Addis Ababa University

College of Natural Sciences

Morphology Based Spell Checker for Kafi Noonoo Language

Fikru Tafesse Bekele

A Thesis Submitted to Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia

October, 2018

Addis Ababa University

College of Natural Sciences

Fikru Tafesse Bekele

Advisor: Yaregal Assabie (PhD)

This is to certify that the thesis prepared by *Fikru Tafesse Bekele*, titled: *Morphology Based Spell Chekcer for Kafi Noonoo Langauge* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the examining committee:

Name	Signature	Date
Advisor: Yaregal Assabie (PhD)		
Examiner: Solomon Atnafu (PhD)		
Examiner: Solomon Gizaw (PhD)		

Abstract

There are a number of NLP tools that are used in processing texts and other human languages. Among these tools spell checker is one that check the validity of words in the document. Spell checker is NLP application that is needed for every word processing document that analyze the input text for misspelled words and then provides possible suggestions for misspelled word for making correction. Two class of error in spelling error check: non-word error and real-word error. Non-word error is an error word that is misspelt and have no meaning in that specific language. Real-word error is a word that have meaning in that specified language but semantically and syntactically incorrect. Real word error is difficult to detect and provide suggestions and it needs syntactic and semantic analysis of the text. Dictionary look up and N-gram analyses are the most common used spelling error detection approaches. Edit distance, noisy channel model, neural network, rule-based, N-gram, phonetic based techniques are applied to generate suggestions for error correction. In spell checking area, a lot of work has been done in English, Arabic and Asian languages. Kafi Noonoo is one of the language spoken in South West part of Ethiopia by Kaffecho people. It is morphological rich language. There is no available spell checker for Kafi Noonoo language to analyze text written using this language which we were work on it.

This thesis work is aimed to design and implement a spell checker system for Kafi Noonoo language. The proposed architecture of spell checker contains four main components: tokenization, error detection, word suggestion and error correction and with two backend components. Dictionary look up approach and morphology based approaches are used to implement the spell checker for Kafi Noonoo language.

The prototype of the system is developed to test and evaluate the functionality and performance of the spell checker system. To test and evaluate the system, we used 2743 unique words collected from different sources. To measure the accuracy of the spell checker system lexical recall, error recall and precision evaluation metrics were used. Based on these evaluation metrics we get promising result of 95.91% lexical recall, 100% error recall and 62.76% precision.

Key words: Error Detection, Error Correction, Spell Checker, Morphology, Non Word Error, Real Word Error, Kafi Noonoo, edit distance, dictionary lookup

Acknowledgments

First of all, my thanks goes to an almighty God for giving me strength and wisdom to complete this work. Then after, my deep thanks goes to my advisor Yaregal Assabie (PhD) for his guidance, assistance and provide a valuable comments, suggestions on this thesis work from starting to end without tiredness.

I would like to thank Ambo University for giving me this opportunity and support me financial to complete this work and also deep thanks to Addis Ababa University specially Computer Science Department for their support with different educational materials and others.

I also wish to thank my beloved family for their love, encouragement and support throughout my academic life. I would like to express my deepest thanks to my mother Abeba Beyecha who love, encourage and lead me to this success both as mother and father. Also my endless love and thanks goes to Asmamawu W/Mariam, Diriba Gudisa, Boja Hunde, Sisay Eshetu, Endale Assefa, Tejitu Tafesse and others for the time being I forgot their names who supports and encourage me to achieve this goal.

Finally, but not least, I would like to thank Bonga College of Teachers Education, department of Kafi Noonoo Language instructors especially Kifle Meshesha, Abebech Assefa for supporting me with data for this work, providing and discussing on the linguistic structures of Kafi Noonoo language.

