CONTEXT-BASED SPELL CHECKER FOR SIDAAMU AFOO

USING HYBRID APPROACH

MASTER'S PROGRAM(MSc)

FRANCIS BEZABIH BEYENE

MAJOR ADVISOR: ANDARGACHEW M. (PhD)

CO-ADVISOR: EPHREM A. (MSc)

HAWASSA UNIVERSITY

INSTITUTE OF TECHNOLOGY

FACULTY OF INFORMATICS, DEPARTMENT OF COMPUTER SCIENCE

HAWASSA, ETHIOPIA

November, 2024

CONTEXT-BASED SPELL CHECKER FOR SIDAAMU AFOO

USING: HYBRID APPROACH

FRANCIS BEZABIH BEYENE

MAJOR ADVISOR: ANDARGACHEW M. (PhD)

CO-ADVISOR: EPHREM A. (MSc)

THESIS SUBMITTED TO

HAWASSA UNIVERSITY DEPARTMENT OF COMPUTER SCIENCE

INSTITUTE OF TECHNOLOGY, SCHOOL OF GRADUATE STUDIES,

HAWASSA, ETHIOPIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTERS OF SCIENCE IN COMPUTER SCIENCE

November, 2024

DECLARATION

I hereby declare that this thesis, titled **Context-Based Spell Checker for Sidaamu Afoo Using Hybrid Approach**, represents my work and effort. It has been developed independently under the guidance and supervision of Andargachew Mekonnen (PhD) and Ephrem Alamerew (MSc). All the information and ideas obtained from external sources have been appropriately cited and referenced.

Francis Bezabih

November, 2024

SCHOOL OF GRADUATE STUDIES

HAWASSA UNIVERSITY

ADVISORS' APPROVAL SHEET

This is to certify that the thesis entitled "**Context-Based Spell Checker for Sidaamu Afoo Using Hybrid Approach**" was submitted in partial fulfillment of the requirements for the degree of Master's with specialization in Computer Science for the Graduate Program of the Department of Computer Science, and has been carried out by **Francis Bezabih**, (ID. No: GPCoScW/0003/14), is conducted under my supervision.

Therefore, I recommend that the student has fulfilled the requirements and hence hereby can submit the thesis to the Department.

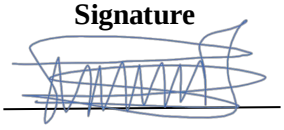
Andargachew Mekonnen (PhD)		
Name of Major Advisor	**Signature**	**Date**
Ephrem Alamerew (MSc)		
Name of Co-Advisor	**Signature**	**Date**

SCHOOL OF GRADUATE STUDIES

HAWASSA UNIVERSITY EXAMINERS' APPROVAL

SHEET-1

We, the undersigned, members of the Board of Examiners of the final open defense by **Francis Bezabih** have read and evaluated his thesis entitled "**CONTEXT-BASED SPELL CHECKER FOR SIDAAMU AFOO USING HYBRID APPROACH**", and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirements for the degree.

Andargachew Mekonnen (PhD)		
Name of Major Advisor	**Signature**	**Date**
Ephrem Alamerew (MSc)		
Name of Co-Advisor	**Signature**	**Date**
Degif Teka (PhD)		
Name of Internal Examiner-I	**Signature**	**Date**
Name of Internal Examiner-II	**Signature**	**Date**
Michael Melese (PhD)		29/11/2024
Name of External Examiner-I	**Signature**	**Date**
SGS Approval	**Signature**	**Date**

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

Stamp of SGC

DEDICATION

To my beloved parents, whose unconditional love, sacrifices, and belief in my potential have been the driving force behind my pursuit of knowledge. This thesis is dedicated to you, as a token of my deepest gratitude and appreciation.

ACKNOWLEDGMENTS

First of all, I would like to thank almighty GOD for keeping me until this date and I would like to express my deepest gratitude to my advisors, Andargachew Mekonnen (PhD) and Ephrem Alamerew (MSc), for their invaluable guidance, insightful feedback, and unwavering support throughout the entire process of this thesis. Their expertise, encouragement, and dedication have been instrumental in shaping this research work.

I extend my heartfelt appreciation to the faculty members of Informatics for their valuable insights, stimulating discussions, and continuous encouragement. Their expertise and commitment to academic excellence have played a significant role in broadening my understanding of the subject matter.

I am immensely grateful to the participants who generously contributed their time, knowledge, and experiences to this study. Their willingness to share their insights and perspectives has enriched the findings of this research, and I am deeply thankful for their contributions.

I would like to acknowledge the support and encouragement from my family and friends throughout this journey. Their unwavering belief in me, patience, and understanding have been a constant source of motivation and inspiration. I am grateful for their presence in my life.

Finally, I extend my appreciation to all those who have played a part, however small, in the completion of this thesis. Your encouragement, advice, and assistance have made a difference, and I am truly grateful for your contribution.

TABLE OF CONTENTS PAGE

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AI: Artificial Intelligence

CBSCSA: Context-Based Spellchecker for Sidaamu Afoo

CV: Consonant Vowel

DL: Deep Learning

ER: Error Recall

EP: Error Precision

FDRE: Federal Democratic Republic of Ethiopia

FN: False Negative

FP: False Positive

GPU: Graphical Processing Unit

HMM: Hidden Markov Model

LR: Lexical Recall

LP: Lexical Precision

LSTM: Long Short-Term Memory

MT: Machine Translation

NLP: Natural Language Processing

OCR: Optical Character Recognition

POS: Part of Speech Tagging

SCMIL: Sequence-to-sequence text Correction Model for Indic Languages

SMT: Statistical Machine Translation

SNRS: Sidama National Regional State

SOV: Subject Object Verb

SVO: Subject Verb Object

SR: Speech Recognition

TN: True Negative

TP: True Positive

ABSTRACT

Spellcheck involves identifying and suggesting corrections for incorrectly spelled words within the text. Its integration spans various applications such as digitally correcting handwritten text, aiding user word corrections during retrieval, and more. This thesis outlines the creation, implementation, and assessment of a model intended to rectify both non-word and real-word errors. The central objective of this research is to devise a context-based spellchecker for Sidaamu afoo. This system relies on the language's error patterns, deduced from word sequences within input sentences. The chosen technique for this spellchecking entails an unsupervised statistical method, which is particularly beneficial for languages like Sidaamu afoo by enabling analysis without the need for extensive tagged datasets. The process of rectifying spelling unfolds through distinct phases: identifying errors, proposing potential corrections, and arranging these suggestions by priority. Error identification hinges on a combination of dictionary lookup and bigram analysis. Data for the dictionary and Bigram model, essential for error detection and correction, were collected from diverse sources by the researcher. Addressing non-word errors involves computing the similarity between the misspelled word and tokens in the dictionary, measured using the Levenshtein distance, resulting in ranking and correction suggestions. In cases of real-word errors, bigram frequency aids in error detection, while bigram probability informs the correction process for misspelled words. The experimental phase encompassed the utilization of 52,093 tokens and 5,788 tokens for model learning and testing, respectively. The outcome revealed a spellchecker recall score of 92.4% and an accuracy rate of 92.5% for both non-word and real-word errors. These findings, aligned with the gated result accuracy of 92.5%, underscore the system's capability to rectify Sidaamu afoo misspellings. Future enhancements could explore advanced neural architectures to improve model quality further.

Key words: context-based spellchecker, Levenshtein distance, bigram model, real-word errors, non-word errors, N-gram methods, dictionary lookup.

CHAPTER ONE
INTRODUCTION

1.1 BACKGROUND

Natural Language Processing (NLP) stands as a domain within computer science, specifically in computational linguistics. Its purpose revolves around facilitating interaction between humans and computers through the application of formal rules to various components of human language. This interaction involves enabling a computer system to comprehend and generate both spoken and written language, encompassing diverse languages like English, Amharic, Afan Oromoo, Tigrigna, Spanish, and Chinese. (Schutze, 2008) emphasize NLP's potential to construct software that can analyze and interpret natural language, while simultaneously being adept at crafting responses in the same manner. Furthermore, NLP demonstrates its proficiency in comprehending and handling human language intricacies, ranging from precise real-time language translation to the extraction and summarization of information from an array of data sources. The effectiveness of NLP in human communication is underscored by its focus on devising and enacting tools, methodologies, and frameworks rooted in natural language intricacies (Zue,, et al., 2000).

NLP's origins trace back to the late 1940s when its initial focus lay in machine translation endeavors. By 1958, NLP had become associated with information retrieval through the Washington International Conference on Scientific Information (Hindle,, Barr,, Su,, Gabel,, & Devanbu,, 2012). During this era, the foundational concepts for creating tools that could identify and rectify textual errors began to take shape. NLP's allure remains potent to this day due to its pivotal role in human-computer interaction. It stands as the crossroads where linguistics converges with artificial intelligence, enabling the programming of machines to manipulate human language (Nadkarni, 2022).

Natural Language Processing (NLP) boasts a plethora of practical applications, spanning from automatic summarization and Machine Translation (MT) to Part-Of-Speech (POS) tagging, Speech Recognition (SR), and Optical Character Recognition (OCR). Furthermore, among these applications, we find the early-mid 20th-century emergence of spell-checkers, an innovation credited to Lee Earnest at Stanford University in the USA. It's worth noting that Ralph Gorin, a student of Lee Earnest, holds the distinction of developing the inaugural spell-checker application.

His pioneering work hinged on a 10,000-word English dictionary, culminating in the design of this spell-checking application (Al-bakry, 2014).

Commonly, the process of rectifying misspelled words involves several key stages (Xie,, et al., 2015). The initial phase revolves around spotting words with incorrect spellings within a given document or those provided by the user. The prevalent approach for identifying these misspelled words revolves around employing either a dictionary lookup or cross-referencing against a roster of accurate words.

The technique of dictionary lookup proves invaluable for pinpointing non-word errors—those words that do not find a place within the dictionary's compilation. Given the extensive lexicon of natural languages, compiling an all-encompassing dictionary presents a significant challenge. Consequently, the task of scrutinizing each word against the dictionary's contents results in substantial time consumption, thereby delaying the identification of inaccuracies (Achenkunju, 2014).

Another phase in the spellchecking process involves generating potential corrections for misspelled words. This step is crucial for addressing both nonword errors and real-word errors. To rectify nonword errors, a pool of candidate words is drawn from the dictionary. The selection of candidates is based on evaluating the likeness between the misspelled word and words in the dictionary.

In the case of real word errors, candidate words are generated using a Bigram model that relies on probabilistic information. These candidate words are chosen based on the context provided by the surrounding words and their probabilities within the language.

Subsequently, the generated candidates undergo a ranking process. For nonword errors, this ranking is determined by assessing the similarity scores between the candidates and the misspelled word. On the other hand, the ranking of real word error candidates is influenced by the probabilities derived from the Bigram model.

The culmination of this process results in a list of candidates for both types of errors. These candidates are then presented to the user, who can make an informed decision and select the most suitable correction based on the displayed information (Mishra,, Kaur,, & Technique,, 2013).

Contextual spell-checking for identifying misspelled words is an innovative approach as highlighted by (Dickinson, 2013). In contemporary times, numerous applications have been developed with the capability to contextually rectify spelling errors. For instance, Microsoft Word 2007 introduced a context-based spell-checker, operating within the Microsoft Word interface, effectively addressing both non-word and real-word errors (Flor, & Futagi,, 2012). Additionally, Google offers Google Suggester, which aids in resolving misspelled words by considering the contextual nuances of user queries (Inkpen, 2009).

Various algorithms are employed in the creation of context-based spell-checkers, which encompass the detection and correction of both non-word and real-word errors (Bassil, & Semaan,, 2012). The selection of the most appropriate algorithm hinges on several factors, including corpus size, performance considerations, and runtime efficiency. These factors collectively contribute to the successful implementation of context-based spell-checkers with the goal of effectively identify and rectify errors, as identified by (Daiber,, Jakob,, Hokamp,, & Mendes,, 2013).

Contextual error detection and correction significantly enhance the efficacy of spell checkers. This approach not only addresses errors stemming from non-existent words but also those arising from correctly spelled words that are contextually inappropriate (Cucerzan, & Brill,, 2002). Accordingly, this investigation employs context-based principles to rectify misspelled words resulting from both genuine lexical errors and the absence of entries in the dictionary. In essence, the primary objective of this research is to devise an unsupervised, statistically driven context-based spell checker proficient in rectifying misspelled words.

To capture contextual cues for individual words, data was gathered from a corpus, and diverse algorithms were employed to imbue the model with this knowledge. Among these methods, the most renowned technique for identifying misspellings involves using bigrams along with a dictionary lookup mechanism to pinpoint inaccuracies within user input words (Ahmed,, Ernesto,, De,, & Andreas,, 2009).

On another note, the Levenshtein edit distance has been employed for rectifying misspelled words. This method calculates the similarity between the misspelled words and all other words, aiding in the identification of the correct word. This involves generating a candidate list for correcting non-word errors, as demonstrated by (Nejja, & Yousfi,, 2023). Furthermore, N-gram models leverage

probabilistic and frequency-related information. These models consider various N values such as unigrams, bigrams, trigrams, etc. The probability associated with these n-grams is then used to propose potential corrections for real-world errors, as highlighted by (Seo, 2012).

The approach involves a context-sensitive spell-checking technique capable of identifying and rectifying various types of errors in human-generated text, including both real-word and non-word errors. Following the suggestion by (Bassil, & Semaan,, 2012), the initial stage of this context-sensitive spell-checker employs a dictionary-based error detection and correction system, which focuses on rectifying non-word errors. For the context-aware spell-checker in Sidaamu afoo, a fusion of dictionary lookup and bigram analysis is utilized to effectively address both non-word and real-word errors.

In cases where the available corpus size is limited, such as in the absence of a comprehensive Sidaamu afoo corpus, the utilization of bigram analysis proves advantageous. This preference arises from previous findings indicating that for smaller corpora, bigram models outperform trigram models (Ahmed,, Ernesto,, De,, & Andreas,, 2009). Consequently, for the detection and correction of real-word errors in Sidaamu afoo, the chosen approach revolves around assessing the significance of bigram probabilities and frequencies within the collected sample corpus.

Furthermore, the method employs a dictionary lookup strategy to identify and rectify non-word errors, relying on predefined lists of acceptable corrections for each instance.

Spell checking and correction techniques, along with various NLP tools, have achieved a higher level of acceptance, effectiveness, and accuracy in the English language, compared to the progress made for the Sidaamu afoo language. In the case of Sidaamu afoo, there is ongoing and completed research aimed at bridging the gap and addressing the challenges across different NLP domains.

Various types of spelling checkers, including probabilistic and rule-based approaches, have been created for widely used languages like English, French, Spanish, Chinese, and Arabic (Cucerzan, & Brill,, 2002). This study aims to develop an interactive context-sensitive spell-checker model for the Sidaamu afoo language. The model utilizes bigram probability data to effectively address real-word errors and employs Levenshtein edit distance for non-word error solutions.

Sidaamu afoo, a member of the Cushitic language family, serves as the principal language in the Sidama Regional State. It is utilized as the primary medium of instruction in both primary and secondary schools within the region. Sidaamu afoo is also extensively employed in official documents across governmental and private sectors in Sidama. The language holds prominence in private-sector communication as well. While individuals proficient in Sidaamu afoo can generate written content using word processing applications, mobile platforms, search engines, and various other tools, there remains uncertainty regarding the accuracy of spelling within these texts. To address this issue, the present research makes a significant contribution by identifying and rectifying spelling errors in written Sidaamu afoo text. The ultimate goal is to ensure precise information transfer by providing appropriate corrections for misspelled words.

Nowadays, the sharing of knowledge heavily relies on the writing system. Consequently, the writing system bears the crucial responsibility of effectively communicating information between writers and readers without any misspellings. When a language demonstrates insufficient spelling, it can result in information gaps, the erosion of crucial knowledge, and misunderstandings. This situation poses a notable risk to the advancement and growth of the language.

1.2 MOTIVATION OF THE STUDY

Our initial motivation for undertaking research on spell-checkers for Sidaamu afoo arises from the existing and completed research efforts aimed at addressing challenges and bridging gaps across various NLP domains. Given the widespread use of Sidaamu afoo in schools, offices, and various media, a substantial volume of electronic data is processed daily. Within the domain of electronic data processing, spelling errors possess the capacity to change the intended message of the writer. Spell-checkers are instrumental in detecting and rectifying these errors during the processing of documents. The absence of a dedicated spell-checker for Sidaamu afoo has resulted in texts retaining spelling errors, which can hinder the effective communication of the writer's message.

Another key motivation is the preservation and promotion of linguistic heritage. Sidaamu afoo is a language spoken by a significant community, and as digital transformation accelerates, the development of tools like spell-checkers becomes essential for ensuring the language's relevance in the digital era. Without proper tools to assist in maintaining orthographic accuracy, the language risks fragmentation or a gradual shift towards inconsistency in written forms. This study not only helps in maintaining linguistic standards but also supports the growth of Sidaamu afoo in

education, government institutions, and online platforms where language preservation is critical for cultural identity. Additionally, such tools can assist in creating more accessible learning materials and resources for future generations, enhancing both literacy and communication.

Furthermore, the researcher, as a community member of this language, feels compelled to contribute to its enrichment within his discipline. Another significant motivation for this study is that a spell-checker serves as a foundational tool for anyone interested in developing NLP applications for Sidaamu afoo. NLP applications such as grammar checkers, information retrieval, parts of speech tagging, text-to-speech synthesis, dialogue systems, question-answering, machine translation, etc., can integrate spell-checkers to enhance their overall performance.

1.3 STATEMENT OF THE PROBLEM

The writing system plays a crucial role in facilitating the exchange of information between authors and readers. Incorrect spelling can create barriers to communication between these two groups (Treiman, & Kessler,, 2005). This can lead to a lack of clarity in transferring an author's knowledge to readers. Moreover, poor spelling can also impede the evolution of languages. This issue is evident in the Sidaamu afoo writing system. Due to misspelled words, the Sidama Cultural and Tourism Bureau has taken corrective measures. These measures apply to government institutions, non-governmental organizations, and businesses, where misspelled words on banners displayed publicly have been addressed.

Sidaamu afoo lacks extensive tagged datasets, which are typically necessary for training advanced spellcheckers. Without such resources, it's challenging to develop a model that can understand the language's unique structure, vocabulary, and grammar.

Effective spellchecking in Sidaamu afoo requires understanding context to distinguish correct words from real-word errors.

Detecting and correcting non word and real word errors requires a more sophisticated approach that can suggest contextually appropriate corrections for misspellings.

Until now, there has been research (Wagara, 2022) aimed at creating a context-based spelling checker to address misspellings within the Sidaamu afoo writing system. Thus, the primary

objective of this study is to formulate a prototype context-based spellchecking solution intended to rectify the prevalent issue of misspellings within the Sidaamu afoo writing system.

1.4 RESEARCH QUESTIONS

This study addresses the following research inquiries to develop a solution for the misspelling challenges, aiming to enhance the effectiveness of a context-based spell-checker for Sidaamu afoo: -

- Which algorithm is best suited for designing a context-based spell checker?
- To what extent does the context-based spell checker improve the correction of misspelled Sidaamu afoo words?
- Is the approach efficient and practical in accurately detecting and correcting errors?

1.5 OBJECTIVE OF THE STUDY

1.5.1 GENERAL OBJECTIVE

The general objective is to develop a context-based Sidaamu afoo spell checker using a hybrid approach (rule-based and statistical).

1.5.2 SPECIFIC OBJECTIVES

In pursuit of the overall goal, this study aims to address the following specific objectives:

- To determine appropriate approaches for implementing a spell checker.
- To compile and refine the Sidaamu afoo corpus, essential for training and validating the proposed model.
- To collect and prepare Sidaamu Afoo corpora.
- To formulate a context-based spell-checker model tailored for Sidaamu afoo text.
- To design an architecture of spell checker model.
- To assess the effectiveness of the prototype spell checker's performance.

1.6 SCOPE AND LIMITATION OF THE STUDY

This study was conducted to create a context-based spell-checking model for the Sidaamu afoo language. The approach relies on the statistical frequency of words, utilizing bigram probabilities of word occurrences in relation to each other, along with a dictionary lookup method. The primary focus of the model is to address spelling errors arising from typographical mistakes, encompassing

scenarios like character deletions, insertions, substitutions, and transpositions, as well as errors involving real words. Essentially, two categories of spelling errors exist in the spell-checking process (Ahmed,, Ernesto,, De,, & Andreas,, 2009). The first category involves non-word errors, which occur when a word is not found within the dictionary listing.

Conversely, another type of error is a real-word error, which occurs when a word is correctly spelled (matching the bigram dictionary), but it appears in the wrong position within the sentence. In this study, both non-word and real-word errors are addressed. However, it's important to note that not all Sidaamu afoo words were included in the corpus preparation. Consequently, the model could not suggest candidates for every misspelled Sidaamu afoo word. Additionally, the model did not automatically rectify misspelled words even when a candidate list was provided; it required user interaction to input candidates for the misspelled words. Furthermore, the model did not address compounds and abbreviations, treating compound words as separate entities.

1.7 METHODOLOGY

1.7.1 RESEARCH DESIGN

This study employs an experimental research approach, as outlined by (Gersten,, Fuchs,, Coyne,, & Greenwood,, 2005). This methodology relies on empirical evidence and involves both direct and indirect engagement with the subject matter to acquire knowledge. Accordingly, the researcher utilized the experimental method to construct a model, develop a prototype, and assess its performance within this study. The typical procedures for prototype system development were followed, encompassing stages such as literature review, collection and preparation of a corpus, system design, and experimenting, culminating in the evaluation of the system's performance.

1.7.2 LITERATURE REVIEW

Prior research was examined to gain a more profound comprehension of the spelling structure of the Sidaamu afoo language. This encompassed an exploration of how the corpus and user words are prepared, as well as the development of spelling check mechanisms and underlying principles pertinent to this endeavor. An evaluation of various methodologies employed in spell-checking systems was also conducted with the aim of pinpoint the most effective approach.

1.7.3 DATA PROCESSING

Gathering a substantial corpus of Sidaamu afoo words, comprising 449,788 entries, is essential for creating an effective spell-checker and correction model. A comprehensive word collection and an extensive dictionary within the corpus database are crucial elements that contribute to the enhancement and successful development of such a spell-checker and correction model.

Constructing a textual corpus plays a pivotal role in identifying and rectifying misspelled words within the framework of spell checking (Han, & Baldwin,, 2011). In this study, a manual creation of a Sidaamu afoo text corpus was undertaken for utilization in the Sidaamu afoo Spell Checker system. This choice was necessitated by the absence of a standardized corpus for Sidaamu afoo. Consequently, we sourced a free-text corpus from diverse outlets including the Sidama Media Network (SMN), the Sidaama Culture and Tourism Office, Sidaama national regional state constitution, the Sidama Region Education Office, the Ethiopian Press Agency Sidaamu afoo Facebook page, and the web site of Bible Society. This amassed corpus encompasses a wide array of subjects such as culture, society, politics, sports, and economics. This strategy was adopted to avert data scarcity and to formulate an extensive dictionary, while also serving as a testing ground for the model. The dictionary derived from the compiled corpus underwent preprocessing through techniques like tokenization and normalization. The accumulated words are stored in text format, facilitating a comparative analysis with user input words.

