JIMMA UNIVERSITY

INSTITUTE OF TECHNOLOGY

SCHOOL OF COMPUTING

DEPARTMENT OF INFORMATION TECHNOLOGY

CONTEXT BASED SPELL CHECKER FOR AMHARIC

BY

NIGUSU YITAYAL

JUNE, 2016

JIMMA, ETHIOPIA

JIMMA UNIVERSITY

INSTITUTE OF TECHNOLOGY

SCHOOL OF COMPUTING

DEPARTMENT OF INFORMATION TECHNOLOGY

CONTEXT BASED SPELL CHECKER FOR AMHARIC

A Thesis submitted to the school of graduate studies of Jimma University in partial fulfillment for the degree of the Master of Science in Information Technology

BY

NIGUSU YITAYAL

Principal Advisor: Getachew Mamo (Assistant Professor)
Co-Advisor: Teferi Kebebew (Msc)

JUNE, 2016

JIMMA UNIVERSITY

INSTITUTE OF TECHNOLOGY

SCHOOL OF COMPUTING

DEPARTEMENT OF INFORMATION TECHNOLOGY

CONTEXT BASED SPELL CHECKER FOR AMHARIC

BY

NIGUSU YITAYAL

This thesis entitled "CONTEXT BASED SPELL CHECKER FOR AMHARIC" has been read and approved as meeting the requirements of Department of Computing in partial fulfillment for the award of the degree of Master of Science in Computing (Information Technology), Jimma University, Jimma, Ethiopia.

Name	Title	Signature	Date
______________________	Chairperson	____________	__________________
______________________	Principal Advisor	____________	__________________
______________________	Co- Advisor	____________	__________________
______________________	External Examiner	____________	__________________
______________________	Internal Examiner	____________	__________________

DECLARATION

I declare that this thesis is my original work and it has not been presented for a degree in any other universities. All the material sources used in this work are duly acknowledged.

____

NIGUSU YITAYAL

JUNE, 2016

This thesis has been submitted to the department for examination with our approval as university advisors:

Principal Advisor: Getachew Mamo (Ass. Professor) ____________________

Co- Advisor: Teferi Kebebew (MSc) ____________________

DEDICATION

This thesis was dedicated to my lovely parents and family members Yitayal Fered, Yesgede Sentayehu, Yeshmbele Lakew, Bosna Sentayhu and Beleyneh Yitayal for their assistance and constructive comments until the end of the study.

Acknowledgment

First of all, and foremost, I would like to thank Almighty God. Secondly, I would like to acknowledge my advisor Getachew Mamo (Assistant Professor) for spending his precious time to direct me. He always gave me a clear direction to follow and also taught me with his generous patience. I have no word to thank him for his precious comments and immeasurable assistance throughout my study. Next I would like to convey my deep appreciation and gratefulness to my co-advisor Teferi Kebebew (Msc) for his immeasurable assistance from the starting up to the end of the work. I am gratefully thanking him for his concern, unreserved assistance throughout my study and for his guidance in keeping me on the right track and the accomplishment of the study. I also extend my gratitude to the entire staff of the department of computing for their assistance in various ways. My specials thank goes to all staff members in all directions by helping me continuously from beginning to end of thesis submission. Similarly, I would also like to acknowledge my parents and family members ,they always advice and support me in many ways to accomplish my study successfully. Additionally, I would like to express my special thanks for Yeshambele Lakew and Bosena Sentayhu for their contribution to conduct the study by supporting and advising during the study. Finally, I would like to thank my colleagues and classmates for their assistance and constructive ideas to succeed this paper. It’s also my great pleasure to thank Jimma University for providing financial support to conduct this study and the chance they gave for me to learn my postgraduate study.

Abstract

Developing language applications or localizations of software is a resource intensive task that requires the active participation of stakeholders with various backgrounds. Spell checking is the one and significant application of computational linguistics. Spell checking is the process of detecting and sometimes providing spelling suggestions for incorrectly spelled words in a text. The text data in local languages is also increasing fast, requiring text-processing tools for text documents to be available in local languages. This application is vital to detect and correct spelling errors in under resource languages like Amharic. This thesis describes the development, implementation and testing of a model that have been developed to detect and correct non-word and real word typing errors made by writers for Amharic language. The aim of this study is to develop context based spell checker and corrector for Amharic depends on the spelling error patterns of language based on the sequence of words in in the input sentences contextually.

Training and testing data sets were collected from various sources describes different issues to balance the inclusiveness of the corpus. The texts were prepared and cleaned manually from any kind of unnecessary errors which are not necessary for detection and correction like numbers and punctuations. Experimental research design was used to evaluate the performance of developed prototype system. To conduct experiment 10,000 and 500 sentences were used to learn and test the model respectively. According the experimental result, the spell checker can correctly classify Amharic words with prediction accuracy of 95.62%, lexical recall of 95.52% and lexical precision of 35.18% for non-word spelling errors. The performance of the context sensitive spell checker was measured and scored a value of prediction accuracy 64.93%, lexical recall 63.42% and error precision 5.49% to resolve real word errors. Finally, as a comprehensive spell checker system has to be capable of detection, resolving and ranking correction possibilities using complementary contextual and linguistic knowledge, we are planning to extend the coverage level of the system considering more syntactical and semantic knowledge to improve and complete the quality of the developed system through rule based approaches.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

OPERATIONAL TERMS

Computational linguistics: is a branch of language technology leaning towards the linguistic aspects of the computational handling of language.

Spelling Error Detection: is the process of detecting the spelling error during typing

Natural language: is a language spoken, written or otherwise used by people as a means of communication.

Non-word error: a spelling error where the mistyped string is not a valid word form in a dictionary of the language.

Real-word error: a spelling error where the mistyped string is another valid word form in a dictionary of the language.

Spell checker: is software capable of detecting and correcting spelling errors in word forms.

Spell checking: is the task of verifying that the word forms of the text are correctly written word forms in the language of the spell checker. Spell checker can, however, refer to software capable of both spell checking and correction.

Spelling correction: is the task of correcting misspelled word forms in the text by correct ones, or suggesting alternatives in an interactive application.

ACRONYMS

NLP	Natural Language Processing
SR	Speech Recognition
IR	Information Retrieval
CBSCA	Context Based Spelling Checker for Amharic
POS	Part of Speech
OCR	Optical Character Recognition
CV	Consonant Vowel
AMSPELL	Amharic Spell Checker
ASCII	American Standard Cod for Information Interchange
ENA	Ethiopia News Agency
SOUNDEX	SOUND indEX
FSA	Finite State Automata
WIC	Walta Information Center
MS	Micro Soft
HC	Habeas Corpus
ECOSA	Ethiopian Computer Standards Association
ISO	International Standard Organization
UTF	Unicode Transformation Format
MLE	Maximum Likelihood Estimation

CHAPTER ONE

INTRODUCTION

1. Background of the study

One of the fundamental features of human behavior is the natural language. It is a vital component through which we communicate about the world that affects our daily lives. Most human knowledge is recorded using natural languages, therefore, only computers that have the capability to understand natural language can access the information contained in the natural language efficiently.

Natural language processing (NLP) can be described as the ability of computers to generate and interpret natural languages. It is a major subfield of study in computer science. The applications that will be possible when NLP capabilities are fully realized are impressive as computers would be able to understand and process natural language, translate languages accurately in real time, or extract and summarize information from a variety of data sources, depending on the users' requests [29]. Natural Language Processing, as a field of scientific inquiry, plays an important role in increasing computer capability to understand natural languages, the language by which most human knowledge is recorded. NLP focuses on designing and implementing of tools, techniques, frameworks to enable computers communicate effectively as and with humans.

Additionally, NLP encompasses a set of related disciplines like psycholinguistic, linguistic and computational linguistic and other related fields to study and design effective components like morphological analyzer, syntax parser, semantic analyzer, speech recognizer and many more applications that can help computers easily understand text, sounds, images and communication material as humans do. NLP has also many applications, which includes Automatic Summarization, Machine Translation, Part-of-Speech Tagging (POS), Speech Recognition (SR), Optical Character Recognition (OCR), grammar checker, spell correction and Information Retrieval (IR).

In computing, spell checking is the process of detecting and sometimes providing spelling suggestions for incorrectly spelled words in a text. Spell checking is a significant application of computational linguistics whose research extends back to the early seventies when Ralph Gorin built the first spell checker for the mainframe computer at Stanford University [47]. By definition, a spell checker is a computer program that detects and often corrects misspelled words in a text document

[44]. It can be a standalone application or an add-on module integrated into an existing program such as a word processor, search engine or mobile application.

Fundamentally, a spell checker is made out of three components: An error detector that detects misspelled words, a candidate spellings generator that provides spelling suggestions for the detected errors, and normally, choosing the correct word is done by humans rather than by computers. This component may be required for auto correction systems which are commonly used in search engines and mobile systems. All these three basic components are usually connected underneath to an internal corpus or dictionary of words that they use to validate and look-up words present in the text to be spell checked. However, as human languages are complex and contain countless words and terms, as well as domain-specific idioms, proper names, technical terminologies, and special jargons, regular dictionaries are insufficient to cover all words in the vocabulary of the language.

Word error is a major hindrance to the real world applications of natural language processing. In textual documents, word-error can be of two types. One is a non-word error which has no meaning and other is a real word error which is meaningful, but not the intended word in the context of the sentence. Of these, a non-word error has been widely studied and algorithms to detect and suggest the correct word for the error have been proposed. Errors, particularly spelling and typing errors are abundant in the human generated electronic text. Search engines like Google do spell checking and correction automatically. This would prevent wasted computational processing, prevent wasted user time and make any system more robust as spelling and typing errors can prevent the system identifying the required information.

The idea of using context of a misspelled word to improve the performance of a spell checker is not new [46]. Moreover, recent years have seen the advance of context-aware spell checkers such as Google Suggest, offering reasonable corrections of search queries. Errors detected by such advanced spell checkers have a natural overlap with those of rule-based grammar checkers because grammatical errors are also manifested as unlikely n-grams. Methods used in such spell checkers usually employ the noisy-channel or winnow-based approach [26]. It makes extensive use of language models based on several morphological factors, exploiting the morphological richness of the target language.

Most word processors have a built-in spell checker that highlights misspelled words in some way and offers the facility to correct these misspellings by selecting an alternative from a list. To detect these misspellings in the first place, most spell checkers take each word in a text in isolation and check it against the words stored in a dictionary. If the word is found in this dictionary it is accepted as correct without regard to its context. Although this approach is adequate for detecting the majority of typos, there are many errors that cannot be detected in this way. These spelling errors are real-word errors which are correctly written that are not the word the user intended. Real-word spelling errors are errors that occur when a user mistakenly types a correctly spelled word when another was intended. Errors of this type generally go unnoticed by most spell checkers as they deal with words in isolation, accepting them as correct if they are found in the dictionary and flagging them as errors if they are not. The problem of real-word error is a more complex one. Usually, such error disturbs the syntax and semantics of the whole sentence, which requires human-being to detect it.

Since spelling error detection and correction on word level cannot solve this problem, research into automatic context-sensitive spell checking is going on to develop spell checker and corrector based on the context of the text. Spelling error detection and correction now focuses on the development of spell checking algorithms that make use of context. Spelling error detection and correction techniques that aim at detecting and correcting interactive real-word errors are thus also referred to as context-sensitive spell checking. In this research, we tried to create a context-sensitive spell checking method that is able to detect and correct human-generated real-word errors. The context-sensitive spell checker can be combined with corpus based spelling error detection and correction application in order to create an application that is able to detect and correct non-word errors as well as real-word errors.

As per researchers knowledge the above mentioned spell checker and corrector techniques and many NLP tools have been developed for English language to more degree of acceptance, efficiency and correctness's than that of Amharic language. Regarding Amharic language, there are numerous numbers of researches being undergoing and done to improve the gap and alleviate the problem in different areas of NLP. Today, spelling checker of various kinds (e.g. Probabilistic, rule based) have been developed for different languages, which have relatively wider use nationally and or internationally (e .g. English, German, Chinese, Arabic etc.) [2]. The major goals of this research is finding out and develop a model of interactive context-sensitive spell checking for Amharic language using unsupervised n-gram probability information to provide a valid solution to the problem of real-word errors.

1.1 Statement of Problem

Problem of interactive and automatic spell checking is not new in the areas of information retrieval and language processing. The research started as early as the 1960s [16]. Many different techniques for detection and correction of spelling errors are proposed during last 40 years. Some of these techniques exploit general spelling error trends while others use the phonetics of misspelled word to find likely correct words. Spell checkers and correctors are either stand-alone application capable of processing a string of words or a text, or as an imbedded tool which is part of larger applications such as a word processor.

Spell error and correction are closely related to exact and approximate pattern matching respectively. Checking words that are valid in some language is a difficult task since it has many vocabulary and morphology in one specific language. On the other hand, correcting errors with one or more alternative suggestions also considered when a misspelled word identified in the written text. Spell checking involves non-word error and real word error detection and spelling correction performed with respect to the writers need. So, to write and convey ideas language learners and native writers understand and recognize the language features if not the spell checker replayed valid words for them for missed words.

One of the inevitable activities of any government or private office worker needs to edit a document that has been written by someone. Computers have considerably minimized this activity since they automatically detect and correct spelling as well as grammatical mistakes. Thanks to this, office workers not only save considerable amount of time and money but they have also started relatively producing better documents.

Unfortunately, Ethiopians do not benefit from this, unless they use English or one of the many foreign languages for which electronic spell checkers have been developed. This is because no software provides spell checker for Ethiopian languages. Everyone makes spelling mistakes at one time or another. Mistakes can be caused by not taking the time to proofread or lack of knowledge about what the correct spellings are, and other times it's from confusion about usage.

Quite a few of spelling correction techniques are being used with text editors and other text handling applications and are showing reasonably good performance. Nevertheless the problem of spell checking is still considered open for further research and improvements in Amharic language. There are many reasons for considering this problem still unsolved and the main question to be asked while developing a spell error detector and corrector has to improve and enhance the gap of spell checker with respect to the Amharic language.

The first reason is that as the research in the area of natural language processing advanced over the years, the need of automated spell checking is being felt for many tasks other than simple proof reading of computer generated text. Many NLP applications like machines translation systems, Text to Speech Systems, information retrieval and Optical Character Recognizers require automated spell checking of text. Amharic language should have its own spell checker to those applications applicable and solving these problems by developing new models for spelling error and corrector is crucial. The demands that are implied by these applications are much more challenging than the ones implied by human users of spell checkers. The major difference is that, for a human user it is adequate if the errors are detected and for every error a small number of suggestions are proposed from which user can select the required one. Whereas, in automated spelling correction it becomes the spell checker's responsibility to decide on what is required, spell checker should be able to find one best correction for an error.

The second reason for considering the spell checking problem unsolved is that most of the techniques proposed so far are based on English or some other Latin script based language. Since every language has its own writing system and alphabets, the techniques that perform well in one language may not perform that well for some other language, they may even totally fail. The writing system of a language also governs the types and trends of spelling errors of that language. Therefore, existing techniques which are designed mainly focusing English language are limited in their scope.

Another challenge is really needs a spell checker that can detect and correct a given spelling error when the users write incorrect words that are not exist from the dictionary or corpus. Basically the spelling errors can be real word errors which exist in the corpus and non-real word error which are not exist in the corpus or dictionary. Non-native Amharic language writers cannot correct the spelling errors correctly since they are not familiar with languages and even does not now the semantic of the sentences. Hence, developing Amharic spell checker can assist those language users and learners even if for native Amharic writers to correct misspelling without spending time and efforts to correct the misspelling during writing.

Using word processors in Amharic writing is growing in a very fast pace. Currently, government ministries and departments, legal institutions, business offices, media channels, universities all use word processors in their daily work. This growth is expected to continue as computers and electronic devices become more and more prevalent in Amharic. Spelling is an important aspect of language writing. Poor spelling can interfere with communication between the writer and the reader. Word processer use spell checkers to suggest corrections to misspelled words. Unfortunately, existing word processors do not come with built-in spell checkers for every language. Individual nations create their own customized dictionaries and add them to the word processors for error correction. Currently, Amharic language lacks a reliable spell checker. This limitation need to be resolved. Overcoming the lack of spell checker problem would flourish Amharic writing and helps word spelling standardization. Word recognition and automatic correction techniques have been studied in a large spectrum of computer applications. These include word processors, machine translation, search engines, and voice recognition. While almost all modern human spoken language has one or more spell checkers, Amharic language lacks even a very basic one. Hence, building an Amharic spell checker would have an outstanding effect on Amharic language processing applications.

Furthermore, the Amharic spell checker is very important in learning environment. For example, children’s can learn the spelling errors during writing by themselves without interacting teachers. But there is no spell Amharic spell checker that can detect and correct spelling errors. As per researchers knowledge there is no Amharic spell checker and corrector application that incorporate Amharic spell checker like search engines and mobile application that can detect and correct the misspellings during writing and the result after writing can be meaningless and users cannot understand what they wrote and the message can be changed.

From above discussion, it can also be conjectured that in order to propose a new spell checking technique or fit an existing one in a language having a writing system significantly different from English, one has to clearly identify the language specific issues and deeply investigate general spelling error trends of the language, only then a reasonably effective spell checking approach can be proposed. The study described and developed the details of a study performed on Amharic language to identify the problem areas of Amharic spell checking and to test the effectiveness of spell checking techniques on Amharic.

Kukich [40] in a comprehensive survey of spell checking techniques claimed that "Developing context-based correction techniques has become the foremost challenge for error correction in text". The fact that her paper remains the definitive survey is perhaps indicative of the small amount of progress that has been made in the last decade or so. The research described in her work takes up this challenge to detect and correct errors. It considers both syntactic and semantic approaches to the problem and assesses their performance when applied to real-word errors produced by dyslexics.

Interactive context sensitive spell checker and corrector were developed for Amharic language that can detect misspelling words based on the sequence of words in the sentences with in the given corpus. The context based spell checker mostly used for real word errors regard to the structure of the given sentences and non-word errors would be solved by providing word suggestion with respect to probabilistic information.

Currently, there almost no software or web-services applications that are used for implementing language specific features for Amharic language [45]. Even if a lot of researches are going on, the language is not studied in detail manner to integrate with the computer technology. In addition, we strongly share the conclusion by Daniel Yacob [64] Amharic orthography reflects the spoken phonetic features to a large extent. So this can be lead to believe that there is no notion of “spelling” in Amharic. The rule generally followed is “if a word sounds right when read aloud then it was rightly written”. Upon closer inspection, we quickly realize that Amharic spelling rules are just very forgiving when compared to the strict, albeit irregular, conventions of English.

Spell checking has been researched into a great depth in various development languages like English, Arabic and French while there are states of the art spell checker tools available for English language. Various documents, novels, newspapers are typed in Amharic and there is a need for development of spell checking tools for Amharic. These thesis aims at building a spell checker application for Amharic language. By doing so, we believe, our work gives yet another perspective for current research and strengthens the attempts already made on the NLP of Amharic. Therefore, the major concern of this research was to investigate an unsupervised machine learning approach for Amharic spell checker and corrector, test the results in order to develop an interactive context sensitive spell checker and corrector for Amharic texts using the local context of the words.

1.2 Objective of Study

1.2.1 General Objective

The general objective of the study is to develop an interactive context sensitive spelling checker that can detects and correct spelling errors for Amharic language.

1.2.2 Specific Objectives

To achieve the general objective, the study attempts to address the following specific objectives:

- To review the concept of spelling error detection and correction
- To understand the basic characteristics of the Amharic spelling and its writing system
- To collect and prepare Amharic sentence corpora for training and testing model
- To design context sensitive spell checker model for Amharic text
- To develop prototype to demonstrate the effectiveness of designed model
- To test and evaluate the performance of spelling checker system
- To draw conclusions and forward recommendation for further research

1.3 Research Methodology

Different methods were studied to get detail information for spelling checker application from various sources. To develop and design the model, the researcher would use different methods and techniques that are related to spelling error detector and corrector according to the features of the Amharic language. In order to achieve the objectives of this research, the following methods and techniques would be employed.

1.3.1 Literature Review

An extensive literature review was conducted to understand the general n-gram approach to spell checking and select a suitable n value type with a suitable n-gram generation and extraction to be used in the experiment. Evaluation techniques for testing the effectiveness of the method were also determined from this review. Printed materials like books, journal articles, and previous related research work as well as electronic materials on the web were consulted for this purpose.

Additionally, Materials concerning Amharic languages and other related languages spelling correction and candidate suggestion mechanisms were reviewed. Since there are several approaches used in spelling error detection and correction, the literature review was carried out on different aspects of spell checker techniques that are focused on different aspects. The writing system and structure of the Amharic sentences was reviewed to detect and correct the spelling errors. In addition to this we were identifying the types spelling error patterns to develop a better interactive context sensitive spell checker and corrector that correct the misspelled words in the corpus.

1.3.2 Proposed Model

The approach to be followed in this work is unsupervised statistical approaches, since supervised machine learning requires laborious and costly manual preparation of tagged and annotated text which is not ideal for under resourced languages like Amharic. The proposed method does not require any human annotated knowledge resources. In this thesis the researcher has used unsupervised statistical approaches to detect and correct spelling errors efficiently in the written Amharic words and sentences. N-gram statistical methods were used for detecting and correcting the spelling errors contextually depend on the neighboring words sequentially in Amharic sentences within the corpus which are collected from various resources. The value of n value for n-gram is chosen depend on the size and collections of the corpus that have been collected. Since Amharic does not have sources to train and test the model, bigram were selected and uses for detection and correction purposes by bigram words rather than using other trigram and above.

Multiple approaches that have been developed to solve the non-word spell checking and correcting problem include n-gram analysis and the dictionary lookup for identifying the errors and edit distance approach for suggestion generation. The lexicon based approach for identifying the errors combined with the shortest edit distance approach is used spell checker applications. Firstly, a dictionary words were constructed from a corpus and lexicon based spell checker build by a dictionary lookup from the available dictionary. A corpus based dictionary developed and used for identification of errors from a test corpus. Errors detected using dictionary lookup and corrections suggested on the basis of minimum edit distances. Levenshtein minimum edit distance was used for dictionary checking and candidate suggestions were sorted based on the edit distance operation.

Detection and correcting real-word errors is crucial in spell checking that are important to understand the context of the terms in the given sentences. Context based checking for real-word errors done using the n-gram approach. The n-gram approaches constructed from the given text to be checked searched for in a set of n-grams constructed from available corpus. The frequencies of these n-grams probability used to suggest possible corrections for real world errors. The method for context-sensitive spelling error detection and correction that is used in this thesis considers a number of word sequences instead of single words.

To accomplish a task of spell checker and corrector, one has to have probabilistic information such as the sequential probability of occurrence of words in the sentence. The suggestion words are ranked based on the sequential probability occurrence of words in the given corpus. This can be achieved by preparing training and testing a corpus. Since Amharic corpus, mostly not readily available, we are prepared comprehensive and balanced Amharic corpus in order to design and test the spell checker and corrector model. The corpus prepared from various sources that include newspapers, books covering wide domain areas such as agriculture, politics, religion, history, sports, love and others.

1.3.3 Corpus Preparation

For training and testing purpose sample Amharic sentences were important to measure the performance and accuracy of the interactive context sensitive spell checker during detecting and correcting both non-word and real word errors. The model was implemented, trained by 10,000 Amharic sentences which are not annotated.

This Amharic text data set collected from various resources to reflect the semantic and syntactic structure of the Amharic language features. Sample datasets were collected from different sources to make complete and balanced the coverage of words if the language. The corpus was collected from Walta Information center, ENA, Newspapers, blogs and books that are available in electronic format. Walta Information Center is a government information center that distributes news for broadcast over television and radio for local consumption. These were selected as sample to make it representative and balanced. A corpus is said to be representative of a language variety if the content of the corpus can be generalized to that variety. Basically, if the content of the corpus, defined by the specifications of linguistic phenomena examined or studied, reflects that of the larger population from which it is taken, then we can say that it "represents that language variety." These are considered as consisting different issues of the users like social, economic, technological, health, political and other issues. This could reduce the possibility of making the corpus biased toward some specific words that do not appear in everyday life [18].

1.3.4 Modeling and Tools

The spell checker and corrector algorithms of Amharic were trained on the Amharic training corpus. The corpus is divided into training set and test set. The algorithms trained on the training sets evaluated on the other sets of test corpus. The results were analyzed to evaluate the impact and draw conclusions. Java NetBeans programing environment was used for implementing the model and algorithm prototype of spell checker. Java is used as a programming language in this study since it is a general purpose programming language. It is optimized for software quality, developer productivity, program portability, and component integration.