Dedications

I would like to dedicate this paper work to my mother Abeba Beyecha.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Acronyms and Abbreviations

BCTE	Bonga college of teachers education
BSA:	Binary search algorithm
F:	Female
GPO:	Government printing office
HBSA:	Hashing based searching algorithm
KN:	Kafi Noonoo
KNSC:	Kafi Noonoo spell checker
KSs:	Knowledge sources
LED	Levensthein edit distance
M:	Male
NLP:	Natural language processing
OCR:	Optical character recognition
POST	Part of speech tagger
SSCS:	Smart spell-checking system
W7:	Webster's seventh

Chapter One: Introduction

1.1 Background

Natural Language Processing (NLP) is a field which employs computational techniques for the purpose of learning, understanding and producing human language content [1]. NLP is the use and ability of systems to process sentences in a natural language such as English, Amharic rather than in a specialized artificial computer language such as C++, Java [2]. It sits at the intersection of computer science, artificial intelligence and computational linguistics. Early computational approaches to language research focused on automating the analysis of the linguistic structure of language and developing basic NLP applications like machine translation, speech recognition, speech synthesis, spelling and grammar correction. The goal of NLP is to design and build software that will analyze, understand and generate languages that humans use naturally. Natural language system is easiest for humans to learn and use but hardest for a computer to master [1, 2]

Among the application areas of NLP tasks, the task of spell checking important for processing a text that is error free [3]. Spell checker is an application that identifies misspelled words, provides appropriate suggestion to misspelled word and ranks the suggested words for correction with the words that has high probability in the given list. Spell checker first scans the text and selects the words contained in it, after that it compares each word with a known list of correctly spelled words that are found in the given list or dictionary [3, 4, 5]. The work of developing spell checker began on computer techniques for automatic spelling correction and automatic text recognition and it has continued up to the present day. The reason for its continuity was that there is a limitation in their scope with the existing spelling correction techniques [6, 7].

The spell checker will provide two major functionalities. The first one is spelling error detection and the second one is error correction. Spelling error detection is to detect that the word was misspelled in the given language and whereas error correction is to suggest correct word form for the misspelled word to replace it [8, 9].

Kafi Noonoo is one of the language spoken in South Western part of Ethiopia by Kafecho people. It belongs to the Afro Asiatic language super family of the North-Omotic Southern Gonga Sub-group. The emphasis was not given to the language before 1987 by the previous governments. The language started as official working language of the Kafecho people since 1987 and offered as independent course both at primary and secondary school

level in Kafa zone [10, 11, 12, 13]. Kafi-Noonoo uses Latin Script for writing purpose and has 22 consonant phonemes and 5 vowel phonemes. Out of these, six of them are both long and short consonants. Among the 22 consonants, five of them are borrowed from English and Amharic languages. In Kafi-noonoo, a sentence is a set of words that contain subject and a predicate. Kafi Noonoo Language follows Subject -> Object ->Verb grammatical rule. For example: Kabadi doyee kexooch hammite (Kebede went to school). Kabadi->Subject, doyee kexoo-> object, hammite-> verb. More discussion about Kafi Noonoo word formation and morphological structure is presented in Chapter 2, Section 2.5.

1.2 Motivation

Natural language processing plays a great role in developing applications that learn, understand and produce human language. Computational approaches applied to the language research focused on automating the analysis of the linguistic structure of the language. NLP tools such as spell checker, grammar checker, POST, machine translation, text to speech synthesis, speech to text synthesis, sentiment analysis, dialogue system, information extraction, text summarization developed for the languages such as English [4], Turkish [14], Arabic [15, 16], Spanish [17] and Asian languages [3, 8, 18, 19]. Kafi Noonoo language lacks these NLP tools for processing natural language. Among these tools, Spell checker is one of the NLP application that detects misspelled words and generate suggestion list as a correction candidate. NLP applications such as machine translation system, text to speech synthesis system, information retrieval system, dialogue system, question-answering system require automated spell checker.

The development of spell checker application for this language is required for easy preparation of documents. Huge amount of electronic data is produced every day, since the language is used in schools, colleges, offices and other media. So, the development of Kafi Noonoo spell checker will contribute a lot in the areas of natural language processing application. So, the spell checker system have the ability of detecting errors and providing suggestion for correcting error words while processing a document

To the best of our knowledge, there is no research done in the area of spell checker for Kafi Noonoo language. Due to the absence of the spell checker in the language, texts are not analyzed and processed with error which means documents are kept containing misspelled word that leads to misinterpretation of word's meaning.