1.7.4 DEVELOPMENT OF SYSTEM PROCEDURE

The approach taken in this study involves using unsupervised statistical methods to address footpath-related issues. This approach is particularly advantageous for languages like Sidaamu afoo that have limited linguistic resources. The study utilizes a gathered collection of texts, or corpus, to identify and rectify both real-word and non-word errors. To achieve this, N-gram statistical techniques are employed, aiding in the identification and correction of spelling errors by considering the words surrounding the target word. In the N-gram model, the likelihood of a word appearing in a sentence is estimated based on its occurrence in proximity to neighboring words. Typically, sequences of two or three words, known as bigrams or trigrams, are analyzed, and their probabilities are derived from a sizable dataset. These probabilities are then combined to calculate the prior probability of various possible interpretations of spoken phrases, enhancing the selection of the most plausible interpretation, particularly when dealing with real-word errors.

Moreover, the model employs a technique of utilizing dictionary lookup to identify non-word errors within the provided text. These errors are then marked by highlighting the incorrect words and presenting potential replacement suggestions through the calculation of distances between the misspelled words and words found in the dictionary. Furthermore, the model utilizes N-gram analysis to identify and rectify real-word errors in the given text, marking these errors by applying highlights.

The algorithm and prototype for the Sidaamu afoo context-based spell checker were developed using the latest version of the Python 3.10.11version programming environment using the Kivy open-source library. Python was chosen as the programming language for this project due to its versatility, platform independence, and user-friendly interface.

1.7.5 EVALUATION PROCEDURE

The created prototype underwent an evaluation process to gauge its model accuracy. An evaluation metric was employed to assess the accuracy of the context-based spell checker. Both recall and precision evaluation metrics were utilized to gauge the practical effectiveness of the spell-checker in identifying both non-word and real-word spelling errors, as noted by (Momtazi, 2012). The Sidaamu afoo Spell-checker also adopted these recall and precision metrics to effectively evaluate the model's performance.

1.8 SIGNIFICANCE OF THE STUDY

Writing with correct spelling is a significant activity that everyone engages in daily, and it has become an integral part of contemporary life. It serves as a reliable method for transmitting information from one source to another. Errors in spelling can impede the smooth exchange of information between authors and readers. In the present era, the importance of spelling checks cannot be overstated, as they are now an essential component of various processes, including text editing tools, word prediction, machine translation, information retrieval, word processing, search engines, and mobile applications.

Consequently, developing a spellchecker for Sidaamu afoo holds immense significance in nurturing the language. The outcomes of this study offer experimental evidence for identifying and rectifying misspelled words entered by users. The model presented not only detects and corrects these errors but also enhances effective communication between authors and readers.

Consequently, the language's users reap the initial benefits by reducing information gaps and facilitating seamless knowledge exchange. Furthermore, such research related to natural language processing contributes to the advancement of the Sidaamu afoo language.

1.9 THESIS ORGANIZATION

This thesis comprises six chapters. The first chapter provides a succinct introduction to the study, including the background of the problem area, both the general and specific objectives, the research methodology, scope and limitations, the programming tools utilized for prototype development, the evaluation procedure, and the significance of the study. Chapter two offers an overview of pertinent research, delving into concepts such as spell checking and n-gram techniques. It thoroughly explores the motivating concepts behind this research and reviews n-gram approaches for context-sensitive spell checking, as well as related works in the spell-checking domain. Moving to Chapter Three, the focus shifts to discussing characteristics of the Sidaamu afoo writing system that pertain to the research. Chapter four outlines the modeling and text preprocessing procedures for the spell-checker. The fifth chapter centers on the experimentation and evaluation of the model across various domain areas. Lastly, chapter six encapsulates the research's concluding remarks and provides recommendations for future studies.

CHAPTER TWO

LITERATURE REVIEW AND RELATED WORKS

2.8 OVERVIEW

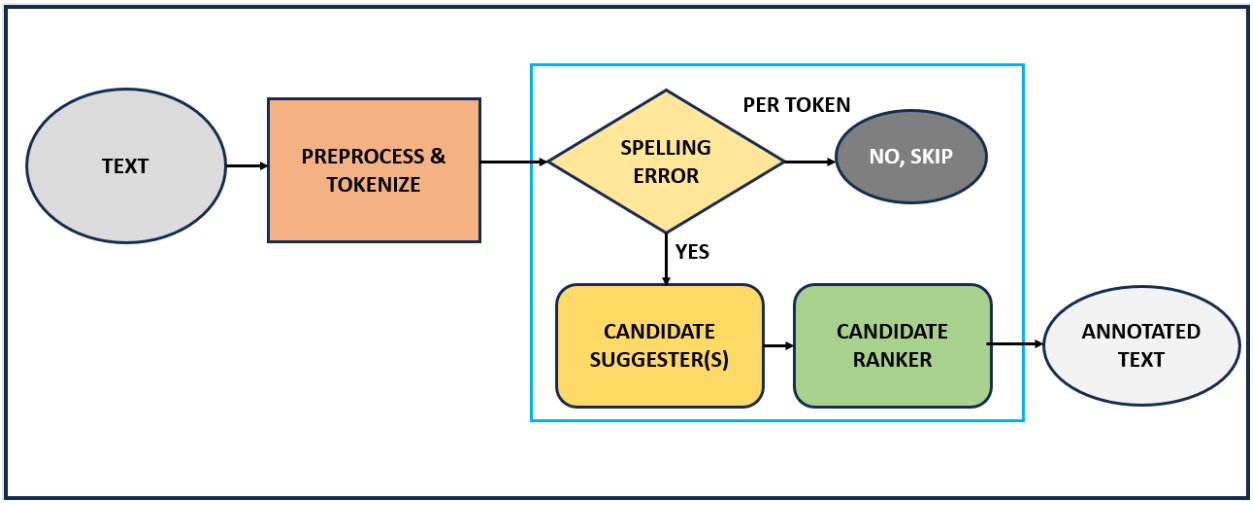
This section starts with 2.1 history of spell checkers, which traces the evolution from dictionary-based systems to advanced context-aware models.2.2 approach of spell checkers explores various techniques, including rule-based, statistical, and hybrid approaches.2.3 text errors focus on the types of spelling errors and their impact on text processing.2.4 spelling error detection techniques examines methods like n-gram analysis that improve error detection by considering word context. 2.5 spelling error correction techniques discusses context-based correction methods for more accurate results. 2.6 context-based spelling checking techniques highlights the significance of using context to enhance spelling correction accuracy. 2.7 performance of spell checker evaluation methods covers evaluation metrics such as accuracy and precision. Finally, 2.8 literature review and 2.9 related works provide a synthesis of previous research, positioning this study within the broader field and identifying gaps the current work addresses with its hybrid approach for Sidaamu afoo.

2.1 HISTORY OF SPELLCHECKER

A spellchecker is a software tool designed to identify and correct misspelled words in a given text (Ratnasari,, Kusumadewi,, & Rosita,, 2017). It can operate autonomously or be incorporated into various applications such as search engines, email clients, electronic dictionaries, or word processors.

The concept of spellchecking is not novel; its origins date back to the 20th century when the first spell checker was created for mainframe computers during the 1970s. This application initially supported six languages for IBM (International Business Machines Corporation). Its primary role was to highlight errors rather than suggest correct word replacements. Furthermore, personal computer spell checkers were introduced in 1981 by IBM. By the mid-1980s, popular word-processing software such as WordStar and WordPerfect had incorporated integrated spell-checking functionality (Sidoti,, Metsky,, & Somogyi,, 2008).

The primary functions of spell-checkers encompass preprocessing (tokenization), identifying and rectifying errors, as well as prioritizing suggested alternatives. The error detection phase primarily centers on recognizing and exhibiting incorrectly spelled words within the text. On the other hand, the error correction component offers potential replacement suggestions for these misspelled words, aiming to substitute erroneous terms with accurate ones, all while arranging these candidate alternatives based on their ranking (Tijhuis, 2014).



Source: Tijhuis, 2014

Figure 2.1: Spelling check procedure

2.2 APPROACH OF SPELLCHECKER

The preceding discussion revolved around spell-checking and its associated concepts. This portion provides concise explanations of diverse methodologies employed in spell-checking. Different algorithms and techniques were developed to address the issue of misspellings. Numerous strategies exist for designing spell check models, with the two most prevalent being the rule-based approach and the statistical approach (Ahmed,, Ernesto,, De,, & Andreas,, 2009).

2.2.1 RULE BASED APPROACH

Rule-based approaches stem from an expert-established theory of morphology. These methods allow for the integration of advanced linguistic theories. Such techniques employ a series of rules to manipulate misspelled words according to common error patterns, thereby transforming them

into valid words. Once all relevant rules have been applied, the resulting set of generated words, which are recognized as valid within the dictionary, are put forth as candidates to aid in rectifying misspelled words (Liang, 2008).

As outlined by (Martin, 2009), this algorithm operates through a two-stage architecture. The initial phase involves determining all potential tags for each word within the sentence, while the subsequent phase focuses on identifying the correct tag using manually formulated rules. This stands as the central element of the rule-based tagging system. Each word receives distinctive codes based on the categories of their contextual words.

These rules, known as tag-changing rules, furnish insights into the suitability of specific words based on context, effecting the transformation of erroneous words into accurate ones. These rules might pertain to words that follow particular patterns. Rules fall into either contextual or lexical categories. Contextual rules alter a word's tag based on the words surrounding it, whereas lexical rules capitalize on the morphological traits of the word itself (Xie,, et al., 2015).

2.2.2 STATISTICAL BASED APPROACH

The term "statistical approach" indicates its reliance on deriving word characteristics from training data, such as frequency, probability, or statistics (Koehn,, et al., 2007). Unlike rule-based methods, these approaches don't adhere strictly to explicit linguistic theories. Instead, they rely on test corpora for input. Within this category, algorithms are employed to grasp the language's word formation patterns from the given corpus. Subsequently, analysis is performed based on this acquired knowledge. Notably, these algorithms can be adjusted and refined during operation (Markou, 2022).

Statistical approaches are categorized into supervised and unsupervised methods based on their utilization of test corpora. Supervised approaches necessitate annotated text corpora and a provided training input. Conversely, unsupervised approaches utilize heuristics or probability data from test corpora to construct the morphological analysis system (Hilas, 2022).

2.2.2.1 SUPERVISED STATISTICAL APPROACH

In the supervised approach, semantically annotated corpora are employed to train machine learning (ML) algorithms. These algorithms learn to determine the appropriate word sense for given

contexts. Within these annotated corpora, words are manually tagged with semantic classes derived from a specific lexical semantic resource, often WordNet. These supervised methods are referred to as Corpus-based techniques when they learn from pre-annotated sense data. Consequently, they typically demand significant human effort to annotate the training data (Navigli, 2022).

In a supervised statistical approach, the model's training involves minimizing the discrepancy between its predicted outputs and the actual labeled outputs in the training dataset. This is usually done using optimization techniques that adjust the model's parameters to improve its predictive performance.

2.2.2.2 UNSUPERVISED STATISTICAL APPROACH

Unsupervised learning methods ascertain the categorization of individual items within a dataset without relying on labeled training instances. This entails grouping the contextual instances of an erroneous word into various clusters and distinguishing between them without explicitly assigning labels (Navigli, 2022). The differentiation between senses, in the context of misspellings, can be accomplished using unsupervised clustering techniques. As a result, the present study opted for an unsupervised approach in its implementation.

In an unsupervised statistical approach, the model's objective is to identify inherent structures or groupings within the data, often through techniques like clustering or dimensionality reduction. Clustering involves grouping similar data points based on certain criteria, while dimensionality reduction aims to capture the most important features of the data while reducing its complexity.

2.3 TEXT ERROR

A word can be erroneously mistaken in two primary ways. The first arises from either an incorrect spelling due to insufficient information about the word's correct form or un intentional manipulation of symbols within the word. This type of mistake is referred to as a "non-word error," where the word cannot be located in the language's lexicon. The second type involves using a correctly spelled word in an inappropriate position within a sentence or an unsuitable context. These errors are categorized as "real-word errors," where an incorrect word is accepted within the language's lexicon (Ahmed,, Ernesto,, De,, & Andreas,, 2009).

Detecting non-word errors is comparably easier, while real-word errors necessitate a deeper understanding of language syntax and semantics. Consequently, correction techniques are divided into isolated word error detection, which focuses on non-word errors, and context-based error correction, which addresses real-word errors (Hirst G. a., 2003).

(Christen, 2012) further classifies spelling errors into two distinct categories: Typographical errors and Cognitive errors. Typographical errors occur when the correct spelling of a word is known, but a mistake is made while typing it. These errors are primarily tied to keyboard usage and lack linguistic criteria. On the other hand, cognitive errors occur when the correct spellings of words are unfamiliar, leading to a lack of knowledge about the correct spelling in the target language. In these instances, the intended word's pronunciation is what guides the misspelling. The user simply lacks the correct spelling knowledge or holds misconceptions about it, resulting in an erroneous form. For example, instead of "Computer files," a user might write "Computer flies." Cognitive errors in spelling often stem from sound similarities between the erroneous word and the correct word. For instance, the confusion between "Creak" and " Creek" arises due to their similar sounds (Christen, 2012).

2.3.1 NON-WORD ERRORS

Non-word errors refer to spelling mistakes that are not found among the words listed in a dictionary (Tavast, 2012). These errors occur when a word is inaccurately typed due to various factors such as the presence of additional spaces, extra characters, misspelled words, or other similar issues. Detecting these errors is relatively straightforward since a simple comparison between the words in a text and the dictionary entries can weed out the incorrect words. According to (Liang, 2008), around 80% of misspelled words categorized as non-word errors occur due to single occurrences of letter insertions, deletions, substitutions, or transpositions.

Insertion: Adding an extra letter, e.g., ‘Nainkehu’ instead of ‘Ninkehu’ which means ours. An important special case is a repeated letter, e.g., ‘Muundee’ instead of ‘Mundee’ related with blood in English context.

Deletion: Missing a letter, e.g., ‘lme’ instead of ‘lame’. An important special case is missing a repeated letter, e.g., ‘ogeyye’ instead of ‘ogeeyye’.

Substitution: Substituting one letter for another, e.g., ‘kaase’ instead of ‘kaasi’. The

most common substitutions are incorrect vowels.

Transposition: Swapping consecutive letters, e.g., 'hnaqi' instead of 'hanqi'.

The utilization of a prepared dictionary is central to non-word spell checkers, serving both error detection and correction purposes (Ratnasari,, Kusumadewi,, & Rosita,, 2017). According to (Misha R., 2013), methods based on dictionary lookup rely on identifying non-word errors. This dictionary comprises words compiled from a gathered corpus and serves as a reference for accurate, error-free words. For rectifying misspelled words, many models employ the Levenshtein distance, which quantifies the similarity between the misspelled word and potential corrections (Ratnasari,, Kusumadewi,, & Rosita,, 2017).

2.3.2 REAL WORD ERROR

These errors arise when one word is mistakenly used in place of another word that is accepted by the dictionary. Such mistakes can occur when a correctly spelled word is used incorrectly within a sentence or in an unsuitable context. This type of error occurs when an incorrect word is considered valid in the language's lexicon, and it can stem from cognitive factors (Ahmed,, Ernesto,, De,, & Andreas,, 2009).

Correcting real-word errors presents a significant challenge for spell-checking systems. Nevertheless, various scholars have endeavored to address the issue of real-word errors. (Brill, 2000) proposed employing a noisy channel to predict the accurate correction for a real-word error. Their approach involved compiling a corpus of approximately 100 million words and utilizing n-gram statistics to rectify real-word errors. Another approach was suggested by (Sundby, 2009), who utilized an n-gram model to anticipate the appropriate correction for such errors. The central concept was to generate potential spellings for each misspelled word by applying basic edit operations like insertion, deletion, and substitution. Subsequently, n-gram statistics derived from a corpus were used to compute the probability of the word. Notably, the spell checker in the recently released Microsoft Word 2007 has demonstrated the capability to identify and rectify certain real-word errors (Jurafsky, 2009).

In terms of correction techniques, context-sensitive error rectification pertains to real-word errors and is frequently addressed through n-gram analysis. Generally, the process of correcting real-

word errors is context-dependent, as it necessitates assessing the neighboring words and sentences prior to proposing potential replacements.

2.4 SPELLING ERROR DETECTION TECHNIQUES

Determining the correctness of a word depends on the type of correction procedure employed. Non-word detection primarily involves verifying a word's presence in the language dictionary (lexicon). Any words that do not match are deemed incorrect. On the other hand, identifying real-word errors is a more intricate task, necessitating a comprehensive examination of significant portions of the text to generate potential misspelled candidates (Sundby, 2009).

As outlined by (Ratnasari,, Kusumadewi,, & Rosita,, 2017), there exist two primary methodologies for detecting non-word errors. These are character n-gram analysis and Dictionary lookup. Character n-gram techniques function by analyzing each character n-gram within an input string and cross-referencing it with a precompiled table of n-gram frequencies. Words that are not found in this precompiled table are classified as misspelled. However, since the compilation of the n-gram frequency table requires a substantial lexicon or corpus, dictionary lookup techniques are more prevalent compared to character n-gram techniques for identifying errors made by humans (Cohen, 2001).

2.4.1 DICTIONARY LOOK UP METHOD

Dictionary lookup is a straightforward method that involves directly checking the presence of each input text in the dictionary. However, the speed of response becomes problematic when the dictionary size becomes larger, going beyond a few hundred words. In tasks like document processing and information retrieval, the number of entries in a dictionary can vary widely, ranging from 25,000 to well over 250,000 words. In such cases, to ensure rapid access to the dictionary, hash table techniques, as proposed by Pagh and Friche in 2004, are preferred. This approach involves looking up an input string by calculating its hash address and retrieving the corresponding word stored at that address in the hash table structure.

Moreover, dictionary lookup techniques tend to be more precise compared to character n-gram techniques, as pointed out by (Baldwin, 2011.), particularly when it comes to correcting non-word

errors. As a result, the majority of methods designed for identifying and rectifying errors generated by humans rely on dictionary-based approaches.

The methods used for dictionary lookup include hashing, binary search trees, and finite state automata. Within these techniques, hashing stands out as the most significant and efficient strategy for performing dictionary lookups swiftly. This strategy revolves around minimizing computational time by relying on the input string to determine the location where a matching pattern is found, as discussed by (Liang, 2008).

2.4.2 N-GRAM ANALYSIS

An n-gram refers to a sequence of n variable-length subsequences of words or characters. Common values for n include one, which produces unigrams, two for bigrams, and three for trigrams. Sometimes, larger values are used. This method identifies errors by analyzing each n-gram within a given string and comparing it with a precompiled table of n-gram statistics. The decision is influenced by the presence of such n-grams and their frequency. If a particular n-gram is absent, the words or strings are considered misspelled. In this study, we focused on bigrams extracted from input sentences at the word level, as opposed to the character level, for error detection. The presence of a word as a bigram is taken as a valid correction, whereas the absence of bigram words indicates misspelled words.

2.5 SPELLING ERROR CORRECTION TECHNIQUES

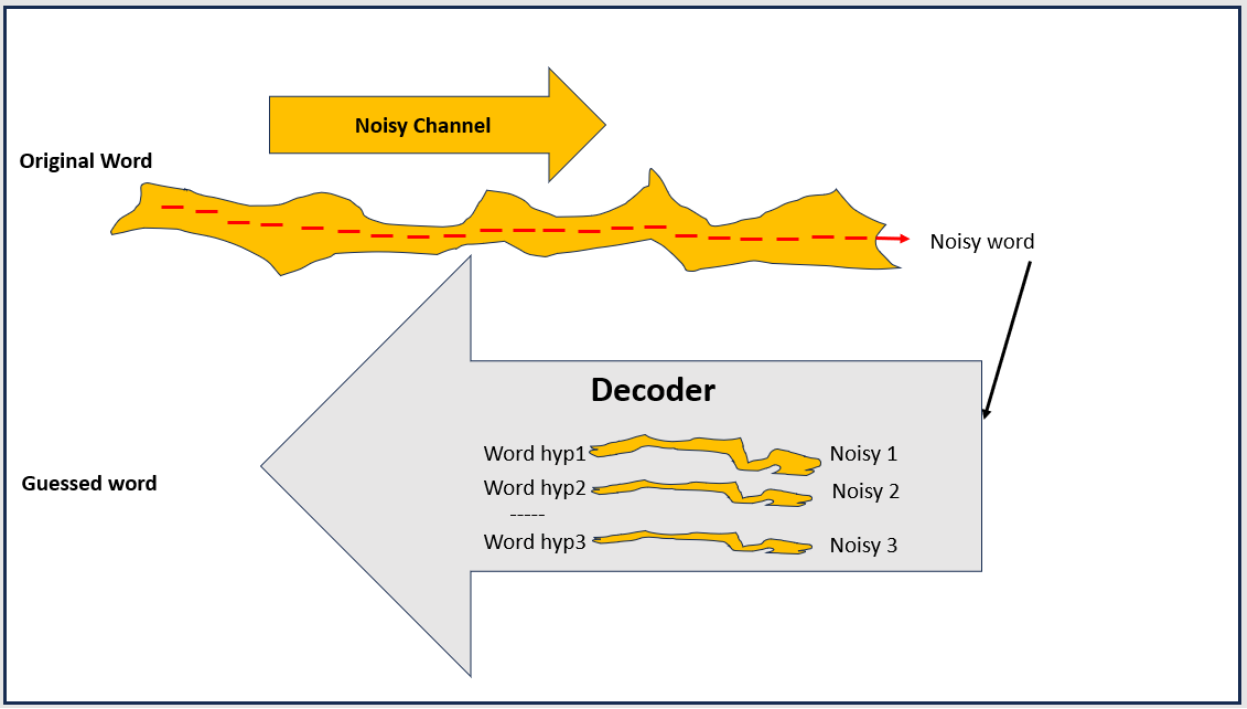
The process of error correction involves identifying potential replacement options for misspelled words, which is accomplished by spell checkers. Various strategies have been suggested for rectifying spelling mistakes, including techniques like Levenshtein edit distance, the noisy channel model, Similarity Keys, and probabilistic methods (Ahmed,, Ernesto,, De,, & Andreas,, 2009).

```
Algorithm: Spell_Correction
Input : word w
Output : suggestion(s) a set of alternatives for w
Begin
    If (is_mistake(w))
        Begin
            Candidates=get_candidates(w)
            Suggestions = filter_candidates(candidates)
            Return suggestions
        End
    Else
        Return is_correct
End
```

Figure 2.2: Spell correction Algorithm

2.5.1 THE NOISY CHANNEL MODEL

The fundamental idea underlying the noisy channel model is to view a spelling error as a signal that has encountered noise or distortion while being transmitted. The crux of this methodology is that if we could determine the nature of the distortion that affected the original word, identifying the correct replacement becomes a straightforward task (Jurafsky, 2009). The noisy channel model aligns with Bayesian inference (Wilde, 2017), representing a form of classification model. This model examines given observations, categorizing and sorting them into relevant classes.



