop a spelling checker for different languages. Among them, N-gram analysis and dictionary search approaches are applied for error detection and edit distance, similarity key, neural network, N-gram, rule-based, probabilistic and noisy channel model are techniques for error correction.

During the process of creating the Guragina Language spelling checker, this study used a rule-based approach for error correction along with the dictionary search method and the morphological analyzer (a morphology-based spelling checker) to identify error words and lengths. Only typographical errors, or non-word errors, in the language are intended to be detected by this spell checker. The focus on non-word error detection in this research was a strategic decision given the complexity and limited resources available for the Guragina language. Addressing the non-word errors is a crucial first step that will provide valuable insights and lessons that can then be applied towards the more complex task of real-word error detection and correction. Addressing non-word errors first provides a solid foundation before tackling the more challenging task of real-word error

detection and correction. This phased approach allows meaningful progress on this under-resourced language, rather than attempting to solve the entire spell checking problem all at once. The intention is to use the non-word spell checker as a starting point, then leverage that knowledge and capability to progressively tackle real-word error handling in the future, as the abstract outlines.

As a result, real word errors are not taken into account in this study. Future research will need to analyze the language's semantics and grammar before introducing real word errors. You can use the created GLSC independently or as a component of other word processing programs.

By manually entering text in the text editor's provided space or a user can provide a block of text. The tokenizer separates the text block into individual words, and the morphological analyzer examines each one using stem and affix dictionaries. The Guragina spelling checker was developed using more than 6000 stem words from this root dictionary.

Lastly, evaluations and tests are conducted to verify the spelling checking system using sample data of Guragina Language words randomly collected from papers and books. To verify the effectiveness of the system's lexical recall, error recall and precision evaluation metrics were conducted. Based on these evaluation metrics, we obtain accuracy of 98.27% precession of 98.07%, recall of 97.75%, and F1 Score of 95.45%.

In general, we conclude that our morphology-based spelling checker can still perform well, which needs to be improved, especially more work on the morphology of the language.

6.2. Contribution of the Thesis

The main contributions of this study are described as follows.

- We made up Guragina Stem Dictionary
- We prepared affix pattern of different word categories.
- We introduced a hybrid algorithm by combining both Ratcliff algorithms with a new distance calculator algorithm that has advanced key features select highly approximate suggestions, which can be adapted to other morphological complex languages.

6.3. Future Work

There are various research areas within natural language processing (NLP) that can be explored for local languages, and spell checking is one such task. A spell checker serves as a foundation for other NLP applications, including part-of-speech taggers, grammar checkers, machine translation, question-answering systems, text-to-speech synthesis, speech-to-text synthesis, anaphora resolution, text summarization, dialogue systems, and more. For future work in this field, we recommend the following key activities:

- Increasing the coverage of the stem dictionary's lexicon size would enhance the performance of the spell checker. This expansion would involve including more stem words in the dictionary.
- The proposed spell checker system, which relies on morphological analysis, would benefit from the development of a comprehensive morphological analyzer specifically designed for the Guragina Language. Currently, a simple morphological analyzer is used due to the absence of a fully developed analyzer for Guragina.
- Expanding the spell checker to detect and correct real-word errors by considering the semantics and grammar of the Guragina Language would make the system more interactive. This would involve incorporating language-specific knowledge to handle errors beyond non-word errors.
- Extending the work by testing and evaluating the spell checker with a large corpus that encompasses diverse data sources would provide a more comprehensive assessment of its performance.
- Developing a standard Guragina Language corpus would be beneficial, as it is currently a resource-scarce language. Such a corpus would motivate researchers to explore various NLP applications beyond spell checking and reduce the time required for corpus collection when evaluating system performance.
- While this work presents a spell checker system as a demo, further projects could involve developing a fully functional Guragina spell checker that can be seamlessly integrated with other NLP applications such as machine translation, text summarization, and more.

By pursuing these recommendations, future research in Guragina spell checking and NLP applications can make significant advancements and contribute to the development of language technology for local languages. In this work, the researcher concluded that the implementation, testing and evaluation of system modeling for non-standard word Guragina spelling checker and correction has fulfilled the objectives of providing a tool with a reasonably good suggestion support in Guragina language for spelling checker text entry.