1.4 Scope and Limitation of the study

There is a supervised and unsupervised machine learning techniques for spell checking and correcting, due to time and linguistics constraint to prepare and train the model unsupervised machine learning algorithm were used to build and evaluate spelling checker and corrector model. The study was limited to developing interactive spelling checker and corrector for Amharic language that can correct typographical and cognitive spelling errors .The prototype was developed based on the context of the sentences on the immediate surroundings of the words considering the local context. The non-real word misspelling errors were corrected based on dictionary lookup before checking and contextual spelling errors were corrected by considering n-gram extraction of words in the n-gram lists. This thesis focuses only local surrounding context of words in the sentences and it's dependent on the Amharic texts. The spell checker could not check and correct the errors automatically and errors weren't detected intelligently.

1.5 Significance of Study

The spelling checking and correcting have become a part of everyday life for today generation. Those are an inevitable part of the process such as text editing tools in various areas like word processor, search engines and mobile applications. In addition to being an academic exercise to fulfill the requirement of the program, this research is believed to produce results that can indicate the possibility for the development of a general Amharic spelling checker software for both non-word and real word spelling errors. The results of this study were expected to produce experimental evidences that demonstrate different application areas of unsupervised machine learning technique to spelling checker and corrector of Amharic texts.

As researchers' knowledge, there numerous limitations to Amharic NLP and not much research is undergoing by researchers in this field as compared to the need for NLP tools in Amharic language. The significance of the study can be considered very important in the Amharic language, we don't really have this kind of context based spell checker and corrector developed so far, this study could provide a lot of possibilities to enhance error detection and correction capability of Amharic spelling checker in sentences and transform one step ahead to our Amharic spelling checking and correcting applications. This study assists easy and more accurate way of detecting and correcting spelling errors for Amharic texts.

Since Amharic is an official language of Ethiopia a complete spell checker and corrector is vital to develop and promote the linguistic features of the language. So any one in country who writes spelling on the computer can detect and correct the typing errors during writing or inputting spells from the keyboard. The user can select appropriate, suggested words from the given suggested related terms based on the sequence of words in the sentences. This Amharic Spelling checking can be used in various applications like machine translation and information retrieval and recently it be used to develop a context sensitive spell checker and corrector for mobile application for mobile users and search engine queries.

On the other hand, the spelling checker can increase the speed and efficiency of checking spelling in the text and writers cannot spend more time correcting the word errors and gets appropriate terms that are depending on the context of the sentences. Developing the context dependent spelling checker model can facilitate to detect the non-real word spelling errors which do not exist in the dictionary which is not context sensitive and real word spelling errors which are actually found in the dictionary and that are not contextually correct. The developed Amharic context spell checker and corrector would be used that detect typing errors both non-real word and real word spelling errors and suggest related terms based on the syntax or context of sentences in the corpus and dictionary by considering the distance similarity of terms among in the written text.

The development of the system was one way of formalizing linguistic knowledge and thus can be considered as a form of documentation for poorly investigated languages. Prevention of culture breakdown is necessary due to languages represent the culture and diversity of different people around the world. Unavailability of language resources eventually leads to extinction. Failing to salvage the language will lead to extinction of the culture and consequently the people. Reduction of the language technology divide created between the languages of the developed nations and those of the less developed. Languages must endeavor to keep up with and avail of language technology advances if they are to prosper in the modern world.

Besides, the results of this study produce experimental evidences that demonstrate different application areas of machine learning technique to check and correct spelling in Amharic texts. The study contributed to future researches and development in the area of NLP specifically in machine translation, speech processing, text processing, information retrieval, grammatical analysis, content and thematic analysis as those areas require accurate spell checker and correction mechanisms. Therefore, the spell checker could be used as input for other NLP applications and can be integrated with them to resolve the challenges behind the applications.

1.6 Thesis Organization

This thesis is organized in six chapters. Chapter one, the present chapter, gives a general overview of the research with the research problem statement, objectives and methodology. Chapter two is devoted to a literature review. It discusses the concepts in spell checking and n-grams techniques in two sections. In the first section, concepts that underlie the experiment in this research are discussed in adequate detail. In the second, the n-gram approaches to interactive context sensitive spell checking and a review of works on n-gram related spell checking is presented. In chapter three, the characteristics of the Amharic writing system that are applicable to the research area are discussed to represent and process the Amharic texts in electronic format. The modeling and text preprocessing procedures for spell checker including described in chapter four. The experimental settings, the process of the experimentation and the findings are presented in chapter five. Finally, in chapter six general conclusions and recommendations are made based on observations and results from the experiment.

CHAPTER TWO

LITERATURE REVIEW

2. Introduction

This chapter deals with the state of the art relating to context based spell checking and correcting with its spelling error types in the written texts. In order to complete this study, literature reviews are necessary to analyze and understand the previous researches that have been done in spelling checking and correcting techniques. It plays an important role as the early phase to develop this study. The literatures can be explained different aspects of information on spell correction techniques and algorithms from various resources that have been done before the current spelling checker.

2.1 Spell Checking

The main tasks of a spell-check module are tokenization, error detection and correction, and ranking the suggestions. Tokenization is a language specific task that splits a text into meaningful elements called tokens. Most methods use dictionary directly to detect and correct non-word errors, although there are methods that work without using a dictionary for detecting and correcting real word errors in the given text [15]. Methods that use dictionary directly can differ in the way of storing their dictionaries. From this point, the whole methods can either use minimal redundancy or full listing approaches [36]. There are some other ways of saving the word list like using a dictionary as bitmap [49] or Ternary search tree [8]. The patterns of the errors can be categorized into four groups, (1) multi-word token and split errors, (2) typographical errors, (3) cognitive errors, and (4) phonetic errors [10]. Multi-word token errors are those errors, which happen due to missing space between two distinct words like ‘ofthe’ and split errors refer to having extra space between the letters of a word like ‘sp ent’.

Typographical errors deal with regular forms of mistyping like pressing a key on the keyboard twice or hitting the adjacent key mistakenly. Cognitive errors refer to those errors that happen because of misconception or lack of knowledge of the user like typing ‘recieve’ instead of ‘receive’. Phonetic errors are those errors that happen due to pronunciation similarities between the letters like typing the word ‘naturally’ as ‘nacherly’. There are many algorithms for correcting the errors such as Soundex, SPEEDCOP described in [49] and Metaphone just deal with phonetic errors and do not rank the list of suggestions.

There are also some other works that use n-gram models and neural networks [77] for error correction. Another method in spelling correction is finding minimum edit distance. Analysis of typographical errors in [16] states that about 80–95% of the errors in English texts are single errors that are caused by wrong insertion, deletion, substitution of one single letter, or transposition of two adjacent letters. The Damerau-Levenshtein distance refers to the minimum number of insertions, deletions, substitutions, or transpositions need to convert a word to the other word. In this model [16] after detecting the erroneous word, all the words that could be converted to this word with only single error are extracted from the lexicon.

A language-independent approach based on finite-state automata is introduced in [30] for automatic correction of spelling mistakes, using a dictionary and text data. In [48] a comparison of different strategies for finding the best spelling correct, including ranking heuristics, various correction algorithms, and priority strategies by using error types, syntactic information, word frequency statistics, and character distance is demonstrated.

2.2 Spelling Errors

Spelling errors are a rich source of information. Systematic spelling failures are thought to reveal aspects of the cognitive mechanisms of spelling and learning to spell. Moreover, spelling errors may be strongly dependent on the language-specific orthographic system and on the individual level of competence.

In the current research spelling errors are defined as human-generated writing errors. The term spelling error sometimes refers to both spelling errors and typing errors: automatic spelling error detection and correction aims at detecting and correcting both spelling errors and typing errors. In the current research this ambiguous denotation of the term spelling error is adopted. Also, the current research considers only human-generated spelling errors. While some techniques for detecting and correcting errors of optical character recognition (OCR) devices have been studied in the literature, most research has been done into techniques for detecting and correcting human-generated errors.

Detecting whether or not a word is correct seems simple, why not to look up the word in a set of all words. Unfortunately, there are some problems with this simple strategy. Firstly, a lexicon containing all correct words could be extremely large, which entails space and time inefficiency. Secondly, in some languages it is practically impossible to list all correct words, because they are highly productive. Thirdly, making a spelling error can sometimes result in a real word, which belongs to the lexicon such an error is called a real-word error. It is impossible to decide that this word is wrong without some contextual information. Fourthly, the bigger the lexicon, the more esoteric words it contains, making real-word errors more likely. Techniques for spelling error detection were designed on the basis of different spelling error trends these are also called error patterns. Studies were performed to analyze various trends in spelling errors. According to Damerau [16] spelling errors are generally divided into two types which are typographic errors and cognitive errors.

Typographical errors could be occurred when the correct spelling of the word is known, but the word is mistyped by mistake. These errors are mostly related to the keyboard and therefore do not follow any linguistic criteria. Whereas, cognitive errors were produced when the correct spellings of the words are not known and lack of knowledge about correct spelling of the target language. In these types of errors, the pronunciation of the misspelled word is the intended as correct word.

Phonetic errors were also a special class of cognitive errors in which the writer substitutes phonetically correct but orthographically incorrect sequences of letters for the intended word. This spelling errors mostly occurs when the language that have same pronunciation of alphabets with different orthographical writing system.

According to Bhagat [10], large number of spelling errors commonly encountered in human generated text and these errors were categorized on of the following error types. The first errors are substitution error occurs when at least one character is substituted by other character and the maximum of misspellings contains substitution errors in Punjab texts. The second type of errors is deletion errors produced when at least on character is deleted in the desired word. Whereas, when at least on character is inserted in the desired word insertion error could be produced. Also two adjacent characters can be transposed and the desired word can be changed which produces transposition errors. On the other hand, spacing is very important to identify and separate one word with others during typing. Based on this run-on errors can be produced when there is missing space between two or more words and split errors were occurred when there is extra space is inserted between parts of a word. All this errors were produce a real and non-word errors in the document writing which needs spell checker to detect and correct in desired language.

Non-words errors are spelling errors resulting in words that do not appear in the reference dictionary and real-word errors are words that are in the reference dictionary but are actually erroneous spellings of some other words [79]. A spelling checker would detect a misspelled word and depending on the level error, fine tune the word to provide a set of suggestions. These suggestions are a set of words a user probably intended to type. Non-word errors are relatively easier to detect and eradicate. Real word errors are more intricate ones. Usually, such error affects the syntax and semantics of the whole sentence, which in some cases requires human-being involvement for detection [78]. The use of spelling correctors should be handled with care.

2.3 Spelling Error Detection

The first part consists of identifying the errors in the typed text. This part uses a language model which accounts for the words allowed in the language. Language models may vary from a simple list of permitted words to finite state graphs that accept words with valid spellings in the language**.**

Error detection is the procedure of finding incorrectly spelled words in a text. A word that is considered incorrect is flagged by the spell checking application. Techniques to detect non-word spelling errors in a text can be divided into two categories: dictionary lookup and n-gram analysis. A non-word refers to a continuous string of characters and/or numbers that cannot be found in a given dictionary or that is not a valid orthographic word form. Dictionary lookup technique employs efficient dictionary lookup algorithms and/or pattern matching algorithms. N-Gram analysis makes use of frequency counts or probabilities of occurrence of N-Grams in text and in a dictionary or a corpus.

For the real-word case, however, detection necessarily involves having some model of what we expect the text to be like, so that we can tell whether those expectations have been violated. Real word spelling errors includes those errors where the misspelled word fits into the language model but, occurs as a misspelling of some other correct word. In other words, the word does not fit into the context of the sentence.

2.3.1 Dictionary Lookup Technique

The most popular mechanism of detecting errors in a text is simply to look up every word in a dictionary. Dictionary lookup is a straightforward task because it directly checks the presence of every input text in the dictionary. If that word present in the dictionary, then it is taken as a correct word. Otherwise, it puts into the list of error words. However, response time becomes a problem when dictionary size exceeds a few hundred words. In document processing and information retrieval the number of dictionary entries can range from 25000 to more than 250,000 words [10].

The most significant dictionary lookup techniques are hashing, binary search trees and finite state automata. Hashing [33] is a technique used for searching an input string in a pre-compiled hash table via a key or a hash address associated with the word and retrieving the word stored at that particular address. In spell checking problem, if the word stored at the hash address is same as the input string there is a match. If the word stored in the hash table is null the input word is indicated as a misspelling. This technique eliminates the large number of comparisons required for lookups. The following the techniques were used for gaining fast access to a dictionary.

Binary Search Trees are useful [33] for checking if a particular string, i.e. an input word exists within a large set of strings i.e. the dictionary. The main goal of binary search trees, particularly median split tree is to make access to high frequency words faster than to low frequency words. It is efficient compared to the lookup time of a linear search technique on a large data representation, although it is slower compared to the lookup time of hashing.

Finite state automata also used as a basis for string matching or dictionary lookup algorithms that locate elements of a dictionary within an input text. One specific form of the FSA that has used for spell checking and correcting purposes is a tree data structure. Tries are also known as prefix trees [33]. Finite state approaches are used for spelling correction for agglutinating languages or languages with compound nouns.

The most common technique for gaining fast access to a dictionary is the use of a HashMap data structure.to look up an input string one simply computes its hash address and retrieves the word stored at that address in the pre-constructed hash table. If the word stored at the hash address is different from the input string or is null, a misspelling is indicated. The main advantage is that the random access nature of a hash code eliminates the large number of comparisons needed for sequential searches of the dictionary. Therefore, concerning these issues we have chosen HashMap structure for gaining access quickly from the dictionary lists.

2.3.2 N-gram analysis

N-gram analysis is used to detect incorrectly spelled words in a mass of text. Here instead of comparing the complete word in a text to a dictionary, only the n-grams are compared with a dictionary because comparing each single word with dictionary is a time consuming process. It uses n-dimensional matrix, where the actual n gram frequencies are toured is used for spell checking [51].

If a non-existent or rare n-gram is detected the word is flagged as an error or misspelled, otherwise not. An n-gram is a set of consecutive characters taken from a string with a length of n. If the value of n is set to one, then it is called unigram, if n is two, then it is a bigram, similarly if n is three then the term is trigram. Every string that is involved in the comparison process is split up into sets of adjacent n-grams. The major advantage of n-grams algorithms are that they require no knowledge of the language that it is used with and so it is often called language independent algorithm [51].

In general n-gram detection techniques work by examining each n-gram in an input string and looking it up in a precompiled table of n-gram statistics to ascertain either its existence or its frequency of words or strings that are found to contain nonexistent or highly infrequent n-grams are identified as either misspellings. N-gram techniques usually require either dictionary look up techniques or a large corpus of text in order to pre-compile an n-gram table. Dictionary lookup techniques work simply checking to see if an input string appears in the dictionary that is a list of acceptable words. In this thesis bigrams were chosen and extracted from the input sentences at word level rather than character level during error detection. The nonexistent bigram words were detected as errors in the input sentences and needs a correction form the bigram list by considering the distance between each bigram errors with dictionary words.

2.4 Non-word Error Correction

This is the second spelling checker tasks consist of rectifying the spelling mistakes made by the user. Spelling error correction is the procedure of correcting an error once it has been detected. An error is corrected when the spell checking application or the user replaces an erroneous word by the word that the user intended. Sometimes, the term error correction is used to refer to the processes of error detection and correction together. In this research, consistently adopt the distinction between error detection and correction.

Spell correcting refers to finding the subset of dictionary or lexical entries that are similar to the misspelling in some way. Spell checking can be categorized by isolated word error correction and context dependent error correction. Isolated word error correction thus refers to spell correcting without taking into account any textual or linguistic information in which the misspelling occurs. Therefore, the corrections are based only on the misspelled word itself. A context dependent corrector would correct both real word errors and non-word errors involving textual or linguistic context. Real-Word errors are those spelling errors, which result in valid words of language that are not the actual intended words, for example writing “form” for “from”. Such errors can never be caught without using contextual information. Contextual information can be used for ranking the suggested corrections, especially when more than one suggestions otherwise seem equally likely for being the actual correction.

Early work in the area of spell checking was more focused on isolated-word error correction, but with the passage of time, the number of such applications increased where auto-correction was a requirement, for example in applications like Text to Speech Synthesis systems, Machines Translation systems or other NLP related systems. In such applications the spell checker should be capable of catching real word errors. Moreover it should also be capable of deciding one best correction, and this can be achieved only if the context information is also used for correction.

Kukich [37] pointed out that 80% of spelling errors tend to be single-letter errors, such as insertions, deletions, substitutions and transpositions. Spelling error correction relies on some approximate string matching technique to find a set of correctly spelled words in the dictionary that satisfy a similarity relation. This involves the association of a misspelled word with one word or a set of correctly spelled words in the dictionary that satisfy a similarity relation [33].

According to [16] Error correction consists of two steps: the generation of candidate corrections and the ranking of candidate corrections. The candidate generation process usually makes use of a precompiled table of legal n-grams to locate one or more potential correction terms. The ranking process usually invokes some lexical similarity measure between the misspelled string and the candidates or a probabilistic estimate of the likelihood of the correction to rank order the candidates.

Error pattern analysis of each language helps in developing an efficient spell checker which includes analysis of various error types (insertion, deletion, substitution, transposition, run-on, and split word errors), positional analysis, word length effects, phonetic errors, and keyboard effects. Furthermore, there are many ways of writing the same word and all the ways could correct. So it may be necessary to collect the raw typed text as the data for analysis. Because of the raw texts does surly direct us to the typing and spelling mistake of that word. The main target of this thesis is to analyses the typing and spelling mistake since the study would be used to design a suggestion list for Amharic spell checker.

2.4.1 Minimum Edit Distance

Minimum edit distance is the most studied technique for spelling correction. The minimum number of editing operations (insertions, deletions, substitutions and transpositions) required to transform one string into another. This simplest method is based on the assumption that the person usually makes few errors if ones, therefore for each dictionary word the minimal number of the basic editing operations necessary to convert a dictionary word in to the non-word. The lower edit operation has higher the probability that the user has made such errors. Through the operation of adding, deleting and modifying, edit distance changes a word into the minimum operating frequency of another word. The dictionary word that is at the shortest distance from the misspelling is suggested as the most probable correct word. The words beyond a pre-specified threshold edit-distance are ignored. Wagner [62] introduced the notion of edit distance for spelling correction. Minimum edit distance has different algorithms like Levenshtein algorithm, Hamming, Longest Common Subsequence [54].

Levenshtein algorithm is a weighting approach to appoint a cost of 1 to every edit operations (Insertion, deletion and substitution). Levenshtein edit distance produces a similarity score for the query term against each lexicon word in turn. For instance, the Levenshtein edit distance between "dog" and "cat" is 3 (substituting d by c, o by a, g by t). Hamming algorithm also measure the distance between two strings of equal length. For instance, the hamming distance between "sing" and "song" is 1 (changing i to o). On the other hand, Longest Common Subsequence algorithm is a popular technique to find out the difference between two words. The longest common subsequence of two strings is the mutual subsequence.

2.4.2 Similarity Keys

The essence of similarity key techniques is the mapping of every word in the key. The mapping is chosen so that similarly spelled words would either have similar or identical keys. When a key is computed for a misspelled word it would provide a pointer to all similarly spelled words in the dictionary and these dictionary entries would be returned as candidate correction. This is to say, all words in a dictionary having similar key values compared to the key of the current misspelled word, will be returned as possible correct words. Due to the fact that it is not necessary to compare the misspelled word with every dictionary entry, similarity techniques are fast. Similarity key mechanisms are based on transforming words into similarity keys that reflect the relations between the characters of the words such as positional similarity, material similarity and ordinal similarity [33] of words.

A key is assigned to each dictionary word and only these keys are compared with the key computed for the non-word. The words for which the keys are most similar are listed as suggestions. Such an approach is speed effective only if the words with similar keys have to be processed with a good transformation algorithm. This method can handle keyboard errors.

2.4.3 Rule-based Techniques

Rule based techniques involve algorithms that attempt to represent knowledge of common spelling error patterns for transforming misspelled words into correct ones. The knowledge is presented as rules. These rules can contain general morphological information of words, lengths of words and more. The candidate suggestions are generated by applying all applicable rules to a misspelled word and retaining every valid word in the dictionary that results [33]. Ranking on the suggested words is based on a predefined estimate of the probability of occurrence of the error that the particular rule corrected.it is completely independent of any grammar or parsing formulation. It can be a mere lexical lookup routine.

These techniques have a set of rules that collect common spelling and typographic errors and applying these rules to the misspelled word. Each correct word generated by this process is taken as a correction suggestion. The rules also have probabilities, making it possible to rank the suggestions by calculating the probabilities for the applied rules. Edit distance can be viewed as a special case of a rule-based method with limitation on the possible rules [51].

2.4.4 N-gram Based Techniques

N-gram based technique can be used in two ways, either together with a dictionary or without having a dictionary. N-grams used without a dictionary are employed to find in which position in the misspelled word the error occurs. The performance of this method is limited. Its main virtue is that it is simple and does not require any dictionary. Together with a dictionary, n-grams are used to define the distance between words, but the words are always checked against the dictionary.

Therefore, in this work n-grams were integrated with dictionary to increase the performance of correcting the bigram errors in the input sentences. Any word can be checked for errors for errors by simply looking its corresponding entries in the bigram list to make sure that all are exist.

2.4.5 Probabilistic Techniques

N-gram based techniques led naturally to the probabilistic technique in both text recognition and spelling correction paradigms. This technique is based on some statistical features of the language. Two common methods are confusion probabilities and transition probabilities. Transition probabilities are similar to n-grams. This give the probability that a given letter or sequence of letters is followed by another given letter. Transition probabilities are not very useful when we have access to a dictionary or index. Given a sentence which has to be checked, the system decomposes each string in the sentence into letter n-grams and retrieves word candidates from the lexicon by comparing string n-grams with lexicon entry n-grams. The retrieved candidates are ranked by the conditional probability of matches with the string, given character confusion probabilities. And, a word-bigram model and a certain algorithm are used to determine the best scoring word sequence for the sentence [54].

2.4.6 Neural Networks

Neural networks are also an interesting and promising technique. The current methods are based on back propagation networks, it uses one output node for each word in the dictionary and an input node for every possible n-gram in every position of the word, and where n is one or two. Only one of the outputs should be active, indicating which dictionary words the network suggests as a correction. This method works for small dictionaries, but it does not scale well. In the learning phase the time requirements are too big on traditional hardware.

According to [33] neural networks are potential candidates for spelling correction due to their ability do associative recall based on incomplete or noisy input. Neural networks have that ability to adapt to the specific errors patterns of a certain users domain they can be trained on actual spelling errors. For training a neural net Back Propagation Algorithm is the most widely used one.

2.5 Context Based Error Correction

Context-sensitive spelling error correction is the task of detecting and correcting spelling errors that result in valid words, i.e. real-word errors. For instance, in the sentence "you should constantly backup your computer flies", the word "flies" is a real-word error mostly caused by a typographical mistake. Obviously, the writer didn't intend to mean that computer flies like planes, but he most probably meant "computer files". This slight confusion produced a real-word error that is actually valid in the English dictionary, however invalid with respect to the sentence in which it has occurred. Context-sensitive spelling error correction tries to detect and correct such real-word errors by inspecting their grammatical and semantic contexts.

Error correction based on grammatical context or syntactic context, attempts to apply grammatical rules to detect misspellings, for instance, asserting that the word "play" in the sentence "he play" is a grammatical error is true since in the English language, a third person verb in the present tense must always ends with a "s". In contrast, error correction based on semantic context can correct the word "peace" into "piece" in the sentence "peace of cake". Since the words "peace" and "piece" are valid nouns in the English language, they are hard to be flagged by traditional non-context-sensitive spell checkers.

According to Kukich [40], the problem of spell checking can be classified in three categories of increasing difficulty: non-word error detection, isolated-word error correction, and context-dependent word correction. The real-word errors detection and correction task, the focus of this paper, belongs to the third category. Such errors are the most difficult to detect and correct, because they cannot be revealed just by a dictionary lookup, but can be discovered only taking context into account.

Mays, Damerau and Mercer [46] proposed using the n-gram model to predict the actual correction of a real-word error. The idea centers on generating candidate spellings for every misspelled word by only applying simple edit operations such as insertion, deletion, and substitution, and then using n-gram statistics derived from a corpus to compute P (wn|wn-1). Church and Gale [22] also suggested that the use of a noisy channel to predict the actual correction of a real-word error. The technique harnesses a 100 million word corpus and n-gram statistics to correct errors according to their contextual information.

Apart from Mays et al. [46] and Church et al. [14], several other methods have been proposed to handle real word spelling error problem. They are mainly based on either semantic information or machine learning and statistical method. Different approaches to tackle the issue of real-word spell checking have been presented in the literature. Symbolic approaches [31] try to detect errors by parsing each sentence and checking for grammatical anomalies. More recently, some statistical methods have been tried, including the usage of the word n-gram models [46, 8], POS tagging [23, 27, 76], Bayesian classifiers [22, 65], decision lists [65], Bayesian hybrid methods [28], a combination of POS and Bayesian methods [27], and Latent Semantic Analysis [35].