Source: *(Wilde, 2017)*

Figure 2.3: Noisy channel structure

In terms of mathematics, the Bayesian model functions as a probabilistic framework rooted in statistical assumptions and probability theory. Specifically, it centers around two pivotal probabilities: the prior probability denoted as P(w) and the likelihood probability denoted as P(O|w). These probabilities are commonly determined through the following equation (Bassil, 2012):

$$w = argmax \frac{\frac{P(O}{P(W)}/W)}{P(O)} = argmax P(O \backslash W) P(W) \qquad equ, 2.1$$

The prior probability, P(w), signifies the likelihood of the occurrence of the word w within a specific corpus. In contrast, the likelihood probability, P(O|w), characterizes the likelihood of encountering a misspelling denoted as O, given that the correct word is w. Here, O represents the actual misspelled term, while w stands as a potential candidate for the correct spelling. For each candidate, the product of P(O|w) * P(w) is calculated. The candidate with the highest resulting product is chosen as the correction for O, and it is denoted as w'.

The computation of the prior probability, P(w), is relatively straightforward; it is calculated as P(w) = (C(w) + 0.5) / (N + 0.5), wherein C(w) signifies the frequency or occurrence count of the word w in the corpus, and N represents the total number of words present in the corpus. To prevent instances of zero counts for C(w), the addition of 0.5 is included in the calculation. Conversely, estimating the likelihood probability P(O|w) is more intricate than determining P(w). Precisely calculating the probability of a word being misspelled is challenging; however, it can be approximated by evaluating the likelihood of potential errors such as insertion, deletion, substitution, and transposition in a general context.

2.5.2 MINIMUM EDIT DISTANCE TECHNIQUES

This approach relies on determining the smallest count of basic operations needed to change the source string into the target one. The concept of minimum edit distance entails finding the fewest editing actions (such as adding, removing, and replacing characters) necessary for converting one string into another (Misha R., 2013). The initial idea for a spelling correction algorithm based on minimum edit distance was put into practice by (Kaur N. Mishra, 2013), employing transformations involving character insertions, deletions, and substitutions. A significant benefit of utilizing the minimum edit distance metric is its facilitation of easy ranking. Various algorithms exist for minimum edit distance, including the Levenshtein algorithm, Hamming, and Longest Common Subsequence (Misha R., 2013).

2.5.2.1 LEVENSHTEIN EDIT DISTANCE

The Levenshtein distance stands as a matrix used to quantify the dissimilarity between two sequences. It represents the minimal count of individual character modifications needed to transform one term into another, where each modification (insertion, deletion, substitution) incurs an edit cost of 1. This measure finds applications in tasks such as spell-checking, speech recognition (SR), DNA analysis, and plagiarism detection.

In the realm of spell checking, it gauges the likeness between a misspelled word and every entry within a dictionary, selecting the entry with the lowest score as the most fitting match. This selection is based on tallying the distances incurred by each basic operation. The Hamming Algorithm follows a similar principle to Levenshtein but is constrained to strings of equal length. On the other hand, the Longest Common Substring algorithm identifies the shared substring

between two words. The preference for the Levenshtein algorithm stems from its ability to accommodate various symbol types and lengths without limitations (M., 2011).

Mathematically, the algorithm is defined as follows (Sundby, 2009):

f (i, j) = min [(f(i-1, j) + 1, f(i, j-1) + 1, f(i-1, j-1) + d(ni, mj))]

where d (ni, mi) = 0 if qi = li

d (ni, mi) = 1 otherwise, returning Null.

For every erroneous word letter and every word in the dictionary, the function f(i, j) is computed iteratively. This function tallies the differences between the misspelled words n1, n2, …, ni and the dictionary words m1, m2, …, mj. Each insertion, deletion, or substitution contributes a score of 1.

2.5.3 SIMILARITY KEYS

An alternative approach for rectifying identified errors involves the utilization of techniques centered around similarity measurements using key metrics. This methodology identifies a distinct key for aligning words that possess similarity in their assembly. The computed similarity key pertains to the misspelled term and is subsequently linked to a reference that points towards a cluster of words sharing a resemblance in the rectified vocabulary. The Soundex algorithm plays a pivotal role in determining the similarity key required to amend the misspelled word. This algorithm derives the similarity key based on the phonetics of the words. Similarly, the SPEEDCOP system derives the similarity key through a process involving the rearrangement of letters within the words (Misha R., 2013).

2.5.4 PROBABILISTIC TECHNIQUES

Statistical attributes of the language underlie probabilistic techniques, which find application in two prevalent approaches: transition probabilities and confusion probabilities (Ahmad, 2005). Transition probability hinges on the likelihood of a specific letter being succeeded by another particular letter. This probability estimation draws from n-gram statistics derived from a corpus. In contrast, confusion probability signifies the likelihood of a certain letter being misconstrued or substituted by another. The determination of this probability rests upon the source.

Regarding **transition or Markov probabilities**, these determine the probability that a given letter will be succeeded by another specific letter within a particular language context. The computation of these probabilities can be accomplished by amassing frequency data from n-grams sourced from a sizable corpus (Baik, 2006).

On the other hand, **confusion or error probabilities** ascertain the likelihood of a particular letter being interchanged with another designated letter. This can be evaluated by inputting a sample text into an Optical Character Recognition (OCR) device and subsequently compiling error statistics (Evermann, 2000).

2.5.4.1 HIDDEN MARKOVA MODEL

The Hidden Markov Model (HMM) stands as one of the most widely utilized probabilistic techniques that relies on Markov Models. Markov models represent state-space models suitable for depicting a sequence of random variables that may not be entirely independent. A Markov chain denotes a mathematical structure that undergoes transitions from one state to another based on specific probabilistic regulations (Blunsom, 2022). The fundamental aspect of a Markov chain does not concentrate on the pathway taken to reach its current state; rather, the potential forthcoming states are predetermined. Put differently, the likelihood of moving to any particular state hinges solely on the ongoing state and the time that has passed. The array of potential states, or the complete set of feasible states, can encompass various elements: letters, numbers, weather conditions, baseball scores, or stock performances. Precise application of Markov models necessitates dependable and strong estimation of transition probability matrices (TPMs).

In a broader context, the Hidden Markov Model is characterized by its components {S, V, A, B, π}, which are outlined as follows:

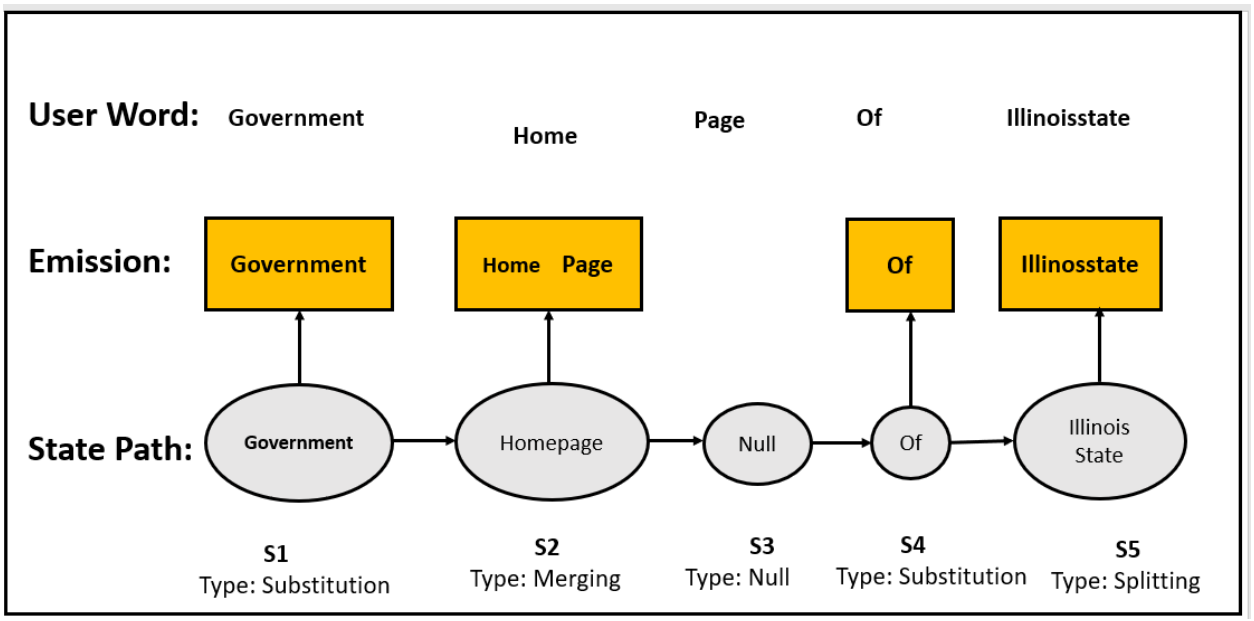
- S represents the collection of states, denoted as {S1, S2... Sn}.

- V stands for the output alphabet, encompassing {v1, v2... Vm}.

- π(i) signifies the probability of being in state qi at time t = 0, signifying initial states.

- A refers to the transition probabilities, expressed as {aij}, where aij stands for the likelihood of transitioning to state qj at time t+1 while being in state qi at time t. Notably, this probability doesn't hinge on prior states in preceding times, illustrating the Markov property.

- B denotes the output probabilities, depicted as bi(k), where b signifies the probability of generating vk at time t while in state qj at time t.



Source: *(Li,, Duan,, & Zhai,, 2012)*

Figure 2.4: Markova Model steps

This sequence of states is alternatively referred to as a Markov Chain. Let's consider X = (X1... XT) as a sequence comprising random variables with values drawn from a finite set S = {S1… SN}, representing the state spaces. The characteristics of the Markov properties can be delineated as follows:

1. **Limited Horizon**: In essence, this signifies that forthcoming elements within the sequence are conditionally autonomous from preceding elements, provided the current element.

$$P(Xt+1= Sk/X1…Xt) = P (Xt+1= Sk/X1…Xt)$$

2. **Time Invariant (Stationary)**: The aforementioned limited horizon dependence remains constant over time.

$$P(Xt+1= Sk/X1…Xt) = P (X2/X1)$$

Where X denotes the representation of a Markov Chain.

2.6 CONTEXT BASED SPELLING CHECKING TECHNIQUES

Contextual spelling error correction involves the task of identifying and rectifying mistakes that lead to a user typing a correctly spelled word that is different from the intended one (Yousfi, 2015). Due to the occurrence of real-word errors, an alternative approach becomes necessary to pinpoint errors. When leveraging context for error detection, the focus extends beyond individual words, encompassing the words that surround them to determine correctness. Presently, we consider pairs of words and attempt to generate analogous words. To illustrate, consider the sentence "you should constantly backup your computer flies." In this context, the term "flies" represents a real-word error primarily triggered by a typographical error., the author did not intend to convey that computers possess the ability to fly like airplanes; the intended meaning is most likely "computer files." This slight confusion has resulted in a real-word error that holds validity within the English dictionary, yet remains inappropriate within the sentence where it has emerged.

Context-aware spelling error correction aims to identify and rectify real-word errors by analyzing their contextual structures. For such words, we compute a probability that indicates the likelihood of their co-occurrence. Subsequently, the word with the highest probability of appearing together is chosen as the correct word and can be proposed as a suggestion to the user. As outlined by (Inkpen D, 2009),there are three approaches available for addressing context-sensitive spelling detection and correction: techniques based on semantic information, methods rooted in machine learning, and approaches founded on probability information (Inkpen D, 2009). In this study, the chosen methodology revolves around probability information.

2.6.1 METHODS BASED ON SEMANTIC INFORMATION

The notion of utilizing semantic information was initially introduced by Hirst and St-Onge in 1998 (Budanitsky A. Hirst G, 2003)and it was further refined by Hirst and Budanitsky (Hirst G. a., 2003). This approach was established on the observation that a writer's intended words are semantically connected to the words surrounding them. In contrast, certain types of real-word errors lack such coherence. An example from Hirst and Budanitsky's paper illustrates this: "It is

my sincere (hope) that you will recover swiftly." Such errors can lead to concerns about the text's consistency and coherence. Hirst and Budanitsky employ semantic distance metrics from WordNet to identify words that might be contextually anomalous—words that possess a semantic separation from neighboring words. If a spelling variation results in a word that holds closer semantic proximity to the context, the hypothesis is that the original word contains an error, and the nearer word serves as its correction (Inkpen D, 2009).

2.6.2 METHODS BASED ON MACHINE LEARNING

The machine learning approach is recognized as a task for resolving lexical ambiguities, and confusion sets are employed to simulate the uncertainty among words. Typically, machine learning and statistical techniques rely on predetermined confusion sets, which consist of commonly interchanged words like (their, there) and (principle, principal). These methods grasp the contextual traits specific to each set member and identify scenarios where one member appears in a context more aligned with another. Although these methods are confined to a predefined set of prevalent errors, these errors can encompass both words of substance and meaning. When presented with an instance of one of its confusions set members, the role of the spellchecker is to anticipate the most fitting member of that confusion set within the given context (Inkpen D, 2009).

2.6.3 METHOD BASED ON PROBABILITY INFORMATION

Furthermore, as outlined by (Inkpen D, 2009), Mays introduced a statistical technique that employs trigram probabilities (a sequence of three words) to identify and rectify real-word errors, all without necessitating predefined confusion sets. In this approach, if the trigram analysis reveals that the likelihood of an observed sentence is reduced compared to the sentence created by substituting one of the words with a spelling variant, then the initial form is classified as a real-word error, while the variant aligns with the user's intended expression (Inkpen D, 2009).

In the current study, the probability information approach was selected due to research indicating its relatively high accuracy (Inkpen D, 2009). As detailed in the preceding chapter, this research centers on context-sensitive spell-checking utilizing bigrams probability. The appropriate algorithm for detecting and rectifying real-word errors within a spelling checker application is the N-grams probabilistic approach (Bassil, 2012). This technique identifies errors by evaluating each n-gram within the provided string and cross-referencing it with a pre-compiled n-gram statistics

table. N-gram techniques typically necessitate either dictionary lookup methods or an extensive corpus of text for pre-compiling an n-gram table. Notably, the primary advantage of n-gram techniques lies in their language independence, meaning they don't require language expertise for the development of a spellchecking application (G., 2006). Additionally, n-grams contribute by offering probability information, estimating the likelihood of a given letter being followed by another to identify plausible solutions for real-word errors (Deksne, 2011).

A language model retains information regarding words within a context. Probabilities are established through the analysis of extensive text samples, quantifying the frequency of each n-gram occurrence. Given the token count and N value, the maximum quantity of sequences for tokens can be determined using the (Jurafsky, 2009):

Ns = Nt - (N - 1)

Where Ns represents the sequence count, Nt denotes the token count, and N signifies the N value within the N-gram.

Let's assume Ns constitutes a collection of sequences for a particular sentence. Among these sequences, one contains W1 and W2 (implying N = 2). The sequence's probability, specifically the probability of W2 given W1, is computed in a subsequent manner (Jurafsky, 2009)

P(W2/W1) = Count(W1W2) / Count(W1)

Here, W2 stands as the second word, and W1 represents the first word in the sequence.

For N = 3 (trigram), the probability of sequences can be computed by tallying the instances of the three words appearing together relative to the number of occurrences of the first and second words together. Assuming W1, W2, and W3 correspond to the initial, second, and third words in a given sequence, the sequence's probability (pertaining to W3 given W1 and W2) is evaluated as demonstrated (Grinstead, 2012):

P(W3/W1W2) = Count(W1W2W3) / Count(W1W2)

The general formula representing sequence probability for N = n is as follows:

P(Wn/W1W2...Wn-1) = Count(W1W2...Wn-1Wn) / Count(W1W2...Wn-1)

2.7 PERFORMANCE OF SPELLCHECKER EVALUATION METHOD

To gauge the efficiency of the developed prototype, an evaluation is imperative. Evaluation metrics provide the means to assess the genuine performance of spell checkers, encompassing both non-word and real-word spelling errors (Borah, 2017). The assessment of the prototype system involves employing evaluation parameters aligned with the count of user-generated words, classifying them as either correctly spelled or misspelled. To appraise the context-based spell-checking effectiveness for Sidaamu afoo Writing, we utilize metrics such as precision, recall, and accuracy as measures of effectiveness.

2.7.1 RECALL

Recall serves as an indicator of the spell checker's comprehensiveness (Borah, 2017),with its focus on assessing how effectively the model identifies valid words from the corpus.

Lexical Recall (LR) quantifies the valid words recognized by the spelling checker (true positives (TP)) within the text, relative to the overall count of correct words in the text (true positives and false negatives (FP)) (Olson, 2008), (Borah, 2017).

Mathematically:

Lexical Recall (LR) = TP/TP + FN

Error Recall (ER), on the other hand, addresses the number of invalid words flagged by the spelling checker (true negatives (TN)) within the text, in proportion to the total count of incorrect words in the text (true negatives and false positives (FP)) (Olson, 2008).

Mathematically:

Error Recall (ER) =TN/TN + FP

2.7.2 PRECISION

Precision stands as a gauge of the correctness of the spell checker's responses (Olson, 2008) (Borah, 2017).

Lexical Precision (LP) is determined by dividing the count of all correct non-flags (true negatives) by the total count of non-flags (namely, true positives plus false positives).

Mathematically:

Lexical Precision (LP) =TN/TN + FP

Error Precision (EP) signifies the count of correct flags (true negatives) in relation to the overall number of flags assigned by the spelling checker (comprising true negatives plus false negatives), offering an insight into the performance of the spelling checker (Olson, 2008)

Mathematically:

Error Precision (EP) =TN/TN + FN

The **F-measure** represents a performance metric that amalgamates Recall and Precision into a unified assessment. It is computed by considering the product of Precision and Recall divided by their sum. The formula for F-measure is as follows (Borah, 2017):

F-measure =2 * precision * recall/precision + recall

2.7.3 ACCURACY

Accuracy, which derives from both precision and recall, provides a comprehensive assessment of the spell checker's overall quality. Predictive Accuracy embodies the spell checker's holistic performance, encompassing calculations and the capability to accurately manage both correct and incorrect words. These metric gauges the spell checker's efficacy in both identifying and rectifying spelling errors within input texts (Borah, 2017).

Mathematically:

Accuracy =TP + TN/TN + TP + FN + FP

2.8 LITERATURE REVIEW

Correcting spelling errors is a crucial function in writing software programs. Nonetheless, many spell-checking tools are created for languages with abundant digital assets, like English. Creating

a context-sensitive spell-checker for languages with restricted digital resources, such as Sidaamu afoo, could enhance the precision of writing in these languages.

In recent times, there has been a growing interest in crafting language technology tools for languages with limited resources, like Sidaamu afoo. Various research endeavors have concentrated on devising spell checkers for diverse languages. However, the necessity persists for spell checkers tailored to individual languages, considering their distinct attributes.

Computational linguistics encounters significant constraints when it comes to resources for Sidaamu afoo, including text collections and lexical databases. This situation not only presents difficulties in creating a spell checker for the language but also underscores the significance of this tool in enhancing literacy rates and safeguarding the language for future generations.

In a study conducted (Mariawit, Sep 2020), an Amharic spelling error detection and correction system was explored using a morphological approach. The research highlighted those unlike other languages, the word distance in Amharic relies significantly on the arrangement and character sequence, due to its intricate morphology. Consequently, employing these methodologies for Amharic, with its complex linguistic structure, may not yield the expected outcomes. While previous Amharic morphology analyzers demonstrated satisfactory precision for valid words, they struggled to offer viable suggestions for misspelled words that needed correction.

The proposed model introduces a morphology-oriented approach to devise an error detection and correction system for non-word errors in Amharic writing. This approach proves capable of scrutinizing words containing multiple prefixes, suffixes, prepositions, and conjunctions. Furthermore, the system underwent testing on derived nouns and exhibited proficiency in both analysis and spell-checking. This suggests its efficacy in scrutinizing complex Amharic words. A system proficient in handling intricate words could also effectively manage simple derivations in other parts of speech, such as adjectives, adverbs, and compound nouns.

In conclusion, the researcher inferred that the developed system possesses the capacity to effectively spell-check a majority of words within the Amharic language.

The work conducted (P. Etoori, July 2018) is titled "Automatic spelling correction for resource scarce languages using deep learning." The study introduces a Sequence-to-sequence text

Correction Model for Indic Languages (SCMIL) aimed at automated spelling correction in Indic languages. This model employs a recurrent sequence-to-sequence attention mechanism to learn spelling corrections from pairs of both accurate and erroneous words. The researchers developed a parallel corpus of correctly spelled words and their corresponding misspelled versions by deliberately introducing spelling mistakes into accurate words.

SCMIL's effectiveness was assessed on these artificially generated datasets for Hindi and Telugu languages. The researchers used the Moses statistical machine translation (SMT) system as a baseline model for spelling correction and compared their system's performance with existing methods tailored for Indic languages. The results demonstrated favorable outcomes for their approach.

Additionally, the researchers discussed potential expansions to enhance the capabilities of SCMIL and conduct more comprehensive evaluations. Notably, while many deep learning models require training on extensive datasets comprising billions of instances, SCMIL achieved competitive results using a dataset of less than a million parallel examples. This highlights the suitability of their approach for automatic spelling correction in languages with limited resources.

In a study by Verberne in January 2002, the focus was on context-sensitive spell-checking using word trigram probabilities. The research aimed at developing a method for spell checking and correction that considers context, allowing it to identify and suggest various types of real-word errors made by humans, including misspellings. The foundation of this context-aware spell checker and correction system was built upon a lexicon-based application for correcting spelling errors, primarily targeting non-word errors.

The innovation lay in combining the context-aware spelling checker and corrector with the aforementioned lexicon-based correction application. This combined approach enabled the creation of a system capable of addressing both non-word errors and real-word errors, which commonly arise in the context of spell checker applications. Non-word error detection pertains to identifying errors that don't correspond to entries in a lexicon. Techniques for detecting non-word errors are often referred to as isolated-word error detection techniques.

The method employed in this application for error detection and correction relied on analyzing three-word sequences, or trigrams, instead of individual word tokens. This choice was motivated

by the observation that misspelled words often lead to improbable sequences of three words occurring sequentially.

In the research conducted by (Lorraine, November 2008) titled "Spell Cheekers and Correctors: A Unified Treatment," a comprehensive approach was taken to address spell checking and correcting challenges. The study encompassed four main stages.