REFERENCES

[1] R. Egger and E. Gokce, “Natural Language Processing (NLP): An Introduction,” in Applied Data Science in Tourism: Interdisciplinary Approaches, Methodologies, and Applications, R. Egger, Ed. Cham: Springer International Publishing, 2022, pp. 307–334. doi: 10.1007/978-3-030-88389-8_15.

[2] C. I. Ratnasari, S. Kusumadewi, and L. Rosita, “A Non-Word Error Spell Checker for Patient Complaints in Bahasa Indonesia,” Int. J. Inf. Technol., p. 5, 2017.

[3] K. Kukich,“Techniques for automatically correcting words in text,” ACM Comput. Surv., vol. 24, no. 4, pp. 377–439, Dec. 1992, doi: 10.1145/146370.146380.

[4] F. J. Damerau, “A technique for computer detection and correction of spelling errors,” Commun. ACM, vol. 7, no. 3, pp. 171–176, Mar. 1964, doi: 10.1145/363958.363994.

[5] S. Singh and S. Singh,“Systematic review of spell-checkers for highly inflectional languages,” Artif. Intell. Rev., vol. 53, no. 6, pp. 4051–4092, Aug. 2020, doi: 10.1007/s10462-019-09787-4.

[6] R. C. de Amorim and M. Zampieri, “Effective Spell Checking Methods Using Clustering Algorithms,” Proc.Recent Adv. Nat. Lang. Process.Pages 172–178 Hissar Bulg. 7-13 Sept. 2013, p. 7, 2013.

[7] A. A. Patil and P. R. Sharma,“Study and Review of Selective Spell Checking Approaches,” vol. 8, no. 2, p. 8, 2019.

[8] S. Sooraj, K. Manjusha, M. Anand Kumar, and K. P. Soman,“Deep learning based spell checker for Malayalam language,” J. Intell. Fuzzy Syst., vol. 34, no. 3, pp. 1427–1434, Mar. 2018, doi: 10.3233/JIFS-169438.

[9] Derib Ado, Almaz Wasse, and Janne Bondi, Eds., “The languages of Ethiopia: Aspects of the sociolinguistic profile,” in IMPACT: Studies in Language, Culture and Society, vol. 48, Amsterdam: John Benjamins Publishing Company, 2021, pp. 1–12. doi: 10.1075/impact.48.int.

[10] S. Völlmin,“Towards a Grammar of Gumer - Phonology and Morphology of a Western Gurage Variety,” University of Zurich, 2017. doi: 10.5167/UZH-149973.

[11] A. Melaku Tilahun and Tesfa Tegegn,“AUTOMATIC SPELLING CHECKER FOR AMHARIC LANGUAGE,” p. 92, 2017.

[12] K. Peffers, T. Tuunanen,M. A. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” J. Manag. Inf. Syst., vol. 24, no. 3, pp. 45–77, Dec. 2007, doi: 10.2753/MIS0742-1222240302.

[13] K. Kiran, B. V. Kiran, D. C. Sai, G. V. Vamsi, and P. R. Salomi, “Face Mask Detection Using Machine Learning.” Rochester, NY, Sep. 17, 2021. doi: 10.2139/ssrn.3925736.

[14] “(PDF) Inherent Intelligibility among Guragina Varieties.” https://www.researchgate.net/publication/312093428_Inherent_Intelligibility_among_Guragina_Varieties (accessed Apr. 12, 2022).

[15] R. Meyer, “Gurage,” Stefan Weninger Ed Semit. Lang. Int. Handb. Berl. N. Y. Gruyter Mouton 1220-1257, Jan. 2011, Accessed: Jan. 03, 2023. [Online]. Available: https://www.academia.edu/5533734/Gurage

[16] T. L. Feleke, “Ethiosemitic languages: Classifications and classification determinants,” Ampersand, vol. 8, p. 100074, Jan. 2021, doi: 10.1016/j.amper.2021.100074.