Thus, we are motivated to conduct a research on developing spell checker for Kafi Noonoo language which is morphological rich language.

1.3 Statement of the Problem

The advancement of the technology increases the need of the users for processing of information. Information (documents) are processed using computers in everyday life. During information processing humans may make errors in case of spelling words, due to not knowing how to spell the word or pressing adjacent keys or in other cases [3]. This leads to incomplete or miss interpretation of information, which motivates researchers to develop spell checker system to overcome the problem of spelling errors. For languages like English [4, 6, 7, 20, 21], Myanmar [8], Finnish [9], Swedish [22], Indian [3, 18, 19], Turkish [14], Spanish [17], Arabic [15, 16] and for the other western languages spell checker was developed. In case of Ethiopian languages like Afaan Oromo spell checker system was attempted by Gaddisa Olani [23]. Kafi Noonoo language is one of a morphologically complex language [12]. Only dictionary look up approach cannot enough to develop Kafi Noonoo spell checker system because of morphologically reach language. Storing all words (inflected, derivated and compounded) of the language in dictionary is difficult as it consuming space, time and probability of missing some words. Thus, morphology based approach for the development of Kafi Noonoo spell checker can be applied. Morphology based spell checker have ability of recognizing inflected, derivated and compounded word of the language. Due to the characteristics of the language, purpose and objective, the spell checker developed for one language cannot work well for the other language [3]. So, as far as the knowledge of the researcher is concerned, there is no spell checker for Kafi Noonoo language.

1.4 Objectives

General objective

The general objective of this research is to design and implement morphology based spell checker for Kafi Noonoo language.

Specific objectives

The specific objectives of this research to meet the general objective is as follows:

- Review literatures of spell checker developed for other languages for better understanding.
- Study and understand the behaviors of morphological structure and word formation of Kafi Noonoo language.
- Collecting Kafi Noonoo corpora and preparing training data.
- Studying spelling errors in Kafi Noonoo text documents.
- Assess different techniques and approaches employed so far in spell checking tasks and select the ones that are appropriate to the development of spell checker for Kafi Noonoo language.
- Design Kafi Noonoo spell checker system.
- Develop a prototype to check the performance of the designed system.
- Evaluate the performance of the developed system with different data sets.

1.5 Methods

To achieve the objectives of the research, we use a number of different techniques. Among the techniques, we are going to employ are discussed below.

Data Collection

Kafi Noonoo does not have publicly available annotated corpus text for spell checker and any other NLP applications. In this research, we collect electronic text form linguistic department of Bonga College of Teachers Education and other sources like media. We can communicate with the linguistics experts to understand the structure and characteristics of the language.

Literature Review

Different literatures that are relevant to our research are reviewed, which helps us to understand the current state of spell checker in different languages. We will review papers like journals, articles and previously done thesis that are related to our research area.

Prototype Development

Microsoft Visual Studio 2010 is used as a development environment for developing the prototype in windows 8 operating system. C# is used as programming language because it is object oriented, simple and flexible, automatic memory management and cross platform interoperability [24, 25].

Evaluation

Prototype will be tested and evaluated to measure the performance of the developed system.

1.6 Scope and Limitation

There are a number of error types that a spelling system can be modeled to detect and correct. Generally, human-generated misspellings can be distinguished into four groups: Typographic Errors (Non-word errors), Sequence Errors, Phonetic Errors (Cognitive errors) and Context Errors (Real word errors) [4, 8, 16]. Among them, our research mainly focuses on recognizing, detecting and correcting non-word errors. Detecting and correcting real-word error misspelling types are not considered in this work.

1.7 Application of Results

Spell checker is a useful program in many NLP applications as stated in Section 2. The application of this study will benefit anyone who want to process Kafi Noonoo documents. This research opens the way for the other researchers who want to develop other NLP applications like machine translation, speech recognition, information retrieval, text summarization using Kafi Noonoo Language and they will benefit from this research. The result of application can also be used

- In learning and teaching of Kafi Noonoo language
- Provides a list of most of the spelling as suggestions and one of which may be the correct one, which provides the writer to save time
- In developing other NLP applications such as machine translation, information retrieval, text summarization, text extraction, text to speech synthesis, grammar checker and so on.