Firstly, the denotation or semantics of spell checking and correcting problems were provided, building upon the work by (F. Ahmed, 2020). Secondly, techniques employed to accomplish the tasks of spell checking and correction were identified.

The third step involved an exploration of different spell checking and correcting algorithms. Notably, algorithms such as spell, correct, a spell, fsa package, agrep, speed cop, aura, DNS Spell Checker, and CInsunSpell were thoroughly investigated.

Lastly, the research delved into empirically assessing the performance of the various spell checking and correcting algorithms, subsequently classifying them based on their effectiveness. Towards the conclusion, an in-depth mathematical investigation encompassing numerous other spell checking and correcting packages was carried out. The researcher proposed a classification of these models based on their implementation strategies, system functionalities, and performance outcomes. Ultimately, the developed and tested system demonstrated satisfactory functionality within the defined scope of the study.

In a study conducted (F. Ahmed, 2020) titled "Revised N–Gram Based Automatic Spelling Correction Tool to Improve Retrieval Effectiveness," an enhanced approach for automatic spelling correction was introduced to enhance retrieval effectiveness. According to Ahmed's findings, they introduced a language-agnostic spellchecker that builds upon a refined version of the original n-gram model. The research also involved evaluating this approach on datasets containing misspelled words from both English and Portuguese languages. Impressively, the results obtained surpassed the performance of other contemporary methods.

The research outlines plan for future improvements, aiming to optimize both the algorithm and data structures used for computing similarity scores. The core process of the spell check and correction involves selecting the most promising candidates from a ranked list of suggestions

derived from lexical resources and n-gram probability statistics. This ensures the efficacy of the developed application.

The study's proposal and subsequent evaluation of the developed system demonstrated its superior performance compared to alternative methods tested during the system's testing phase. This research showcases the potential of the approach in significantly enhancing the functionality of automatic spelling correction systems.

In a study conducted (Tirate, 2018) titled "Context-Based Spelling Checking for Afaan Oromo Writing," a context-sensitive spell checker for Afaan Oromo has been conceptualized, developed, and subjected to testing for both non-word and real-word errors. The research employs two distinct approaches for spelling checking: the dictionary lookup approach and the N-gram approach.

The dictionary lookup approach is used to identify and rectify non-word errors. It functions by cross-referencing each user word with a dictionary. Conversely, the N-gram approach targets real-word errors and is designed to operate using N-gram statistics. This approach necessitates a high-quality training dataset and aids in detecting real-word errors within given words. Both of these approaches have the capability to address both non-word and real-word errors present in the Afaan Oromo writing system.

The study outlines three main phases encompassing the spelling correction process: error detection, candidate suggestion, and ranking of suggested candidates. The culmination of testing indicates the successful performance of the developed system, yielding substantial accuracy in the context-based correction of misspelled words within Afaan Oromo text when used on computer platforms.

In a study conducted (Patil, Aug, 2020.) titled "Auto-Spelling Checker Using Natural Language Processing," the focus was on the process of spell checking, which involves identifying and proposing corrections for incorrectly spelled words within a paragraph. The developed spell-checking system operates by initially detecting incorrect words within the written paragraph and subsequently offering appropriate corrections when such errors are identified.

This system is a fusion of several language rules required for constructing a spell-checking system, combined with a comprehensive dictionary containing accurate and error-free words. The research

suggests that the effectiveness of the developed system can be enhanced by employing more refined rules and utilizing an extensive and accurate dictionary. This approach aids in elevating the error detection rate and providing relevant suggestions for word corrections. In cases of wrongly spelled words, the system presents multiple suggestions.

Multiple systems are available for text detection and correction. This particular system is designed to perform spelling checks using a word list present within a text file.

In a study carried out (Petros, Oct 25, 2020.), the focus was on developing a spellchecker and corrector system tailored specifically for the Tigrigna language, employing a rule-based approach specifically designed for mobile phone devices. The researchers compiled a corpus from various sources such as newspapers, magazines, and books, acknowledging that Tigrigna is a language with limited resources. Creating such a system is particularly significant for the language's advancement.

In this study, the researchers proposed a system model for spellchecking Tigrigna language using a rule-based method, specifically addressing word-level spelling errors. To evaluate the system's performance, they utilized a corpus containing 430,379 Tigrigna words, which was collected and stored in dictionary format as a reference.

Through the development of a prototype and the formulation of an algorithm, the research successfully demonstrated the viability of the model, yielding impressive outcomes in the system's output. The paper's ultimate proposition was a dictionary-based spellchecker for Tigrigna language, and the experimentation phase showcased a high accuracy rate of 92% for the prototype of the Tigrigna spellchecker and correction system designed for mobile phone devices.

In a study conducted (Debela, 2011.) titled "A Rule-Based Afaan Oromo Grammar Checker," the focus was on addressing grammar errors in Afaan Oromo and proposing probable corrections. Detecting and suggesting the most likely corrections for grammatical errors, which involve considering neighboring words in a sentence or even entire sentence structures, poses significant challenges for computational linguists and software developers compared to the relatively simpler task of checking the orthography of Afan Oromo language.

Grammar errors often involve violations of syntactic laws or rules pertaining to sentence structure and language syntax. One prominent example of such a rule is the agreement requirement between nouns and adjectives in terms of gender and grammatical number, which holds significance for writers and users of the Afan Oromo language.

The researcher utilized three prevalent methods for grammar checking in a language: syntax-based checking, statistics-based checking, and rule-based checking. These methods collectively contribute to the development of a system capable of identifying and addressing various types of grammar errors within the context of Afaan Oromo.

In a study conducted (Getnet, June, 2018.) titled "Automatic Amharic Spelling Error Detection and Correction Using a Hybrid Approach," the researcher developed and implemented a spell checker model utilizing the visual basic programming language. During the prototype's creation, fundamental algorithms were applied at various stages of the development lifecycle. Notably, the edit distance algorithm was employed to select the most probable replacement for misspelled words while users typed with errors. Additionally, the Metaphone algorithm was integrated for spelling error detection and to determine the most likely correct word for a misspelled entry.

To assess the prototype's functionality and usability, the implemented system was integrated as an add-in within Microsoft Office Word 2010, allowing the exploration of how well the model performed. Rigorous testing was carried out using a test dataset containing 500 words, out of which 100 were intentionally misspelled. A dictionary comprising 125,000 words was used in these evaluations.

During the final testing phase of the development lifecycle, the model exhibited exceptional performance in both error detection and the provision of accurate suggestions for error correction. This research showcases the successful integration of a hybrid approach for automatic Amharic spelling error detection and correction, yielding promising results for practical applications.

In a study conducted (Addisu, 2016), a part-of-speech tagger was developed specifically for the Sidaama language. The findings of the research indicate that the accuracy of the Brill tagger varies depending on the proportion of corpus data used for training the lexical and contextual rule learners. The study highlights that optimal accuracy is achieved by allocating a larger amount of

data to the lexical rule learner. This suggests that lexical or morphological rules hold greater prominence within the language compared to contextual rules.

The annotation of the corpus was conducted using an incremental approach, resulting in the creation of a relatively small corpus comprising approximately 2,842 sentences (with 51,870 tokens). The tagger was trained on 2,558 sentences (41,822 tokens) and tested on 284 sentences (4,613 tokens).

According to Addisu's findings, the experimental results yielded a relatively high level of accuracy. Specifically, 93.64% of the total words were correctly tagged by the developed tagger. This outcome underscores the effectiveness of the part-of-speech tagging approach for the Sidaama language, as outlined in the study.

2.9 RELATED WORKS

Various works have been reviewed from a range of journals and web pages, focusing on different methodologies and techniques for spell checking, as well as the advancements made by scholars in this field. (Liang, 2008) categorizes spelling check techniques into two main groups: Dictionary Look Up and N-gram analysis. The Dictionary Look up method stands out as a widely utilized approach for identifying misspelled words. In fact, many spell-checking processes rely on this method to detect errors in text (Mihov S., 2004). This method relies on a dictionary that encompasses an exhaustive collection of correctly spelled words within the language. In the process of Dictionary Look Up, a tokenized word is initially inserted, and then the token is cross-referenced with the list of stored words.

(R.C., 2010) demonstrated that the size of a dictionary plays a pivotal role in the efficiency of identifying misspelled words within a text. A smaller dictionary might result in numerous accurate words being flagged as erroneous. On the other hand, opting for a larger dictionary can mitigate this issue but at the cost of increased word lookup time. To tackle the extended access times, various optimized storage mechanisms like hash tables and tries can be employed, as indicated by gated results. In the contemporary context, owing to the enhanced computational capabilities available, the time required for dictionary word lookups has become largely inconsequential.

(S., 2013) introduced a novel approach to spelling correction, treating words as "documents" and framing spelling correction as a type of document retrieval. In this model, the goal is to retrieve

the most suitable correctly spelled word that matches the given input. This involves representing words as small document-like entities composed of bits. The Hamming distance metric is employed to predict the nearest bit string match from a dictionary containing the correctly spelled words represented as bit strings. This knowledge-free model solely maintains a list of accurate words.

(Bassil, 2012) developed a parallel spell-checking algorithm for detecting and rectifying spelling errors. The algorithm draws on information from the Yahoo! N-gram dataset 2.0. It follows a shared memory model, enabling concurrent execution among threads across parallel multi-processor and multi-core machines. The algorithm's key components—error detection, candidate generation, and error correction—are designed to operate in parallel. The error detector focuses on unigrams and identifies non-word errors. The candidate generator utilizes bi-grams, while the context-sensitive error corrector relies on 5-gram information.

(Ahmed,, Ernesto,, De,, & Andreas,, 2009)developed a model for correcting spelling errors in real-word sentences. They treated the sentence as a signal passed through a noisy channel, representing the typist's input with errors. Each word's correctness probability was associated with the sentence's overall correctness probability, and each word had a set of potential correct spellings. The correction process involved replacing a wrongly spelled word with a candidate from its set of valid spellings that yielded the highest probability.

(Mihov S., 2004) outlined a method for post-correction of optical character recognition (OCR) output. Their approach focused on two main aspects: expanding dictionaries with specialized entries and efficiently selecting correction candidates using a Levenshtein automaton-based algorithm. They ranked candidates based on various factors such as frequency and edit distance.

(Verberne, Context-sensitive spell checking based on word trigram probabilities, 2002)proposed a context-sensitive spell-checking algorithm based on BESL spell checker lexicons and word trigrams. The algorithm segmented input text into trigrams, consulting a database containing trigrams and their occurrence frequencies. Trigrams not found were considered to have real-word errors. The BESL spell checker suggested candidates, with priority given to the most frequent trigrams in the database.

(Ringlstetter, 2007)employed a bigram model to rectify misspelled words. They used web-crawled corpora to construct dictionaries and language models, improving correction accuracy by incorporating bigram frequency values into a baseline strategy based on word similarity and frequency. This enhancement required special memory techniques to efficiently handle bigram counts for large dictionaries, leading to more effective results compared to simpler scoring methods based on word similarity and frequency.

(Kirthi J.) proposed a semantic information retrieval system that integrates automatic spell correction into user queries before initiating the retrieval process. The correction process hinges on matching misspelled words against a dictionary of correctly spelled words using the Levenshtein algorithm. When an erroneous word is detected, the system retrieves the most akin word based on the Levenshtein measure and the frequency of occurrence of the misspelled term.

(Ahmed,, Ernesto,, De,, & Andreas,, 2009)devised a language-independent spell checker that enhances the N-gram model. This enhancement involves generating a prioritized list of correction candidates founded on N-gram statistics and lexical resources. Subsequently, the most promising candidates are selected as correction suggestions, and the algorithm assigns weights to potential suggestions to identify nonword errors. The approach relies on the "multiWordNet" dictionary, containing around 80,000 entries.

In the study (Wagara, 2022) they employed a two-pronged approach to identify and rectify spelling errors in Sidaamu afoo's written text. they utilized a dictionary look-up method coupled with a hashing algorithm to detect non-word errors and a character-based encoder-decoder model to correct them. For context-based spelling errors, they harnessed the power of an LSTM model with an attention mechanism and the edit distance metric for detection and correction. Their experiments involved a dataset of 55,440 sentences, with 90% (49,896) allocated for training and 10% (5,544) for testing.

Their results showed that the dictionary look-up approach with hashing achieved an impressive accuracy of 93.05% for detecting errors, with a recall of correct words at 91.51% and a precision of incorrect words at 72.37%. The encoder-decoder model exhibited a recall of 91.76% for correction. In the realm of context-sensitive spell checking, the LSTM model with attention and edit distance yielded an accuracy of 88.8%, a recall of correct words at 86.18%, and a precision of

incorrect words at 62.84% for detection. Furthermore, it achieved a recall of 74.28% for correction. These experimental findings affirm the effectiveness of their chosen model in detecting and correcting both non-word and real-word spelling errors in Sidaamu afoo's text.

In conclusion, to further enhance the model's performance, they recommend augmenting the dataset with additional examples and considering the adoption of state-of-the-art transformer models.

The primary objective of this research was to create a context-based spell checker suitable for the Sidaamu afoo writing system. Unlike many other languages, Sidaamu afoo lacks a spell checker to rectify misspelled words. Such errors can impede effective communication between authors and readers, hindering the transfer of knowledge. Given the unique linguistic structure and letter arrangement of Sidaamu afoo compared to other languages, the development of a context-based spell checker specifically tailored to Sidaamu afoo was essential to address misspelled words within its writing system.

CHAPTER THREE
SIDAAMU AFOO WRITING SYSTEM

3.1 OVERVIEW

Sidaamu afoo language is an Afro-asiatic language that belongs to the Highland Cushitic branch of the Cushitic family. The Sidama people use the Sidaamu afoo language, primarily in southern Ethiopia, with a significant presence in the densely populated Sidama National Regional State (SNRS). Sidaamu afoo is the ethnic autonym for the language, while Sidaminya is its name in Amharic. Although it is not known to have any specific dialects, it shares over 64% lexical similarity with Alaba-K'abeena, 62% with Kambaata, and 53% with Hadiyya, all of which are other languages spoken in southwestern Ethiopia. The word order is typically Subject Object Verb (SOV). Sidaama has over 100,000 people use it as a second language. In terms of its writing, Sidaama used an Ethiopic script up until 1993, from which point forward it has used a Latin script. (RaymondJr R. G., 2005)

Sidama are an ethnic group traditionally inhabiting the Sidama Region, formerly part of the Southern Nations, Nationalities, and Peoples' Region of Ethiopia. On 23 November 2019, the Sidama Zone became the 10th regional state in Ethiopia after a zone-wide referendum. They speak the Sidama Language (Sidaamu afoo), which the language of Cushitic branch of the Afro-asiatic language family. (Allein, 2016).

3.2 DESCRIPTION OF SIDAAMU AFOO ALPHABETS AND SOUND SYSTEMS

Various scholars have recognized that the writing system of Sidaamu afoo utilizes the Latin script, with its alphabet and phonetic sounds being adaptations of the Latin script. Consequently, Sidaamu afoo exhibits many similarities with the English writing system due to certain modifications, which are crafted based on the Latin script. Therefore, it can be observed that Sidaamu afoo shares common letters with the English or Latin alphabets, albeit with distinctions in their phonetic combinations and pronunciation styles, as pointed out by (Kawachi K. , 2007) .

Given that the Sidaamu afoo writing system is essentially an adaptation of the Latin script, it shares many similarities with the English writing system, albeit with some modifications. Consequently, both English and Sidaamu afoo contain the same letters, but their language structures are

fundamentally distinct. In Sidaamu afoo, sentences are structured as Subject-Object-Verb (SOV), whereas English follows the Subject-Verb-Object (SVO) arrangement for sentence construction. For example, the Sidaamu afoo sentence "Beettu hiikkiira ha'ri?" which translates to "Where did the boy go?" places "Beettu" as the subject, "hiikkiira" as the object, and "ha'ri" as the verb.

As previously stated, Sidaamu afoo employs Latin characters encompassing both consonants and vowels, totaling 33 languages specifically refer to Table 3.1:List of Sidaamu afoo language specific, Consonants and vowels. In the fundamental Sidaamu afoo alphabet, "p," "v", "ts", "zh" and "z" are notably absent. This omission stems from the absence of native Sidaamu afoo words that incorporate these characters. Nevertheless, over time, additional letters were introduced to the alphabet to accommodate words that necessitate the use of these particular letters.

3.3 VOWELS (COYISHIISHAANO)

Sidaamu afoo comprises five vowels: 'a,' 'e,' 'i,' 'o,' and 'u.' While they share similarities with English vowels, their pronunciation differs. It's noteworthy that each vowel in Sidaamu afoo maintains consistent pronunciation across all instances within the language's literature. In Sidaamu afoo, words are formed using both vowels and consonants. Vowels are phonetic elements that can stand on their own and produce sounds independently. In Sidaamu afoo, vowels can be classified into short and long vowels ('aa,' 'ee,' 'ii,' 'oo,' and 'uu'), and altering their length can result in changes in the meaning of words.

As an illustration, in Sidaamu afoo, "Maala" signifies Meat, whereas "Mala" represents seeking a solution. In Sidaamu afoo, it's important to note that vowel length, whether short or long, can lead to spelling errors. Moreover, Sidaamu afoo features both double and single consonants, and spelling errors may arise from adding or omitting these consonants, thereby altering the intended meaning.

For instance, in the context of English, "Qallo" translates to "interest," while "Qalo" means "Creativity." Consequently, misspelled words can arise in Sidaamu afoo due to variations in vowel length (short or long), the presence or absence of double consonants, as well as the rearrangement, removal, or addition of characters within words.

Table 3.1:List of Sidaamu afoo language specific, Consonants and vowels

No	Capital	Short, Double	Type
1	A	a, aa	Vowel
2	B	b	Consonant
3	C	c	Consonant
4	D	d	Consonant
5	E	e, ee	Vowel
6	F	f	Consonant
7	G	g	Consonant
8	H	h	Consonant
9	I	i, ii	Vowel
10	J	j	Consonant
11	K	k	Consonant
12	L	l	Consonant
13	M	m	Consonant
14	N	n	Consonant
15	O	o, oo	Vowel
16	P	p	Consonant
17	Q	q	Consonant
18	R	r	Consonant
19	S	s	Consonant
20	T	t	Consonant
21	U	u, uu	Vowel
22	V	v	Consonant
23	W	w	Consonant
24	X	x	Consonant
25	Y	y	Consonant
26	Z	z	Consonant

			(borrowed -Argete Fidalla)
27	CH	ch	Siwilu fidalla
28	DH	dh	Siwilu fidalla
29	NY	ny	Siwilu fidalla
30	PH	ph	Siwilu fidalla
31	ZH	zh	Siwilu fidalla
32	SH	sh	Siwilu fidalla (geminated in the Sidaamu afoo)
33	TS	ts	Siwilu fidalla
34	‘		

Sidaamu afoo uses Latin character consonants and vowels it uses 26 letters of Latin character and **7 Siwilu fidalla**. However, later on a new letter was included in the alphabets as there are words which require the letter.

Table 3.2: Sidaamu afoo consonant phonemes

	Bilabial	Labio dental	Dental	Alveolar	Palatal	Velar	Glottal
Stop Plosive Ejective Implosive	b p’	t d t’ dh			k g k’		
Affricate Ejective			c j sh				
Fricative		f	s z	sh			h
Nasal	m	n	n		ny		
Tap/Flap			r				
Lateral Approximant			l				
Approximant	w				w		

Source: (Wagara, 2022)

Table 3.3: Sidaamu afoo vowel phonemes

	Front	**Central**	**Back**
High	i/ii		u/uu
Front	e/ee		o/oo
low		a/aa	

Source: (Wagara, 2022)

3.4 SIDAAMU AFOO WORD CLASS

Recent studies have outlined that Sidaamu afoo encompasses five primary word classes: nouns, ad-positions, adverbs, conjunctions, and verbs. Each of these classes can be further categorized into sub-classes. For instance, within the noun class, there exist subdivisions like proper nouns, common nouns, pronouns, and more. Likewise, ad-positions include sub-classes such as prepositions and postpositions. This classification hierarchy can continue to branch into further subcategories, depending on the specific level and objectives of the research, as indicated by (Addisu, 2016).

3.4.1 NOUN (SU'MA)

Nouns in Sidaamu afoo encompass any words capable of naming or identifying places, objects, or ideas. These nouns are divided into two grammatical genders, namely masculine and feminine, and all nouns in the language belong to one of these gender categories. Additionally, Sidaamu afoo employs two numbers, singular and plural, which can be distinguished by the morphemes they incorporate. To form the plural form of a noun, various suffixes can be appended to the root noun. It's important to note that these suffixes result in the transformation of a singular noun into its plural form without altering its meaning. Typically, the final vowel of the singular noun is dropped before adding the suffix. Examples of such suffixes include "-a,","-e,","-o,","-no," "-ra," "-mmo," "-nna," ","-shshe," "-cho," and so on.

In contrast to the English language, which uses definite articles ("the") and indefinite articles ('a,' 'an,' 'some' and 'any'), Sidaamu afoo does not employ articles before nouns. However, it does utilize suffixes to convey a similar context to the English definite article, where "-(t)ttu" is employed for masculine nouns and "-(t)tto" for feminine nouns. It's worth noting that the final

vowels of nouns are omitted before these suffixes are affixed to the noun. For example, "beettu" translates to 'boy,' "beettu" to 'the boy,' "beetto" to 'girl,' and "beetto" to 'the girl.'

3.4.2 PRONOUN (‘SU’MU-RIQIWO’)

Pronouns in Sidaamu afoo are words that serve as substitutes for nouns. Much like nouns, pronouns in the language also possess gender and number distinctions. For instance, " ise," signifying 'she,' is feminine and singular, while "isi," meaning 'he,' is masculine and singular. On the other hand, "insa," translating to 'they,' can be either masculine or feminine in the plural form. Pronouns can further be categorized based on their roles and meanings within sentences, including personal pronouns, possessive pronouns, demonstrative pronouns, relative pronouns, or reciprocal pronouns.

3.4.3 VERB (‘GURDA’)

In Sidaamu afoo, verbs serve as words used to signify actions or events occurring within specified time frames, as explained by (Kawachi K. , 2007). These verbs can be categorized into various types, including transitive verbs, intransitive verbs, modals, and auxiliary verbs. Transitive verbs are those that convey a message to a complement or object, while intransitive verbs do not convey a message to a complement and, consequently, do not require a complement or object. The following examples serve to illustrate this distinction: "Manchu saa hidhi," which means ‘He bought a cow’.' In this instance, since the action of buying was directed toward the object 'Saa' (Cow), 'hidhi' is identified as a transitive verb.

3.4.4 ADVERBS (‘GURDI-LEDO’)

In Sidaamu afoo, adverbs are words that serve to clarify or modify verbs, as explained by (Kawachi K. , 2007). These adverbs can pertain to various aspects such as time, place, manner, or frequency. In Sidaamu afoo, adverbs are positioned before the verbs they are modifying. For instance, consider the sentence: "Shubaare rake dagu," which translates to 'Shubaare came quickly.' Here, ' rake ' means 'quickly' and functions as an adverb. In another example, "Baqqali ga’a daano," which means 'Bekele came tomorrow.' In this case, 'ga’a' means 'tomorrow' and serves as an adverb.