[17] R. Meyer, “Non-verbal predication in East Gurage and Gunnän Gurage languages,” Crass Joachim Ronny Meyer Eds 2007 Deictics Copula Focus Ethiop. Converg. Area Afr. Forschungen 15, Jan. 2007, Accessed: Jan. 03, 2023. [Online]. Available: https://www.academia.edu/467045/Non_verbal_predication_in_East_Gurage_and_Gunn%C3%A4n_Gurage_languages

[18] G. Gragg, “The Gunnän-Gurage Languages. Robert Hetzron,” J. East. Stud., vol. 41, no. 3, pp. 231–234, Jul. 1982, doi: 10.1086/372958.

[19] B. Araya Keleta,“Gura Documentation and Description of Morphology and Syntax,” Thesis, AAU, 2020.Accessed: Dec. 23, 2022. [Online]. Available: http://etd.aau.edu.et/handle/123456789/24901

[20] Daniel, “(12) A Review of Shifts in Gurage Orthography | Daniel Yacob and Fekede Menuta - Academia.edu.” https://www.academia.edu/87133918/A_Review_of_Shifts_in_Gurage_Orthography (accessed Nov. 28, 2022).

[21] R. Meyer, “Language standardization efforts in Gurage,” in 20th International Conference of Ethiopian Studies, Mekelle, Ethiopia, Sep. 2018. Accessed: Sep. 16, 2022. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02047237

[22] F. Menuta, “Verbal Extension and Valence in Gumer Variety of Gurage,” Stud. Afr. Linguist., vol. 51, Aug. 2022, doi: 10.32473/sal.v51i1.121891.

[23] W. LESLAU,“TOWARD A CLASSIFICATION OF THE GURAGE DIALECTS,” J. Semit. Stud., vol. 14, no. 1, pp. 96–109, Mar. 1969, doi: 10.1093/jss/14.1.96.

[24] “Gurage Adds | PDF | Typefaces | Orthography,” Scribd. https://www.scribd.com/document/504623447/21037-gurage-adds (accessed Mar. 22, 2022).

[25] A. K. Simpson,“The Origin and Development of Nonconcatenative Morphology,” UC Berkeley, 2009.Accessed: Nov. 04, 2022. [Online]. Available: https://escholarship.org/uc/item/7t18f7jw

[26] W. Gewe and M. Gasser, “Learning Morphological Rules for Amharic Verbs Using Inductive Logic Programming,” May 2012. doi: 10.13140/2.1.5171.2001.

[27] E. Assefa and E. Addis Ababa University,“Major Morphophonemic Operations in Ezha (Ethio-Semitic),” Macrolinguistics, vol. 7, no. 10, p. 29, 2019.

[28] M. Miestamo, Standard negation: the negation of declarative verbal main clauses in a typological perspective. Berlin: De Gruyter Mouton, 2005.

[29] C. M. Ford, “Notes on the Phonology and Grammar of Chaha-Gurage,” J. Ethiop.Stud., vol. 19, pp. 41–80, 1986.

[30] “17670-EN-morphology-of-the-english-noun.pdf.”

[31] B. W. Debela, “Morphology and Verb Construction Types of Kistaniniya,” Doctoral thesis, Norges teknisk-naturvitenskapelige universitet, Det humanistiske fakultet, Institutt for språk- og kommunikasjonsstudier, 2010. Accessed:Dec. 01, 2022.[Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/244002

[32] F. Menuta,“Morphology of Eža,”Thesis, Addis Ababa University, 2002. Accessed: Jan. 13, 2023.[Online]. Available: http://etd.aau.edu.et/handle/123456789/6370

[33] “IJCTT - Survey of Spell Checking Techniques for Malayalam: NLP.” https://ijcttjournal.org/archives/ijctt-v17p133 (accessed Jan. 16, 2023).

[34] P. Gupta, “A context sensitive real-time Spell Checker with language adaptability,” ArXiv191011242 Cs Stat, Oct. 2019, Accessed: Mar. 23, 2022. [Online]. Available: http://arxiv.org/abs/1910.11242

[35] H. L. Liang, “SPELL CHECKERS AND CORRECTORS: A UNIFIED TREATMENT,” p. 119.