The main problem with word n-grams are data sparseness, even with a fairly large amount of training data. In fact, a recent study [27] reported better performances using word bigrams rather than word trigrams, most likely because of the data sparseness problem. POS based methods suffer less of sparseness problem, but such approaches are unable to detect misspelled words that are of the same part of speech. Bayesian methods, on the other hand, are better able to detect this case, but have worse general performance. These last two methods give better results when combined together.

Additionally, different researchers proposed statistical method based on a language model that is a combination of the word-trigrams model and the POS-trigrams model which is a mixed trigram model. The main linguistic motivation behind this model is to represent fine-grained lexical information at a local level, and summarize the context with syntactic categories. The main advantage of this model is a great reduction of the data sparsity problem. A slightly different application area in which statistical contextual spell checking have been also studied is Optical Character Recognition

(OCR). For this application, Markov Model based approaches using letter n-grams have been shown to be quite successful [59].

On the other hand, Golding and Schabes [27] introduced a hybrid approach called 'Tribayes' combining Trigram and Bayes' method. Trigram method uses part-of-speech trigrams to encode the context, whereas Bayes' is a feature-based method. They use two types of features: context word and collocations. Their method worked better than MS-Word on a predefined confusion set. Another approach was proposed by Demetriou, Atwell and Souter [19], based on semantic knowledge and large vocabulary to correct spelling errors. A semantic model was built based on semantic association between words in a text to largely decrease the semantic ambiguities in natural languages.

Later Golding with Roth [26] proposed a Winnow-based method for real word detection and correction. They modified the previous method [65] by applying a winnow multiplicative algorithm combining variants of winnow and weighted majority voting and achieve better accuracy. However, they used a small data set in their experiment. Liu and Curran [42] also employed n-gram statistics to correct real-word errors using a big corpus of text collected from crawling the web. As a result, huge improvements were achieved due to the large volume and generality of web corpuses.

Hodge and Austin [77] proposed a supervised learning spell checking methodology based on a Hamming distance algorithm and on an n-gram model for detecting isolated word errors. The generated candidate spellings are ranked based on their Hamming distance and n-gram statistics. In due course, candidates having the highest score are selected as correct for the detected real-word errors.

Carlson and Fette [12] employed the same previous technique, but instead a memory-based learner was used to correct cross domain errors. The system was trained using n-gram data tokens extracted from the web. The experiments yielded high precision real-word and non-word error correction.

All real word error correction techniques either require matured knowledge of the syntax of the language or extensive balanced corpus of the language. The languages, for which neither of the two is available, cannot reach the goal of real word error correction.

2.6 Candidate Corrections

Once a string has been detected as an error, an error correction technique aims at finding candidate corrections for the erroneous word. Several algorithms for finding candidate corrections have been explored. The most popular method by far is computing the minimum edit distance between the detected string and a lexicon entry. The minimum edit distance has been defined as the minimum number of editing operations (i.e. insertions, deletions and substitutions) that is required for transforming one string into another. The first minimum edit distance spelling correction algorithm based on these three types of character transformation was implemented by Damerau [19]. Levenshtein [41] developed a similar algorithm for correcting deletions, insertions and transpositions. Other researchers developed variants of the algorithms that were developed by Damerau and Levenshtein. Wagner and Fischer [62] generalized it to cover also multi-error misspellings and Lowrance and Wagner [43] extended the algorithm to account for some additional transformations, such as the exchange of nonadjacent characters. Some minimum edit distance algorithms that have been explored do not only use orthographic distance scores, but also phonetic similarities.

Veronis [61] devised an algorithm that calculates weights for the orthographic edit distance based on phonetic similarity. These weights are important to be able to find phonetic misspellings, because often, phonetic misspellings are a large number of editing operations removed from the intended word. If only orthographic information is taken into account, the intended word will most probably not be among the candidate corrections. Minimum edit distance techniques have been applied to virtually all spelling correction tasks. An advantage of using a minimum edit distance measure is the fact that ranking can be performed easily.

Dictionary and context dependent spelling checker have a better performance and accuracy to detect and correct spelling errors in proper manner. For this research dictionary and context based spelling checker were integrated to suggest candidate alternatives using the Levenshtein minimum edit distance and ngram probabilities for non-word and real word errors respectively.

CHAPTER THREE

AMHARIC WRITING SYSTEM

3. Introduction

This chapter gives a brief description of the Amharic writing system by focusing mainly on the electronic representation of Amharic characters, punctuation and numbers. Amharic uses its own alphabets, numbers, punctuation marks, etc., for its writing system.

Amharic was the national language of Ethiopia until 1983 E.C. Currently it is the official language of the Federal Government of Ethiopia. Moreover, it is the working language of different governmental and non-governmental organizations throughout the country. Mass Media like radio, television broadcasts and the press are also using it for disseminating information to the public.

As a result of its wide application, large Amharic documents are compiled both in hard copy and electronic forms. Like any documents of another language, the contents or meanings of these documents are represented using important features of the language. For the purpose of this research since Amharic spells are considered, it is important to investigate these potential features are the capability of representing the contents of the texts, which in turn demands one to understand the characteristics of the language in particular. Hence, under this section, important features of the Amharic spells that are believed to be pertinent to the current research will be reviewed.

Since the Amharic language is phonetic, it is really important to deal with the spelling and phonology of the language. This provides us a better insight to explore ideas that are directly related to this. Despite the large number of speakers, the language has very few computational linguistic resources. This has a direct impact specially to implement research works that are done so far and to be done also in the future. Spell checker for Amharic language is one of those areas that are not well explored. Even if this thesis targets on Amharic spelling checker and corrector for Amharic words, it would discuss related issues which are important for the research work. We looked at the Amharic alphabets and the spelling of the language in the coming sections.

3.1 History of Amharic language

Amharic is a Semitic language and the official language of the Federal Government of Ethiopia. It is the second most spoken Semitic language in the world, next to Arabic and is estimated to be spoken by over 20 million people as their first or second language [71]. The current Amharic writing system was adopted from the Ge'ez writing system, which was the classical language of the Axum Empire of Northern Ethiopia [6]. It existed between the 1st Century AD and the 6th Century AD that the ancient Sabaean script is in turn attributed as the source of the Ge'ez script. As the Sabean script descended into Ge’ez and later into Amharic, the numbers of symbols in its original Sabean script and their shapes have been changed [67].

When the power base of Ethiopia shifted from Axum to Lalibela between the 10th and 12th Century AD, the use of the Amharic language spread its influence, hence became the national language of the country until 1983 E.C. A wide variety of Amharic literatures including books, religious writings, fiction, poetry, plays, and magazines are available both in printed and machine readable format.

Ethiopia is a linguistically diverse country where more than 80 languages are used in day-to-day communication. Amharic is the working language of the Federal Government of Ethiopia and is spoken and written as a first or second language in many parts of the country [6]. Amharic, like other languages that use the Geez script (Gurage, Harari, Tigre, and Tigrinya), use characters derived mainly from Geez. It is the second most spoken Semitic language in the World (after Arabic) and today probably one of the five largest on the African continent (albeit difficult to determine, given the dramatic population size changes in many African countries in recent years) [63].

Amharic uses a unique script, which has originated from ancient language, the Ge’ez alphabet, which is the liturgical language of the Ethiopian Orthodox Church. Manuscripts in Amharic are known from the 14th century and the language has been used as a general medium for literature, journalism, education, and so on [25]. Amharic language script has 33 core characters and of each 32 of them are consonants having seven orders to show the seven vowels. Out of the seven derivatives six of them are CV (Consonant vowel) combinations while the sixth is the consonant itself [7]. Other symbols representing labialization, numerals, and punctuation marks are also available.

3.2 Amharic Writing System

Today, Ge’ez is no longer the mother tongue of any living person in Ethiopia. Ge’ez is classified as a sacred language that is still used in the culture of highland Ethiopia as the traditional language of literature and religion. Today, people speak Amharic in their daily life. Amharic is born from original Ge’ez script and has further evolved to include more characters in the character set [68].

The present Amharic writing system was adopted from the Ge’ez writing system. Ge’ez, which belongs to the class of Semitic languages, was the language of literature in Ethiopia in earlier times [6]. According to Bender et al. [6], three writing systems are in use in Ethiopia, the Ethiopic (Ge’ez) syllabary, the Roman alphabet, and Arabic script. The widely used Ethiopic syllabary, which is derived from the writing system of ancient South Arabian alphabet, is used for Ge’ez, Amharic, Tigrigna and other semantic languages. The writing system has a similarity with some Semitic languages like Arabic in having vowel marks added to basically consonant letters.

Moreover, some new symbols have been added to Amharic. Amharic did not discriminate in adopting the Ge’ez fidel; it took all of the symbols [1] and added some of its own. Although Sabaean is not used currently, Ge’ez is still used especially as a language of liturgy (mass) in the Ethiopian Orthodox and Catholic churches and in church literature. When Ge’ez became the spoken and written language, it took over only twenty-four of the twenty-nine symbols from Sabaean script [25]. In Ge’ez, two new symbols were created to represent sounds of Greek and Latin loan words, ጰ/p’/ and ፐ /p/ (e.g. ጳጳስ and ፖሊስ Baye) [1].

Ge’ez in turn took its script from the South Arabian mainly attested in inscriptions in the Sabaean dialect [6]. The original Sabaean alphabet is said to have had 29 symbols. When Ge’ez became the spoken and written language in common use in northern Ethiopia, it took only 24 of the 29 Sabaean symbols, modify most of them and add two new symbols to represent sounds of Greek and Latin loan words not found in Ge’ez, these symbols are ጸ and ፐ. The style of the writing was also modified to left to right. By the time Ge’ez ceased to be a living spoken and written language and replaced by Amharic and other languages, further changes took place. Amharic did not discriminate in adopting the Ge’ez fidel; it took all of the symbols and added some new ones that represent sounds not found in Ge’ez. The added alphabetic characters are ቸ, ጨ, ጀ, ኘ, ቨ, ሸ, ኸ, and ዠ.

One of the results in the development from Ge’ez is redundancy in the number of symbols with the same pronunciation. For example, the three different symbols ሀ, ሐ and ኀ (all with the same pronunciation; h) are used interchangeably in text written in Amharic although they gave different meanings to words in the Ge’ez language. Likewise, ሰ and ሠ, and አ and ዐ have the same pronuounstion which have different symbols. This redundancy has been recognized in literature as a problem of the language [25].

Currently, the language’s writing system contains 33 base characters each of which occur in a basic form and six other forms known as orders. The seven orders represent syllable combinations consisting of a consonant following vowel. This is why the Amharic writing system is often called syllabic rather than alphabetic, even if there is some opposition. The 33 basic characters and their orders give 231 distinct symbols. In addition, there are forty others that contain a special feature usually representing labialization e.g. ጇ, ቋ. In Amharic there is no Capital-Lower case distinction. There are also punctuation marks and numeration system.

Unlike other Semitic scripts such as Arabic and Hebrew, Amharic is written from left to right, there are also no systematic variations in the form of the symbol according to its position in the word [58].

3.3 Amharic Alphabets

The transformation of the base form into the non-basic forms indicates that the Amharic writing system does not use independent symbols for vowels in representing a syllable. As Bender [6] explains, this is a characterization known as syllabic. However, currently there is a debate whether the language is actually syllabic or alphabetic [1, 34]. Alphabetic writing systems are systems that present the consonants and the vowels separately such as the English and Greek language.

On the other hand, syllabic writing systems are systems that combine both the consonant and the vowel together (e.g. Amharic writing system). However, [1] argues that Amharic is alphabetic on the grounds that each symbol can be broken down into consonant and vowel phonemes which can be independently represented by separate symbols. In fact, he describes the Amharic script in terms of 27 consonant and 7 vowel phonemes.

The current Amharic writing system consists of a core of 33 base characters (ፊደል, FIDEL) each of which occurs in a basic form and in six other forms known as orders [25]. The non-basic forms are derived from the basic forms by more-or-less regular modifications. Thus there are 33 core characters which give 231 distinct characters. Therefore, the FIDEL has 275 characters (letters) to be used in the writing system. The seven orders represent syllable combinations consisting of consonant and following vowel. This characteristic according to [6] makes the Amharic writing system a syllabic writing system. The seven orders (the first basic order and the other six orders) of the Ethiopic script represent the different sounds of a consonant-vowel combination (a characterization known as syllabic). In addition to the 231 basic characters, there are also four labio-vellars (like ቈ ኈ ኰ ጐ) each having five orders and twenty additional labialized consonants. (Refer Appendix 1 for a complete list of Amharic character)

Amharic has borrowed most of its characters from Geez and thus the Amharic writing uses characters created by a CV fusion. Out of the 33 basic forms, two of them represent vowels in isolation (**አ** and **ዐ**) [73]. In this thesis we consider the Alfa-A/**አ**/ and its variation while considering the Amharic vowels. Seven vowels are used in Amharic each of which comes in seven different forms (orders) reflecting the seven vowels sounds (አ ኡ ኢ ኣ ኤ እ ኦ). A character or a symbol is used to represent a phoneme, which is a combination of a vowel and a consonant. Six of them have this CV combination while the seventh is the consonant itself [**72**].

C/e/ C/u/ C/ii/ C/a/ C/ie/ C C/o/

From the above representation, we can see that the sixth order in the orthographic symbols, which do not have any vowel unit associated to it in the written form (CV transcription of the orthographic form), may associate the vowel /**ix**/ in its spoken form, which has important role during syllabification of the word in the language which allows splitting impermissible consonant clusters. Even if it may have different representation in some other literatures [74], in this work we preferably follow the transliteration presented in [31].

Order	**1st**	**2nd**	**3rd**	**4th**	**5th**	**6th**	**7th**
V C	**E**	**U**	**Ii**	**A**	**ie**	**ix**	**o**
/m/	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
/b/	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ

Table 3.1: Amharic constants /በ/and/መ/ with their associated vowels [70]

The IPA (International Phonetic Association- responsible for standardizing representation of the sounds of spoken language) defines a vowel as a sound, which occurs at a syllable center. A chart depicting the Amharic vowels in the IPA representation is shown Figure 3.1 [72]. The IPA maps the vowels according to the position of the tongue. The vertical axis of the chart is mapped by vowel height. Vowels pronounced with the tongue lowered are at the bottom, and vowels pronounced with the tongue raised are at the top. For example, / **አ** /-[a] (as the [a] in ‘Beal"/ **በአል** /)is at the bottom because the tongue is lowered in this position. However, [ix] (said as the vowel in "Enat"/ **እናት /**) is at the top because the sound is said with the tongue raised to the roof of the mouth.

In a similar fashion, the horizontal axis of the chart is determined by vowel backness. the tongue moved towards the front of the mouth (such as the [ie] vowel in "Bet") are to the left in the chart, while those in which it is moved to the back (such as the vowel [o] in "Sost" / **ሶስት** /) are placed to the right in the chart. As mentioned earlier, in places where vowels are paired, the right represents a rounded vowel (in which the lips are rounded) while the left is its unrounded counterpart. The central vowels are also considered to be unrounded.

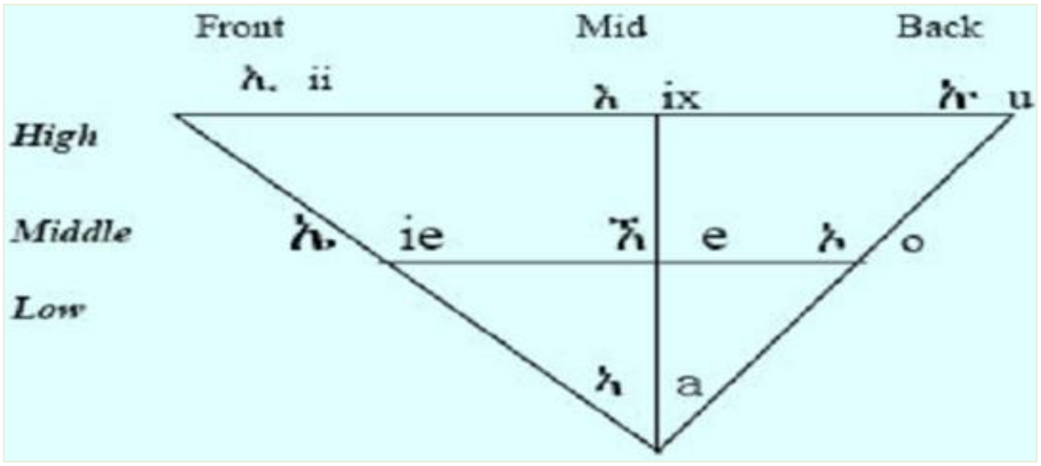


Figure 3.1: IPA maps of the Amharic vowels [72]

3.4 Amharic Number

Numbers in Amharic consist of single characters for one to ten, for multiples of ten (twenty to ninety), hundred, and thousand. According to [6] these characters are derived from Greek letters, and some were modified to look like Amharic character. Each of the symbols has a horizontal stroke above and below. There is no symbol for zero in the Amharic script. Also it is widely applied in the environment of Ethiopian Orthodox Church. Thus, arithmetical computations using the symbols are very difficult, if ever done. As a result, people generally use the Hindu-Arabic numerals. Ethiopic numbers are used mostly in writing dates and page numbers in text.

፩	፪	፫	፬	፭	፮	፯	፰	፱	፲
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
፳	፴	፵	፶	፷	፸	፹	፺	፻	፼
20	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**	**1000**

Table 3.2: Amharic number system

3.5 Amharic Punctuation Marks

Analysis of Amharic texts reveals that different Amharic punctuation marks are used for different purposes. The Amharic writing system uses some indigenous and foreign punctuation marks (signs) in addition to the Amharic characters [20]. There are a number of symbols for punctuation in Amharic. According to Beletu [4] (as quoted in Zelalem [66]) there are about 17 punctuation marks. Only some of them are commonly used and have representations in Amharic software. The following are the most commonly used both in handwritten and computer written text.

The word-separator (hulet Neteb), two square dots arranged like colon (:), and sentence-separator (arat netb), four square dots arranged in a square pattern (: :), are the basic punctuation marks in Amharic writing system that are used consistently. Lists in Amharic text are separated by an equivalent of comma, 'netela serez (፣) followed by ASCII space and 'derib serez' (፤), which is the equivalent of semi-colon. The use of '...' for question mark is not used rather a '?' which is borrowed from English is used. Table 3.1 lists the most commonly used Amharic punctuation with their equivalent in English which is adopted from [63]. Others include borrowed symbols like?,!, ", ", ', / and \.

As far as the application of these punctuation marks is concerned, the word delimiter (two dots) is mostly used in handwritten text but it is becoming a common practice to exclude it from computer written text. Hulet Neteb (:) is no longer used and space is being used as word separator. In case of sentence delimiter, the four dots continue to be used. The remaining punctuation marks are used where appropriate.

Amharic	English
፡	**White space**
።	**.**
፤	**;**
፣	**,**
…	**?**

Table 3.3: Commonly used Amharic punctuation marks corresponding English marks

However, Amharic words in a text are separated by above punctuation marks, there punctuation symbols that are important to connect and separate words in the sentences. Consequently, In Amharic texts the punctuation mark '-' the equivalent of hyphen in English, is used to form compound words. However, in the test collection this punctuation mark was not used consistently. The same compound words were found written both as separate words without the hyphen mark and as compound words with hyphen (example, ጸረ ሙስና and ጸረ-ሙስና). To keep consistency throughout the test collection, a decision was made to replace the one character space with hyphen mark and split compound words into their constituent terms.

3.6 Processing Amharic Texts

This research uses the unsupervised approach to detect and correct spelling errors foe Amharic texts. In order to check and correct spelling mistakes identification of word features is neccecery to represent the texts. As there can be millions of words in text datasets storage and processing time costs require document processing for efficient and reliable spell checking and correction. Text processing is therefore an important task to get features that adequately represent a document without being redundant and irrelevant. In this research the nature and characteristics of the Amharic writing system are considered during the processing of the Amharic texts of the source dataset.

3.7 Characteristics of Amharic Spelling

As it was discussed in many literatures, the Amharic writing system has many features, which may cause some problem from the perspective of computation [1, 6]. The characteristics of the Amharic writing system considered in this section are limited to those that are common to the dataset.

Character Redundancy: Amharic took the whole Geez alphabet (all seven orders of the 26 symbols of Geez) without considering whether all the 26 characters have meaning in the Amharic writing system. It then added some more symbols for some other sounds that it has and that could not be represented by the symbols of the Geez alphabet. This unsystematic borrowing from Geez has resulted in redundant characters in the Amharic FIDEL.

The different symbols with the same pronunciation also pose a problem in making words appear different (not in meaning, but in spelling.) Although in the Ge’ez language, these different symbols give each word different meanings, in the Amharic language they have been used [1, 6]. As a result, in Amharic writing system, there has been found different symbols with the same pronunciation and meaning (i.e., in Geez those symbols are different in meaning as well as in spelling, which is not the case for Amharic) and they have been used interchangeably [1, 6]. As [1] noted, however, for the case of Amharic there is no defined rule that differentiates their proper usage.

In Amharic, these consonants with the same sound falls into two categories: (1) the first and the fourth order alphabets of the same base form having the same sound and (2) different alphabets with the same sound.

For the first case, for instance, it is not clear whether one should write "ሀይማኖት" (`religion) and "ሃይማኖት” since both "ሀ" and "ሃ" have the same sounds. Those alphabets that exhibit such characteristics are listed in table 3.4.

First order	Fourth order
ሀ	ሃ
ሐ	ሓ
ኀ	ኃ
አ	ኣ
ዐ	ዓ

Table 3.4፡ Amharic different alphabets with same sound at first and forth order

Similarly, table 3.5 shows lists of different alphabets that have the same meaning and sound. Here, not only the base forms listed have the same sound, but also all the corresponding orders (6 orders) of them have the same sound too. For example, writing "ሰላም" and “ሠላም" to mean "Peace" does not make a difference in meaning even though "ሰ" and "ሠ” are used interchangeably. The same holds true for "አጥንት" (Bone) and "ዐጥንት” although "አ" and "ዐ" are two different alphabets with the same sound.

Alphabet	Other alphabet with same sound
ሀ	ሐ , ኀ
ሰ	ሠ
አ	ዐ
ጸ	ፀ

Table 3.5: Different alphabets having the same sound

Spelling variations of a word would unnecessarily increase the number of words representing a document which could reduce the efficiency and accuracy of the spelling checker. Moreover, a complex case comes when the same word appears to be in many forms (more than two forms) by using interchangeably these alphabets having the same sound. We can take "ፀሀይ", "ጸሀይ", "ጸኀይ", "ፀሐይ", "and "ፀኀይ" as a good example, which refers to the name of a sun (tsehay). As all the above discussion indicates, there arise some confusion and inconsistencies in Amharic alphabet and as a result these redundant consonants add their contribution to make the vocabulary to be large. Even if the size of dictionary and corpus size increases the spell checker of Amharic should consider to all this variation of word. In this research the character redundancy is included in the text that indicates the canonical and common Amharic forms of a language. In the case of canonical Amharic the word writing systems for redundancy characters were taken from Geez language. For example, for example, the word “Alem” which means world is written as ዓለም taken from Geez vocabulary and which is a canonical Amharic. But አለም is a common Amharic in which most Amharic writer was used to write this word. Furthermore, the above word has a possibility to be written as in different forma like ዐለም and ኣለም which is an improbable Amharic.

According to Bethlehem [9]Uniform substitutions may be made for similar sound letters in worlds to group words by shared strings since such substitutions do not make any changes in meaning in the Amharic language, unlike the Ge’ez in which they have significance for the meaning (e.g. ሰረቀ and

ሠረቀ) ` meaning “he stole” and “it penetrated, or rose (as in the rise of the sun)” respectively in Ge’ez, and both meanings either of the two in Amharic).

Compound Words: In the Amharic writing system, inconsistency is often observed regarding the representation of compound words. The writing system there is no agreed upon standard in spelling compound words. There are different ways of writing compound words without affecting their meaning [5]. That means, at one time the compound words can be written as two separate words and at another time as single words (either by fusing the two words or by inserting a hyphen between them). For instance, it makes no difference in meaning at all while writing the compound word "*ወጥ ቤት*" as one word "*ወጥቤት*" which is to mean that "Kitchen”. Additional examples of such Nouns are mentioned in table 3.6.