3.5 SIDAAMU AFOO PUNCTUATION MARKS

Punctuation plays a crucial role in text by enhancing clarity and facilitating easier reading. An examination of Sidaamu afoo texts reveals that they follow a punctuation pattern similar to English and other languages that utilize the Latin Writing System. Much like in English, Sidaamu afoo employs several commonly used punctuation marks as described by (Addisu, 2016)). For instance, the 'Taxxeessu malaate' comma (,) is used to separate lists of ideas, concepts, names, or items. The full stop(period) 'Uurrinshu Malaate' (.) is employed to conclude statements. The 'Xa'mote malaate' question mark (?) marks interrogative sentences, the 'Ittisu malaate' (;) semi-colon to join independent clauses, separating items in a list, the 'Maqishshu malaate' ("") is used to indicate direct speech, defining terms, avoiding confusions and the 'Darbu Malaate' exclamation mark (!) signifies the end of command or exclamatory sentences.

3.6 SIDAAMU AFOO MORPHOLOGY

Much like several other African and Ethiopian languages, Sidaamu afoo possesses a highly intricate and robust morphology, as observed by (Kawachi K. , 2007). It exhibits the foundational characteristics of agglutinative languages, characterized by extensive inflectional and derivational morphological processes. In the context of agglutinative languages like Sidaamu afoo, the bulk of grammatical information is effectively conveyed through affixes, including prefixes, infixes, and suffixes, which are affixed to the root or stem of words. It's worth noting that while Sidaamu afoo words do incorporate certain prefixes and infixes, suffixes hold the predominant role as the primary morphological features within the language.

In Sidaamu afoo, nearly all nouns within a given text exhibit person, number, gender, and possession markers, which are concatenated and affixed to the stem or singular noun form. Moreover, when it comes to plural markers or forms for Sidaamu afoo nouns, there exists a variety of alternatives, unlike the English noun plural marker "-s" or "-es." In Sidaamu afoo, there are over fifteen major and commonly used plural markers, including "-mmo," "-no," "-ra," "-si," "eti," "nna," "-di," "-cho," "-shshe," and more. As an illustrative example, consider the Sidaamu afoo singular noun "mine" (house), which can take various plural forms such as " mineemmo" (mine + emmo), "mineeti" (mine + eti), and " mineemmosi " (mine + emmosi). The utilization and application of these alternative affixes and attachments adhere to the morphological and syntactic

rules inherent in the language, as described by (Kawachi K. , 2007). Additionally, Sidaamu afoo nouns exhibit diverse case and gender suffixes contingent upon the grammatical level and classification system employed for their analysis.

Table 3.4: The construction of gender and suffixes in Sidaamu afoo letters

No	**Sidaamu afoo Word**	**Construction**	**Gender**	**English**
1	*Rosiisaancho*	*Roso + saancho*	Male	Teacher
2	biilloonynye	Biilo + nynye	Male	Authorization
3	Qineessaancho	Qineessa + aancho	Male	Coordinator
4	Duuwaleette	Duwo + Leette	Female	Arrogant
5	Albisa	Alba + isa	Male	To precede

3.7 NUMERALS (‘KIIRO’)

Numerals encompass words denoting numbers or quantities, as noted (Addisu, 2016). These can take the form of cardinals like "mite" (one) or "lame" (two) or sase(three) or ordinals such as "umiha" (first) and "layinkimeeshiha" (second). As discussed (Kawachi K. , 2007), numerals in Sidaamu afoo are positioned in a sentence according to the category they describe in terms of quantity or amount. Ordinal numerals in the language are created by appending the suffix "-ha" to cardinal numerals. To illustrate, please consider the following sentences:

1. In the sentence "***Baqqali sase handuwa hidhi***," it signifies that 'Bekele bought three oxen.' Here, the word 'sase' functions as a cardinal numeral, specifying the quantity of 'handuwa' (oxen).

2. In the sentence "***Selomme mitikkita fultino***," it conveys that 'Selome stood first in her classes.' In this instance, the word 'mitikkita' serves as an ordinal numeral, indicating 'first.' It is derived from the cardinal numeral 'mite' ('one') through the addition of the affix '-kkita.'

Creating a spellchecker for Sidaamu afoo presents challenges due to the absence of a standardized corpus. The spellchecker relies on the Sidaamu afoo corpus it has collected to identify and correct misspelled words. However, the lack of a well-established corpus poses a significant obstacle in designing a context-based spellchecker for Sidaamu afoo. Additionally, the language exhibits complexities related to the repetition of different letters that represent the same word. In our data preparation efforts, we aimed to address these complexities by including repeated letters and accounting for morphological intricacies.

CHAPTER FOUR

MODEL OF CONTEXT BASED SPELLCHECKER FOR SIDAAMU AFOO

4.1 INTRODUCTION

In the preceding sections, we outlined various studies pertaining to mechanisms aimed at detecting and rectifying spelling errors across different languages. Additionally, we highlighted fundamental characteristics of the Sidaamu afoo language that must be taken into account before devising the Sidaamu afoo spell checking and correction model. Moving forward, this section delves into the approach adopted in this research, encompassing the model's architecture, employed techniques, and the chosen algorithm for constructing a context-based spell checker tailored to Sidaamu afoo.

4.2 SPELL CHECKING MODEL

The development of the Context-Based Spellchecker for Sidaamu afoo (CBSCSA) model aimed to identify and rectify both non-word and real-word errors. The design of CBSCSA followed a three-step approach, encompassing: (a) detecting errors to identify misspelled words within user input, (b) error correction to propose potential candidates for rectifying these misspelled words, and (c) ranking the candidates to select the optimal suggestion based on a high similarity score and the maximum bigram probability.

In essence, Figure 4.1 outlines the sequential architecture for creating a context-based spellchecker tailored to the Sidaamu afoo writing system. As depicted in Figure 4.1, the initial phase involves the model detecting misspelled words within preprocessed user input. For non-word errors, a dictionary lookup method from a prepared dictionary is utilized for identification. In contrast, real-word errors are pinpointed using bigram sequence analysis derived from a bigram model.

Moving to the subsequent step of the CBSCSA model, the error correction module attempts to provide potential candidates for the detected misspelled words. For non-word error correction, the model employs the Levenshtein algorithm to compute the distance between the user's misspelled word and dictionary words, subsequently suggesting corrections. Meanwhile, bigram probabilities are applied to rectify real-word errors by calculating statistical information for each individual word.

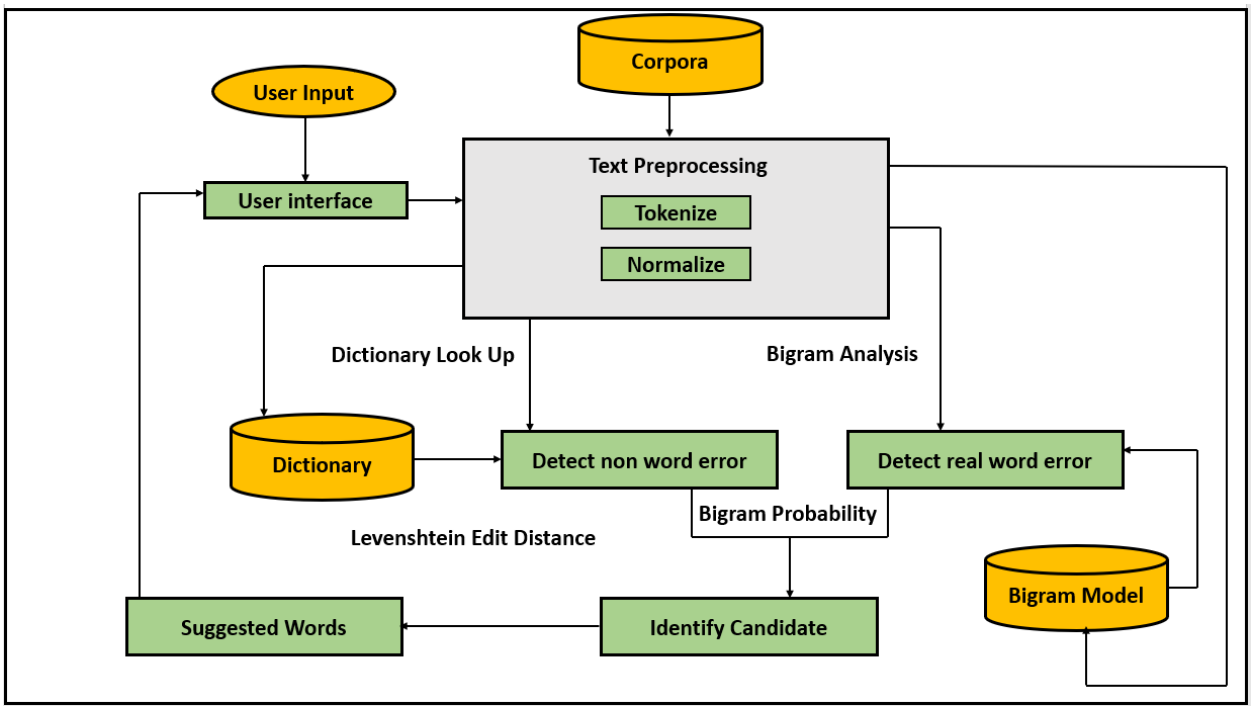


Figure 4.1:Context Based Spell Checker for Sidaamu afoo Architecture

Detail description of the architecture

User Input

- The process begins with text input provided by the user through a User Interface. This is where users type the text that needs to be checked for spelling errors.

Corpora

- A Corpora is a large collection of correctly spelled words, phrases, and language patterns specific to the Sidaamu Afoo. This serves as a linguistic database that the spell checker refers to for identifying common words, phrases, and correct usage.
- It's essential for understanding context and helps in real-word error detection.

Text Preprocessing

- The text input is sent to a Text Preprocessing module, which consists of two key steps:
- Tokenization: The input text is split into individual units, usually words or tokens, to be analyzed separately.
- Normalization: This involves standardizing the text format. It includes converting all letters to lowercase, removing punctuation, and handling language-specific

transformations. This ensures consistency, making it easier to check each word against the dictionary and bigram model.

Dictionary Lookup

- Each tokenized and normalized word is then checked against a Dictionary that contains valid words in Sidaamu Afoo. This helps identify whether a word is valid or potentially misspelled.

Error Detection

- Non-Word Error Detection: If a word is not found in the dictionary, it is flagged as a non-Word Error, which typically indicates a misspelling. The system then looks for possible corrections for this word.
- Real-Word Error Detection: For words that are found in the dictionary but may be incorrect in the specific context, the system applies Bigram Analysis (contextual analysis). This involves analyzing word pairs to detect Real-Word Errors—words that are correct in isolation but incorrect given the surrounding words.

Bigram Model and Probability

- Bigram Model: A Bigram Model is a statistical model that stores the likelihood of word pairs based on their frequency in the corpora. It helps in determining the appropriateness of word sequences.
- Bigram Probability: Using the bigram model, the system calculates the Bigram Probability of word pairs to decide if a word makes sense in its specific context. If a word pair has a low probability, it's flagged for review, suggesting that it might be contextually inappropriate.

Candidate Identification and Correction:

- Identify Candidate: For each detected error (non-word or real-word), the system identifies possible corrections. For non-word errors, the Levenshtein Edit Distance is applied to suggest words that are similar in spelling but correctly listed in the dictionary.

- For real-word errors, the bigram probabilities guide the system to find contextually appropriate alternatives.
- Suggested Words: Based on identified candidates, the system presents Suggested Words to the user. These suggestions are based on words with the smallest edit distance for non-word errors and the highest bigram probabilities for real-word errors.

Output:

- The system outputs a list of suggested corrections for each identified error. These suggestions are then presented to the user, who can decide whether to accept or reject the recommended corrections.

4.3 DICTIONARY AND BIGRAM MODEL CONSTRUCTION

As outlined in section 1.6.3, the corpus was gathered from diverse sources. This collected corpus underwent preprocessing, involving tokenization and normalization, as well as meticulous manual inspection to eliminate any superfluous errors. Consequently, every word within the corpus was devoid of spelling errors, rendering them suitable for representing Sidaamu afoo words, as indicated in the aforementioned Figure 4.1.

The meticulously prepared words were compiled into a dictionary format, serving as a repository of validated and corrected words. Simultaneously, the corpus was utilized to construct a bigram model aimed at identifying and rectifying real-word errors.

The dictionary functions as a compilation of accurate words, aiding in the identification and rectification of misspelled words attributable to non-word errors. As explained in section 2.4.1, the size of the dictionary can influence response time, as it involves cross-referencing user words against the entirety of dictionary entries, as depicted in Figure 4.4.

Conversely, the bigram model comprises pairs of words arranged in sequence, representing contextual associations within the Sidaamu afoo writing system. This model is instrumental in pinpointing and correcting real-word errors within user-preprocessed text.

4.3.1 PREPROCESSING

To facilitate the detection of user words, as well as the creation of a dictionary and the computation of bigrams, it becomes essential to partition sentences into smaller units referred to as tokens (Asghar et al., 2013). This initial module within the framework serves to segment sentences into tokens or words, concurrently identifying typographical errors—referred to as focal words—through dictionary comparison (Kundi, et al., 2014).

Recognizing the boundaries of words is a pivotal aspect of the tokenization process (Jurish, 2013)According to the insights presented by (Tesema W., 2017),the Sidaamu afoo writing system boasts its own distinctive word boundaries, with sentence boundaries in Sidaamu afoo sharing similarities with those in the English language. In Sidaamu afoo, spaces denote the conclusion of a word, and additional indicators such as exclamation points (!), periods (.), question marks (?), parentheses (), and quotation marks (‘’) are employed to delineate word boundaries.

As depicted in Figure 4.2, the process involved in splitting a Sidaamu afoo sentence utilized a tokenization algorithm facilitated by Python’s built-in String split function. To illustrate, if we consider the sentence ' kaaliiqi mannasi baxanno ' within our corpus, this sentence would be broken down into a collection of words, including ' kaaliiqi ', ' mannasi ', and ' baxanno '.

```
Input : Corpus (C)
Output : Token (T)

Begin

Step 1: Parse input Corpus(C);
Step 2: For input C;

    Extract word(EWI), where word delimit for Sidaamu Afoo (space,.,!,?) and i=1,2,3,---,n;
Step 3:
      for each EWI;
      frequency_count(WCI)=EWI;
   // for the total number of existence(occurrence) of each word.
    Return (WCI)
Step 4:
   Token (EWI) =Ti Return(TI)
End
```

Figure 4.2:Tokenized Algorithm

On the flip side, the normalizer component primarily focuses on dissecting the words within a document into individual constituents, typically by taking into account spaces and punctuation marks (where the use of punctuation marks in Sidaamu afoo resembles that of English) (Seretan, 2004). Additionally, text normalization aids in rectifying the issue of capitalization discrepancies in the given words. To automatically identify misspelled words, the words are converted to lowercase. For example, if users input "hawALLE kEEre daggiNi," the model automatically normalizes it to " hawalle keere daggini." Moreover, non-standard words like "Xuma35" are also normalized to "Xuma," and this aspect was managed using Python's Pycld2.

Input : Input Sidaamu Afoo words (SWi)
Output : Processed words (PWi)

Begin
For each SWi
Do:

Substitute any sequence of two or more consecutive punctuation characters with the initial punctuation character.

Transform each instance of consecutive multiple whitespace characters into a solitary whitespace character.
If the given term, SWi, includes a number or digit, exclude the numerical component.

Convert every appearance of a capitalized letter to its corresponding lowercase form.
Iterate through the following steps until the input text is fully normalized
End

Figure 4.3:Normalized Algorithm

4.4 ERROR DETECTION ALGORITHM

As elaborated upon in section 2.4, the process of error detection typically involves assessing whether an input string corresponds to a valid entry within a dictionary (in the case of non-word errors) and verifying the availability of bigram combinations (in the context of contextual errors). Instances where the input string is neither valid nor available result in the model categorizing them as incorrect words, subsequently flagging them as misspelled words for the user's attention. As

illustrated in Figure 4.1, the technique of dictionary lookup was utilized to identify non-word errors, while the analysis of bigram combinations was employed to detect errors involving real words within the user's input.

4.4.1 DICTIONARY LOOKUP ALGORITHM

As outlined in section 2.4.1, the technique of dictionary lookup is employed to discern words absent within the lexicon entries (termed non-word errors) within the user's input. The process involves subjecting preprocessed user words to the dictionary lookup method to identify instances of non-word errors, as depicted in Figure 4.1*:Context Based Spell Checker for Sidaamu afoo Architecture*.

In line with Figure 4.4*: Dictionary lookup method*, the dictionary lookup mechanism involves a comparison between each token (word) within the user's input and the words present in the dictionary containing correctly spelled terms. Tokens that find matches among the dictionary's elements are considered accurate words, while tokens that fail to match any entry in the dictionary are flagged as misspelled words.

In the execution of dictionary lookup, the utilization of a Hash function was adopted in this research. As elaborated upon in section 2.4.1, the preference for a Hash function stems from its capacity to reduce the response time in dictionary lookup procedures. Consequently, for the purpose of this study, the choice was made to employ a dictionary lookup method integrated with the Python function, as illustrated in Figure 4.5.

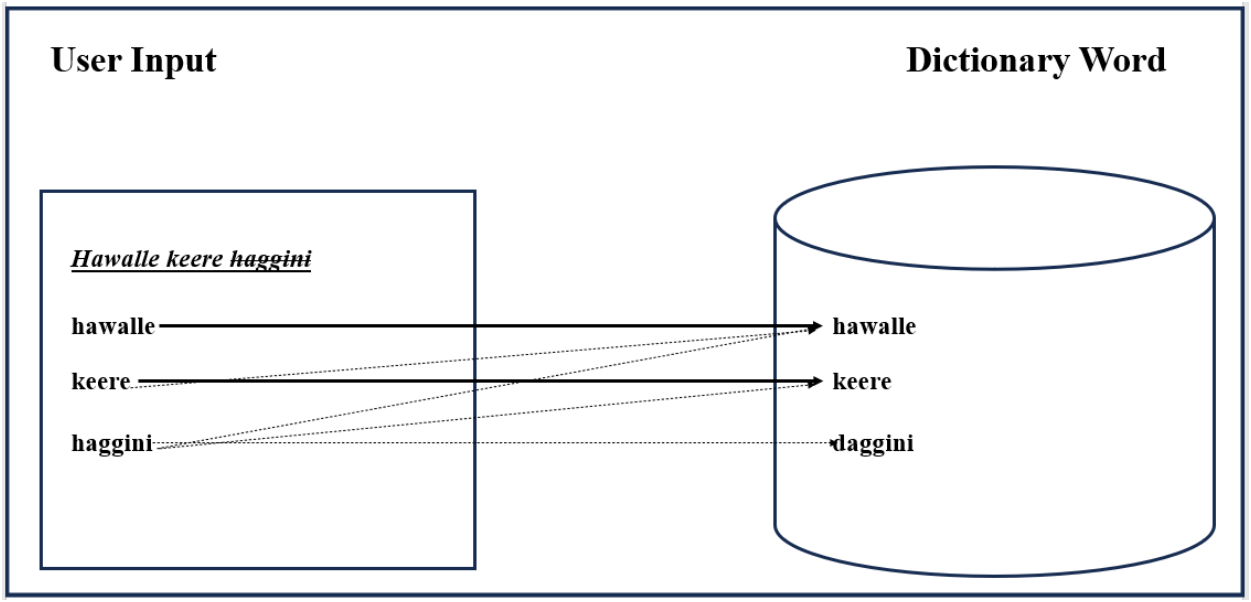


Figure 4.4: Dictionary lookup method

In the context of the depicted Figure 4.4, the terms "hawalle," "keere," and "haggini" are categorized as user-provided words, whereas "hawalle," "keere," and "daggini" belong to the list of corrected dictionary words. Through dictionary lookup, the system identifies that "hawalle" and "keere" are accurate terms, whereas "haggini" is deemed a misspelled word. Based on the example strikethrough in the aforementioned figure, the bold line signifies the alignment between user words and dictionary words, while the dotted line indicates that the user word did not correspond with dictionary words. Consequently, the model identifies these discrepancies as misspelled words.

Input : (1) Sidaamu Afoo Target Word(STW): (2) Hash table
Output : (1) Available Word(AW): Not_Available_Word(NAW)

Step 1: Set primary Hash Table Information
Xi:address, where i=1,2,------n
Step 2: Examine Xi:
for Xi>0 then primary index=X1
if prefix(token at primary_index in Hash Table) = Prefix(STW)
then return AW
else
Xi – iterated_index=Xi
if prefix(token at iterated_index in Hash Table)=Prefix(STW)
then return AW
Step 3: if no match was found at step 2 return NAW
End

Figure 4.5: Dictionary Lookup Algorithm

4.4.2 BIGRAM ANALYSIS ALGORITHM

The objective of real-word error detection is to pinpoint errors that arise when a user inadvertently types one correctly spelled word when another was intended, as discussed by (Hirst G. a., 2003). As elucidated in section 2.6, our approach to identifying real-word errors within sentences involves analyzing the context of words both before and after the target word. For instance, in the sentence 'She can’t bare the thought of leaving,' instead of 'She can’t bear the thought of leaving,' the word 'bare' is erroneously flagged as incorrect, even though it exists in the English lexicon.

In this study, we have employed Bigram statistics frequency count (as depicted in equation 4.1) to facilitate the detection of real-word errors. This technique involves tallying the occurrences of word bigrams appearing together to differentiate misspelled words from correctly spelled ones. If a given word combination does not occur, the model identifies it as a real-word error.

The formula for calculating the Bigram of a sequence, denoted as $Bigram(W1W2W3)$, is defined as the product of consecutive word pairs, such as (W1W2)(W2W3).

For instance, consider the sentence " kaaliiqi mannasi baxanno gara." In this sentence, the frequencies of various bigrams are as follows: frequency (kaaliiqi mannasi) = 1, frequency (mannasi baxanno) = 1, frequency (baxanno gara) = 1. These frequencies are determined based on the occurrences of the corresponding word pairs within the sentence. When the frequency of a specific bigram is zero, it indicates that the associated word is a real-word error within the given context.

4.5 ERROR CORRECTION ALGORITHM

Error correction involves the process of rectifying an identified error, as outlined by (Hirst G. a., 2003) Upon identifying a string as erroneous, an error correction algorithm is employed to generate potential corrections for the inaccurate word. Typically, the correction mechanism utilized in this research consists of employing the Levenshtein edit distance for addressing non-word errors, and the Bigram probability for addressing real-word errors, as illustrated in Figure 4.1.