[36] M. M. Al-Jefri and S. A. Mohammed,“Arabic spell checking technique,” US9037967B1, May 19, 2015 Accessed: Jan. 16, 2023 [Online]. Available: https://patents.google.com/patent/US9037967B1/en

[37] N. Hossain, S. Islam, and M. N. Huda, “Development of Bangla Spell and Grammar Checkers: Resource Creation and Evaluation,” IEEE Access, vol. 9, pp. 141079–141097, 2021, doi: 10.1109/ACCESS.2021.3119627.

[38] D. Hládek, J. Staš, and M. Pleva, “Survey of Automatic Spelling Correction,” Electronics, vol. 9, no. 10, p. 1670, Oct. 2020, doi: 10.3390/electronics9101670.

[39] F. Tafesse, “Morphology Based Spell Checker for Kafi Noonoo Language,” Thesis, Addis Ababa University, 2018.Accessed: Mar. 28, 2022. [Online]. Available: http://etd.aau.edu.et/handle/123456789/19584

[40] M. Shimelis, “Amharic Spelling Error Detection and Correction System:,” p. 130.

[41] A. Brhanu,“Tigrigna language spellchecker and correction system for mobile phone devices,” Int. J. Electr. Comput. Eng., vol. 11, Jun. 2021, doi: 10.11591/ijece.v11i3.pp2307-2314.

[42] R. Altarawneh, “Spelling Detection Errors Techniques in NLP: A Survey,” Int. J. Comput. Appl., vol. 172, pp. 1–5, Aug. 2017, doi: 10.5120/ijca2017915176.

[43] S.-S. Kang,“Word Similarity Calculation by Using the Edit Distance Metrics with Consonant Normalization,” J. Inf. Process. Syst., vol. 11, no. 4, pp. 573–582, Dec. 2015, doi: 10.3745/JIPS.04.0018.

[44] R. Kumar, M. Bala, and K. Sourabh,“A study of spell checking techniques for Indian Languages,” p. 9, 2018.

[45] D. Yacob, “Application of the Double Metaphone Algorithm to Amharic Orthography.” arXiv, Aug. 22, 2004. doi: 10.48550/arXiv.cs/0408052.

[46] Andargachew Mekonnen, Binyam Ephrem, and A. Nürnberger, “Portable Spelling Corrector for a Less-Resourced Language: Amharic,” in Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, May 2018.

[47] G. Assefa, “Automatic Amharic spelling error detection and correction using hybrid approach,” undefined, 2018, Accessed: Apr. 01, 2022. [Online]. Available: https://www.semanticscholar.org/paper/Automatic-Amharic-spelling-error-detection-and-Assefa/ccaf7f78c884c394b8f84f58b4f0433d99118cae

[48] B. Hamza, A. Yousfi, H. Gueddah, and M. Belkasmi,“For an Independent Spell-Checking System from the Arabic Language Vocabulary,” Int. J. Adv. Comput. Sci. Appl., vol. Vol. 5, p. 113, Jan. 2014, doi: 10.14569/IJACSA.2014.050115.

[49] K. Shaalan, M. Attia, P. Pecina, Y. Samih, and J. van Genabith, “Arabic Word Generation and Modelling for Spell Checking,” in Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC’12), Istanbul, Turkey, May 2012, pp. 719–725.Accessed: Jun. 16, 2022. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2012/pdf/603_Paper.pdf

[50] G. O. Ganfure and D. D. Midekso, “Design And Implementation Of Morphology Based Spell Checker,” vol. 3, no. 12, p. 8, 2014.

[51] “GNU Aspell.” http://gnu.ist.utl.pt/software/aspell/ (accessed Jun. 13, 2022).

[52] Vibhakti V. Bhaire, Ashiki A. Jadhav, Pradnya A. Pashte, and Mr. Magdum P.G, “SPELL CHECKER,” 2017. https://www.ijsrp.org/research-paper-0415.php?rp=P403950 (accessed Jun. 13, 2022).

[53] D. Seth and M. M. Kokar, “SSCS: A Smart Spell Checker System Implementation Using Adaptive Software Architecture,” in Adaptive Software Architecture. IWSAS, 2005, pp. 187–197.