Compound word as Single word	Compound words used as Separate words	Literal English meaning
ብረትድስት	**ብረት ድስት**	**Metal cooking pot**
አዲስአበባ	**አዲስ አበባ**	**Addis Abeba**
ቤተመቅደስ	**ቤተ መቅደስ**	**Temple**
ትምህርትቤት	**ትምህርት ቤት**	**School**
ማዕድቤት	**ማዕድ ቤት**	**Dining room**

Table 3.6: Writing compound words in Amharic texts

Occasionally, the constituent terms may have completely different meaning from the compound word formed from them. For example, the word 'hode-sefee’ (*ሆደ-ሰፊ*) which means ‘tolerant’ has a different meaning from the constituent terms ‘hode’ which means ‘stomach’ and ‘sefee’ which means ‘wide’.

The inconsistent usage of compound words could result in redundant word features by creating more words when a compound word (example አዲስ-አበባ) is treated as two separate words አዲስ and አበባ. It may also result in a semantic loss by confusing a document about the city, Addis Ababa (አዲስ-አበባ) with the one talking about the floral industry.

Variations of Pronunciation: usage of foreign language words in Amharic (transliteration) is also found to be another source of word spelling variations. The Transliteration of foreign words into Amharic writing system is one of the main causes of this irregular spelling of words. Amharic language lacks some basic English sounds. When foreign terms are transliterated in Amharic, different spellings may be used as varied as the number of possible pronunciations [5]. As [6] stated, about six vowels and three consonant sounds common to English are absent in Amharic. Due to this a native Amharic speaker may fail to correctly pronounce some English words. The situation is similar to other foreign languages. Hence, each writer has a tendency to write a foreign word the way he/she pronounces it. The cause of the difference in the Amharic spellings of these foreign language words seems to be the difference in the pronunciations of these words. The following table shows examples of spelling variation in the writing of foreign words in Amharic.

Foreign word	**Equivalent word in Amharic**
Meteorology	ሚትሪዮሎጂ ,ሚትዎሮሎጂ ,ሚትዮሮሎጂ ,ሚቲዎሮሎጂ, ሚቲዎሮሎጅ ,ሚቲዎሮሊጂ , ሜትዎሮሎጂ , ሜትሮዎሎጂ
Electricity	ኤሌክትሪክ, ኤሌትሪክ
Computer	ኮምፒውተር, ኮምቲዩተር, ኮምፒተር
Airplane	አውሮፕላን, አይሮፕላን
Director	ዳይሬክተር, ዲሬክተር

Table 3.7: Spelling variation of words translated from foreign words

Different ways of writing (spelling) the same Amharic word are also exercised. Regional and dialect variations can also impact word formation in the basic level where the words are more likely to be written following their spoken form. Moreover, there are word spelling variations that could be attributed to variations in pronunciations at different parts of the country, like for example using the two words ጠባይ and ፀባይ to mean temperament or using the three words ጢንዚዛ, ጢንዝዛ and ጥንዚዛ to mean beetle.

This is a problem also exists when the language has some words having different forms of writing system. In Amharic disjoint labiovelars words the ጡዋት word “tuwate" (morning) may be spelled as "ጥዋት, ጠዋት, ጧት" which are different variants of the same word.

In addition to the phonetic redundancy of characters, Amharic suffers slightly from visual redundancy in a few cases. Most prominently the vowel markers of; ‘ው’ and ‘ዉ’ , ‘ፕ’ and ‘ኘ’, ‘ፖ’ and ‘ፓ’, ‘ዪ’ and ‘ዩ’, ‘ጉ’ and ‘ጒ’ are similar enough that the former characters are often interchanged in words with the latter. This problem is exacerbated at small print sizes and with the lack of visual clarity often found on computer screens. Additionally, the letter ‘ቁ’ is often used in place of ‘ቍ’ (e.g. “ቍፐር” vs “ቁፐር”) which may owe more to phonetic proximity and decay than to visual. Often times the writer may simply be choosing the form that is easiest to write by hand or type into a computer.

Assimilation and Alternations of Character: There are a number of common cases where phonology clashes with Amharic orthography. For instance, ‘ም’ may be exchanged for ‘ን’ before ‘በ’, as in “አንበሳ” vs “አምበሳ”. Likewise ‘ም’ may also replace ‘ን’ before forms in ‘ፈ’ (e.g. “ላንፋ” vs “ላምፋ”). On the other hand ‘ሀ’ may replace ‘አ’ at the beginning of the word like ‘ሀገር’ vs ‘አገር’ Spoken Amharic has a great many alternations, whole and partial assimilations. Not all spoken occurrences will also manifest themselves in written form. This alteration can produce a spelling error during writing the words by confusing the writers and creates inconsistency due to alter and replace the character by other character.

Orthography Elisions: Difference in word affixing has also been observed to cause word spelling variations. For example difference in suffixing would result in the two writings ኢትዮጵያአዊ and ኢትዮጵያዊ to refer to Ethiopia while differences in prefixing would give the two writings በአገር and ባገር to mean ‘for country’.

Amharic Abbreviation: In Amharic, it is also found that there is no consistency while spelling abbreviations. For instance, the phrase "ዓመተ ምህረት" can be abbreviated as "ዓም", "ዓ.ም" and “ዓ/ም". Similarly, the use of the hyphen is also not consistent. The same word "ዓመተ ምህረት" can also be written as "ዓመተ-ምህረት". Hence there should be a mechanism to handle these problems while representing Amharic documents.

3.8 Amharic Spelling Errors

Techniques for spelling error detection were designed on the basis of different spelling error trends these are also called error patterns. Spelling and typing errors are common in documentation made by human. The problem of detecting error in words and automatically correcting them is a great research challenge. The word error can be divided in two types i.e., non-word error and real-word error [40]. Errors may be of missing letters, extra letters or disordered letters.

Real word error produced where the word in question is valid yet in appropriate in the context, and hence not giving the intended meaning. The word found in the desired language and hold by the corpus but during sentence formation it does not fit to provide a complete meaningful ideas for the user either semantically, syntactically and structurally. It is kind of errors occurring by the cause of semantic, syntactic, typographic, improper spaces, cognitive and phonetics of the linguistic context which are the reasons to generate real word errors. Real word errors can change the complete meaning sentences and makes an ambiguity to recognize the input sentences.

Typographical errors are occurring when the correct spelling of the word is known but the word is mistyped by mistake. These errors are mostly related to the keyboard adjacencies and therefore do not follow any linguistic criteria. The most common of these typographic errors is the substitution error substitution error is mainly caused by replacement of a letter by some other letter whose key on the keyboard is adjacent to the originally intended letter's key. There are large numbers of errors commonly encountered in human generated Amharic text and this error mostly belongs to one of the following error types based on Bhagat [10] spelling error classification.

Insertion error occurs when one or more extra letters are inserted in the required word. For example: መሳት to መነሳት.In the above examples, መነሳት is also valid word but it is not required word. These types of errors can give rise to real word errors which means words are valid but not required for instance ቁጥር to ቁጥጥር. In addition to this, adding an extra letters to the intended word can produce non-real word errors such as, ደራሲ to ደራሲይ. This errors are not found in the word list in Amharic texts which is no required in the document.

A deletion error occurs when one or more letter is removed from the required word. For example: ሥርዓት to ሥርት, መምህር to መምር, ሥርት and መምር are non-word errors that are not required. These types of errors can also give rise to real word errors. For example: ምስክር to ምክር

Substitution error occurs when one or more letters are substituted by some another letter. For example: አገር to ሀገር, አንበሳ to አምበሳ, ብሎአቸው to ብሎዋቸው. In the above given examples, አ to ሀ, ን to ም, አ to ዋ are the various substitution pairs. In addition to tis this kind of errors can be produced by the characters that have same pronunciation but different writing system such as ሥጋ to ስጋ, ጸባይ to ፀባይ, ታኅሣሥ to ታህሳስ, ዓይን to አይን, ሕግ to ህግ.

Transposition error occurs when two adjacent letters are written in swapped way. For example: መሰለ to መለሰ, ክምር to ምክር, ትዳር to ትርዳ. In the above explained examples, ሰ to ለ, ዳ to ር, ብስ to ስብ are transposed .Transposition errors also give rise to real word errors (the word which are valid but not required).

Not only real word errors, non-word errors are produced by swapping the letters in the written texts which are no required in the texts. For instance, በመጠቅ to መጠበቅ, ማሚረያ to ማረሚያ in which ማሚረያ and በመጠቅ invalid words that does not exist in the language that generates an error through swapping the letters in the words.

The splitting or run-ons words, keyboard effect, copy, paste and space might induce word repetitions, omission, splitting and run-on error words the input text. Writing repeated words and giving space between words can produce errors that cannot require in the document texts.

A run-on error detected relating to word boundary and occurs when two or more valid words are erroneously typed side by side without a space in the middle of the word. For example: መደበኛ ተማሪ to መደበኛተማሪ, ዕውን መሆን to ዕውንመሆን. In the above explained examples, መደበኛ, ተማሪ, ዕውን, መሆን are four different words, and መደበኛተማሪ and ዕውንመሆን are non-word errors that does not include in the language and it's considered as one word. In some cases these words can also give rise to real word errors. For example: in አለ መመለስ to አለመመለስ, the word አለመመለስ created which is a real word that does not require.

Split word error is opposite of run-on error. These types of errors occur when there is some additional space is embedded between the parts of the word. It can be simply removed by deleting the additional space such as ወይን ሸት to ወይንሸት. In some cases, split word errors can also give rise to real word errors in addition to real word errors. For instance, from አለ መመለስ to አለመመለስ real words አለ and መመለስ are two valid words and from ወይን ሸት to ወይንሸት, the first is real word and the second one is non-word.

Cognitive error also occurs when the correct spellings of the word are not known. In these types of errors, the pronunciation of misspelled word is the intended as correct word. Cognitive errors are orthographic errors occur when writer does not know or has forgotten the correct spelling of a word in the language. It is assumed that in the case of cognitive errors, the pronunciation of misspelled word is the same or similar to the pronunciation of intended correct word. These errors are occurring when the correct spelling of the word is known but the word is mistyped by mistake due to same pronunciation which is a phonetic error.

Phonetic errors are that type of errors in which the writer substitutes a phonetically correct but orthographically incorrect sequence of characters for the required word. For example, the word ፀባይ can be written as ጠባይ. In the case of Amharic language there are alphabets that have same sound with different spelling which makes an error in the documents by changing and modifying the semantic and syntactic of the texts. Basically, real word errors are produced when the sequence of words in the input sentences are not convey the meaningful sentences. Since Amharic has various ways of writing the same word there is no mechanism to handle such kind of errors without developing and integrating a rule based approaches into the model.

In our case the phonetic errors were detected and corrected by incorporating words in the corpus that have different writing systems with the same language. Because the user may be writing the same words in different ways and the corpus were prepared by considering those types of errors. For example, ሰዓት can be written ሠአት, ሠዓት, ሰአት, ሰዐት, and ሠዐት.

Non-word error can generally produce in different ways that have been discussed in above, in which the intended words are not in exist in the text documents even in the Amharic language vocabulary and users cannot recognize and understand the words to describe their documents and texts in different applications. In this research the words that does not included in the dictionary is considered as non-word errors and this kind of errors are corrected without considering the linguistic context like semantic and grammatical aspects of the given text. Therefore, if the required words not found, the spell checker model can detect words as an error and can provide suggestion alternatives by comparing the Amharic dictionary lists with distance similarity between words.

To sum up, Typographic errors are the typos, when people know the correct spelling, but makes a motor coordination slip when typing. The cognitive errors are those caused by a lack of knowledge of the person. Finally, phonetic errors are a special case of cognitive errors that are words that sound correctly but are orthographically incorrect.

4.9 Amharic Fonts

As reported in [66], Amharic alphabets do not have a representation in the ASCII (American Standard Code for Information Interchange) code table. As a result, font developers have tried to develop their own keyboard driver programs that make use of the existing English keyboard (ASCII codes) for writing Amharic. The English QWERTY keyboard were used in various combinations to produce Amharic characters. ECoSA (Ethiopian Computer Standards Association) is a professional association established in 1998 to solve problems that result from the disparity in the available different Amharic software. In order to solve the problem, ECoSA is currently working on standardization issues on Ethiopic including character definition, keyboard layout, character encoding and transliteration. This standardization projects are sponsored by the Ethiopian Quality and Standards Authority. Each one of the projects is handled by a sub-committee consisting of members from various professions (linguists, software developers, etc) from various governmental and non-governmental organizations (EcoSA Newsletter, 2000).

Different Amharic fonts have been produced over the years (e.g. Alpas, Brana I, Brana II, Power Ge'ez, Geez, Agafari, Alxethiopian, Visual Ge'ez …) but they all use the existing symbol sets differently so that an Amharic text written in Ge'ez font cannot be read in another one of the fonts. The need for standardization has been felt and as a result an association has been established in order to undertake the task. The Amharic text used for this research is written in the Nyala font. The UTF-8 Unicode standard was used for Amharic texts and saved in Notepad by file extension text (.txt) format. In this thesis we were used a Sabaean script directly which is not necessary to convert Amharic script to Latin script.

.

CHAPTER FOUR

MODEL OF AMHARIC SPELL CHECKER

4. Introduction

The decreasing cost, increasing capacity, pervasive feature and the increasing availability of applications are the major factors that initiated us to think a Amharic spell checker and corrector system is a requirement in Ethiopia. The usage of increasing applications and usage of handheld computers and mobile devices of demands a spell checker was one of the major input methods. Taking this initiative, we started our work by investigating the existing systems developed for other characters such as Latin, Chinese and Arabic.

In the previous two chapters we describe some of the related works on spell error detecting and correcting mechanisms for different languages and basic features of the Amharic language to be taken into consideration before designing the model of the Amharic spell checker and corrector are discussed. This chapter gives the detail description of the model designed for this thesis work. As discussed in the previous chapter, unsupervised machine learning was selected for this study and the procedures to develop and design the model was explained in this section. The brief description of techniques to detect spelling mistakes for context sensitive spelling error detection and corrections are discussed with respect to dictionary lookup techniques and n-gram statistical analysis. Dictionary lookup and n-gram statistical probabilities are used to detect and correct the non-real word and real word spelling errors by considering its dictionary and probabilities of the word in the corpus. Furthermore, the method to detect and correct the spelling errors and the algorithm that have been used to give appropriate word suggestion for spelling errors are described in a brief manner. The detail description of the Amharic spell detection and correction model is described as follows.

4.1 Spell Checking Model

The present disclosure addresses the problem of real-word errors using context words and n-gram language models. An unsupervised machine learning model was applied for real-word error detection and correction for Amharic text in which n-gram language models was implemented. N-gram language models detect real-word errors by examining the sequences of n words. The same language models are also used to choose the best correction for the detected errors. Conventional spell checking systems detect typing errors by simply comparing each token (word) in a text against a dictionary that contains correctly spelled words. The tokens that match elements of the dictionary are considered as correctly spelled words; other tokens are flagged as errors and corrections are suggested. A correctly spelled token that is not the one that the user intended cannot be detected by such systems.

In this research developing and designing a context based Amharic spelling checker is crucial issue to detect and correct real word errors using the Amharic sentences at sentence level. Since non-word errors are detected and corrected by dictionary lookup, real word errors in the sentences are detected and corrected by considering the preceding and following of words. Context-sensitive error detection and correction aims at detecting and correcting real-word errors, which cannot be detected by isolated-word detection and correction techniques. As described in the previous section, modern lexicon-based spelling error detection and correction systems correct approximately 50% of all errors. As 25% to 40% of all errors are real-word errors, a method for detecting and correcting real-word errors would be useful [56].

The model that developed is an interactive spelling checker for spelling mistakes that suggest a number of possible corrections and allow the user to choose the word that should replace the erroneous word. The problem of real-word errors for spelling error detection and correction was described as isolated word (lexicon-based) detection and correction techniques cannot detect and correct real-word errors. Spelling error detection and correction methods that aim at detecting and correcting real-word errors are referred to as context-sensitive error detection and correction methods. A context-sensitive spell checking method should aim at detecting and correcting all of these real-word error types and creating an interactive system for detecting and correcting human-generated real-word errors.

AMharic SPELLing checker and corrector (AMSPELL) model was developed to detect and correct non-word errors and contextual word errors using dictionary words and bigram lists respectively. The non-word errors are checked and corrected by Levenshtein minimum edit distance between strings of error words in the sentences and dictionary candidates and the minimum distance between was selected and suggested as suggestion of correct words. Simple dictionary words are constructed from the corpus and used as candidate for suggestion for spelling errors in the sentences. Whereas, real word errors are detected by the bigram sequence in the sentences and each bigram are generated from the corpus that have statistical probability. The errors are detected by considering the bigram list in the sequence of words in the sentences and the suggestion lists for errors are provided by the highest probability that each bigram has and the left and right side of the words are considered during suggestion.

A method for context-sensitive spell checking should aim at detecting and correcting all types of real-word error relayed the word category without tagging of the words. The developed spelling checker and corrector model can detect and correct real word errors using the bigram lists. This means that sequences of two words are considered instead of words in isolation. To check whether a specific bigram in the text contains a real-word error, the information of that bigram is determined from the input sentences.

First, the model detects the non-word spelling errors from the input words and correct by providing a list of candidate suggestions before checking contextual real words errors of bigram words. But if the sentences free from non-word errors, it starts to detect and correct the real word errors by splitting the sentences into list of bigrams. The overall context based spelling checker and corrector model is designed as follows.

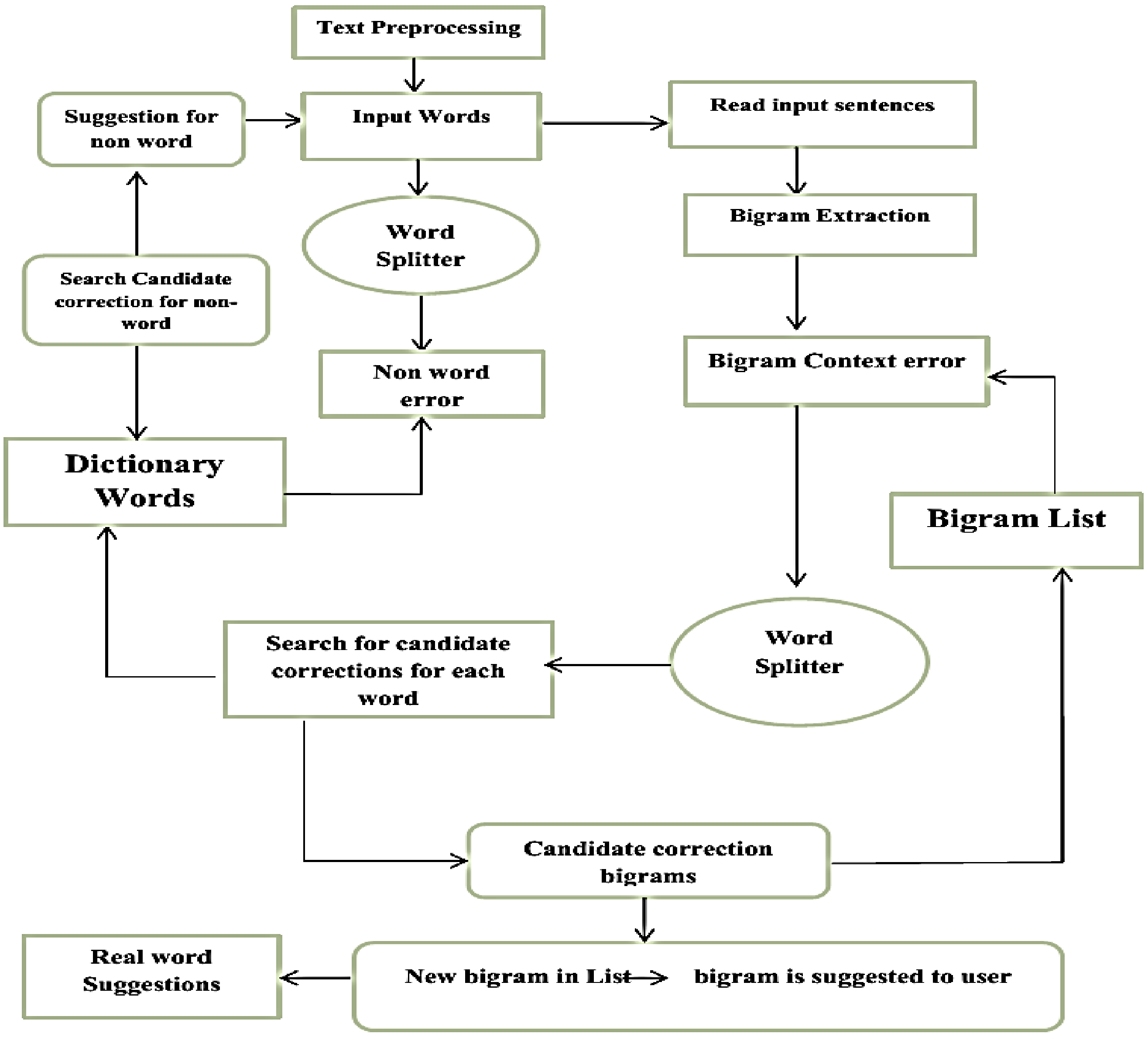


Figure 4.1: Amharic context based spell checker model

As described above, the context-sensitive spell checking model uses probability information to determine whether a specific word bigram contains a real-word error. The context-sensitive spell checking performs detection and correction.

The detection model performs three main steps. First, the input sentences that has to be spell checked is split up in bigrams. At every word a new bigram starts, resulting in a number of bigrams equal to the number of words in the sentences minus one. For example, the five-word sentence “ቅድሚያ መስጠት ላለብን ቅድሚያውን እንስጥ።” split up in the four bigrams ቅድሚያ መስጠት, መስጠት ላለብን, ላለብን ቅድሚያውን and ቅድሚያውን እንስጥ.

Second, for each bigram it is checked whether all two words are in the dictionary spell checker lexicon. This check would not have to be executed when the lexicon-based spell checker and the context-sensitive spell checker would have been combined into one spell checking application. In that case, the lexicon-based spell checker would perform non-word error detection and correction before the context-sensitive spell checker would perform real-word error detection. Then the input of the context-sensitive spell checker couldn’t contain any kind of non-word errors and this second step would not have to be executed. In this research, a stand-alone context-sensitive spell checker is built in order to be able to test with lexicon-based spell checker. Thus, the lexicon check is performed, if one or more words from the bigram are not in the spell checker lexicon, the bigram contains a non-word error and is it not considered further, because non-word errors are not in the scope of context-sensitive spell checking. This means that in this research, there are still non-word errors in the text after the context-sensitive spell checker checked it for real-word errors. Looking up each word of every bigram in the lexicon implicates that most words from the text are checked three times (once for every bigram it is part of). This way, the program does not have to remember which word is correct and which one is not. This is done to save memory space. In the developed model, the memory of the system is restricted to the bigram and dictionary under consideration.

Third, every bigram is looked up in a precompiled database containing a list of bigrams and their number of occurrence in the corpus used for compiling the database. If the bigram is in the bigram database, the bigram is regarded correct and it is not considered further. If the bigram is not in the bigram list, then the bigram is considered too unlikely and therefore detected as an erroneous bigram containing a real-word error.

The correction performs an additional three steps. When a bigram has been detected, one or more of the words is considered erroneous, but which of the two is not known. Therefore, candidate corrections for all words are sought. The dictionary spell checker lexicon is used to find candidate corrections for all words of the bigram. This is the first step of the correction mechanism. When all possible candidate corrections for all two words have been found, these are all put together resulting in candidate corrections for the bigram as a whole. The third step is looking up each of these candidate correction bigrams in the bigram list. The bigrams that are in the bigram lists are considered more likely to be intended by the user than the detected bigram and suggested to the user.

4.2 Error Detection

Text enhancement systems are used in the area of human language technology where manual correction of text is time consuming and creates a bottleneck in human language technology applications. Systems in human language technology like document understanding systems and speech recognition systems depend on reliable automatic misspelling correction capabilities. Although spell checkers are widely available for a number of languages, most spell checkers only detect errors and propose corrections regardless of their context, which increases ambiguity and incorrect suggestions for misspelled words. Also, the available systems are not able to detect and correct all kinds of errors, in addition to having other constraints. Conventional spell checking systems detect typing errors by simply comparing each token in a text against a dictionary that contains correctly spelled words. The tokens that match elements of the dictionary are considered as correctly spelled words; other tokens are flagged as errors and corrections are suggested. A correctly spelled token that is not the one that the user intended cannot be detected by such systems.

Dictionary look up techniques were employed to compare and locate input strings in a dictionary. There is standards string mechanism with the aim of reducing dictionary search time. In order to serve the purpose of spelling error detection exact pattern matching techniques are used. If strings are not present in the chosen dictionary it is considered as a misspelled or invalid word. In this research we assumed that all words are included in the corpus and dictionary which are complete and balanced. Hashing data structure is the most significant and efficient lookup strategy relies on input string to detect where a matching pattern found. More specifically, hashing is used for this research to search an input string in a pre-compiled hashing via a key or a hash address associated with word and retrieving word stored in the hash function.