4.5.1 LEVENSHTEIN EDIT DISTANCE

As explained in section 2.3.1, the Levenshtein edit distance method is utilized to rectify non-word errors. In the context of this study, the Levenshtein algorithm was implemented to determine the minimum number of operations required, encompassing insertion, deletion, and substitution. These operations are undertaken to identify potential corrections for misspelled words, as demonstrated in Figure 4.6. An insertion operation entails adding a letter to a misspelled word to yield a correctly spelled word. Conversely, a deletion operation involves removing a letter from a misspelled word to arrive at a correctly spelled word. Substitution denotes replacing a letter in the erroneous word with the appropriate letter to result in a correctly spelled word.

```
Input : Sidaamu Afoo word1(SW1), Sidaamu Afoo word2(SW2)
Output : Edit operation Number

Step 1: Declaration
        distance(length of SW1,SW2) =0, min1=0, min2, min3=0, cost=0
Step 2: Calculate the distance
     if SW1 is NULL return Length of SW2
     if SW2 is NULL return Length of SW1
      for each symbol x in SW1 do
          for each symbol y in SW2 do
Begin
   if X=Y  cost=0
       else
  cost=1
  r=index of x, c=index of y
  min1 = (distance(r-1,c) +1 //deletion of character
  min 2= (distance(r,c-1) + 1)// insertion of character
  min 3= (distance(r-1,c-1) +cost)// Substitution of character
   distance(r, c) = minimum(m1,m3,m3)
End
Step 3: Return the value of the cell in the matrix
Return distance(Length of SW1 and SW2)
End
```

Figure 4.6: Levenshtein Algorithm

As depicted in the aforementioned Figure 4.6, the (minimal) edit distance between two strings, namely s1 and s2, signifies the smallest count of fundamental operations required to transform s1 into s2. Fundamentally, within the model, there exist two strings: the provided word (an error-ridden term) and the corrected word (sourced from the dictionary). In the model's context, the algorithm is tasked with transitioning the provided word to match the dictionary word. To illustrate, consider s1="xoma," representing a word misspelled by the user and requiring correction. Conversely, s2="xuma" represents the accurate word sourced from the dictionary.

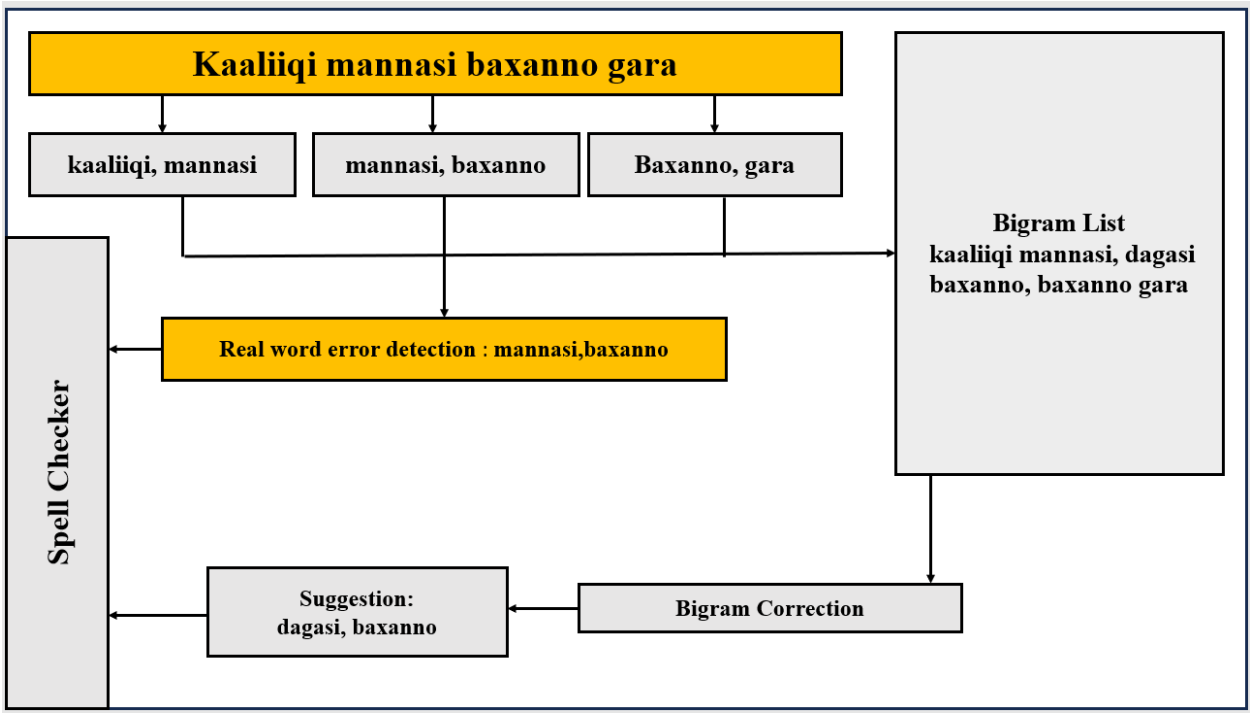


Figure 4.7: Bigram word detection and correction

4.5.2 BIGRAM PROBABILITY ALGORITHM

Alternatively, genuine spelling errors within words can be rectified through the utilization of a bigram language model. In this study, a bigram word probability derived from the amassed corpus was implemented to address real-word errors.

In this context, the count (w1) denotes the frequency of occurrences of the term w1 within the corpus, while count (w1; w2) represents the instances where the term w2 immediately follows w1. The bigram word probability, represented as $P(W2/W1)$, is computed using the formula:

$P(W2/W1) = count\ (w1, w2)/count(w1)$

This formulation captures the probability of word w2 occurring following word w1, derived from the collected corpus.

As illustrated in figure 4.7, the user inputted the phrase ‘kaaliiqi mannasi baxanno gara’. However, the bigram module prepared from the corpus comprises the sequence mannasi, baxanno '. Regrettably, ' mannasi ' is identified as an error, despite being a correct Sidaamu afoo word. This

discrepancy arises due to the limitations of the corpus. When we apply equation 4.2 as outlined above, the resulting outcomes are as follows:

P(mannasi/baxanno) =1(mannasi,baxanno)/2(mannasi)= 0.5

4.6 CANDIDATE RANKING

Candidates refer to tokens exhibiting a significant resemblance to the incorrect word, a concept explored by (Ratnasari,, Kusumadewi,, & Rosita,, 2017). When addressing non-word errors, the candidate list is derived from the stored dictionary. This process involves gauging the similarity between the misspelled word and a compiled list of correctly spelled words.

Conversely, for real-word errors, the candidate token is the one deemed most probable to have been intended. The candidate list is generated through a bigram model, which is established from the collected corpus. The generation hinges on calculating the likelihood of preceding words occurring after the initial word's entry.

CHAPTER FIVE
EXPERIMENTATION AND EVALUATION

5.1 INTRODUCTION

In the preceding chapter, an effort was undertaken to outline the blueprint for a Context-based spell checker model tailored for Sidaamu afoo writing. This chapter delves into the practical implementation of the tools and algorithms detailed in the previous section, aimed at constructing the model. Subsequently, an experiment was executed to showcase the precision of spelling error detection and correction. The outcomes of this experiment are expounded upon in this section, allowing for an interpretation of the achieved results. The efficacy of the spelling error detection and correction can be gauged through a designated evaluation approach. Precision and recall metrics were employed to assess the accuracy, efficiency, and reliability of the spelling error detection and correction process. These metrics were applied to the training and testing texts utilized within this experiment.

As outlined in section 1.4, a key goal of this project is to create a prototype capable of identifying misspelled words among the user-input words. To realize this objective and conduct the necessary experiments on the system, the latest version of Python was employed. The resultant interfaces, designed for both detecting misspelled words and suggesting potential corrections in Sidaamu afoo, encompass a variety of user interface elements essential for fulfilling the objectives of this research. These elements are visually depicted in Figure 5.1.

5.2 TRAINED DATA

The construction of the corpus involved the expertise of linguistic professionals, following the approach outlined by Vincze et al. (2008) to capture spelling intricacies. Similarly, in the case of Sidaamu afoo, we meticulously curated an error-free corpus. This process was undertaken in collaboration with linguistic experts well-versed in the nuances of Sidaamu afoo spelling features. The aim was to optimize the model's accuracy and performance by crafting coherent and authentic words from Sidaamu afoo vocabulary. Considering that the gathered corpus contained superfluous characters and words, it became imperative to engage in text preprocessing to purify the corpus and enhance its clarity.

Furthermore, the corpus that was gathered served as the foundation for compiling a dictionary. This dictionary encompasses a comprehensive list of 57,881 Sidaamu afoo words, thoughtfully organized in alphabetical order. Additionally, the dictionary comprises 291,941 bigram words derived from the collected corpus. It's noteworthy that these bigrams were generated at the level of complete words, as opposed to individual characters. This approach is particularly beneficial for identifying and rectifying genuine word-level errors, as illustrated in Table 5.1.

Table 5.1: Top ten most frequently occurring bigram from the corpus

No	**Bigram words**	**Frequency**	**Probability**
1	Hakkunni gedensaanni	2,314	0.26%
2	Lowo geeshsha	1,392	0.16%
3	Konne coye	1,378	0.16%
4	Hatte yannara	1,142	0.13%
5	Israeelete manni	780	0.09%
6	Kunni garinni	680	0.08%
7	Baattote aana	624	0.07%
8	Qullaawa mine	526	0.06%
9	Gede assi	518	0.06%
10	Qullaawu mini	464	0.05%

Table 5.2: Top ten most frequently occurring unigram from the corpus

No	Unigram words	Frequency	Probability
1	Gede	11714	1.32%
2	Kaaliiqi	8534	0.96%
3	Kayinni	5764	0.65%
4	Daafira	5012	0.57%
5	Ani	4720	0.53%
6	Manni	4468	0.50%
7	Baala	4438	0.50%
8	Maganu	4404	0.50%
9	Geeshsha	4326	0.49%
10	Ledo	4278	0.48%

5.3 EXPERIMENT AND RESULT

As indicated in section 1.6, an experimental phase was undertaken within this study to gauge the precision of the Context-based spell checker for Sidaamu afoo. This section outlines the process of data collection for testing purposes and presents the results yielded by the conducted experiment. The interface of the Context-based Spell Checker for Sidaamu afoo (CBSCSA) comprises three distinct components:

1. **Input Area**: This area serves as a space for entering desired words using an input method, with a keyboard being the chosen method in this case.

2. **Button**: A button is included to initiate the verification process for determining the correctness of the entered words.

3. **Mouse Pointer**: This component comes into play when the user clicks on misspelled words. It provides potential candidates for correction, aiding in the correction process. This functionality is illustrated in Figure 5.1.

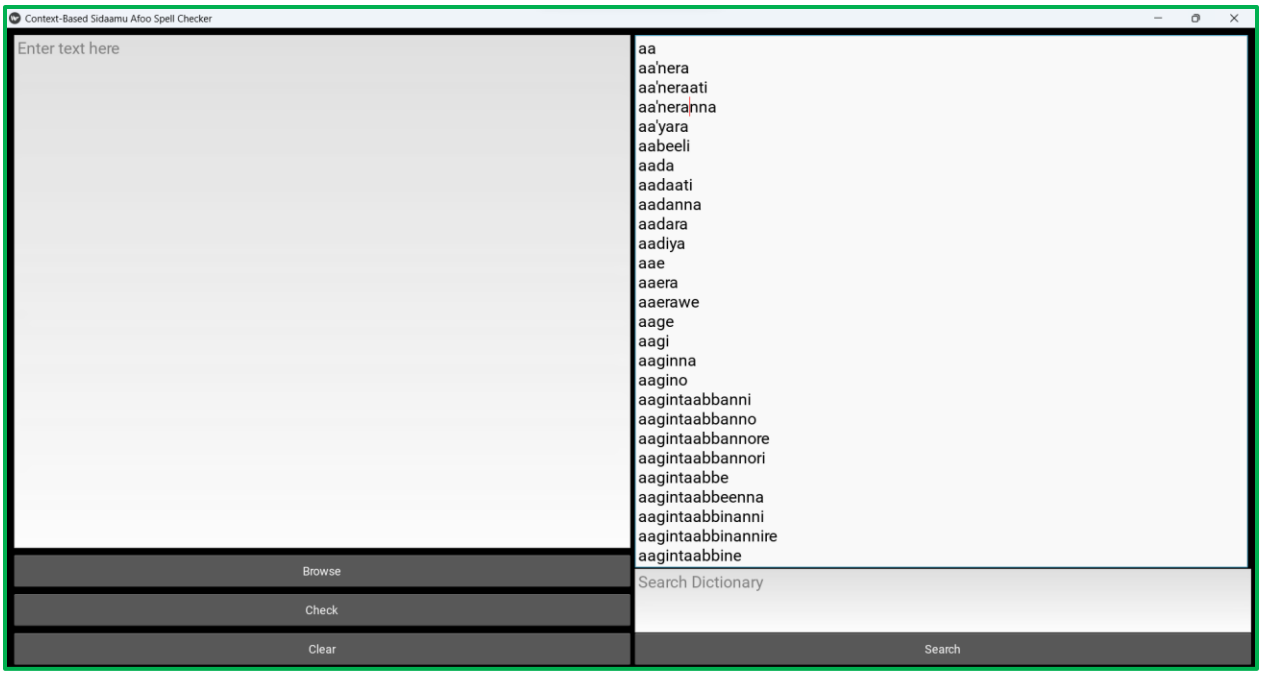


Figure 5.1: Context Based Sidaamu afoo Spell Checker

The CBSCSA model includes a text area where users can input text directly. This interactive model waits for users to click the "Check" button to detect errors. The error detection module is responsible for preprocessing and comparing the entered words with a dictionary and a bigram model. Any words not found in the dictionary are considered misspelled, while others are left unchanged. As mentioned in section 2.4, error detection is carried out through two methods: the first involves dictionary lookup for nonwords, and the second involves bigram analysis for real words.

5.3.1 ERROR DETECTION

The model employs a word-level error detection method to identify non-word spelling errors that are not present in the dictionary, as detailed in section 2.4.1. In the user's input, any word that isn't found in the dictionary is flagged as a misspelled word and font color in red. Conversely, words found in the dictionary are considered correct. As illustrated in Figure 5.2, words like 'geede,' ' kaliiqi,' 'kayini,' 'dafira,' 'anni,' 'mani,' 'baalla,' 'kone,' ' qulawu,' 'womanka,' and 'gedensanni' are identified as misspelled words in the sample test because they are not present in the prepared dictionary.

For instance, 'womanka' is detected as a misspelling because apostrophe '’' has been missed in the middle of 'wo,' and ‘ma’ instead of 'wo'manka,' which means 'Whole.' The model categorizes words not found in the dictionary as non-word errors and highlights them accordingly.

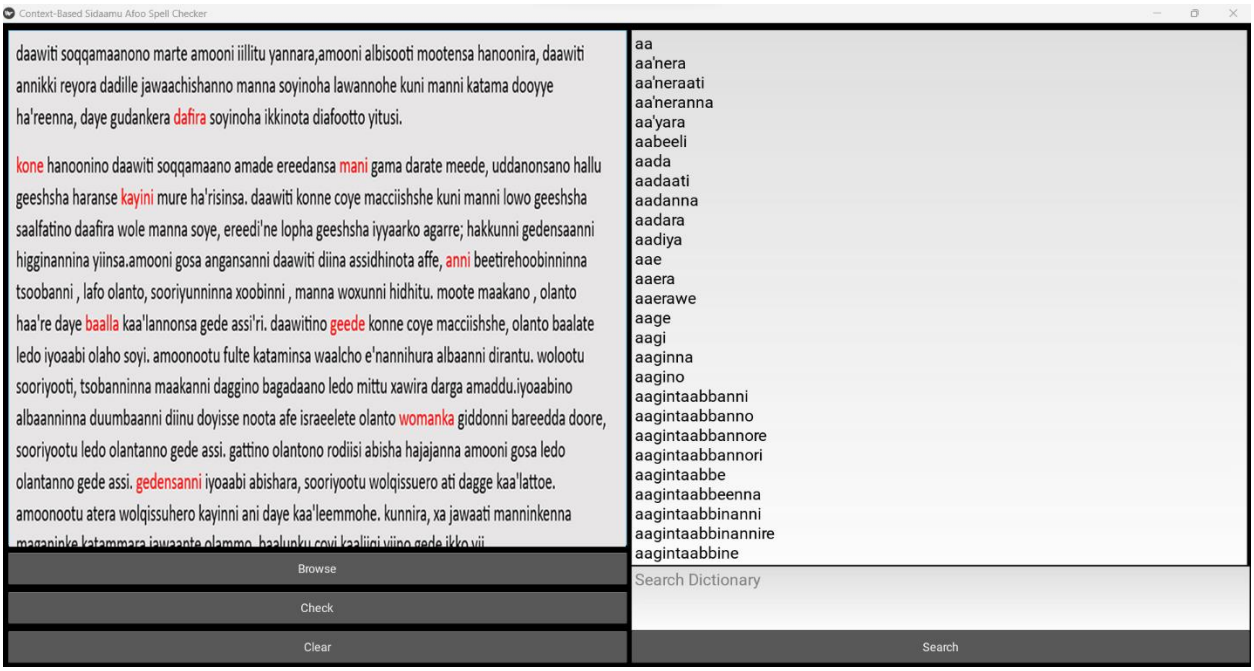


Figure 5.2: Non-word error detection

Conversely, real word errors are detected by examining the sequences of bigram words that appear together. If a sequence of bigram words is not found in the bigram list, it is identified as a real word error. The input sentences are broken down into bigrams, and for each bigram, the associated probability information is generated. This information is used to rank candidate suggestions for correcting errors. When all the bigram words are found in the bigram model, there is no error, and the words are considered valid. However, if at least one of the words is not present in the bigram word list, it is marked as a misspelled word. The model changes these misspelled word fonts in to green and displays them, as shown in Figure 5.3 below.

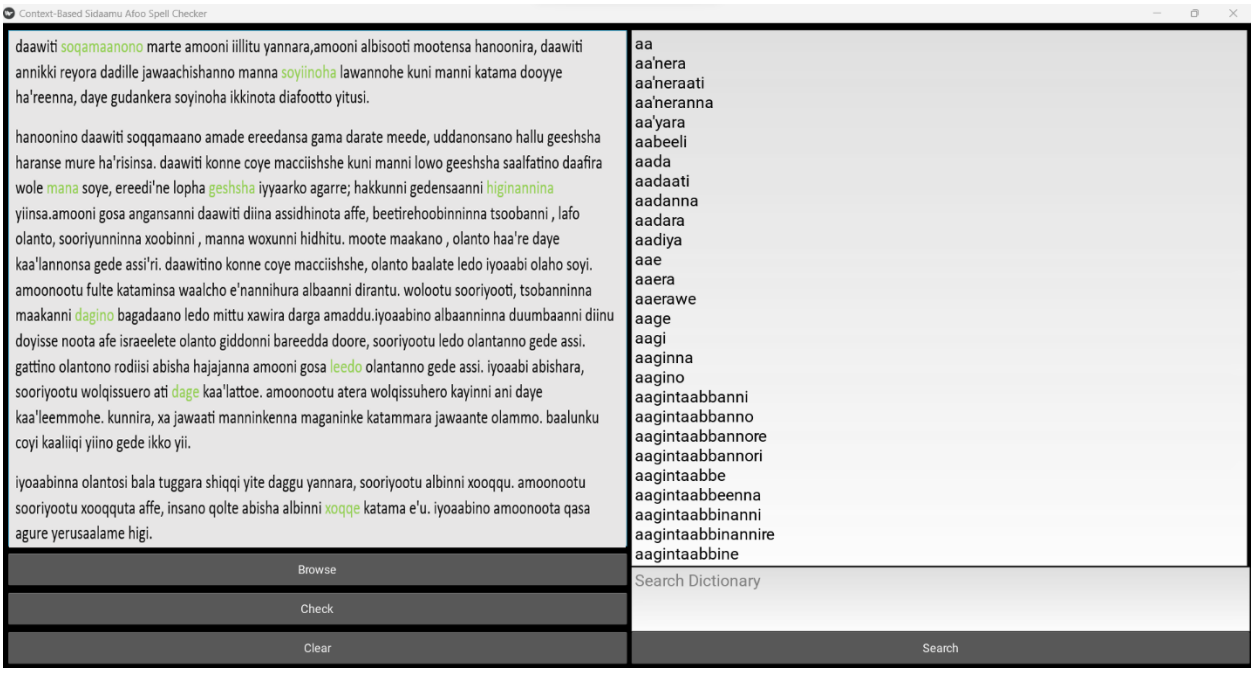


Figure 5.3: Real-word error detection

As depicted in Figure 5.3 above, the model identifies 9 words as misspelled. Take, for example, the word ' mana,' which is flagged as an error. This determination is made by analyzing the bigram of ‘wole mana and soye’. In context, 'mana' does not fit with the previously mentioned words. In Sidaamu afoo, ' wole mana soye ' is not a coherent phrase, whereas ‘wole manna soye’ is contextually accurate. Consequently, the model identifies ‘mana’ as a real word error based on its learning from the collected corpus.

5.3.2 ERROR CORRECTION

To rectify misspelled words, the correction algorithm offers a range of potential candidate corrections, as illustrated in Figure 5.4 below. When a word is identified as misspelled, the algorithm presents a set of suggestions to the user. In the case of non-word errors, the Levenshtein edit distance is employed to generate suggestions for misspelled words. The algorithm calculates the minimum edit distance and ranks these suggestions accordingly.

The correction module utilizes a dictionary list to offer spelling suggestions for words flagged as misspelled in the input. These errors are rectified and altered using the suggested words presented in a popup menu. In CBSCSA, the model provides a maximum of three candidate lists based on

the computed distance values between them. Furthermore, if the user is aware of the misspelled words, the model allows for the inclusion of these words as corrected entries in the dictionary.

5.4 EVALUATION

Olson and Delen (2008) assert that Precision, Recall, and F-measurement are commonly employed fundamental statistical metrics extensively utilized for gauging the efficiency of spell checking. This section involves an evaluation of the spell-checking application developed within this study to assess its performance. The outcomes of the ongoing research can be dissected into two distinct components: the effectiveness of the dictionary-based spell checker and the effectiveness of the context-aware spell checker.