[54] Banik, Debajyoty and Roy Choudhury, Ritabrata and Ekbal, Asif, “Spelling Checking Mechanism Based on Layered Language Model Complied with Google Web.” Available at SSRN: https://ssrn.com/abstract=4517425 or http://dx.doi.org/10.2139/ssrn.4517425

[55] Singh, S., Singh, S. HINDIA: “a deep-learning-based model for spell-checking of Hindi language.” Neural Comput & Applic 33, 3825–3840 (2021). https://doi.org/10.1007/s00521-020-05207-9

APPENDICES

Appendix 1 Modren Guragina Fider (Alphabet) basic letters

No.	Consonant (Transliteration)	-ə	-u	-i	-a	-e	-(i)	-o
1.	h	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
2.	hʲ	ሗ	ሗ	ሗ	ሗ	ሗ	ሗ	ሗ
3.	hʷ	ሐ		ሒ	ሓ	ሔ	ሕ	
4.	l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
5.	m	መ	መ	ሚ	ማ	ሜ	ም	ሞ
6.	mʷ	መ		ሚ	ማ	ሜ	ም	
7.	r	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
8.	s	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
9.	ʃ	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
10.	kʼ	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
11.	kʲ	ቘ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
12.	kʷ	ቈ		ቁ	ቋ	ቈ	ቅ	
13.	b	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
14.	bʷ	በ		ቢ	ባ	ቤ	ብ	
15.	t	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
16.	ʧ	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
17.	n	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
18.	ɲ	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
19.	ʾ	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
20.	k	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
21.	kʲ	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
22.	kʷ	ከ		ኪ	ኻ	ኬ	ክ	
23.	w	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
24.	ʔ	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
25.	z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
26.	ʒ	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
27.	j	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
28.	d	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
29.	dʒ	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
30.	g	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
31.	gʲ	ጘ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
32.	gʷ	ገ		ጊ	ጓ	ጌ	ግ	
33.	tʼ	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
34.	ʧʼ	ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
35.	sʼ	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
36.	f	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
37.	fʷ	ፈ		ፊ	ፏ	ፌ	ፍ	
38.	p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ

39.	*pʷ*	ፕ		ፒ	ፓ	ፔ	ፖ	

Appendix 2 Numeral (Cardinal) Pronunciation in Guragina

Numeral		**Translation**	**Tens**		**Translation**
at	**ኣት**	One	*asiɾəm*	**ኣስረም ኣት**	eleven
hʷɛt	**ሑት**	Two	*asiɾəm*	**ኣስረም ሑት**	twelve
sost	**ሶስት**	Three	*asiɾəm*	**ኣስረም ዝጠ**	nineteen
arbət	**ኣርበት**	Four	*hʷuja*	**ሕሥያ**	twenty
amɨst	**ኣምስት**	Five	*sasa*	**ሳሳ**	thirty
sidɨst	**ስድስት**	Six	*arba*	**ኣርባ**	Forty
səbat	**ሰባት**	Seven	*amsa*	**ኣምሳ**	Fifty
sɨmʷɨu	**ስምሞት**	Eight	*silsa*	**ስልሳ**	Sixty
ʒətʾə	**ዝጠ**	Nine	*səba*	**ሰባ**	seventy
asɨr	**ኣስር**	Ten	*səmanəj*	**ሰማንያ**	eighty
			ʒətʾəna	**ዘጠና**	ninety
			bəqr	**በቅር**	hundred
			Hum	**ሕም**	thousand

Appendix 3 Derivation of Ordinal numbers form Cardinal

Cardinal Number			Ordinal Number		
Transl	Guragina	Translatio	*Transliteration*	Guragin	Translati
at	**ኣት**	one	*atənə*	**ኣት-ኣነ**	first
hʷɛt	**ሑት**	two	*hʷɛtənə*	**ሑተ-ኣነ**	second
səbat	**ስባት**	seven	*səbatənə*	**ስባት-ኣነ**	seventh

Declaration

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

Declared by:

Name: Mengistu Gebre Bokane

Signature: _____

Date: _____

Confirmed by advisor:

Name: Dr. Yaregal Assabie

Signature: _____

Date: _____