In spell checking context if the words stored at hash address is the same as the input string which is the value of hash address. However, if the input string/word and retrieved word are not the same or the word stored in the hash address is null, the input word is indicated as a misspelling. The random access match of a hash eliminates a large number of comparisons retrieved for lookups and faster than other searching methods in large data representation. Therefore, hashing mechanism is used to detect spelling errors by matching hash address and the input string that are retrieved.

The majority of undetected errors are real-word errors where the word produced is in the computer's dictionary but is not the word the user intended. This type of error is largely ignored by most computer spell checkers as they rely on isolated word look-up to detect misspellings. Real-word spelling errors may be caused by the writer's ignorance of the correct spelling of the intended word or by typing mistakes. Such errors generally go unnoticed by most spell checkers as they deal with words in isolation, accepting them as correct if they are found in the dictionary, and flagging them as errors if they are not [69]. Therefore, the detection of real-word errors requires the spell checker to make some use of the surrounding local context.

In this research contextual spelling errors were developed to detect real word errors using the sequence of words in the given sentences. Bigram probabilistic information was used to detect the spelling errors during the formulation of complete sentences that should be semantically meaningful for users. Even if words existed in the dictionary some words are embedded in without considering the syntax and semantics information which change the meaning of sentences. Bigram words are not goes together in the given sentences the errors are detected as real word errors. Bigram model were used to detect words that are found in the dictionary but which doesn't co-occur and misspelled with other words. So if the words that does not combine and allocate together to form a complete full sentences words are checked as spelling errors based on n-gram words. Dictionary lookup and n-gram probabilistic models are integrated to detect non-word errors and real word errors that increase the accuracy and performance of the spelling checker. The method tries to detect an error by noting bigrams constituted by immediate left and right neighbor of candidate word and then generate some suggestions according to probability calculated for the correction set of words.

4.3 Error Correction and Suggestion

Spelling error correction attempts to endow the spell checkers to correct detected words to find the subset of dictionary or bigram entries that are similar to the misspelling in some way. As we have seen in the above, the spell correcting tasks can be described by a function that maps a misspelled word to a set of possible correct spellings. Spelling correction can be involving a dictionary or bigram lists, since the set of possible corrections are defined in terms of membership in the chosen dictionary or bigram lists. Spelling error corrections can be interactive or automatic based on the user intervention to generate suggestions for erroneous words. The simplest spelling corrector is interactive which provides the user with a list of candidate corrections and leaves to retrieve predicted word choice to the user. Whereas, the automatic approaches of spelling correction needs a significant level of machine intelligence as it is expected to correct spelling errors automatically without user involvement. In this thesis, we develop and design an interactive spelling corrector that the user could retrieve and select the best candidate correction from the suggestion lists by interacting prototype user interface and replace spelling errors with selected valid words.

On the other hand, spelling error correction could be isolated word error correction and context sensitive correction depend on the types of spelling errors. Isolated word error correction is a spell correcting mechanism without taking account any textual or linguistic information in which the erroneous word occurs. Similarly, a context sensitive word error corrector would correct both real word and non-word errors involving textual or linguistic context. In this paper both isolated and context sensitive correction are used to correct misspellings in the given sentences.

To achieve isolate error correction task Levenshtein minimum edit distance is used to transform one string to another. Levenshtein edit distance used in the current research is applied to find the minimum operation which includes insertion, deletion, substitution and transpostions to modify one word to other. Insertion occurs when a letter needs to be inserted a misspelled word resulting in a correctly spelled word. But deletion occurs when a letter needs to be deleted from a misspelled word in order to result in a correctly spelled word. Substitution indicates to the replacement of a letter in the erroneous word by a correct letter, thus the resulting in the correctly spelled word. The last one is transposition which takes place when the positions of the two adjacent letters are reversed and need to be swopped in order to result in a correctly spelled word.

Therefore, Levenshtein edit distance between two spellings words w1 and another w2 is the smallest number of edit operation that needs to take place in order to transform w1 to w2. The distance is used to search appropriate candidate corrections for misspelled word and the distance measured between a misspelled words in the sentences to a word in the dictionary. The smallest distance between them is taken as a valid word and replaces error words in the given sequences of sentences. Candidate suggestion lists replied to the user based on the smallest edit distance operation and ranked based on the distance found between errors and dictionary words. The word with smallest distance is ranked at the top as best suggestions because it has small distance between words to edit and replace the error words in the sentences. Additionally, word frequency could be taken into consideration in order to rank suggestion lists if the two words have same distance measurement. After the list of suggestions is composed, it should be ordered so that the user doesn't have to scroll through it, searching for a perfect match. The implemented solution makes use of the Levenshtein minimum edit distance algorithm to calculate the word distance. This distance becomes a parameter for list ordering. The user makes his/her choice from the list of suggestions. The misspelled word can be replaced with a word from the suggestion list, ignored, or edited by the user.

On the other hand, context sensitive spelling errors could be corrected using bigram language model. In this research bigram model was applied for error detection and correction of real word errors in the constructed sentences. In context based error correction the semantic and syntactic features of sentences were not considered for spelling error suggestion and correction that checks and corrects typing mistakes from simple n-grams depend on the sequence of words in the sentences. Real words errors corrected by bigram similarity relied on the probability information of words. The highest probabilities of bigrams were more useful to correct the misspelled word and each individual word generates its own candidate suggestion that consist at least one bigram words. The candidate corrections could be ranked depend on the probability value of each bigram and substituted by correct bigram words.

Furthermore, deciding the number of candidate suggestion generated for each word errors were basic to save searching time and space to view all possible suggestion in the list. The numbers of suggestion lists were different based on the types of errors detected and corrected in the texts. For dictionary based correction, the maximum numbers of suggestion lists display for users were limited only up to ten suggestions for each word errors that are flagged as non-word errors in the sentences based on the smallest distance between words. In the case of contextual error corrections only ten candidate lists were suggested as correct word that had highest probability of bigrams to replace misspelled words. Spell checker system generated up to ten candidate suggestions for one word in the bigrams lists that comes with the second bigram level word and the second word generates at most ten candidates in the popup menu that comes with the preceding words in the bigram which is suggested for users and the users could select the best top to correct misspelled words.

4.4 Tokenization

Word boundaries in most spelling error detection and correction techniques are often defined by inter-word separation such as spaces and punctuation marks. The input of a spell checker is words. When a document is to be spell checked it is tokenized in order to separate words. This tokenization is generally done on word delimiters which are considered as an identifier of individual words in the texts of the sentences. Identification of word is different from language to language and most are used white spaces to separate words which depend on language features. Amharic language has different delimiters to bound words in the text in addition to white spaces. Word boundary issues are the basic challenge to identify the words which are allowed in the language. The Amharic word separators are described in (Amharic Punctuation marks 3.6).

Defining the word boundaries between tokens is crucial to detect and correct the word boundary errors particularly if the texts separated by white spaces. There are incorrect splits and run-ons are the most basic word boundary problems which affects the process of detection and correction mechanisms. Incorrect splits and run-ons therefore yield a deviant number of words in the resulting sentence. This difference in number of words can give problems for error detection and correction. Run-ons are mostly a problem for correcting errors, whereas incorrect splits are a problem for both detecting and correcting.

Incorrectly putting two words together, like mistyping አለ መመለስ as አለመመለስ or misspelling ፍርድቤት as ፍርድ ቤት, often yields a string that is not a lexicon entry. Therefore, the word is detected as an error. In order to correct this error the spell checking application should be able to add white spaces at any position within the incorrect string. If adding a white space yields two lexicon entries, a valid suggestion has been found. Unfortunately, adding white spaces at any position within the incorrect string results in many possible combinations of words that have to be checked against the spell checker lexicon. This decreases the speed of the application.

If a word has incorrectly been split up and results in two strings, detecting and correcting the error is more difficult. Incorrectly splitting a word often results in one or more strings that are lexicon entries themselves. Suppose an incorrect split results in two words of which one is a lexicon entry. Then this string is not detected, but the other string which was not a lexicon entry is. However, it is very difficult to correct this erroneous string, since neighboring words are not taken into consideration for finding suggestions. For example, suppose መታሰቢያ had been written as መታ ሰቢያ. Since ሰቢያ is not a lexicon entry, the string is detected as an error. But when searching suggestions the preceding word መታ is normally not taken into account, as a result of which መታሰቢያ probably is not found as a suggestion because its penalty is too high. A solution could perhaps be found in taking into consideration adjacent words when searching suggestions. Unfortunately, this can also yield incorrect suggestions. More research investigation needs to be done in order to find out how big this problem is. Moreover, taking into account adjacent words will decrease the speed of the application. When an incorrect split yields two lexicon entries instead of one, the error cannot be detected by dictionary based spell checking techniques at all.

A tokenizes can generally remove punctuation characters attached to the start or end of each word and store them as separate text tokens in Amharic; it is also found that there is no consistency while spelling abbreviations and compound words. Identifying abbreviation and compound words were important to recognize the spelling errors in the detection and correction process. Amharic abbreviations words are mostly written using the full stop (.), forward slash (/) and without any punctuations. To make consistency throughout the text common and formal abbreviation words are written without any punctuation marks and white spaces. For instance ዓመተ ምህረት was written as ዓም. In addition to this compound words were the other issue in spell checking operation. Compound words were written using white space and hyphen marks like ጸረ ሙስና and ስነ-ስርአት. Therefore, compound words were converted into only using happen marks to consider as one word in the sentences. The accuracy and effectiveness of spell checkers increases by identifying the abbreviations and compound words in the sentences and those words were processed and separated manually depend on language features.

4.5 Sentence Segmentation

Special care needs to be taken at the time of selecting well-formed sentences due to formal and informal day to day communication and most of the sentences were simple and representative for writers. We assumed that the syntax and semantics of the collected sentences are correct as they are mostly collected from different sources which are normally edited and proofread.

This is the process of segmenting texts into sentences using sentence markers of a language that are useful to obtain a meaningful and well grouped sentences. Once the sentence tokenizer split the text into sentences, it needs to group these texts into sentences. The sequence of input words were ended by word delimiters and the end of sentence was delimited by sentence markers. The sentences of Amharic texts are delimited by Arat netb (**::**), question mark (?) and exclamation mark (!) depend on the linguistic features of the language.

4.6 Dictionary Construction

The first step of developing a spell checker is construction of a dictionary and the important issue is the size of the dictionary. If the size of the dictionary is very small, it will annoy the user with many false alarms, and if it is too large, it will skip the mistyped errors that have been converted to rare words in the language. The popularity of dependent spell checking methods grew in tandem with the increase in available computer memory. While increasing accuracy, they are not infallible. The accuracy of a spell checking method based on a dictionary look-up program is directly related to the accuracy of the dictionary; it must be both valid and contemporary.

A simple dictionary list of words is adequate for non-word error detection and can also be used to produce suggestions for correction by finding words that closely resemble the misspelling. The most straightforward and widely used method for a computer spell checker to detect non-word errors is dictionary look-up. For a simple implementation the dictionary need be no more than a word list. The spell checker then looks up each word in the text to be checked in its list and flags as misspelled any that are not found. The question to be answered at this stage is how many and which words should be included in the list. The dictionary word lists are prepared from the Amharic corpus that are collected from various sources and prepared to increase the performance and accuracy of a spell checker for detecting and correcting the typing errors. The lexical dictionary comprised additional extra words including the training words in the sentences and integrated together to enhance the completeness and comprehensiveness of words of the Amharic words. Appropriate suggestions are suggested for non-word errors that are not exist in the dictionary list by comparing the string distance similarity between words.

The HashMap data structure was chosen by the developer because not only were they fast, efficient and accurate, the structure suited the suggestion mechanism used. They were also efficient in storage space, making it ideal for use in situations where memory was limited. The words were stored in the disk and positioned in the hash function for fast retrieving and accessing of data from the lists of words. The invalid words were swiftly compared with the valid dictionary words that could display relevant alternative suggestion for correction.

4.7 Bigram Generation

In this thesis dictionary lookup and n-gram probabilistic approaches are combined together to detect and correct non-word and real word spelling errors in the given sentences. Depending on the size and comprehensiveness of training corpus bigram language model were applied for contextual spelling errors for this spell checker. Even if the corpus size is less unigram model was not preferable because it’s difficult to get the context of the sentences for neighbor word which is like a normal text to detect and correct the contextual errors in the input sentences. Therefore, Bigrams are generated from the training data that are appropriate to check and correct real word errors. This method extends the token list concept by using a large corpus of text from the desired language. Bigrams are sequences of two characters extracted from adjacent characters in a word or sequence two words extracted from adjacent words in the given sentences. In this thesis word level bigrams are generated from the texts and stored in the memory with its probability information. A statistical measurement is given to each bigram word in the text file being spell checked based on the possibility of the bigrams found in the sentences. Bigram words with low probability information are considered as being potentially erroneous in the given sentences.

The relevancy and effectiveness of bigram words are measured by the input sentences to predict best suggestion for real word errors. Therefore, before generating bigram words from sentences the corpus are segmented in the sentences using Amharic sentence markers. The Amharic sentence markers used in this work were Arat netb (**::**), question mark (?) and exclamation mark (!) which are used to bound the end of the sentences in the corpus. Bigrams are generated using Amharic word delimiters from segmented sentences and each bigram words separated by comma (,) like (ዕቅድ, ብቻ) during bigram generation. During bigram building the occurrence of each bigram words were counted and stored in the hash map with its frequency to increase efficiency of processing and loading bigrams to find bigram probability.

One of the ways to calculate probability of the words or sentence in n-gram model is using Markov chain rule. According to Markov assumption, probability of some future word depends only on a limited history of preceding words. The bigram probability of bigram language model for a sentence of m words W_1, W_2, W_2, …, W_m can be calculated as

$$P\ (W_1, W_2, W_3, \ldots, W_m) = P(W_1)P(W_2|W_1)P(W_3|W2)\ldots P(W_m|W_{m-1})P(W^m)$$

In our model we do not calculate the sentence probability. We assume bigram model that the occurrence of any words depends on its previous and next words only and independent of other words in the sentence. The bigram probability of words was calculated by Maximum Likelihood Estimation (MLE) based on the sequence of words in the sentence as follows. Let say W_1 and W_2 comes sequentially in the sentences and the bigram probability computed as

$$P\ (W_2/W_1) = \text{Count}\ (W_1W_2)/\text{Count}\ (W_1)$$

In addition to this unigrams are generated to from the given corpus and that are used to calculate the probability of bigrams. The occurrences of each unigram words were determined and each unigram words are associated with its frequency in formation to find bigram probability. Estimated bigram probabilities of each bigrams were computed using the given sentences to produce better suggestions for spelling errors and ranking the correct suggestion options.

CHAPTER FIVE
IMPLEMENTATION AND EXPERIMENTION

5. Introduction

A series of experiments is conducted in order to assess the quality of the spelling checker applications. In this chapter, the tools and environments that are used to implement the designed algorithm and the experiment that is conducted to demonstrate the spelling error detection and correction accuracy could be presented. The result of the experiment would be interpreted in this section and the performance of the spelling error detection and correction application could be evaluated using different evaluation method and parameters. Precision and recall were used to evaluate the accuracy, effectiveness and validity of detecting and correcting spelling errors based on the training and testing texts that have been used in this experiment.

5.1 Tools and Development Environment

Developing a prototype to demonstrate the validity and usability of the proposed context based spelling checker system is one of the objectives of this work. In order to implement the model and make the necessary experiment on the system we have used different tools and development environments. This section would talked about the tools and development environments used to implement the model and the interfaces used for training and testing purposes. In addition to this the users can interact to the interface developed to choose and correct candidate suggestions for spelling errors after detecting the invalid words. To take the input texts and display spelling suggestions for the user the spell checker prototype interface was developed using Java NetBeans (NetBeans IDE 8.0.2) coding environment with window 8.1 operating system.

The second tool used for this research was AntConc which is a freeware multiplatform tool for carrying out linguistic research and data driven learning. It runs on any computer running Microsoft windows, Macintosh and Linux system environment. It is developed in Perl using various compilers to generate executable for different operating system. This tool could be used to generate wordlist that are useful as word tokens and types in a spelling application. The tool takes the input sample test corpus and produces a list of words as an output for word lists. Lexicons were prepared using AntConc tool and handles unnecessary duplicate words, numbers and punctuation marks that are not reliable for spell correction and used to reduce storage space of wordlist files. Therefore, the tool can normalize and eliminates any duplicate frequency of words that exists in the dictionary and lists could be ordered by the frequency information that each term appears in the input corpus. Not only word lists, bigram words with its probabilistic information also generated using this tool and the bigram words arranged and ranked using probability of bigrams. This tool used only to prepare and process the test dataset for testing purpose that have been taken in the experiment.

On the other hand, Notepad++ also plays a vital role to develop a good spelling checker by editing and changing unnecessary and invalid words during corpus preparation that does not process automatically by java programs. Since the collected Amharic texts written in different Amharic writing system this tool helps to modify and correct the spelling variation and errors manually. The corpus processed and organized manually in proper manner with linguistic expert to create clear and understandable spelling features words of Amharic language that have been taken as input for spelling checking operations.

5.2 Data Collection and Preparation

Corpus is a large and structured set of texts. It is used to spell checker, checking occurrences or validating linguistic rules on a specific universe. Besides it is a fundamental basis of many researches in NLP. Building of the text corpus is very helpful for the development of spell checking. In this work, Amharic text corpus is created manually to apply in Amharic spell checker system. It contains various sense meanings of ambiguous Amharic words, compound words and training sentences.

Amharic texts were collected from various sources to reliability train and test the spell checker model that have been developed depending on the linguistic spelling features of the Amharic language. The texts were collected from ENA, WIC and books that have discussed various issues to balance the corpus distribution. In addition to the above sources HC corpora was used for this research to conduct the experiment which is collected from Ethiopia newspapers like Ethiopian reporter, zehabesha.com and amharic.voanews.com, and the blogs such as wordpress.com and blogspot.com. The HC Corpora contain text from publicly accessible sources and collected from internet by web crawler which was used for any purpose excluding commercial use without prior consent.

The most important part of any natural language processing task is a proper training corpus preparation. To the researcher's knowledge, there is no standard established training and testing collection texts for Amharic spell checker and corrector testing. The experiment in Amharic spell checker and corrector has conducted by a prepared text and the texts were prepared by the researchers themselves. The corpus were collected from various sources and manually cleared from any kind of unnecessary errors and each word in the corpus were free from spelling errors which is valid to represent Amharic vocabulary words. Word tokenization, sentence boundary identification and bigram generation were important in training corpus preparation.

Text preprocessing is very important aspect of corpus preparation to clear unnecessary ambiguity and errors before training and testing the spell checker model. Since the data collected from various sources and written with their own writing system, preparing complete error-free corpus is a challenging task in the text preparation. Even if unsupervised approaches used for this thesis it needs more effort to modify and edit that errors corpus manually by linguistic professionals based on the spelling nature of the Amharic language. The corpus were prepare with linguistic expert depending on spelling features of Amharic to maximize the accuracy and performance of the model by making well understandable and pure words of Amharic vocabulary. The training and testing texts were prepared manually based on the Amharic spelling error patterns and its writing system before texts processing that are used as input for training and testing the spell checker model.

The validity and relevancy of dictionary words were evaluated by linguistic professionals to check whether the words that prepared for correction were valid or not based on the vocabulary of the desired language. The training text has included almost all type of spelling alphabets and words including its morphological variation of the words in the corpus. It is true that large size corpus has produced high accuracy performance than a small collection of corpus and we tried to collect and prepared a balanced corpus for training and testing the spell checker model. Punctuation marks, numbers, white space and any special characters that appeared were manually and automatically processed to reduce the ambiguity and necessity of words in the corpus.

A sufficiently large corpus is essential for training and testing of any spelling checker and corrector application. One of the major problems of building corpus from learners' data is that the process is very time consuming and requires linguistic knowledge to examine each sentence of learners' text to determine nature and frequency of errors. To overcome this problem, error sentences that consists non-word and real word error has been collected from different sources considering the performance error and language learner's error that occurred frequently.

Amharic language learners often commit spelling mistakes while writing text because of their lack of language knowledge (language learning error) and due to oversight, carelessness or tiredness (performance error). Performance errors can occur mainly due to four operations: insertion, deletion, transposition and substitution. There are two primary concerns at the time of error sentences, first one being linguistically realistic and the second one is to mimic the error scenarios that happen normally.

Real word errors were checked contextually, in which each sentence in the corpus is relevant to represent and convey meaningful information for writers. So every sentence in the corpus was evaluated along with its meaning and syntactic structure of the language. It means that the input sentences should provide clear and understandable ideas for writers.

Using the model described in the previous chapter, list of words and bigrams along with probabilities were generated and stored in tables along with their frequency information. The sample content of corpus used to conduct this experiment was described as follows.

Word	Frequency
ነው	2362
ላይ	1446
ውስጥ	710
ግን	696
ወደ	585
ጋር	540
እና	525
ነበር	507
ነገር	477
ደግሞ	460

Table 5.1: Sample Words counts table in the corpus

The corpus was also used to generate a bigram that used to find the probability of the bigrams in the given corpus. The bigrams are generated at word level rather than character level which are used to detect and correct the real word errors.

Bigram word	Frequency	Probability
ነገር, ግን	112	0.235
ብቻ, ሳይሆን	79	0.385
ማለት, ነው	73	0.369
ምን, ያህል	45	0.152
ዓለም, አቀፍ	39	0.310
በአሁኑ, ወቅት	38	0.559
ላይ, ነው	32	0.022

Table 5.2: Sample Bigrams table from the Corpus

As can be observed, the number of list of word types generated for a word is less than the number of bigrams. Because one word may co-occur with many words sequentially in the sentences which indicates the total occurrence of words was greater than the bigrams generated from the corpus. As the value of n gets higher, so would the number of n-grams generated. For the training set used in learning the model, the following table presents a statistics of the counts of words, bigrams and sentences in the experiment.

Terms	Frequency of word tokens	Frequency of word types
Unigram	342,560	295,527
Bigram	143, 037	124,862

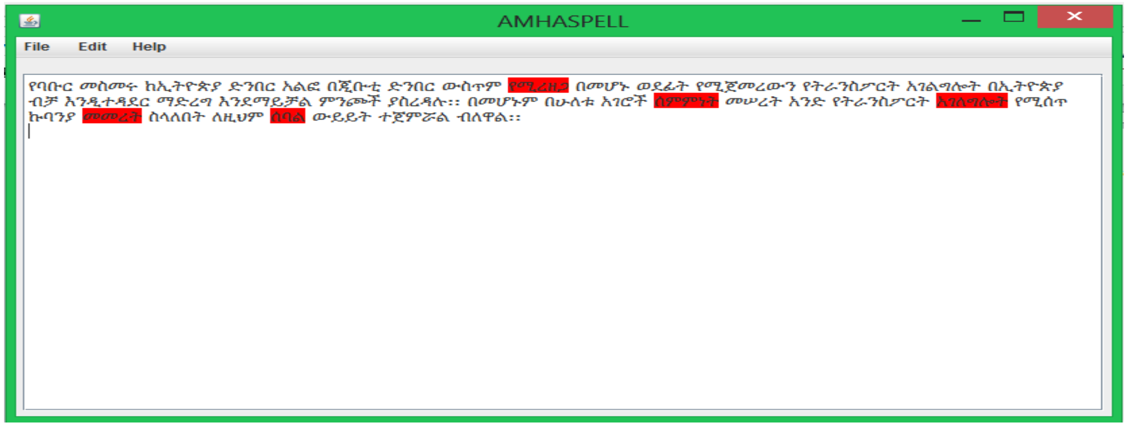
. Table 5.3: Count of generated terms (words and bigrams)

Texts from multiple domains have been collected to a void the skewed distribution of data. In the experiments, texts belonging to several domains including technology, computing, economy, medicine, engineering, politics, love, health, literature, history, religious, sports and other aspects were prepared for testing and training purpose. These corpuses encompass around **47,033** word types and **152**,**423** word tokens which consists regular dictionary words, domain-specific terms, proper names, technical terminologies, acronyms, jargons, and expressions.

The texts used to conduct this experiment were classified into training and testing set. Therefore, 10,000 sentences were collected and prepared from various sources which are used to train and learn the model. On the other hand, for testing purpose 500 sentences were prepared and used in the experiment to verify the efficiency and accuracy of detecting and correcting the spelling errors.