The evaluation method employs four distinct categories, namely true positive (TP), false positive (FP), false negative (FN), and true negative (TN). TP represents valid words correctly identified by the spell checker, leading to accurate non-flagging. TN signifies invalid words correctly recognized by the spell checker, resulting in proper flags (flagged words). FN occurs when valid words go unrecognized by the spell checker, resulting in incorrect flags (erroneously flagged words). FP pertains to invalid words not detected by the spell checker, causing incorrect non-flagging (also referred to as missed flags). The performance of the developed spell checker was assessed using evaluation metrics for both non-word and real-word spelling errors in written texts. Additionally, Error Precision (EP) is another metric used to gauge the effectiveness of a spell checker in identifying valid words that the system fails to recognize because they are absent from the training set but present in the test set. To assess the system's performance, researchers compiled a testing dataset to evaluate the model. This dataset was created by collecting text from various sources and introducing artificial spelling errors into the test set. These errors were then marked to assess the system's efficiency. The dataset consisted of **37,362** correctly spelled words and **2262** misspelled words. In this sample, **35,052** words were correctly accepted as valid, while **2262** words were flagged as misspelled by the system due to their absence from the dictionary. On the other hand, for real-word errors, **39,624** words were prepared, out of which **38,130** were spelled correctly, and **1494** were misspelled real words. Among the sample test data, **34,764** were correctly identified as valid words, while **3366** were not detected by the system. Typically, the test results are summarized in a table format.

Table 5.3: Experimental results of Sidaamu afoo spell checker

Error	TP	TN	FN	R	EP	P	Accuracy
Non-word	35052	2262	2310	0.938	0.495	1	0.94
Real-word	34764	1494	3366	0.91	0.30	1	0.91
Average	-	-	-	0.924	0.40	1	0.925

5.5 DISCUSSION

The CBSCSA system has demonstrated remarkable proficiency in identifying, rectifying, and offering alternative suggestions for both non-word spelling errors and real word errors. According to the experimental findings, the system's test predictions resulted in 35,052 corrected words for non-word errors and 38,130 corrected words for real word errors. Additionally, the system successfully identified 2262 misspelled non-words and 1494 instances of real word errors. As a result, the system achieved an impressive performance accuracy of 92.5% for correcting both types of errors.

The outcome underscores the model's efficacy in effectively addressing non-word errors and real word errors. The evaluation of the model's coverage of corrected words was gauged by recall, resulting in a value of 92.4%. Conversely, the model's inability to recognize valid words scored at 49%, primarily attributed to the limited size of the dictionary. This suggests that the model would benefit from expanded data to bolster recall scores and mitigate error precision.

Remarkably, the model impeccably identified all misspelled words, achieving a perfect score of 100% according to the presented experiment. In line with the study by Attia et al. (2012), which constructed a dictionary for non-word errors and employed a bigram model for real word errors, yielding an 82.86% accuracy rate, the performance of CBSCAO was even more robust.

Conversely, the system's performance in detecting real word errors reached 91%, influenced by the corpus size used in training text. This finding reveals that the training corpus didn't encompass all words present in the testing data, affecting the accuracy due to the sequence of words in sentences, which relies on generated bigrams. Thus, augmenting the training corpus is essential for enhancing the system's efficiency in identifying spelling errors.

The evaluation of the model's corrected Sidaamu afoo word coverage was determined by recall, yielding a value of 91%. However, an error precision rate of 49% was observed from the sample test set employed for model learning. This underscores the necessity for enhancing the prototype system, as real word errors were often linked to the arrangement of words in sentences. Notably, all misspelled words in the tested data were comprehensively addressed, with the system adeptly detecting all errors not accounted for in the bigram lists.

The outcomes of the current study substantiate that the utilization of a bigram probabilistic approach yields favorable results in rectifying real word misspellings. This conclusion is echoed by a study conducted by (Ghotit , 2011), which also demonstrated that approximately 97% of real words were successfully corrected using the bigram probabilistic method.

The assessment of the spellchecker's performance, as presented in Table 5.4, underscores the optimistic nature of the Context-based spellchecker for Sidaamu afoo. This system serves as a promising solution for addressing misspelled words, boasting an impressive accuracy score of 92.5% in rectifying both non-word and real word errors.

CHAPTER SIX

CONCLUSION AND RECOMMENDATION

This thesis has showcased the progress and empirical outcomes of context-based spell checking for the Sidaamu afoo language. In the preceding sections of this document, we delved into the theoretical underpinnings of spell checking and, as an essential step in building our model, outlined the creation of the Sidaamu afoo text corpus. Furthermore, we detailed the establishment of a dictionary and a bigram model derived from this corpus. The focal point of our accomplishments lies in the experimental outcomes of our efforts, which are elucidated in chapter 5. Within this chapter, we encapsulate our final reflections and provide recommendations for prospective endeavors in this field.

6.1 CONCLUSION

The overarching objective of Natural Language Processing research is to analyze and comprehend language, a goal that remains incompletely attained. Consequently, NLP research has emphasized intermediary tasks that extract meaning from certain inherent language structures, all the while not demanding comprehensive understanding. Among these tasks, one notable example is spelling correction.

Spelling checking and correction have seamlessly integrated into the daily routines of today's generation. Whether engaged with text editing tools like MS Word or composing text messages on mobile devices, the process inevitably involves spell checking and correction. This study is dedicated to the design, development, and testing of a context-based spell checker for Sidaamu afoo, addressing both non-word and real-world errors. The research employs two distinct approaches for spelling correction: the dictionary lookup method and the N-gram method. The dictionary lookup approach identifies and rectifies non-word errors by cross-referencing with a dictionary. Conversely, the N-gram approach is tailored for real-word errors, relying on quality training data. This statistical approach aids in identifying errors in actual words. Both methodologies adeptly manage errors in both the non-word and real-word domains within the Sidaamu afoo writing system.

The context-based spell checker developed for Sidaamu afoo exhibited a performance of 92.5% accuracy, as demonstrated by the results. This outcome underscores the system's proficiency in identifying and rectifying misspelled words. However, with regard to real-word errors, the system displayed 91.3% accuracy and 91% recall, indicating the need for further enhancement in its performance.

The Sidaamu afoo Spell Checking model, although effective, does not recognize compound words as individual words. Instead, it identifies them as separate words through word splitting. Additionally, it does not treat capitalized abbreviations and words containing numbers in the same manner as Microsoft Word (MS) does. In MS, every abbreviation with capitalized letters and words containing numbers are accepted as correct, even if not found in the English dictionary. For instance, in MS, 'ask1' is considered a correct word, regardless of its presence in the dictionary. In contrast, within the CBSSA model, 'keere1' is flagged as a misspelled word because numbers are removed before verification.

6.2 RECOMMENDATION

The outcomes of this study effectively address the issue of misspelling when users compose Sidaamu afoo text. Nonetheless, the potential for further enhancement lies in the amalgamation of various research domains. This segment provides a concise rundown of potential areas for improvement in this research endeavor.

Drawing from the discoveries of this investigation, the subsequent research pathways are recommended for future exploration:

- Based on the conducted experiments in this study, it is evident that the absence of a supervised approach diminishes the effectiveness of real-word spell checking. To enhance the accuracy of identifying and rectifying real-word spelling errors, there arises a requirement for the incorporation of supervised methodologies. These supervised approaches should be employed through the integration of annotated and tagged texts, combining dictionary-based techniques with N-gram analysis.
- Accumulating a substantial corpus holds significant potential for elevating the efficiency of the spelling checker, particularly in furnishing optimal candidate suggestions for real-

word spelling errors. Therefore, a crucial undertaking for the future involves the assembly of a substantial and superior-quality corpus. Additionally, the establishment of a comprehensive standard dictionary encompassing a wide array of words is of paramount importance to amplify the accuracy of the spell checker.

- Another avenue for enhancing performance could involve harnessing word probability information beyond the scope of a bigram, such as utilizing trigrams. This could serve as a means to effectively rectify real-word errors and provide highly probable suggestions.
- The distance metrics employed in this model exclusively facilitate the correction of individual errors, encompassing deletion, insertion, and substitution. In order to accommodate multiple errors, there is a necessity to adapt and revise the distance metrics. This adaptation is crucial for generating optimal suggestions to rectify misspelled words effectively.
- Moreover, the CBSSA model is primarily centered on the tokenization and normalization preprocessing procedures. Nevertheless, to further optimize space utilization during dictionary creation, an additional preprocessing step is required – stemming. This step aims to capture the root form of words, facilitating efficient storage.
- Finally, future enhancements could explore advanced neural architectures to improve model quality further.

REFERENCES

Achenkunju, A. a. (2014). An Efficient Refromulated Model for Transformation of . *International Journal of Engineering Research and Applications (IJERA)*, 2248-9622.

Addisu, A. B. (2016). Development of part of speech tagger for sidaama language.

Ahmad, F. a. (2005). Learning a spelling error model from search query logs. *In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 955-962.

Ahmed,, F., Ernesto,, W., De,, L., & Andreas,, N. (2009). *Revised N-gram based Automatic Spelling Correction Tool to Improve Retrieval Effectiveness.* Berlin: Technical University of Berlin.

Al-bakry, P. A. (2014). Enhanced Levenshtein Edit Distance Method functioning as a String-toString Similarity Measure. *Enhanced Levenshtein Edit Distance Method functioning as a String-toString Similarity Measure*, 48–54.

Allein, L. L. (2016). *The Politics of Ethinicity in Ethiopia.* BRILL.

Baik, H. S. (2006). Estimating transition probabilities in Markov chain-based deterioration models for management of wastewater systems. *Journal of water resources planning and management*, 15-24.

Baldwin, T. H. (2011.). Lexical Normalisation of Short Text Messages. *Makn Sens a # twitter*, 368–378.

Bassil, Y. Y. (2012). Parallel spell-checking algorithm based on Yahoo! N-grams dataset. *arXiv preprint arXiv:1204.0184.*

Bassil,, Y., & Semaan,, P. (2012). ASR Context-Sensitive Error Correction Based on Microsoft NGram Dataset,. *ASR*, 4(1), 34–42.

Blunsom, P. (2022, August 19). *Hidden Markov Models*. Retrieved from pcbl@cs.mu.oz.au.

Borah, B. (2017). *For Query-Based Text Summarization*. Retrieved from https://doi.org/10.1007/978-3-319-62407-5

Brill, E. a. (2000). An improved error model for noisy channel spelling correction. *In Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, 286-293.

Budanitsky A. Hirst G, a. B. (2003). Correcting real-word spelling errors by restoring lexical cohesion.

Christen, P. (. (2012). *A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication.*

Cohen, J. D. (2001). *U.S. Patent No. 6,169,969. Washington, DC: U.S. Patent and Trademark Office Cucerzan.*

Cucerzan,, S., & Brill,, E. (2002). *Spelling correction as an iterative process that exploits the collective knowledge of web users.*

Daiber,, J., Jakob,, M., Hokamp,, C., & Mendes,, P. (2013). Improving efficiency and accuracy in. *In Proceedings of the 9th International Conference on Semantic Systems*, 121-124).

Debela. (2011.). A rule-based Afaan Oromo Grammar Checker,. *International Journal of Advanced Computer Science and Applications, , CA , , ,* Vol. 2, No. 8 pp. 126–130.

Deksne, D. a. (2011). *CFG Based Grammar Checker for Latvian.*

Dickinson, B. (2013). *Developing A Real-Word Spelling Corrector.*

Evermann, G. a. (2000). Posterior probability decoding, confidence estimation and system combination. *In Proc. Speech Transcription Workshop*, 78-81.

F. Ahmed, E. W. (2020). *Revised N–Gram based Automatic Spelling Correction Tool to Improve Retrieval Effectiveness.* Otto–von–Guericke ,Germany: Institute for knw and Lang Engi., Univ. of Magdeburg.

Flor,, M., & Futagi,, Y. (2012). On using context for automatic correction of non-word misspellings. *On using context for automatic correction of non-word misspellings*, 105–115.

G., N. (2006). *Pratisthamathur Spell checking techniques in NLP: a survey, Department of computer science.* Banasthalividyapith, India.

Gersten,, R., Fuchs,, L., Coyne,, M., & Greenwood,, C. (2005). *Quality Indicators for Group Experimental and Quasi-Experimental Research in Special Education.*

Getnet, A. (June, 2018.). *Automatic Amharic Spelling Error Detection and Correction using hybrid approach.* Patiala, India: Dept. Comp. Sci. Punjabi Univ., , Volume 5, Issue 6, ISSN-2349-5162 .

Ghotit , l. (2011). Ghotit Dyslexia Contextual Spell Checker.Retrieved from.

Grinstead, C. M. (2012). *Introduction to probability. American Mathematical Soc.*

Han,, B., & Baldwin,, T. (2011). Lexical Normalisation of Short Text Messages . *Makn Sens a # twitter*, 368–378.

Henok, H. D. (2022). Context Based Afaan Oromo Language Spell Checker For Handheld devices.

Hilas, C. S. (2022, July 11). *Knowledge-Based Systems An application of supervised and unsupervised learning approaches to telecommunications fraud detection q, 21,* . Retrieved from https://doi.org/10.1016/j.knosys.2008.03.026

Hindle,, A., Barr,, E., Su,, Z., Gabel,, M., & Devanbu,, P. (2012). On the naturalness of software,In Software Engineering (ICSE). *34th International Conference on*, 837-847.

Hirst, G. a. (2003). Correcting real-word spelling errors by restoring lexical cohesion.

Hirst, G. a. (2003). Correcting real-word spelling errors by restoring lexical cohesion.

Inkpen D, I. D. (2009). Real-Word Spelling Correction using Google Web. *1T 3-grams*, 1241–1249.

Inkpen, D. D. (2009). Real-Word Spelling Correction using Google Web 1T 3-grams. *Real-Word Spelling Correction using Google Web 1T 3-grams*, 1241–1249.

Jurafsky, D. a. (2009). *An introduction to natural language processing, computational linguistics, and speech recognition*, 1-1024.

Jurish, B. a. (2013). Word and Sentence Tokenization with Hidden Markov Models. *JLCL*, 61-83.

Kaur N. Mishra, R. a. (2013). A Survey of Spelling Error Detection and Correction Techniques. *International Journal of Computer Trends and Technology*, 372-374.

Kawachi, K. (2007). *A Grammar of Sidaama(Sidamo) A Cushitic Language of Ethiopia.* Buffalo: UMI Microform.

Kawachi, K. z. (2007). *A Grammar Of Sidaama (Sidamo), A Cushitic Language Of Ethiopia.*

Kirthi J., N. N. (n.d.). *Automatic spell correction of User query with semantic Information Retrieval and Ranking of search Result using WordNetApproach.*

Koehn,, P., Birch,, A., Callison-burch,, C., Federico,, M., Bertoldi,, N., Cowan,, B., & Herbst,, E. (2007, June). Moses : Open Source Toolkit for Statistical Machine Translation. 177–180.

Li,, Y., Duan,, H., & Zhai,, C. (2012). A generalized hidden markov model with discriminative training for query spelling correction. *In Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval* , pp. 611-620.

Liang, H. L. (2008). *Spell checkers and correctors:A unified treatment (Doctoral dissertation, University of Pretoria).* Pretoria: University of Pretoria.

Lorraine, H. (November 2008). *Spell Cheekers and correctors a unified treatment.* Pretoria, South Africa: Master thesis, facu. Of engi. Dept. IT., Univ. of Pretoria.

Lutkevich, B. (2023, March 4). *What is Natural Language Processing? An Introduction to NLP*. Retrieved from techtarget: https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP

M., H. R. (2011). *Levensshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach.*

Mariawit, M. S. (Sep 2020). *Amharic Spelling Error Detection and Correction System: Morphology-based.* Addis Ababa,Ethiopia: MSc Thesis, AA. Univ., Dept. Info. Scie.

Markou Xie, W. H. (2015). Chinese Spelling Check System Based on N-gram Model. 128–136.

Markou, M. a. (2022, July 19). *Novelty detection : a review — part 1 : statistical approaches*. Retrieved from https://doi.org/10.1016/j.sigpro.2003.07.018

Martin, J. H. (2009). Speech and language processing. *An introduction to natural language processing, computational linguistics,and speech recognition,Prentice Hall series in artificial intelligence*, 1-1024.

Mathglot. (2023, March 18). *Natural Langauge Processing*. Retrieved from Wikipedia.org: https://en.wikipedia.org.wiki/Natural_langauge_processing

Mihov S., S. K. (2004). *Precise and Efficient Text Correction Models.* Bulgarian Academy of Sciences.

Misha R., a. N. (2013). A Survey of Spelling Error Detection and Correction Techniques. *International Journal of Computer Trends and Technology*, 372-374.

Mishra,, R., Kaur,, N., & Technique,, A. (2013). A Survey of Spelling Error Detection and Correction Techniques. *International Journal of Computer Trends and Technology*, 4(3), 372-374.

Momtazi, S. (2012). *Natural Language processing:introduction to Language Technology.* Potsdam: University of Potsdam.

Nadkarni, P. M.-m. (2022, Ju;y 1). *Natural language processing*. Retrieved from https://doi.org/10.1136/amiajnl-2011-000464

Navigli, R. (2022, June 19). *Word Sense Disambiguation : A Survey, 41(2)*. Retrieved from https://doi.org/10.1145/1459352.145935

Nejja,, M., & Yousfi,, A. (2023, June 19). *The context in automatic spell correction. Procedia - Procedia*. Retrieved from https://doi.org/10.1016/j.procs.2015.12.055

Olson, D. L. (2008). *Advanced data mining techniques. Springer Science and Business Media.*

P. Etoori, M. C. (July 2018). *Automatic Spelling Correction for Resource Scarce Languages using Deep Learning.* Melbourne, Australia.

Patil, R. R. (Aug, 2020.). *Auto-Spelling Checker using Natural Language Processing.* Maharashtra, India : Chinmay Patil Xavier Institute of Engineering Mumbai, Volume: 07 No 08, .

Petros, B. A. (Oct 25, 2020.). *Tigrigna language spellchecker and correction system.* Aksum, Ethiopia: Info. Tec., Aksum Univ.Vol 11, No 3, .

R.C., C. (2010). *Artificial Intelligence: Natural Language Processing. .*

Ratnasari, C. I. (2017). *A Non-Word Error Spell Checker for Patient Complaints in Bahasa Indonesia.* Indonesia.

Ratnasari,, C., Kusumadewi,, S., & Rosita,, L. (2017). *Non-Word Error Spell Checker for Patient Complaints in Bahasa Indonesia.* Bahasa.

RaymondJr, R. G. (2005). Ethnologue: Language of the world. *Dallas: Summer Institute of Linguistics*.

RaymondJr, R. G. (2005). Ethnologue: Language of the World. *Dallas: Summer Institute of Linguistics*.

Ringlstetter, C. H. (2007). Text Correction Using Domain Dependent Bigram Models from Web Crawls. 47-54.

S., R. (2013). A Deep Graphic Model for Spelling Correction. *Proceedings of the 25th Benelux Conference on Artificial Intelligence*.

Schutze, M. R. (2008). *An introduction to Information Retrieval.* Cambridge University .

Seo, H. L. (2012). A meta learning approach to grammatical error correction. *In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 328-332.

Seretan, V. N. (2004). *A tool for multi-word collocation extraction and visualization in multilingual corpora. .*

Sidoti,, C., Metsky,, H., & Somogyi,, J. (2008). *Using Google to Create a More Accurate and EasilyExtensible Spelling Corrector.*

Sundby, D. (2009). *Spelling correction using N-grams.* Retrieved from http://www.daviddlewis.com/resources/testcollections/re uters21578/.

Tavast, A. M. (2012). Human Language Technologies–The Baltic Perspective. In *Proceedings of the Fifth International Conference Baltic HLT 2012.* IOS Press.

Tesema W., a. T. (2017). Investigating Afan Oromo Language Structure and Developing Effective File Editing Tool as Plug-in into Ms Word to Support Text Entry and Input Methods. *American Journal of Computer Science and Engineering Survey*.

Tijhuis, L. J. (2014). *Context-Based Spelling Correction for the Dutch Language.*

Tirate, K. (2018). *Context Based Spelling Checking for Afaan Oromo writing.* Jimma, Ethiopia: MSc. Thesis, Dept. Info. Sci., Jimma Univ.

Treiman,, R., & Kessler,, B. (2005). *writing System and Spelling Development.*

Verberne, S. (2002). Context-sensitive spell checking based on word trigram probabilities.

Verberne, S. (January 2002). *Context-sensitive spell checking based on word trigram probabilities.* Nijmegen: MSc thesis, Taal, Dept. Spraak & Info. Univ. of Nijmegen.

Wagara, M. M. (2022). *CONTEXT-BASED SPELL CHECKER FOR SIDAAMU AFOO.* Hawassa: Hawassa Univeristy.

Wilde, O. (2017). Spelling Correction and the Noisy Channel.

Wolassa, W. L. (2016). *History of Sidama People.* Addis Ababa.

Xie,, W., Huang,, P., Zhang,, X., Hong,, K., Huang,, Q., & Huang,, L. (2015). Chinese Spelling Check. *Chinese Spelling Check*, 128–136.

Yaregal, T. A. (2016.). *Development of Amharic Grammar Checker Using Morphological Features of Words and N-Gram Based Probabilistic Methods.* Addis Ababa, Ethiopia, : MSc. Thesis, Dept. Comp. Sci. Addis Ababa Univ., .

Yousfi, A. N. (2015). *The context in automatic spell correction. Procedia - Procedia Computer Science,*. Retrieved from https://doi.org/10.1016/j.procs.2015.12.055

Zue,, V., Seneff,, S., Glass,, J., Polifroni,, J., Hazen,, T., Hazen, , T., & Hetherington,, L. (2000). IEEE Transactions on speech and audio processing. *JUPlTER: a telephone-based conversational interface for weather information.*, 8(1), 85-96.