5.3 Spelling Error Detection

The detector module is responsible for determining if a word is considered misspelled or not with respect to the lexicon and bigram analysis. The input for the prototype is a text file. The text file can be typed directly into the Text Area. The system checked the spelling after the space bar is pressed or sentence delimiters pressed depend on the spelling errors. Non-word errors were detected using space bar where as real word errors detected by using Amharic sentence delimiters. A word that the system believes to be misspelled is flagged with color shading red for non-word errors and yellow are used for real word errors. Candidate suggested corrections are available in a popup menu after right clicking on the erroneous word.



የባቡር መስመሩ ከኢትዮጵያ ድንበር አልፎ በጂቡቲ ድንበር ውስጥም የሚረዘጋ በመሆኑ ወደፊት የሚጀመረውን የትራንስፖርት አገልግሎት በኢትዮጵያ ብቻ እንዲተዳደር ማድረግ እንደማይቻል ምንጮች ያስረዳሉ። በመሆኑም በሁለቱ አገሮች ሰምምነት መሠረት አንድ የትራንስፖርት አገለግሎት የሚሰጥ ኩባንያ መመሬት ስላለበት ለዚህም ሰባል ውይይት ተጀምሯል ብለዋል።

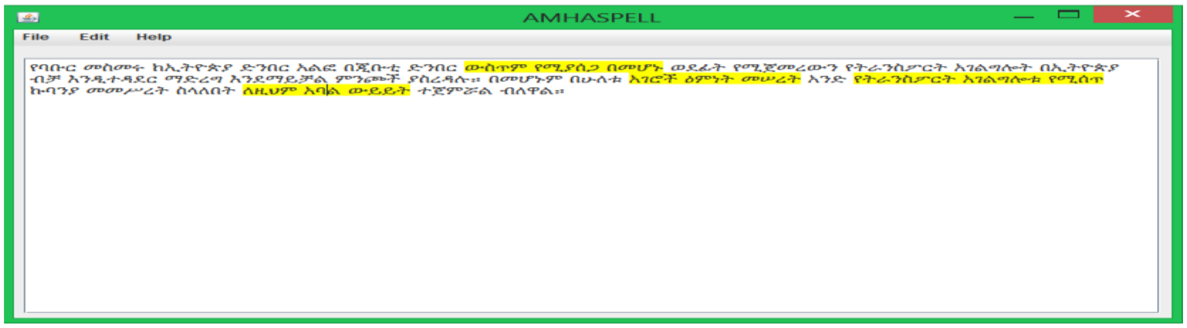
Figure 5.1: A prototype system for non-word error detection

Spelling errors were detected and checked when user types words in the interface by considering dictionary lists and bigram words in the given sentences. As already mentioned above, the detection of spelling errors was designed in two phases. The first one is dictionary based and the second is for bigram words in the input sentences.

The spelling errors were checked at word level to detect non-word spelling errors through the comparison of dictionary lists based on similarity measurement of words using white space and punctuation marks. Word boundaries used for this work were used to determine the validity of words that exist in the dictionary lists. The word that does not exist in the dictionary is detected and highlighted as error word when we type words in the interface that we are developed. Every non-word errors that appeared were detected and corrected before the end of sentences. Therefore, each single word was marked as error and multiple errors were detected in the given sentences using white space rather than Amharic word delimiters. Punctuation marks, numbers and words with numbers such as **ከ1990** were excluded and selected as error by the system. For each input line, a multiple line is written to the standard output for each word checked for spelling on the line.

Whereas, real word errors wear detected after all non-words were corrected and the errors were detected and highlighted as real word error at the end of sentences using the sentences markers of Amharic language. The real word errors were detected under the consideration of bigram words sequences that comes together and sequence of bigram words does not exist in the bigram list, it's detected as real word errors. The input sentences were breakdown into bigram and bigram words were generated along with its probability information which is used to rank the candidate suggestion to correct the errors. If the bigram words found in the bigram list, there is no error which is considered as valid word but if one of the word does not exist in the bigram word list is considered as error and detected to display suggestions alternatives for that bigram.

Real word detection can be afforded the semantic and syntax level of the words in the sentences which is the meaning of sentences was defined by the sequence of words that co-occur in the generated bigram words. The syntactic and semantic nature of sentences were not determined by the syntax rules of Amharic language but defined by the sequence of bigram words that comes together to provide meaningful information for users. Even if words exist in the dictionary words cannot become sequentially with others to convey ideas and the meaning of information were distorted for users which is difficult to recognize and understand the structure of sentences in the input texts.



የባቡር መስመሩ ከኢትዮጵያ ድንበር አልፎ በጂቡቲ ድንበር ውስጥም የሚያሰጋ በመሆኑ ወደፊት የሚጀመረውን የትራንስፖርት አገልግሎት በኢትዮጵያ
ብቻ እንዲተዳደር ማድረግ እንደማይቻል ምንጮች ያስረዳሉ። በመሆኑም በሁለቱ አገሮች ዕምነት መሠረት አንድ የትራንስፖርት አገልግሎቱ የሚሰጥ
ኩባንያ መመሥረት ስላለበት ለዚህም አባል ውይይት ተጀምሯል ብለዋል።

Figure 5.2: How prototype system detects real word errors

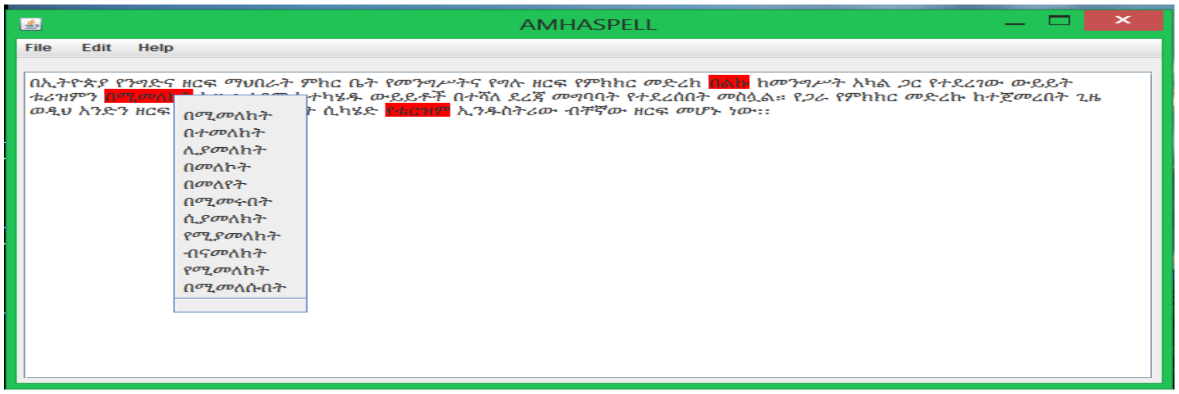
In spelling errors detection phase if one word make errors the surrounding left and right neighbors words were detected as errors and corrected by providing suggestions for each words that are highlighted as errors in the sentences. But this not always true because the errors can be detected and colored only for two words including the error one and one of the right or left side of the words. In the case of word level detection and correction the errors were verified and determined at each word in the sequence of words that appears errors before ending the sentences. The detection and correction word errors were determined at word and sentences level for non-word and real word errors.

5.4 Spelling Error Correction

The corrector module is responsible for providing a set of possible corrections for a misspelled word. After a word is flagged as wrongly spelled, if possible a set of suggestion is availed. Studies point that most misspellings involve at most one character change from the intended word [40]. This means the distance between the correct word and the misspelled word is the character difference. N-grams can also be used for error correction. This is done by assuming certain n-grams within a word are correctly spelled and fix the remaining n-grams. A list of words is established as suggestions. It is also important to rank the words and lift the closest suggestion to the top of the list and presumably trim it. To organize this we need an algorithm that computes the minimum edit distance. A Levenshtein edit distance could accomplish this and show the shortest distance between suggested words and the word with the shortest distance would be considered as the best suggestion.

The spelling correction functions enable to create applications that check if words are spelled correctly. It uses dictionaries that load into the dictionary list and checks words against a specified dictionary. Alternative spelling suggestions were provided for each word errors for correction which is ordered based on the type of errors that occurred in the given inputs of words and sentences. The errors were corrected and modified through the suggested words that displayed in the popup menu. In order to suggest valid words for wrong words two cases were considered depend on the error types.

The first is performed at word level before checking sentences and each word have suggestions from the dictionary words by considering the similarity between the error words with dictionary words. The candidate alternatives were ranked based on the similarity distance which is the minimum edit distance between words and the smallest distance comes at the top and better for correcting error words. The smallest edit distance could be selected and used to correct non-word errors among the suggestion alternatives. Isolated words errors were detected using white space and should be replaced by valid words before the end of sentences that the suggestion and correction taken places immediately before going to correct real word errors.

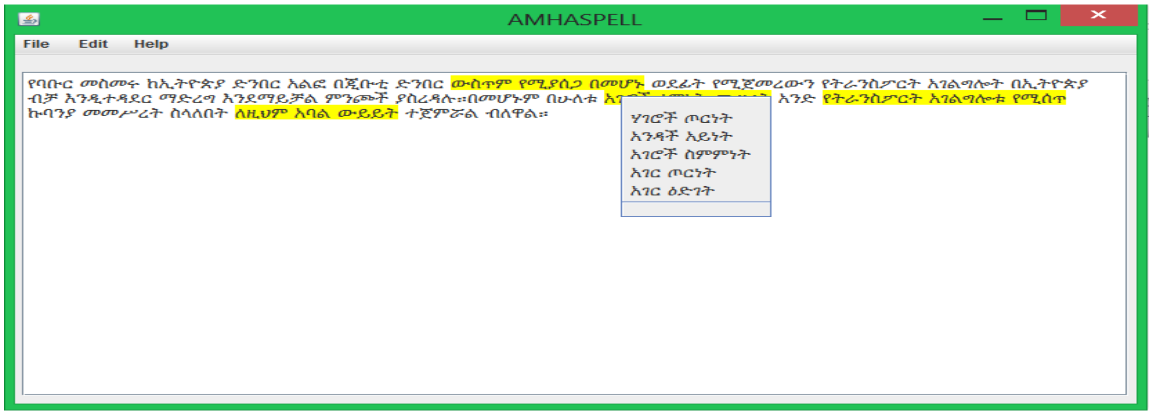


በኢትዮጵያ የንግድና ዘርፍ ማህበራት ምክር ቤት የመንግሥትና የግሉ ዘርፍ የምክክር መድረክ በልኩ ከመንግሥት አካል ጋር የተደረገው ውይይት
ቱሪዝምን በሚመለ ... ተካሄዱ ውይይቶች በተሻለ ደረጃ መግባባት የተደረሰበት መስሏል። የጋራ የምክክር መድረኩ ከተጀመረበት ጊዜ
ወዲህ አንድን ዘርፍ ... ት ሲካሄድ የቱርዝም ኢንዱስትሪው ብቸኛው ዘርፍ መሆኑ ነው።፡

Figure 5.3: A prototype system for generating suggestion for non-word errors

The suggestion replacements were ranked by Levenshtein minimum edit distance which is computed using errors encountered by spell checker with dictionary lexicons. The suggestion popup menus displayed during right click at any position of error word and select one of the target words that replace the errors and correct it.

On the other hand, real word errors were checked and corrected using bigram probabilistic information with similarity between the words that occur with dictionary words. The errors were identified by using sentences boundaries and each words highlighted have its own suggestions. The identified errors that have been detected provide bigram suggestions for each word in the bigram by computing and comparing the Levenshtein edit distance with dictionary lists.



የባቡር መስመሩ ከኢትዮጵያ ድንበር አልፎ በጂቡቲ ድንበር ውስጥም የሚያሰጋ በመሆኑ ወደፊት የሚጀመረውን የትራንስፖርት አገልግሎት በኢትዮጵያ ብቻ እንዲተዳደር ማድረግ እንደማይቻል ምንጮች ያስረዳሉ።በመሆኑም በሁለቱ አገ [illegible] አንድ የትራንስፖርት አገልግሎቱ የሚሰጥ ኩባንያ መመሥረት ስላለበት ለዚህም አባል ውይይት ተጀምሯል ብለዋል።

ሃገሮች ጦርነት
አንዳች አይነት
አገሮች ስምምነት
አገር ጦርነት
አገር ዕድገት

Figure 5.4: A spell checker prototype system for generating candidate suggestion

The real word errors problems were resolved by generating bigram words after computing the distance between each selected word with dictionary and display candidates which is ranked by the probability of the bigrams. Then replace the invalid words by clicking each error words at any position and search the alternatives from the bigram based on the shortest edit distance of the two words. During candidate suggestions the surrounding words were considered depend on the number of errors detected consecutively. If two words detected and clicked on the right side, the suggestion displays the left side and right side replacements from the bigram lists by comparing the distance between dictionary lists and replaces the two errors sequentially. But if three errors occurred sequentially and needed to replace the center of words, the system compares the word with dictionary for both right side and left side of sequences that replaces all the three words including the center one. Therefore, each detected errors have its own suggestions by considering the distance between errors and dictionary lists. In the case of suggestion after comparing the distance erroneous word with dictionary, the system retrieves bigrams words which related to distance of that word.

Appropriate candidates suggestion were necessary to correct the non-target word and the list of candidates should be searched either tha dictionary or bigram list grounded on Levenshtein minimum edit distance between strings. In this thesis minimum edit distance between strings were supposed to less or equal three which indicates the maximum edit operations of strings is three for contextual spelling errors. Whereas, for non-word errors we assumed that the distance between the errors and dictionary lists were not greater than two which is tha maximum editing operation is only two. The majority of the users produce the errors by missing two or three characters of the words and this colud considered for choosing of this value of distance. In the case of generating suggestion omission may occur in which the intended word may not explore and displayed in the popup menu. For this reason, we assumed that the minimum edit distance for this spell checker were two because not only missing target words but also the suggestion includes needless words to modify the errors. These were happened during the experiment and a few words have missed the intended word from suggestion list and includes unintended words the displayed popup menu for both type of errors. On the other hand, the spell checker system refuses to provide suggestion for a few words particularly for real word misspelling since the distance between strings were more than the specified value.

5.5 Performance of Spell Checker

Now a days, spelling checkers are widely available as part of word processors or as standalone components. But there is still a considerable room for improvement in their error handling abilities. In order to quantify any improvement, we need to devise a methodology for evaluating the effectiveness and acceptability of a spelling checker. The increase of competitive proofing tools market, it is becoming ever more important to find evaluation methods and metrics that provide stable and invariable measurements. The performance measure of a particular spelling checker must be constant over a number of evaluations, irrespective of the percentage of mistakes in different texts, the level of difficulty of the texts and the length of the texts [24]. Evaluating spelling error detection and correction system requires various criteria, such as output quality, maintainability and user satisfaction.

The system performance of spelling error detection and correction system is usually measured by metrics like Precision, Recall and Accuracy. These measures generally indicate how often spelling incorrectness is rejected and how often spelling correctness is accepted. Standard metrics for the evaluation of the linguistic performance of spelling checkers, like lexical and error recall, and precision have been widely used for many years [60]. Evaluation methodology and evaluation metrics could be modified to render a more accurate representation of actual spelling checker performance and accuracy of correcting invalid words. The methods and metric used for evaluating the performance prototype system was described as follows.

5.6 Evaluation Metrics and Results

Some experiments were performed in order to quantitatively evaluate our spelling checker mechanism. The designed system must be evaluated to test its effectiveness. In the literature, several methods for evaluating spell checker system have been proposed. A work done by Kukich [40] proposed lexicon size, test set size, correction accuracy for single and multi-error misspellings, and type of errors as evaluation criteria for a spell checker tool. A research described by Paggio et al. [57] recommend error recall, precision recall, interface and suggestion adequacy for the evaluation of a spell checker algorithm. Some of the measurements are subjective and difficult to evaluate. Starlander & Popescu-Belis [32] also came up with some refinements on these metrics, as well as some new metrics for their evaluation of proofing tools, which can be accurately implemented in the evaluation of spelling checkers. The performance of the system was measured using error recall and precision, lexical recall and precision and predictive accuracy of the spell checker. For this research, we are followed Starlander & Popescu-Belis [32] system performance evaluation metrics and suggestion adequacy of the system was not examined and their definitions of the metrics were defined below.

The evaluation technique has four categories: true positive, false positive, false negative and true negative. True positives (TP) indicate valid words recognized by the spelling checker, resulting in correct non-flags. True negatives (TN) invalid words recognized by the spelling checker, resulting in correct flags (Good flags). False negatives (FN) produced when valid words not recognized by the spelling checker, resulting in incorrect flags (False flags). False positives (FP) invalid words not recognized by the spelling checker, resulting in incorrect non- flags (also called "Missed flags").

The efficiency and accuracy of the spell checker and corrector that have been developed were evaluated using evaluation metrics for both non-word and real word spelling errors in the written texts. Therefore, detection and correction of spelling errors were determine using above metrics which are important to measure that actual performance of spell checkers. The following metrics was used to evaluate the accuracy and efficiency of the spell checker by considering the detection and correction of spelling errors in the texts.

EP	ER	LP	LR	Accuracy
TN/TN+FN	TN/TN+FP	TP/TP+FP	TP/TP+FN	$\frac{TP+TN}{TN+TP+FN+FP}$

Table 5.4: Evaluation metrics for spelling error detection and correction

Recall is a measure of the completeness of the spell checker; it tells how much of the language the spell checker covers, the lower the value the more likely it is that the spell checker could complain about correct words. It might also be useful to measure the same useful to measure the same metrics but for the case where the spell checker identifies the word as incorrect instead of correct. In that case recall is called specificity and it tells how likely it is that the spell checker would catch all incorrect words.

Lexical Recall (LR) is defined as the number of valid words in the text that are recognized by the spelling checker (i.e. true positives), in relation to the total number of correct words in the text (i.e. the sum of all true positives and false negatives). The second recall measure was *Error Recall (ER)* which is the number of invalid words in the text that are flagged by the spelling checker (i.e. true negatives), in relation to the total number of incorrect words in the text (i.e. the sum of all true negatives and false positive).

The ideal for any spelling checker would be to recognize all valid words as valid, and all invalid words as invalid, scoring 100% on both LR and ER. The recall scores are mostly an indication of the comprehensiveness of the lexicon of the spelling checker (i.e. how representative it is of the language), as well as how untainted the lexicon is (i.e. whether the spelling checker lexicon contains any erroneous words).

Precision is a measure of the exactness of the spell checker's responses; it basically tells how much trust the spell checker when it tells this word is correct. Precision pertains to the flagging accuracy of a spelling checker how accurate is the spelling checker in assigning non-flags (i.e. to recognize only correct words as correct), and how accurate is the spelling checker in producing good flags (i.e. to flag only incorrect words as incorrect).

Lexical Precision (LP) is computed by dividing all correct non-flags (i.e. true positives) by the total number of non-flags (i.e. true positives plus false positives). On the other hand, *Error Precision (EP)* is the number of correct flags (i.e. true negatives) in relation to the total number of flags assigned by the spelling checker (i.e. true negatives plus false negatives) gives an indication of the spelling checkers.

Once again, the ideal for any spelling checker would be to score 100% on both correction and detection precision, as the end-user expects of a spelling checker to flag all incorrect words, and only incorrect words. This would result in a spelling checker that is 100% accurate in the task at hand.

We also calculate accuracy which is derived from both precision and recall which measures the general quality of the spell checker. *Predictive Accuracy* is the overall performance of the spell checker that have been computed and the likelihood of any given word correct or incorrect being handled accurately by the spelling checker. This metric measure the performance of spell checker for both spelling error detection and correction of a given words in the input texts. Like all the other metrics, this score can also be represented as a percentage value, where 100% would be the ideal.

In this thesis, the performance and effectiveness of spell checker evaluated into two different independent mechanisms manually which is depend on the error types. The frequencies of errors encountered by the system are counted manually for both error types and compute the performance using different metrics. Non-word errors were first test and evaluated independently before real word errors and then the real word errors were evaluated after the non-word errors corrected and replaced by valid words in the sentences. Therefore, each single word checked and corrected using dictionary words before detecting and correcting the real word errors in the given sentences. If we were considering the performance of a non-word error checker that was dealing with each word in isolation, we would only be interested in error types as each occurrence of a particular error would be dealt with in the same way. However, a real-word error checker is dealing with each word in the context in which it occurs and so may make a different decision for the same confusable word appearing in a different context. For this reason, the results reported below use token rather than type counts.

It should be noted that the evaluation methods presented in above works on the dataset that was randomly taken from different sources. The test dataset was prepared to evaluate the number of valid words correctly accepted by the system and the number of invalid words correctly flagged by the system. Since the contextual and non-word errors checked and corrected independently, sentences were prepared for contextual real word errors after checking the invalid words in the dictionary word lists.

Initially, we randomly selected 500 sentences from different sources like magazines, books, news and stories which were belonging to several domains which produced 9115 words. To trim repeated and unnecessary words AntConc text processing tool has been used and reduced to 5350 unique words. Each word lists and sentences are printed out and then manually spell checked by two postgraduate linguistic students. We found that, the data set consists of 5223 correctly spelled words and 127 misspelled words. In this sample, out of 5223 valid Amharic words, 4989 were accepted as a valid word; 234 words were flagged as misspelled words by the system due to the absence of words in the lexicon. On the other hand, all the 127 misspelled words were flagged.

On the other hand, real word spelling errors were checked and corrected after validating non-word errors in the sentences. These errors were checked depend on the sequence of words using bigram information in the sentences which are free from any kind of non-word errors and appropriate alternative suggestions were displayed depend on the probabilistic information of bigrams. The performance of the spell checker is determined at sentence level and we have tested using 500 sentences which are used for non-word errors and free from errors. The bigrams were generated to examine the coverage of text and we found that, the dataset consist 8631 bigram word types and 9114 bigram tokens using AntConc data processing tool. Therefore, the test set data covers 6.92 % of the total training set used to train the spell checker model system that indicate the coverage set of data in training and test set of the text.

As mentioned above, depending on the experiment there are 361 non-word errors which are replaced and corrected by another words for contextual real word spell checking and every real words were tested and checked in each sentences weather invalid or not depend on the predecessor and successor of the words. The sample data set consist 5350 words and out of these of 5303 correctly spelled real words and 47 misspelled real words. According to tested experiment, from the sample 5303 Amharic valid real words 3124 were accepted as valid real words and 1876 words were flagged as misspelled real words in all sentences by the system. Whereas, from the test data 62 invalid real words errors were truly detected this is not the target word in the sentences including 47 invalid errors by the system. Therefore, among total 109 misspelled real word errors, 62 errors were flagged as misspelled real words during the non-word correction and absence of word sequences, and 47 invalid errors were produced by spell checker system in the sequence of words in the sentences.

For both type of errors, any invalid words can't consider as correct and the spell checker system refuses to accept the words. But if the corpus or dictionary contains any errors, the prototype system could recognize and accept as valid words. The recall, precision and prediction accuracy were calculated depending on the data in the in the following table.

Error type	**TP**	**TN**	**FN**	**EP**	**ER**	**LP**	**LR**	**Accuracy**
Non-word error	4989	127	234	35.18	100	100	95.52	95.62
Contextual/Real word error	3365	109	1876	5.49	100	100	63.45	64.93

Table 5.5: Evaluation results of performance of the prototype system

According experimental results, the non-word error correct detection and correction by the prototype system is 4989 and incorrect prediction is 127 words. This indicates the performance of the system scores 95.62% in which the system can identify effectively and efficiently for correct and incorrect non-word erors in the given texts. The coverage of the correct words were determined by lexical recall that has value of 95.52% and incorrect words that correctly flagged is 100% by lexical errors and all non-words errors in the given texts were perfectly detected with the comparison of dictionary words. Therefore, valid errors covered in the sample test dataset was correctly identified and marked in the prototype of the system. According to the result showed in the experiment, the lexical coverage of words accepted as valid words indicates the most of words were covered by the dictionary words. Based on this, even if all words that exist in the language could not include, we can say the training texts used in the experiment were almost complete and covers all words used to test the system except some words which are exist in the language.

Lexical and error precision also another metrics to measure the effectiveness of accepting and ignoring words in the spell checker. In the result, lexical precision shows that all the valid word accepted and recognized by the spell checker scores 100% and no invalid words accepted as correct in the system. Any words in the system is recognized and accepted which indicates there is no way of considering invalid words as correct word in the texts. The developed spell checker can identify incorrect and valid words effectively and efficiently with respect to the dataset used in the experiment. Error precision also have a value 35.18% shows that errors produced by spell checker system for valid words from the total flags because of dictionary size covering all words of the language. Therefore, in the error precision 35.18% of words were valid words which are not recognized by the spell checker system since the words do not exist in the training set used to train the model.

On the other hand, the performances of prototype for recognizing real word errors using bigram words in the sentences were measure using the above metrics. The overall performance of spell checker for real word error is 64.93% due to the size of corpus in training text. This result shows that the corpus size used for training the model is not cover all the words in the sentences. Therefore, the sequence of words in the sentences is depends on the bigram generated and needs more lexical and sentences to increase the efficiency and effectiveness of checking the spelling errors.

The lexical coverage of words was determined by lexical recall has a value 63.45% of from the sample training set used to learn the model. This shows that based on the sample test used for testing the prototype system needs improvement because the real word errors were marked based on consecutive sensitive of words in the sentences. The errors that exist in the sentences were 100% checked and the system was sensitive to detect all errors in which were not included in the bigram lists.

According to the result, the lexical precision shows that 100% distinguished all valid real words from the errors accepted as correct and invalid real words are not accepted and assumed as correct real words in the sentences. The spell checker system can catch real words exactly and accepts the valid one with respect to the training texts. In the experimental result, 5.4% words were truly recognized by the spell checker system and linguistic judgment was applied for each invalid real word errors to determine the context of words in the sequence of words. From this, when the test texts compared to the bigram words the majority of the errors produced were valid real word which is truly needed by the writers but in the corpus excluded and it's ignored to accept those words by the system. The evaluation of the spelling checker showed us that too many valid real words are not recognized due to corpus size of the training set.

The challenges behind evaluation performance of prototype system were related to lexicon size for both types of errors. The success of the spelling checker is directly related to the completeness of the lexicon. If not enough words are contained in the lexicon, too many valid words are flagged as in valid. This means that in theory all morphologically complex words and compounds of the Amharic language should be added to the lexicon. Out of vocabulary occurs due to the problem of incorporating all words of the language in the training data which reduces the performance effectiveness of the spell checker.

Furthermore, the other difficulty for performance evaluation of the spell checker prototype was the Amharic language character features such as redundancy alphabets. In this case the similar word can be written more than once with different alphabets like ሰማይ and ሠማይ. In data preparation, we tried to cover and include repeated alphabets, morphological complex and compound words in the training data to solve morphological variation and completeness of words but it’s difficult to incorporate all words of Amharic language.

Ultimately any approach to Amharic spelling correction is limited by the reliability of the reference used for canonical formations. The establishment of a comprehensive and authoritative lexicon for written Amharic would be the single most valuable resource towards the realization of this eventual goal. There are a number of linguistic complexities specific to non-native learners that a spell-checker would need to handle in order to be successful.

CHAPTER SIX
CONCLUSION AND RECOMMENDATION

6.1 Conclusions

The advancements of industry and information technology are necessary to produce electronic documents that have considerable benefits such as easy organizing and understanding information. The dynamic nature of computer technology increase in the amounts of electronic information and diversity of languages were used to produce and processed to solve recognizing and understanding problems for language users. Spell checking is the one that detect and provide spelling suggestions for incorrectly spelled words in a text. The major objective of this thesis is to develop and design context dependent spelling checker and corrector for Amharic text that detect and correct human generated spelling errors. Since Amharic does not have any spell checker and corrector that can detect and correct real word errors contextually, it plays an important role to edit Amharic texts in different sectors like offices and other areas. The design of the type of spelling checker techniques is heavily influenced by the type of spelling errors patterns. Real word and non-word errors is spelling errors that non-word errors does not exist in the dictionary or even Amharic language and real word error were found in the texts but semantically, syntactical and structural context invalid in the sentences.

The context dependent spell checker can be combined with dictionary based spelling error detection and correction application in order to create an application that is able to detect and correct non-word errors and real-word errors in text documents. Dictionary look up techniques are used to compare and locate an input string with dictionary words. N-gram based approaches was used to build a language model with more complex structure and performing to detect and correct spelling errors in context dependent spelling correction. Candidate replacements were generated for spelling erors and users can interact to the interface to choose the best ranked alternatives among the suggestion lists based on the Levenshtein minimum edit distance and bigram probabilities information.

As the performance indicates the prototype system registers 95.62% of overall accuracy, 95.62% of lexical recall and 35.18% of error recall for non-word errors. This shows that in the study promising result was registered. Whereas, the performance result of the system for context sensitive spell checker was registered 64.93% of accuracy, 63.45% of lexical recall and 5.49% of error precision.

This result indicates improvement will be needed to resolve and enhance the performance the consideration of other language models and lexical coverage of words in the Amharic texts.

6.2 Recommendations

The following recommendations were given for further research based on the observed experiment and uncover areas.

- The collection of more sample texts would be a valuable contribution to enhance the performance of the spelling checker to provide best candidate suggestions for spelling errors especially for real word errors. Thus, preparing adequate and better size corpus must be one task in the future and having a standard dictionary with maximum word size is very important to increase the accuracy of spell checker.
- In this research unsupervised machine learning approaches were used to detect and correct real word errors contextually using n-gram probabilistic information. To maximize accuracy of detecting and correcting spelling errors supervised approaches should be applied using annotated and tagged texts by integrating dictionary based with POS tagging and Morphological analysis.
- Rule based techniques should be incorporated to handle real word errors especially for phonetic spelling errors due to multiple redundant characters. For instance, the same word "Tsehaye" may be written ጸሀይ ጸኀይ ጸሐይ ፀሐይ ፀኀይ ፀሀይ depending on who write it. There should be a rule that integrate canonical and common Amharic together in the spelling checking application.
- The distance metrics should be modified and revised to provide best suggestions for correction especially with respect to the character features of the language to handle multi errors in the text.

References

[1] ባዬ ይማምና ቲም. (1997).”ፊደል እንደገና” የኢትዮጵያ ቋንቋዎችና የሥነ ፅሁፍ መጽሄት ቁጥር 7 (1-32)

[2] Aragew, A. (nd). Automatic sentence parsing for Amharic text an experiment using probabilistic context free grammars: Msc Thesis, Addis Abeba University, Ethiopia.

[3] Barari, L. and QasemiZadeh, B. (2005). Clonizer spell checker adaptive, language independent spell checker.

[4] Reda B. (1982). Graphemes Analysis of the writing system of Amharic. Addis Ababa University: Addis Abeba.

[5] Bender, Marvin, L.(1970).Problems of Transliteration into Amharic. Journal of the Language Association of Eastern Africa, 1(2), 112-115.

[6] Bender, Marvin, L. et al. (1976). The Ethiopian Writing System. In Bender et al (Eds.): Language in Ethiopia. London: Oxford University Press.

[7] Berhane, G. (1992).Word formation in Amharic: Journal of Ethiopian Languages and Literature: p. 50–74.

[8] Berlinsky-Schine, A.(2004). Context-based detection of real word typographical errors using markov models. Technical report, Cornell University, Ithaca, NY

[9] Bethlehem Mengistu. (2002). N-gram based automatic indexing for Amharic text. Msc Thesis' School of Information Studies for Africa. Addis Ababa University, Addis Ababa. (Unpublished)

[10] Bhagat, M. (2007). Spelling Error Pattern Analysis of Punjabi Typed Text: Computer Science and Engineering Department, thesis Report, Thapar Institute of Engineering and Technology Deemed University Patiala, India.

[11] Brockett, C., Dolan, W. B., and Gamon, M. (2006). Correcting ESL errors using phrasal SMT \ Techniques: In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics. Sydney, pp. 249–56.

[12] Carlson, A., & Fette, I. (2007). Memory-Based Context-Sensitive Spelling Correction at Web Scale. Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA).

[13] Chomsky, N. (1956). Three models for the description of language. IRI Transactions on Information

Theory, 2(3), 113-124.

[14] Church, K. W., Gale, W. A. (1991). Probability scoring for spelling correction. Statistics and Computing.

[15] Comeau, D. C. and Wilbur, W. J. (2004). Non-word identification or spell checking without a dictionary. Journal of the American Society for Information Science and Technology, 5(2): 169–77.

[16] Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors, in Communications of the ACM. 7(3): 171-176

[17] Daniel, J. (2000). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition: Prentice-Hall, Inc: England.

[18] Debela T. and Ermias A. (2011). Designing a Rule Based Stemmer for Afaan Oromo Text, M.Sc. Thesis, Addis Ababa University, Addis Ababa, Ethiopia.

[19] Demetriou, G., Atwell, E., & Souter, C. (1997). Large-scale lexical semantics for speech recognition support. In EUROSPEECH, pp. 2755-2758.

[20] Dawkins, C.H., (1969). The Fundamentals of Amharic: A.A Sudan interior mission.

[21] E. Mays, F. J. Damerau and R. L. Mercer (1991). Context based spelling correction. Information Processing and Management, 27(5):517–522.

[22] Gale, W.A., Church, K.W., Yarowsky, D. (1993). A method for disambiguating word senses in a large corpus. Computers and the Humanities:P.415–439

[23] Garside, R., Leech, G., Sampson, G. (1987). The Computational Analysis of English: a corpus-based approach. Long man.

[24] Gerhard B et al. (nd). Evaluating Evaluation Metrics for Spelling Checker Evaluations: Centre for Text Technology. North-West University. South Africa

[25] Getachew Haile. (1967).The problems of Amharic Writing System. Addis Ababa University. (Unpublished).

[26] Golding, A., D. Roth,(1999). A winnow-based approach to context-sensitive spelling correction. Machine Learning, 34(1-3):107–130.

[27] Golding, A., Schabes, Y. (1996). Combining trigram-based and feature-based methods for context-sensitive spelling correction. In: 34th Annual Meeting of the Association for Computational Linguistics.

[28] Golding, A. (1995). A bayesian hybrid method for context-sensitive spelling correction. In: Proceedings of the Third Workshop on Very Large Corpora. 39–53

[29] Grishman, Ralph (1994). Computational Linguistics: An Introduction. New York: Cambridge University Press.

[30] Hassan, A., Noeman, S., and Hassan, H. (2008). Language independent text correction using finite state automata. In Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP'08).

[31] Heidorn, G.E., et al. (1986). The EPISTLE text-critiquing system. IBM Systems Journal 21(3) P. 305–326

[32] Starlander, M. & Popescu-Belis, A. (2002). Corpus-based evaluation of a French spelling and grammar checker. In: Proceedings of the Third International Conference on Language Resources and Evaluation. Las Palmas, Spain. pp. 268-274.

[33] Hsuan orraine Liang, (nd). Spell Checkers and Correctors: A unified Treatment, University of Pretoria, South Africa in Proceedings of the first ICGST International Conference on Artificial Intelligence and Machine Learning AIML Cairo, Egypt, pp. 19–21.

[34] Hudson, G.(2001).Aspects of the History of Ethiopic Writing, IES Bulletin,25,1-10

[35] Jones, M.P., Martin, J.H.(1997). Contextual spelling correction using latent semantic analysis. Fifth Conference on Applied Natural Language Processing.

[36] Jurafsky, D., Martin, J. H., and Kehler, A. (2010). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. New Jersey: Prentice Hall.

[37] Kukich, K. (1991). Techniques for Automatically Correcting Words in Text: Technical report: Bellcore, South Street, Morristown.

[38] Kashefi, O., et al. (2010). Towards Automatic Persian Spell Checking. Tehran, Iran: SCICT.

[39] Kolak, O. & Resnik, P. (2002). OCR error correction using a noisy channel model. In Proceedings of the Second International Conference on Human Language Technology Research, pp. 257–62.

[40] Kukich, K. (1992). Techniques force automatically correcting words in text. ACM Computing Surveys (CSUR), 24(4): 377–439.

[41] Levenshtein, V. I. (1966), Binary codes capable of correcting deletions, insertions and reversals, in .Sov. Phys. Dokl.. 10 (Feb): 707-710

[42] Liu, V., & Curran, J. R. (2006). Web text corpus for natural language processing. In EACL, The Association for Computer Linguistics.

[43] Lowrance, R. and Wagner, R. (1975). An extension of the string-to-string correction problem, in J. ACM., 22, 2 (Apr.): 177-183

[44] Manning, C.D., Schutze, H. (1999). Foundations of Statistical Natural Language Processing. MIT Press .

[45] Dawit Bekele, (2003).The Development and Dissemination of Ethiopic Standards and Software Localization for Ethiopia: The ICT Capacity Building Program of the Capacity Building Ministry of the FDRE and United Nations Economic Commission for Africa: Addis Abeba, Ethiopia.

[46] Mays, E., Damerau, F. J., & Mercer, R. L. (1991). Context based spelling correction. Information Processing and Management, 27(5), 517-522.

[47] Michael Rosner, (nd). Advanced Topics in NLP. Department of computer science.

[48] Min, K., Wilson, W. H., and Moon, Y. J. (2000). Typographical and orthographical spelling error correction. In Proceedings of 2'nd International Conference on Language Resources and Evaluation. Athens: Greece, pp. 1781–5.

[49] Mitton, R. (2010). Fifty years of spellchecking. Writing Systems Research, 2(1): P. 1–7.

[50] Naber, D. (2003). A Rule-Based Style and Grammar Checker. Master Thesis, Computer Science Applied, University of Bielefeld.

[51] Neha G. (nd).Pratistha mathur Spell checking techniques in NLP: a survey, Department of computer science, Banasthali vidyapith, India

[52] Pedler, J. (2007). Computer Correction of Real-word Spelling Errors in Dyslexic Text. PhD thesis, Birkbeck, London University.

[53] Pollock, J.J., Zamora, A. (1984). Automatic spelling correction in scientific and scholarly text. Communications of the A.C.M. 27(4), P.358–368

[54] Ritika, M., Navjot, K. (2013). A Survey of Spelling Error Detection and Correction Techniques: Punjab: India

[55] Shamsfard, M., Fadaee, H. (2008). A hybrid morphology- based pos tagger for Persian. In Proceedings of LREC. Marrakech: Morocco.

[56] Suzan Verberne, (2002). Context-sensitive spell checking based on word trigram probabilities.MA thesis. Nijmegen: Katholieke Universiteit Nijmege.(Unpublished).

[57] Paggio P., et al. (1998). “Evaluation in the SCARRIE project”, In the proceedings of the first International conference on language resources and evaluation.

[58] Teshome, K.(1999). Word Sense disambiguation for Amharic text retrieval: a case study for legal Documents: Master Thesis, Addis Ababa University.

[59] Tong, X., Evans, D.A. (1996). A statistical approach to automatic OCR error correction in context. In: Fourth Workshop on Very Large Corpora.

[60] Van Zaanen M., et al. (2003). Improving a Spelling Checker for Afrikaans. In: GAUSTAD, T. (ed.). Computational Linguistics in the Netherlands 2002: Selected Papers from the Thirteenth CLIN Meeting. Amsterdam: Rodopi.

[61] Veronis, J. (1988).computerized correction of phonographic errors, in .Comput. Hum.., 22: 43- 56.

[62] Wagner, R. A. and Fischer, M. J. (1974), The string-to-string correction problem, in .JACM. 21, 1 (Jan.): 168-178

[63] Yacob, D. (1996). System for Ethiopic Representation in ASCII (SERA): [Cited 2015 Accessed on 12 August]; Available from: *http://www.abyssiniacybergateway.net/fidel/.*

[64] Yacob D., (nd): Application of the Double Metaphone Algorithm to Amharic Orthography: International Conference of Ethiopian Studies XV.Addis Abeba University, Ethiopia

[65] Yarowsky, D. (1994). Decision lists for lexical ambiguity resolution: Application to accent restoration in spanish and french. In: Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics. P.88–95

[66] Zelalem Sintayehu.(2001). Automatic Classification of Amharic News Items: The Case of Ethiopian News Agency. (Master's Thesis). School of Information Studies for Africa. Addis Ababa University. Addis Ababa. (Unpublished).

[67] Bizuneh Mamuye. (2003). The application of WEBSOM for Amharic text retrieval: Msc Thesis, Addis Abeba University: Addis Ababa. (Unpublished).

[68] Daniel Negussie. (2006). Writer Independent Online Handwriting Recognition for Ethiopic Characters: Msc Thesis, Addis Abeba University: Addis Ababa. (Unpublished).

[69] Islam Aminul. (2011). An Unsupervised Approach to Detecting and Correcting Errors in Text: University of Ottawa: Canada.

[70] Nadew Tademe. (2008).Formant based speech synthesis for Amharic vowels. Msc Thesis, Addis Abeba University: Addis Ababa. (unpublished).

[71] Atelach A., Lars Asker. (2007). An Amharic Stemmer: Reducing Words to their Citation Forms: Proceedings of the 5th Workshop on Important Unresolved Matters: Prague, Czech Republic.

[72] Sebsibe H/Mariam, et al. (2005). Unit Selection Voice for Amharic using Festivox: 5th ISCA Speech Synthesis Workshop Pittsburgh, page 103-107.

[73] Hussien Seid. (2005). A Speaker Independent Continuous Speech Recognizer for Amharic", INTERSPEECH, Lisbon Portugal.

[74] Solomon Teferra, et al. (nd). "An Amharic Speech Corpus for Large Vocabulary Continuous Speech Recognition", Fachbereich Informatik, University at Hamburg.

[76] Marshall, I. (1983). Choice of grammatical word-class without global syntactic analysis: tagging words in the lob corpus. Computers and the Humanities 17, P.139–150

[77] Hodge, V. J., & Austin, J. (2003). A comparison of standard spell checking algorithms and novel binary neural approach. IEEE Trans. Know. Dat. Eng., 15(5), Vol. 5, No. 3. P.1073-1081.

[78] Cheng, C. et al. (2007). SpellCheF: Spelling Checker and Corrector for Filipino: Journal of research in Science, computing, and engineering. Vol. 4 No. 3.

[79] Arvi, Tavast. et al. (2012). Human Language Technologies: The Baltic Perspective. Proceedings of the Fifth International Conference Baltic HLT 2012. IOS Pres BV. Netherlands

APPENDICES

Appendix I: List of Amharic Characters

		u	i	a	Y	e	o
H	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
L	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
H	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
M	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
S	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
R	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
S	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
B	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
T	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
C	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
N	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
X	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
K	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
W	ወ	ው	ዊ	ዋ	ዌ	ው	ዎ
Z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
D	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
J	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
G	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
T	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
F	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
P	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
V	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ
Q	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ

		u	i	a	Y	e	o
Sht+ n	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
Sht+ z	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
Sht+ y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
Sht+ s	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
Sht+ h	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
Sht+ p	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
Sht+ t	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
Sht+ T	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
Sht+ x	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
Sht+ c	ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
Sht+ H	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ

Labialized Alphabets

LWA	ሏ	NWA	ኗ	BWA	ቧ	CWA	ቿ
MWA	ሟ	ZWA	ዟ	Sht+TWA	ጧ	TSWA	ጿ
RWA	ሯ	Sht+ZWA	ዧ	Sht+CWA	ጯ	FWA	ፏ
SWA	ሷ	DWA	ዷ	Sht+NWA	ኟ	VWA	ቯ
SHWA	ሿ	JWA	ጇ	TWA	ቷ	Sht+2	ኧ

	O	I	u	A	e
KW	ኰ	ኲ	ኵ	ኳ	ኴ
GW	ጒ	ጒ	ጒ	ጓ	ጔ
QW	ቈ	ቊ	ቋ	ቋ	ቊ
HW	ዀ	ዂ	ዂ	ዃ	ዄ

Appendix II: Sample Amharic Text for testing Prototype of Amharic spell checker

የባቡር መስመሩ ከኢትዮጵያ ድንበር አልፎ በጂቡቲ ድንበር ውስጥም የሚረዘጋ በመሆኑ ወደፊት የሚጀመረውን የትራንስፖርት አገልግሎት በኢትዮጵያ ብቻ እንዲተዳደር ማድረግ እንደማይቻል ምንጮች ያስረዳሉ፡፡ በመሆኑም በሁለቱ አገሮች ሰምምነት መሠረት አንድ የትራንስፖርት አገለግሎት የሚሰጥ ኩባንያ መመረት ስላለበት ለዚህም ሰባል ውይይት ተጀምሯል ብለዋል፡፡ በኢትዮጵያ የንግድና ዘርፍ ማኅበራት ምክር ቤት፣ የመንግሥትና የግሉ ዘርፍ የምክክር መድረክ በልኩ፣ ከመንግሥት አካል ጋር የተደረገው ውይይት ቱሪዝምን በሚመለከት ከዚህ ቀደም ከተካሄዱ ውይይቶች በተሻለ ደረጃ መግባባት የተደረሰበት መስሏል፡፡ የጋራ የምክክር መድረኩ ከተጀመረበት ጊዜ ወዲህ አንድን ዘርፍ በመነጠል ውይይት ሲካሄድ የቱርዝም ኢንዱስትሪው ብቸኛው ዘርፍ መሆኑ ነው፡፡ግብፅና ሱዳን አንፈርም ያሉት ይኼ የትብብር ማዕቀፍ በአሻጥርና ሴራ እንዳይደናቀፍ ልዩ ጥንቃቄና ትኩረት ይደረግ፡፡ከተፋሰሱ አገሮች ጋር በመሆን የናይልን ውኃ በፍትሐዊ መንገድ ለመጠቀም የሚደረገውን እጥረትና የሚያጋጥመውን ፈተና ዓለም አቀፉ ማኅበረሰብ ይገነዘበው አንድ ትልቅ ዘመቻ መደረግ አለበት፡፡ በተለይ ግብፅ ኢትዮጵያ ከሌሎቹ አገሮች ለመነጠል የምታደርገው ዘመቻ አሁንም ውስጥ ለውስጥ የተጠናከረ በመሆኑ፣ የኢትዮጵያ መንግሥት ከፍተኛ የዲፕሎማሲ ሥራ ሥራት እንዳለበት ለደቂቃም ቢሆን መዘጋት የለበትም፡፡ ቦርዱ ስልጣን የለውም የተባለው ጉዳዩ በር ላይ በደረሰ ጊዜ ሲሆን ስልጣንን አመልክቶ ከታች ጀምሮ የተነሳ ክርክር የለም የትምህርት መረጃ በቋንቋ የመጀመሪያ ዲግሪ ይልቁንም መንግሥት በልበ ሰፊነትና በታጋሽነት እነዚህ የጠላት ተላኪ ሊሆኑ የሚችሉ ወገኖች በአስቸኳይ ወደ ድርድር መጥተው በሰላማዊ መንገድ እንዲንቀሳቀስ መደገፍ አለበት፡፡ ከጠላት ጋር እያበሩ ወገንን ከሚያቆስሉና ከሚደሙ ለዲሞክራሲያዊ ሥርዓት ግንባታ የበኩላቸውን እንዲያደርጉ ቢያቀርባቸው ይመጣል፡፡ መንግሥት በኩሉ በሰላማዊ መንገድ ከሚገሉ የፖለቲካ ድርጅቶች ጋር በሆነ ባልሆነው ከመነታረክ ይልቅ ለዲሞክራሲ ማንበብ አስተዋጽኦ ያበረክቱ ዘንድ ድርድር ማድረግ አለበት፡፡ በቅድመ ሁኔታዎች የተጠሩ አለመግባቶች ተፈተው በብሔራዊ ጉዳዮች ላይ የጋራ አቋም እንዲይኖር፣ ካልሆነም የተሻለ መፍትሔ ለማፈላለግ ተገቶ መሥራት አለበት፡፡በፖለቲካው መቆለፍ ምክንያት የተጠሩ ቅሬታዎችና አለመግባባቶች ባሉበት ሁኔታ ውስጥ ስለ ብሔራዊ ጥቅምና ደኅነት መነጋገር ፋይዳ የለውም፡፡ ሐሳብን በነፃነት የመግለጽ መብት፣ የፕሬስ ነፃነት፣ የዲሞክራሲና የሰብዓዊ መብቶች፣ የሕግ የበላይነት፣ የዜጎች ሁለንተናዊ ተሳትፎ፣ ወዘተ ሲበሩ ዜጎች በጋራ ለአገራቸው ህልውና ይቆማሉ፡፡ ይኼ ዘገምተኛ ፖለቲካችን ተነቃቅቶ የሰከነ ድባብ ሲፈጥር እንደ አገር ሁላችንንም ሊያስደስተን ይገባል፡፡ የአካባቢው ነጋዴዎች በአዲስ አበበ ከተማ አስተዳደር የመሬት ልማትና አስተዳደር ቦርድ የሰጠውን ውሳኔ በመቃወም አቤቱታቸውን በተደጋጋሚ ቢያሰሙም ምላሽ ሊያገኙ አለመቻላቸውንና ያለአግባብና ያለአንዳች ምትክ ቦታ እዳ ላይ ሊጣሉ መሆናቸውን ይገልፃሉ፡፡ በአንድ ቤት ውስጥ አምስትም አስር ሃይማኖት ሊኖር ይችላል፡፡ በአንድ ቤት ውስጥ ግን አምስትም አስርም ገር ሊኖር አይችልም፡፡ በአዲስ አበባ የሚገኙ ወጣቶችና ወጣት ጥንዶች እየተዛመተ የመጣው የፍቅረኞች ቀን አከባበር ባለፉት ጥቂት ዓመት ውስጥ አገር በቀል ባዐል የመሰለ መጥቷል፡፡ ኢትዮጵያውያን በበርካታ ባህሎቻችንና የእኛ በምንላቸው መገለጫዎች በዓለም ላይ ብታታወቅም ከነኚህ ዘመንና ትውልድ ተጋሪ ባህሎቻችን መካከል የተወሰኑት ከሌሎች የተቀላቀሉ እንደሆነ የሠርግ ሥነ-ሥርዓታችንን በማንሳት የሚናገሩ አሉ፡፡ በቬሎ፣ በሱፍ፣ በብልጭልጭና በሻምፓኝ እየታጀበ የሚሠረገው የኛው የከተማ ሠርግ አገር በቀል አለመሆኑንና ወደ ፈረንጅ አገር ወር ብለው የተመለሱ የአገር ልጆች የመጡት የውጭ አገር የሠርግ ስነ-ሥርዓት ነው፡፡ የፍቅረኞች ቀንም እዚሁ ከሠርግ ስነ-ሥርዓት ጋር ተመሳሳይነት ያለው ክስተት ነው፡፡ የመጀመሪያው ምክንያት ዓለም አቀፋዊነቱ ሲሆን፣ ሌላው ደግሞ ይህንኑም ባል በነዛ የሠለጠኑ አገራት ለትምህርትና ለሥራ ተጉዘው የተመለሱት የወንዝ ልጆችና የቴክኖሎጂና የኔትዎርክ መስፋት ወዲህ አሻግረውታል፡፡ በእርግጥ ቀኑ አጥር ነበር፡፡ መጀመሪያ መጥሪያ አዘጋጅተን ለየትምህርት ቤቱ በተን፡፡ አንዳንድ ትምህርት ቤቶች ከአንድ በላይ መምር መላክ ቢፈልጉ ያለን አቅም ከየትምህርት ቤት አንድ አንድ መምህር የማሠልጠን ነበር፡፡ ወቅቱ ሰኔ