APPENDIX

I Sample code of context-based spell checker for Sidaamu afoo

```
import kivy

import traceback

from kivy.app import App

from kivy.uix.gridlayout import GridLayout

from kivy.uix.boxlayout import BoxLayout

from kivy.uix.textinput import TextInput

from kivy.uix.button import Button

from kivy.uix.popup import Popup

from kivy.uix.label import Label

from kivy.uix.filechooser import FileChooserListView


kivy.require('1.11.1')


class SpellCheckerApp(App):

    def build(self):

        self.title = "Context-Based Sidaamu afoo Spell Checker"

        self.root = GridLayout(cols=2, padding=10, spacing=10)


        # Left Side: Input and Buttons

        left_side = GridLayout(cols=1, spacing=10)

        self.text_input = TextInput(hint_text="Enter text here",
multiline=True, font_size=25)

        self.browse_button = Button(text="Browse",
on_release=self.browse_file, size_hint=(None, None), size=(950, 50))
```

```
        self.check_button = Button(text="Check",
on_release=self.check_spelling, size_hint=(None, None), size=(950, 50))

        self.clear_button = Button(text="Clear", on_release=self.clear_text,
size_hint=(None, None), size=(950, 50))

        self.search_button = Button(text="Search",
on_release=self.search_dictionary, size_hint=(None, None), size=(950, 50))


        left_side.add_widget(self.text_input)

        left_side.add_widget(self.browse_button)

        left_side.add_widget(self.check_button)

        left_side.add_widget(self.clear_button)


        # Right Side: Display

        right_side = BoxLayout(orientation="vertical")

        self.executed_dictionary = TextInput(readonly=True,
hint_text="Executed Dictionary", font_size=25)

        self.search_input = TextInput(hint_text="Search Dictionary",
font_size=25, size_hint=(None, None), size=(950, 100))

        right_side.add_widget(self.executed_dictionary)

        right_side.add_widget(self.search_input)

        right_side.add_widget(self.search_button)


        self.root.add_widget(left_side)

        self.root.add_widget(right_side)


        # Specify the path to your dictionary file here

        self.dictionary_path = "E:/FRANCIS-BEZABIH/Copora/[9]-Sidaama Corpora
Alphabetically Ordered/ALPHABETICAL-ORDER.txt"
```

```
        # Store the current popup for dismissal
        self.current_popup = None

        return self.root

    def browse_file(self, instance):
        try:
            file_chooser = FileChooserListView()
            file_chooser.bind(on_submit=self.load_file)
            popup = Popup(title="Choose a file", content=file_chooser,
size_hint=(None, None), size=(600, 400))
            popup.open()
            self.current_popup = popup  # Store the current popup for
dismissal
        except Exception as e:
            self.log_error(e)

    def load_file(self, chooser, path, filename):
        try:
            if filename:
                selected_file = filename[0]
                with open(selected_file, 'r', encoding='utf-8') as file:
                    file_content = file.read()
                    self.text_input.text = file_content
        except Exception as e:
            self.log_error(e)
        self.current_popup.dismiss()  # Dismiss the current popup
```

```
    def check_spelling(self, instance):
        try:
            text = self.text_input.text
            words = text.split()
            for word in words:
                if not self.is_word_in_dictionary(word):
                    self.text_input.foreground_color = (1, 0, 0, 1)  # Set
text color to red
        except Exception as e:
            self.log_error(e)

    def is_word_in_dictionary(self, word):
        try:
            if self.dictionary_path:
                with open(self.dictionary_path, 'r', encoding='utf-8') as
file:
                    dictionary = file.read().split()
                    return word in dictionary
            else:
                return False
        except Exception as e:
            self.log_error(e)
            return False

    def clear_text(self, instance):
        self.text_input.text = ""
```

```
        self.text_input.foreground_color = (0, 0, 0, 1)  # Reset text color
to black

    def search_dictionary(self, instance):

        try:

            search_term = self.search_input.text

            suggestions = ["suggestion1", "suggestion2", "suggestion3"]

            self.executed_dictionary.text = "\n".join(suggestions)

        except Exception as e:

            self.log_error(e)


    def on_start(self):

        try:

            # Load and display the dictionary when the app starts

            if self.dictionary_path:

                with open(self.dictionary_path, 'r', encoding='utf-8') as
file:

                    dictionary_content = file.read()

                    self.executed_dictionary.text = dictionary_content

        except Exception as e:

            self.log_error(e)

    def log_error(self, error):

        traceback_str = "".join(traceback.format_exception(type(error),
error, error.__traceback__))

        with open("error_log.txt", "a") as log_file:

            log_file.write(traceback_str + "\n")

if __name__ == '__main__':

    SpellCheckerApp().run()
```

II Sample Trained Data

layinki saamueeli daawiti saaooli reyo macciishshi saaooli reyi. hatte yanna giddo daawiti amaaleqoota qeele hige, tsiqilaagi barra keeshshi. sayikki barra mittu wedellichi olu qachi saaoolihunni kae dayi. isino dadillesi leellishara uddanosi dare umisi aana buko giggishshi're no. daawitiwa shiqe ayirrinyunni umo heeshshi yee keere haa'ri. daawitino, ati maminni dayitto yee xa'misi.isino,israeelootu qachi olunnihunni gate fule dayommo yee dawari.daawitino hanni ikkinore kulie yiisi.isino mannu oluwiinni xooqino olu giddono lowo manni reyino.saaoolinna beettisi yoonaataanino reyitino yii.daawitino,saaoolinna yoonaataani reyitinota hiitto ikkite afitto yee xa'misi.wedellichuno,ani hakkawoyite gilboa ilaali aana noommo. saaooli urdesi siqqi're hee'reenna, diinu faradi gaarenna faradaasine isiwa iillitinore afoommo. isino badhe hige caaqqi yee ane afe woshshinoe.anino,mooticha'ya yee yoommo.ewelumma'ya xa'meennae amaaleqicha ikkoommota kuloommosi. isino, ‹lowo geeshsha madi'roommo daafira reyammoraatina, amo dagge shiie› yiinoe. anino gatannokkita afoommo daafira mare qase shoommo. hakkunni gedensaanni ayirrinyu ba'latto umisi aaninni hoole, gumesino angasinni fushshe haa'roommo. mooticha'ya kune haa'roommoreno atera abboommohe yii.

daawitino dadille uddanosi dari; ledosi noori baaluno isinte gede assitu. olunniwa lowo manni reyino daafira daawitinna ledosi manni saaoolira, yoonaataaniranna kaaliiqi olanto ikkitinorira israeelete manni baalaho hashsha geeshsha ita aga agurte lowo geeshsha hururante wi'litunsa.

daawitino duduwo haa're abbiha hakkonne wedellicha, ateti gobbakki mamaati yee xa'misi.

isino, ani gobbakkira xalatimma hee'ranno amaaleqichi beettooti yee dawarisi.

daawitino, kaaliiqi buurinoha hiitto seyatte shaattora dandiitto yiisi. qoleno, kaaliiqi buurinoha shoottota afiikkinni farci'rootto daafira, ati umikki xa'mamatto yee kae, mannisi giddonni mitto woshshe, amo dagge shii yii. manchuno daye amaaleqicha shii.

daawiti saaooliranna yoonaataanira kule wi'li daawiti saaooliranna beettisi yoonaataanira kule wi'lino. tenneno yihudu manni rosanno gede hajajino. kuni coyino yashaari maxaafira borreessame no. daawiti,israeele, ayirrinyikki gaaru aana qanseenna reyino;wolqaataammu hiitto togo ikkite ubbu filisxeemete meenti hagiidhannona,maganiweelo gosa seennino ilillisannona,

konne coye gaati katamira dudubbinoonte.asiqeloona gudumaalirano hasaabbinoonte.

ki'ne gilboa ilaalla,wolqaataammu wonqo hakko ubbe shollitinona,saaooli wonqo nafa zayitetenni fiinannikki gede ikkite gattinona,xeenunna loju dirroonke'ne;aana'ne mitturino mu'roonke.yoonaataani digaani shiikkinni dihigannoho;saaooli biseno mararro nookkiha guddannote.wolqaataamma gante hiiqqaqqitannote;diinano shitannote.saaoolinna yoonaataani mannu baxanno danchuulleeti;reyo heeshshora dibaxxitannoreeti.allamurchunni roore rakkannoreeti;doobbiichunni roore jawaattinoreeti.israeelete meento,faayya duumo imaana uddisiisino'nehura,culkunninna ilkunni biifisino'nehura saaoolira wi'lesi.wolqaataammu olu giddo hiitto ikkite ubbu yoonaataanino ilaallate aana uwe reyino.rodoo'ya yoonaataani, ani atera manchitoommo;ati anera lubbote jaalaatiwe.ate baxilli anera dhaggenniho;

meyate baxillinni rooreho.wolqaataammu hiitto ikkite ubbu olu bagadino hiitto ikke bai yaanni kule wi'li.layinki saamueeli daawiti amoonootanna sooriyoota qeeli shiima yanna gedensaanni amooni moote naoosi reenna, beettisi hanooni isi darga moohi. moote daawiti, annisi naoosi

assinoente gede anino beettisi hanoonira danchumma asse qoleemmosi yii. kunnira, daawiti annisi reyora dadillinota egensiisate hanooniwa manna soyi.

daawiti soqqamaanono marte amooni iillitu yannara,amooni albisooti mootensa hanoonira, daawiti annikki reyora dadille jawaachishanno manna soyinoha lawannohe kuni manni katama dooyye ha'reenna, daye gudankera soyinoha ikkinota diafootto yitusi.

hanoonino daawiti soqqamaano amade ereedansa gama darate meede, uddanonsano hallu geeshsha haranse mure ha'risinsa. daawiti konne coye macciishshe kuni manni lowo geeshsha saalfatino daafira wole manna soye, ereedi'ne lopha geeshsha iyyaarko agarre; hakkunni gedensaanni higginannina yiinsa.amooni gosa angansanni daawiti diina assidhinota affe, beetirehoobinninna tsoobanni , lafo olanto, sooriyunninna xoobinni , manna woxunni hidhitu. moote maakano , olanto haa're daye kaa'lannonsa gede assi'ri. daawitino konne coye macciishshe, olanto baalate ledo iyoaabi olaho soyi. amoonootu fulte kataminsa waalcho e'nannihura albaanni dirantu. wolootu sooriyooti, tsobanninna maakanni daggino bagadaano ledo mittu xawira darga amaddu.iyoaabino albaanninna duumbaanni diinu doyisse noota afe israeelete olanto giddonni bareedda doore, sooriyootu ledo olantanno gede assi. gattino olantono rodiisi abisha hajajanna amooni gosa ledo olantanno gede assi. iyoaabi abishara, sooriyootu wolqissuero ati dagge kaa'lattoe. amoonootu atera wolqissuhero kayinni ani daye kaa'leemmohe. kunnira, xa jawaati manninkenna maganinke katammara jawaante olammo. baalunku coyi kaaliiqi yiino gede ikko yii.

iyoaabinna olantosi bala tuggara shiqqi yite daggu yannara, sooriyootu albinni xooqqu. amoonootu sooriyootu xooqquta affe, insano qolte abisha albinni xooqqe katama e'u. iyoaabino amoonoota qasa agure yerusaalame higi.

sooriyootu olanto israeele qeeltinonsata affe mittowa gamba yitu. tsooba moote hadaadiezeerino efiraaxisi wayira soojjaatoonni noo sooriyootira sokka soyi. insano hadaadiezeeri olanto hajajaanchi shobaaki albansaanni ha'ranna elaami katama daggu. daawiti konne coye macciishshe israeelete olanto baalanta mittowa gamba assi'ri. yordaanoosi fule sooriyootu noowa elaami dayi. sooriyootuno daawiti aana ola fantu. sooriyootu kayinni, israeelete olanto albinni xooqqu. daawitinna olantosi sooriyootu giddonni faradi gaare gantannorenna , faradaasine shitu. olantote hajajaancho shoobaakino qasse madiissitino daafira hakkiicho olu xawira uwe reyi. hadaadiezeerira giwire baattanno mootoolla baala israeele qeeltinonsata affe araara xa'midhu. israeelete giwireno baattunsa. kunnira sooriyootu amoonoota kaa'la waajjitu.

layinki saamueeli daawitinna bersaabehi diru umira badheessu woggara mootoolla olaho fultanno yannara, daawiti iyoaabi, olantote hajajaanonna olanto baala gaadisiisisi. insano amoonoota qeelte raaba katama doyissite amaddu. daawiti kayinni yerusaalamete gati.

mitte soodo daawiti barra goxinowiinni kae, minu aana minnoonni mini aana ka'anna kawa ha'ramanni hee're mitte mancho biso hayishshidhanni noota afi. tini mancho dana xumate. daawitino ewelummase afara mitto sokkaancho isewa soyi. sokkaanchuno, manchote su'mi bersaabehiiti; annise eeliyaabiiti; ayiddaannise hitichu ooriyooti yee kulisi. daawiti manna soye ise woshshiishi. iseno dagge isiwa e'eenna ledose goxi. hatti yanna manchote godowu daye higeenna seerinsa garinni keeraa'mite no. hakkunni gedensaanni ise minisera hadhu. manchote godowuno gatise. daawitirano, godowu gaatinoe yite sokka sokku.

kunnira daawiti iyoaabi, hiticha ooriyo anewa daanno gede assi yee manna soyi. iyoaabino ooriyo daawitiwa soyi. ooriyono kae daawitiwa dayi. daawitino, iyoaabinna olanto baala keereho oluno hiittooti yee xa'misi. qoleno, minikkira hadhe fooliishshi'ri yiisi. ooriyo daawiti mininni fule ha'ranni hee'reennanni, daawiti hakkawoyintenni hurbaatisi aaninni sagale haa're soyisi. ooriyo kayinni minisira ha'rikkinni mootete mini agaraasine ledo waalchu mule goxe gali. daawiti ooriyo minisira ha'rinokkita afe, minikkinni dibaxxite keeshshoottohu dayitto minikkira ha'ra mayira giwitto yee xa'misi.

ooriyo daware, taabootu, israeelenna yihudu manni dukkanaho no. mootichi'ya iyoaabinna ledosi manni mullu xawira no. ani itammoranna agammora qoleno galte'ya ledo galammora mini'yara ha'ro halaalikki afanna konne coye diasseemmo yiisi.

daawitino qole, maahoyyena, ga'a ha'rattona techo konne hosi yiisi. ooriyono hakko barranna layinki barra yerusaalamete keeshshi. layinki barra, daawiti ooriyora hurbaate itise, dimba geeshsha aganno gede assisi. hakko hashshano ooriyo minisira ha'rikkinni mootete mini agaraasine ledo kadhe gali.

soodita, daawiti borrote sokka qixxeesse ooriyo haa're iyoaabira massanno gede assi. borrote sokkano, ooriyo olu kaajjino widoonni alba qolte sayisse, qasame reyanno gede assi; ki'neno mulisinni ho'le yitannote. iyoaabi katama doyisse hee're, diinunniti kaajjado olanto noo widoonni qole ooriyo albaanni sayisi. katamu mannino fule iyoaabi ledo olami. daawiti bagadaano giddonni gamu reyitu; ooriyono insa ledo reyi.

hakkunni gedensaanni, iyoaabi manna soye olunniwa ikkinore daawitira kulisi. sokkaanchohono, olu ikkito mootete kulittohu gedensaanni, miteekke moote hanqe, ‹olantini yannara katamunniwa gamba mayira yitini huxxu aanaanni qolle maxxa ollanni'neta diaffinoonni gedewooni beetti abemeleeki ikke reyino gara diqaagginanni iso tebesi yinanniwa mitte mancho huxxu aanaanni qolte daammibeetto tugge shitino. tenne affinanni heedhine huxxunniwa gamba mayira yitini› yiiro, ati qolte, ‹olantokki giddonni reyinohu mittichu ooriyooti› yite kulisi yii.

sokkaanchuno kae ha're, iyoaabi hajajinosire baala, diininke wolqise qasankera katamunni fule xawu geeshsha dayino. ninke kayinni katamu waalchi geeshsha shorrinoommo. maxxa oltannori huxxu aanaanni qolte tugge, olantokki giddonni gama manna shitino. soqqamaanchikki hitichu ooriyono reyino yee daawitira kulisi.

daawitino sokkaanchoho, ‹wonanni bagadu aye shaannoro dianfoonnina, hifattooti. hatteentenni jawaatte olante katama amadi› yiinohe yite iyoaabira kulisi; atino ledde jawaachishisi yii.ooriyo galte ayiddaannise reyinota macciishshite wi'litu. wi'la gooffeenna xidda qasidhuhu gedensaanni, daawiti soye minisira dagganno gede assise. iseno dagge galte ikkite labbaa beetto iltusi. ikkollana, daawiti assino coyi kaaliiqi albaanni diseyino.

layinki saamueeli naataani abbino sokkanna daawiti aagintaawa kaaliiqi, himanaancho naataani daawitiwa soyi. naataanino daawitiwa daye, mittu katamira mittu dureessunna mittu buxichu manchi no. dureessaho lowo ge'reewinna lowo lali noosi. buxichoho kayinni hidhinoti mitte ge'rechote goodane calla noosi. iseno oososi ledo itisanninna hayikkisanni hanqafe lossi'rinote. tini goodaneno isira ilasi gedeeti. mitte soodo dureessu minira mittu wosinchu dayino. manchuno wosinchisira umisi saada gorra marare, buxichu goodane haa're gorre itisino yii.

daawitino kunni dureessu manchi assootira lowo geeshsha hanqe naataanira, heeshshote magani halaaleeti, togo assino manchira reyo hasiissanno. togooha mararro nookki coye assino daafira, goodanete darga anga qola hasiissannosi yii.

naataanino daawitira, hakku manchinniwe ateeti. israeelete magani kaaliiqi, ‹israeelete mannira moote ikkatto gede buuroommohe. saaooli angannino gatisoommohe. mootichikki saaooli gashshootenna meentosi angakkira qoloommo. israeelenna yihudu mine baala oommohe. hakku ajoommehero wole ledeemmohe. hajajo'ya mayira mishittoyya konne busha coyeno mayira assitto ooriyo olunniwa shiishiishootto. amoonootu shitannosi gede assootto; galtesino adhootto. ooriyo galte adhatenni aneno mishoottoena, minikkinni jaddote reyo faffoonke. kune minekki busha coye kayiseemmo. meentokki albikkinni haa're woleho eemmo; barra xawoho meentikki ledo wolu goxanno gede asseemmo. ati konne maaxxe loosootto. ani kayinni konne coye barra israeelete manni albaanni reqecci asseemmo› yiino yee kulisi.

daawitino naataanira daware, kaaliiqi aana sao assoommo yii. naataanino daawitira qole, kaaliiqi cubbokki agurannohena direyatto. ikkollana, tenne sao assatenni kaaliiqa lowo geeshsha mishootto daafira, ilaminohu qaaqqikki reyanno yiisi. hakkunni gedensaanni naataani galagale qaesi ha'ri.

daawiti qaaqqi reyo kaaliiqi, daawitira ooriyo galte iltino qaaqqira lowo dhibba tugi. daawitino qaaqqisi daafira magano huucci'ri. itanna agano agure, baalanka hashsha uulla kadhe goxi. ledosi gashshaanono uullanni ka''anno gede assate wo'naaltu. isi kayinni uullanni ka''anna ledonsa sagale ita giwi. qaaqquno mitte lamala gedensaanni reyi. daawiti ledo gashshaanono, qaaqqu lubbotenni hee'reenna yinoommore dimacciishshinonke. xa kayinni qaaqqu reyinota kullummoro hiitto ikka ikka umosi gawajja ikka yite kula waajjitu.

daawiti kayinni ledosi gashshaano hashaashantanna macciishshe, qaaqqu reyinota afi. kunnira, qaaqqu reyini yee xa'minsa. insano, ee reyino yite dawartu.

daawitino uullanni kae bisosi hayishshi're, buudhe, uddanosino soorri're kaaliiqi minira e''e umosi heeshshi asse kumbuuli. minisirano daye, sagale abbe''e yii; abbinoonnisi sagaleno iti. ledosi gashshaano, kuni coyi dileellinonke; qaaqqu lubbotenni hee'reenna ita aga agurte wi'lootto. xa kayinni qaaqqu reya hee'reennanni ka'e sagale ititto yitusi.

isino daware, qaaqqu lubbotenni hee'reenna ita aga agure wi'loommohu, kaaliiqi marare qaaqqo gatisae ikka yeeti. xa kune qaaqqu reyino; ma assi'rammora qatumeemmoyya xa qaaqqo kayisammora dandeemmoni ani isiwa mareemmota ikkinnina isi anewa dihiganno yii.

selemooni ilami daawiti galtesi bersaabehi jawaachishi. ise minira mare ledose gali. iseno godobbe labbaa beetto iltu. su'mano, selemooni yee fushshisi. qaaqqono kaaliiqi baxisi. kunnira, himanaancho naataani soye su'masi, yidiidiya yeemmo yii. daawiti amoonootunniha dannawu katama amadi hatte yannara iyoaabino amooniha dannawu katama raaba amadara rahe no. iyoaabino daawitiwa manna soye, raaba katama qase, wayi fulannowa amadoommo. xa katama ani amadeenna, ane su'minni woshshamannokki gede, gattino olanto gamba assidhe, katama gaangeessite amadi yii. daawitino olanto gamba asse raaba katama mare gaarame qeele amadi. daawiti amoonootu magani melooki yinannihu umi aaninni kiilo ikkannohu dilqu kinchi noo ba'latto haa're umisi aana wodhi. katamu giddonnino lowo jajja adhi. hakko hee'ranno mannano magaazetenni, qottotenninna meesanetenni loosannosi gede assi'ri. xuube seekkitannosi gedeno

assi. amoonootu katamma baalate hee'ranno mannino konne looso loosanno gede assi. hakkunni gedensaanni daawitinna olantosi baala yerusaalame higgu.

aminooninna taamaari daawiti beetti abeseloomira mitte mine assidhinokkiti biifado rodoo noosi. su'mise taamaariiti. anninsa beetti aminooni taamaari lae halchi. ise seemo ikkitino daafira labballu ledo xaada diuyinannise. aminoonino isenni xaadate lowo geeshsha halche hooge dhiwami. aminoonira daawiti rodii saami beetti iyonaadaabi jaalasiiti. isi lowo geeshsha hayyichaho. aminoonira, mootete beetto, barru barrunkunni togo ikkite mayira jaawitto dikkulattoe yiisi.

aminoonino, rodii'ya abeseloomi rodoo taamaari lae halchoommo yii.

iyonaadaabino, dhiwamoommo yite daallasikkira goxi. annikki la"ahera daanno woyite, ‹rodoo'ya taamaari dagge sagale uyitannoe gede, sagaleno ani la"anna worte cuucishshannoe gede assie› yite huucci'ri yiisi. aminoonino yiinosinte gede, dhiwamoommo yee goxi. moote la"asira dayita, rodoo'ya taamaari dagge ani la"anna shiima tima gante angasenni itissannoe gede assie yee huucci'ri.

daawitino, rodiikki aminooni minira marte la"aato uyisi yee taamaariwa sokkaancho soyi. taamaarino ka'e rodiise aminooni minira martu. martanno woyite isi goxe no. bullee haadhe liisse, isi la"anna la"aato wortusi. mixashshotenni fushshite itanno gede shiqqi assitusi. isi kayinni ita giwe, manna baala mininni fushshe"e yii. mannu baalunkuno mininni fuli. hakkunni gedensaanni aminooni taamaari, angakkinni chuucishattaena la"aato haadhe shiqqi assie yiise. taamaarino gantino tima haadhe rodiise aminooniwa shiqqi assitu. sagale haadhe shiqishshuta amade, rodoo'ya ballo, amo ane ledo goxi yiise.

ise qolte, rodoo'ya, dee'ni togoo coyi israeelete giddo dihasiisannona shollishshootie. konne makki'nanni coye assate giddeessitootie. kuni coyi ikkiro mannu albira hiitto ikke fuleemma ate su'mino israeelete mereero bowinoha ikkanno. hatteentenni konne coye mootete kuli; isi ane atera diho'lannohena yitusi. isi kayinni yitinota dihaa'rinose. wolqa roorinose daafira giddeesse macca hunise.hakkunni gedensaanni aminooni tiyyi're giwise. alba baxinosehunni roore giwe, mininni fulie yiise. iseno, mininni fushshite shorrattoeha ikkiro bunshekki te'enninni roore ikkitanno yitusi. isi kayinni yitinota dimacciishshinose.