ወርህ ላይ ነበር፡፡ ለጉዳዩ ባለመገንዘብ ወይም የፈተና ጊዜ ስለነበረ ሊሆን ይችል አንዳንድ ትምህርት ቤቶች መምህር አላኩም፡፡ የመጡትን መምህራን ለስድስት ቀን አሠለጠንን፡፡ ሥልጠናው በጣም ስኬታማ ሆኖ ነው የተጠናቀቀ፡፡ አዘጋጆቹ እንደሚያምኑት በተለያዩ የኪነ ጥበብ ሥራዎች ማለትም በግጥም፣ በሥዕል፣ በዘፈን የማኅበረሰቡን ልብ በቀላሉ መግዛት እንደሚቻልና የተለዩ ጉዳዮችን ለማስተላለፍና እንዲሁም አዎንታዊ ለውጥ ማጣት እንደሚቻል ነው፡፡ ድርጅቱ እንደሚለው ከአዳዲሶቹ ለካንሰር የመጋለጥ አጋጣሚዎች ከሁለት ሦስተኛ ላይ የሚሆነው የሚከሰተው በልማት ወደኋላ በቀሩ ወይም ገና በልማት ላይ ባሉ ሃገሮች ውስጥ ሲሆን አስደንናገጭ ነው ባለው ሁኔታም እየተስፋፋ ነው፡፡ ሁለት ነፍሰጡር ተፈናቃዮች መኪና ላይ ወልደዋል ይሁን እንጂ የእነዚህ ተቋማት እንቅስቃሴ እንደሚጠበቀው የተጓዘ ነው ወይ፡፡ይህ ማንነቴ ማንንም አይመለከተውም፡፡ ምሩቃኑ በተለያየ የዕድሜ ክልል የሚገኙ ወጣትና ጎልማሳ ተማሪዎች ሲሆኑ የሰለጠኑትም በኪቦርድና በክራር፣ በማሲንቆና በጊታር የሙዚቃ መሣርያዎችና በድምፅ መሆኑን የትምህርት ቤቱ ርእሰ መምህር አቶ ብርሃኑ ሳህሉ በምረቃው ዕለት ተናግረዋል፡፡ ወደ ሥልጠናው ከመጡት መካከልም ያለሙያቸው ሥዕል አስተምሩ ተብሎ የተሰጣቸው ይገኙበታል፡፡ ለምሳሌ ከመስቀል አደባይ እስከ ወሎ ሠፈር ያለው መንገድ በጥሩ ሁኔታ የተገነባ ሲሆን፣ ከመስቀል አደባባይ እስከ ጎተራ ያለው መንገድ ደግሞ ሰፋ ብሎ ተገንብቶ አልቋል፡፡ለፍቅር ሲባል አይደለም ገንዘብ እና ጉልበት ቤተ ክርስቲያን የከፈለችው የክርስቶስን ደም መሆኑን መዘንጋት አይገባም፡፡ በመጀመሪያ ደረጀ ማንም ሰው በአንድ አገር ውስጥ መወለዱ ወዶት ወይም አቅዶት የሚያደርገው ነገር አይደለም፡፡ በተጨማሪም ከዚህኛው ወይም ከዚያኛው ብሄረሰብ የወለዱ ያጋጣሚ ጉዳይ እንጂ ከመጀመሪያውኑ የሚታቀድ አይደለም፡፡ ስለዚህም አንደኛው አማራ፣ ሌላው ትግሬ፣ ሌላው ደግሞ ኦሮሞ ስለሆነ በኩራት ወይም በዝቅተኛ ስሜት ራሱን የሚያስጨንቅበት ምክንያት የለውም፡፡ የዚህ ዐይነቱ የመናናቅና የመጠላላት ስሜት የሚመነጨው በመሰረቱ ሰፋ ያለ ጭንቅላትን ብሩህ የሚያደርግ ዕውቀት ስላልተስፋፋና፣ ከዚህም ሆነ ከዚያኛው ብሄረሰብ ተወለድኩኝ የሚል ግለሰብ ራሱን ስለማይጠይቅና ስለማይመራመር ሁልጊዜ በመጠራጠር ዓለም ውስጥ እንድንኖር ተገደናል፡፡ የታንዛኒያው ፕሬዚዳንት ዶክተር ጃካያ ኪክዌቴ በበኩላቸው እንደተናገሩት በኩባ መንግሥት የቴክኖሎጂ ድጋፍ የተገነባው የወባ በሽታ መከላከላያ ማምረቻ ፋብሪካ በአገሪቱ ብሎም በአፍሪካ የወባ በሽታን ለመከላከል የበኩሉን አስተዋጽኦ ያደርጋል፡፡ የአፍሪካ መንግሥታት መሪዎች የወባ በሽታ መከላከል ቅንጅት የወቅቱ ሊቀመንበር ጠቅላይ ሚኒስትር ኃይለማሪያም ደሳለኝ ፋብሪካውን ለመመረቅ በታንዛኒያ በመገኘታቸው በአገሪቱ ሕዝብና መንግሥት ስም ምሥጋና አቅርበዋል፡፡ ሰፋ ያለ በሁሉም አቅጣጫ የሚገለጽ ዕውቀት አለመዳበር የግዴታ አገር የሚለውን ስም ወይም ደግሞ ኢትዮጵያን እንድንጠላ ሊያደርገን በቅቷል፡፡ ኢትዮጵያ የሚለው ስም ከጭቆናና ከብዝበዛ ጋር በመያያዝ የህብረተሰብአችንን የተወሳሰበ የታሪክ ሂደት እንዳንረዳን በማድረግ የሚያሳፍር ሆኗል ፡፡ ብሄረተኝነት እንዳይዳብርና ለአገር በጋራ ተነስተን እንዳንሰራ እንቅፋት ሆኗል፡፡ የብዙዎቻችን ችግር፣ በተለይም በብሄረሰብ መነፅር የታወርን ማንነታችንን ለመግለፅ የምንሞክረው የመጣንበትን አካባቢ አጉልቶ በማሳየት እንጂ፣ እያንዳንዳችን በአንድ ትልቅ ኮስሞስ ውስጥ ተጠቃለን የምንገኝና፣ ማንነታችንም ሆነ ምኞታችን በጠባቡ ኮስሞስ ውስጥ ሳይሆን በሰፊው ኮስሞስ ውስጥ ብቻ ተግባራዊ ማድረግ እንደምንችል አይደለም የምንገነዘበው ፡፡ የብዙዎች በብሄረሰብ መነፅር የታወሩ ግለሰቦችም ሆነ ቡድኖች ዋናው ችግር የተፈጥሮን ህግ በፍጹም ያለመረዳት ችግር ነው፡፡ ለምሳሌ አንድ ልጅ ከተፀነሰ በኋላ እዚያው እናቱ ማህፀን ውስጥ ለዘለዓለም ሊኖር አይችልም፡፡ ከዘጠኝ ወር በኋላ ተረግዞ ሲወለድ ሊያድግ የሚችለው፣ ሙሉ አካል ኖሮት ሊያስብና ዕውነተኛ ነፃነቱንም ሊቀዳጅ የሚችለውና ፈጣሪ የሚሆነው ሰፋ ባለው ኮስሞስ ውስጥ አየር መተንፈስና ሙቀት ማግኘት ሲችል ብቻ ነው፡፡ ይህ ዐይነቱ ባይሎጂካዊ ክንውን ለአገርና ለህብረተሰብ ግንባታም ያገለግላል፡፡ ብሄረሰቦችና ጎሳዎች የሚባሉት ነገሮች ዕውነተኛ ነጻነታቸውን መቀዳጀት የሚችሉት በሰፊው ዕውነተኛ ነፃነትና ብርሃን በሚፈናጣቅበት ኮስሞስ ውስጥ ገብተው መዋኘት የቻሉ እንደሆን ብቻ ነው፡፡ ይህም ማለት ከጠባቡና ከማያፈናፍነው ጥቃቅኑ የብሄረሰብ ዓለም ሰፊው አገር ለማንኛውም ግለሰብም ሆነ፣ የዚህ ወይም የዚያኛው ብሄረሰብ

አባል ነኝ ለሚል የበለጠ ለመዳብር ያመቸዋል፡፡ በሌላ ወገን ደግሞ በአንድ ትልቁ ኮስሞስ ውስጥ የተጠበቀው የፀሀይ ብርሃንና ነፃነት መፈናጠቅ የማይችል ከሆነ፣ ለምን እንደዚህ ዐይነቱ ነፃነትን አፋኛና ዕድገትን የሚቀናቀን ሁኔታ ሊፈጠርቻለ ብሎ በህብረተሰብ ሳይንስ ዘዴና በፍልስፍና መነፅር መመርመሩ እጅግ አስፈላጊ ነው፡፡ ይህን ከማድረግ ይልቅ በጥላቻ ተነስቶ አንድን የህብረተሰብ ሁኔታ በጨቋኝና በተጨቋኝ በመከፍል አጉል ትግል መጀመርና ማካሄድ ወደ ዕውነተኛው ነፃነት ከማምራት ይልቅ ትግሉን ውስብስብ በማድረግ የጭቆናው ዘመን እንዲራዘም ያደርጋል፡፡ ካሊያም ደግሞ ጥራት የጎደላቸው ኃይሎች ስልጣን ላይ በመውጣት ከነፃነት ይልቅ ሌላ የረቀቀ ጭቆናንና ህብረተሰብአዊ መመሰቃቀልን ያስከትላሉ፡፡ የአባይ ዉሃ ለግብፅ የሞትና ህይወት ጉዳይ መሆኑን አድርዉስ አታ አሳሞዋ ከሁለቱ ሃገሮች ጦርነት ግን ማንም እንደማያተርፍ አጽንኦት ሰጥተዋል፡፡ የተፋሰሱ ሃገሮች ቀደም ሲል የተስማሙበትን ስምምነትም ግብፅ ከልቧ የተቀበለችዉ እንደማይመስላቸዉም አመልክተዋል ፡፡ በአንፃሩ የግብፅና የኢትዮጵያ መንግስታት ይህን የአባይን ጉዳይ ከፍ ያደረጉት በወቅቱ በየበኩላቸዉ በአገር ዉስጥ የገጠማቸዉን የፖለቲካ ጫና ለማስተንፈስ ነዉ የሚሉ ወገኖች አሉ፡፡ በተከታታይ በቀረቡት ጽሑፎች ላይ ሁለት አንባቢዎች በተቸገሩባቸው ሀሳቦች ላይ አስተያየቴን ልስጥ፡፡ በመጀመሪያ አርበኛና ባንዳ የሚባሉትን ቃላት የማይወዳቸው አንባቢ ችግሩ አልገባኝም፤ ስገምተው ከቅን መንፈስና ከኢትዮጵያዊ ይሉኝታ ጋር የተያያዘ ይመስለኛል፤ አርበኞችና ባንዶች መኖራቸውን የካደ አይመስለኝም፤ አርበኞች ነበሩ፤ አሁንም አሉ፤ ባንዶችም ነበሩ፤ አሁንም አሉ፤ አርበኛም የለም፤ ባንዳም የለም ለማለት አልችልም፤ ባንዳነት ሲነሣ ኅሊናቸውን የሚቆረቁራቸውና የሚያፍሩ ባንዳዎችና የባንዳ ልጆች ከአሉ፣ ይቆርቁራቸው፤ ይፈሩ፤ የሚያሳፍር ሥራ ውጤት ነው፤ አርበኝነት ሲነሣ ልባቸው የሚያብጥና የሚኮሩ ካሉ፣ ይበጡ፤ ይኩሩ፤ የሚያኮራ ሥራ ውጤት ነው፤ ምንድን ነው ስሕተቱ? የታሪካችን መክሸፍ አንዱ ምክንያት በይሉኝታ ዓይኖችን ሸፍኖ ፍሬውንና እንክርዳዱን ለመለየት አለመፈለግ ነው፤ (በአሥራ ዘጠኝ መቶ አርባዎቹ አጋማሽ ላይ በጾም ጊዜ በምግብ ቤቶች የሚቀርብ የፍስክ ወጥ ነበረ፤ ሽፍንፍን ይባል ነበር፤ ሳይጾሙ የሚጾሙ መስሎ ለመታየት፤) እንክርዳዱንና ፍሬውን እኩል አድርጎ ለማየትና ለማሳየት መ ፈለግ ነው፤ የኢትዮጵያ አርበኞች ታሪክ ተዳፍኖ የቀረው በአጤ ኃይለሥላሴ መንግሥት ውስጥ ባንዶች ተሰግስገውበት ስለነበረ ነው፤ በዚህም ምክንያት ፍሬና እንክርዳዱ ሳይለይ አንዱ ትውልድ ለሚቀጥለው ትውልድ እያስተላለፈ መማርና መሻሻል ያቅተናል፤ አንባቢው ይህንን ይመኛል ብዬ አልገምትም፤ እውነትን የሚገፋ፣ እውነትን የሚያደበዝዝ፣ እውነትን የሚያጨልም ነገር ሁሉ በአገርም ሆነ በግለሰብ ደረጃ መጨረሻው ገደል ነው፡፡

Appendix III: Sample Code of a Prototype Context based spell checker for Amharic

```
package amharictexteditor;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Dimension;

import javax.swing.*;

import java.awt.event.*;

import java.io.*;

import java.awt.Font;

import java.awt.event.KeyEvent;

import java.io.BufferedReader;

import java.util.HashMap;

import java.util.StringTokenizer;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.text.MutableAttributeSet;

import javax.swing.text.SimpleAttributeSet;

import javax.swing.text.StyleConstants;

import javax.swing.text.StyledDocument;

import unicodewriting.GeezInPlainText;

public class spellChecker extends JFrame {
```

```
Dictionary1 dictionary;

ContextChecker contextcheck;

Suggestion suggestion;

Suggestion2 suggestion1;

KeyboardEvent keyevent;

public JFrame frame;

private JPanel panel1, panel3;

public JTextPane fileArea;

public Font textFont;

public static HashMap stringTokens = new HashMap();

public spellChecker() {

   contextcheck = new ContextChecker(spellChecker.this);

   dictionary = new Dictionary1(spellChecker.this);

   AmharicSpeller();

   keyevent = new KeyboardEvent();

   keyevent.setFontSize(16);

   fileArea.addKeyListener(keyevent);

   suggestion = new Suggestion(spellChecker.this);

   suggestion1 = new Suggestion2(spellChecker.this);

   fileArea.addMouseListener(suggestion.mac);

}
```

```
public void AmharicSpeller() {

    frame = new JFrame("AMHASPELL");

    panel1 = new JPanel();

    panel3 = new JPanel();

    fileArea = new JTextPane();

    textFont = new Font("Nyala", 2, 16);

    fileArea.setFont(textFont);

    fileArea.setText("");

    fileArea.setEditable(true);

    fileArea.requestFocus();

    JScrollPane scroller = new JScrollPane(fileArea);

    scroller.setPreferredSize(new Dimension(800, 375));

    JSeparator separator = new JSeparator(SwingConstants.HORIZONTAL);

    separator.setForeground(Color.red);

    JMenuBar menuBar = new JMenuBar();

    JMenu fileMenu = new JMenu("File");

    JMenu editMenu = new JMenu("   Edit");

    JMenu helpMenu = new JMenu("  Help");

    JMenuItem newMenuItem = new JMenuItem("New", KeyEvent.VK_N);

    JMenuItem openMenuItem = new JMenuItem("Open...", KeyEvent.VK_O);

    JMenuItem saveMenuItem = new JMenuItem("Save", KeyEvent.VK_S);
JMenuItem saveAsMenuItem = new JMenuItem("SaveAs", KeyEvent.VK_A);

JMenuItem exitMenuItem = new JMenuItem("Exit", KeyEvent.VK_X);

JMenuItem cutMenuItem = new JMenuItem("Cut");

JMenuItem copyMenuItem = new JMenuItem("Copy");

JMenuItem pasteMenuItem = new JMenuItem("Paste");

JMenuItem helpMenuItem = new JMenuItem("Help");

JMenuItem aboutMenuItem = new JMenuItem("About");

fileMenu.add(newMenuItem);

fileMenu.addSeparator();

fileMenu.add(openMenuItem);

fileMenu.addSeparator();

fileMenu.add(saveMenuItem);

fileMenu.addSeparator();

fileMenu.add(saveAsMenuItem);

fileMenu.addSeparator();

fileMenu.add(exitMenuItem);

editMenu.addSeparator();

editMenu.add(cutMenuItem);

editMenu.addSeparator();

editMenu.add(copyMenuItem);

editMenu.addSeparator();
```

```
        editMenu.add(pasteMenuItem);

        helpMenu.add(helpMenuItem);

        helpMenu.addSeparator();

        helpMenu.add(aboutMenuItem);

        menuBar.add(fileMenu);

        menuBar.add(editMenu);

        menuBar.add(helpMenu);

        frame.setJMenuBar(menuBar);

        panel1.add(scroller);

        panel3.add(BorderLayout.CENTER, separator);

        frame.getContentPane().setLayout(new BoxLayout(frame.getContentPane(),
BoxLayout.Y_AXIS));

        frame.getContentPane().add(panel3);

        frame.getContentPane().add(panel1);

        newMenuItem.addActionListener(new newMenuItemListener());

        openMenuItem.addActionListener(new openMenuItemListener());

        saveMenuItem.addActionListener(new saveMenuItemListener());

        saveAsMenuItem.addActionListener(new saveMenuItemListener());

        exitMenuItem.addActionListener(new exitMenuItemListener());

        cutMenuItem.addActionListener(new cutMenuItemListener());

        copyMenuItem.addActionListener(new copyMenuItemListener());
```

```
    pasteMenuItem.addActionListener(new pasteMenuItemListener());
    helpMenuItem.addActionListener(new helpMenuItemListener());
    aboutMenuItem.addActionListener(new aboutMenuItemListener());
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(950, 550);
    frame.setVisible(true);
    frame.setTitle("AMHASPELL");
}
public static void main(String[] args) {
    new spellChecker();
}
private void clearArea() {
    fileArea.setText("");
}
private void openFile(File file) {
    frame.setTitle(file.getName());
    clearArea();
    try {
        try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
            String line = null, content = "";
            while ((line = reader.readLine()) != null) {
                content += line;
                content += "\n";
            }
            fileArea.setText(content);
        }
    } catch (IOException ex) {
        System.out.println("cannot open file");
        ex.printStackTrace();
    }
}
private void saveFile(File file) {
    frame.setTitle(file.getName());
    try {
        try (FileWriter writer = new FileWriter(file)) {
            writer.write(fileArea.getText());
        }
    } catch (IOException ex) {
        System.out.println("cannot save file");
        ex.printStackTrace();
    }
}
```

```
public void highlightContextualError(int sp, int len) {
   MutableAttributeSet attr = new SimpleAttributeSet();
   StyleConstants.setBackground(attr, Color.yellow);

   StyledDocument doc = (StyledDocument) fileArea.getDocument();
   doc.setCharacterAttributes(sp, len, attr, false);
}
public void highlightDictionaryError(int spd, int len) {
   MutableAttributeSet attr = new SimpleAttributeSet();
   StyleConstants.setBackground(attr, Color.red);
   StyledDocument doc = (StyledDocument) fileArea.getDocument();
   doc.setCharacterAttributes(spd, len, attr, false);
}
public class newMenuItemListener implements ActionListener {
   @Override
   public void actionPerformed(ActionEvent event) {
      clearArea();
   }
}
public class openMenuItemListener implements ActionListener {
   @Override
```

```
    public void actionPerformed(ActionEvent event) {
        JFileChooser fileOpen = new JFileChooser();
        fileOpen.showOpenDialog(frame);
        openFile(fileOpen.getSelectedFile());
    }
}
public class exitMenuItemListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent event) {
        try {
            System.exit(0);
        } catch (Exception ex) {
            ex.printStackTrace();
        } }
}
public class saveMenuItemListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent event) {
        JFileChooser fileSave = new JFileChooser();
        fileSave.showSaveDialog(frame);
        saveFile(fileSave.getSelectedFile());
    }
}
public class cutMenuItemListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent event) {
        fileArea.cut();
    }
}
public class copyMenuItemListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent event) {
        fileArea.copy();
    }
}
public class pasteMenuItemListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent event) {
        fileArea.paste();
    }
}
public class helpMenuItemListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent event) {
    }
}
public class aboutMenuItemListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent event) {
    }
}
class KeyboardEvent extends GeezInPlainText {
    String delimiter = " !?\u1362";
    public KeyboardEvent() {
        super(fileArea);    }
    @Override
    public void keyTyped(KeyEvent key) {
        if (key.getKeyChar() == ' ') {
            String newword = "";
            int position = fileArea.getCaretPosition();
            int start = position - 1;
            String content = fileArea.getText();
            while (true) {
```

```
        //first word
        if (start < 0) {
            start = 0;
            newword = content.substring(start, position).trim();
            break;
        }
        char ch = content.charAt(start);
        if (delimiter.contains("" + ch)) {
            newword = content.substring(start, position).trim();
            break;
        }
        start--;
    }
    boolean correct = dictionary.isCorrectWord(newword);
    if (correct == false) {
        highlightDictionaryError(start + 1, newword.length());
    }
} else if (key.getKeyChar() == '.' || key.getKeyChar() == '?' || key.getKeyChar() == '!') {
    checkContext();
}
super.keyTyped(key);
```

```
    }
    @Override
    public void keyReleased(KeyEvent e) {
    }    }
public void checkContext() {
    int sentstart = -1;
    String content = fileArea.getText();
    int loc = fileArea.getCaretPosition();
    int punc1 = content.lastIndexOf("\u1362", loc - 1);
    int punc2 = content.lastIndexOf("!", loc - 1);
    int punc3 = content.lastIndexOf("?", loc - 1);
    if (punc1 > punc2 && punc1 > punc3) {
        sentstart = punc1;
    } else if (punc2 > punc1 && punc2 > punc3) {
        sentstart = punc2;
    } else if (punc3 > punc1 && punc3 > punc2) {
        sentstart = punc3;  }
    //check if there is space before the current sentence or after the previous sentence
    if (sentstart == -1) {
        sentstart = 0;
    } else if (content.charAt(sentstart + 1) == ' ') {
            sentstart = sentstart + 2;

        } else {

            sentstart = sentstart + 1;

        }

        String sentence = content.substring(sentstart, loc);

        StringTokenizer tokenizer = new StringTokenizer(sentence, " ()[]\u1061");

        boolean nonworderror = false;

        while(tokenizer.hasMoreTokens()) {

            String temp = tokenizer.nextToken().trim();

            if(!dictionary.words.containsKey(temp)){

                nonworderror=true;

                break;

            } }

        if(!nonworderror) {

            contextcheck.checkContext(sentence, sentstart);

        }

    }

}

//
```