1.8 Organization of the Thesis

The remaining part of this thesis is organized as follows. Chapter Two presents review of literature which includes: types of spelling errors, techniques and approaches of spell checking and the linguistic behaviors of Kafi Noonoo languages. Chapter Three presents related works done on spell checking system and related areas. The Fourth Chapter deals with the design of Kafi Noonoo spell checker system. The Fifth Chapter presents the experimental results of spell checking system. Finally, conclusion, contribution and future works are presented in Chapter Six.

Chapter Two: Literature Review

2.1 Introduction

This Chapter discusses approaches and techniques for detecting and correcting spelling error. Section 2.2 describes the approaches and requirement in implementation of spell checker such as dictionary look up approach and morphology based approach. In Section 2.3, we discussed types of spelling errors that can occur during processing a text. The two common error types: non-word error and real word error types are discussed. Non-word error such as typographic error type, cognitive error type and phonetic error type are explored in this section. Next Section 2.4 discusses techniques applied to spell checker for the purpose of detecting spelling error and correcting spelling. Lastly, Section 2.5 describes the linguistic characteristics of Kafi Noonoo language.

2.2 Approaches and Requirements in Implementation of Spell Checker

Different approaches are used to develop spell checker for different language based on the characteristics of the language. One approach that is used to develop spell checker system is dictionary look up approach [6, 26]. Dictionary lookup approach is looking every word in the dictionary. If the word exist in the dictionary, it is considered as correct word and if the word doesn't exist in the dictionary, it is flagged out as misspelled word. Dictionary have characteristics of storage space requirement and time consuming for searching. Large dictionary requires more space and may take longer time to search for specific word and too small dictionary consider many words as invalid words but the words are valid words in that specific language. The reason the valid words flagged out as error words is because the words are not exist in the dictionary. Especially inflected, derivate and compounded words are considered as misspelled words in this case. This problems are solved with the use of morphological analysis that recognizes the inflected, derivate and compounded words as valid words.

The other approach is morphology based approach [23, 27]. Morphology based spell has two main purpose in the development of effective spell checker. One it reduces the dictionary size in effective way because only root words of the language are going to be stored in the dictionary and the other words are recognized through the process of morphological analysis. The second purpose of morphology based spell checker is that it

generate new words that are not exist in the dictionary. Through morphological process new words that are derivated, inflected and compounded words are created based on the rules provided with suffixes to be attached to the root words.

In order to develop morphology based spell checker, one could have the knowledge of the language because knowledge of the language have great role in the development of spell checker and any other NLP applications such as morphological analyzer, part-of-speech tagger, text to speech synthesis, machine translation and so on. Knowledge of the language can obtained by reading books, thesis documents, other language related documents and discussion with linguistics regarding the characteristics of the language.

The other issue to be considered in developing spell checker is that word boundary delimiter. Some languages such as Japanese, Chinese, Thai, and Myanmar have no delimiter between words [28]. This is not the case for English, Amharic, Afaan Oromo and Kafi Noonoo languages which uses white space between words as delimiter.

With more complex morphology of the language to determine whether a word is correctly spelt or not will differ from language to language. Since simply increasing the size of the lexicon in not overcome the problem of error detection and correction. Only adding each and every individual words that means all inflected, derivate and compounded words to dictionary may also leads to further incorrect word input. It needs a morphological processing in detection and suggestion phase of spell checking. It is necessary to analyze each word at word level specifically and accurately to develop high level spell checker.

For designing and implementing spell checker, two main modules are considered. These are the lexicon (which contains a list of words) and algorithm (techniques) that implements lexicon for spell checking. Generally, these techniques will provide three main functions in the process of spell checking.

i. Error Detection: the process usually consists of checking to see if an input string is a valid word or not.
ii. Error Correction: the suggestions are provided by the suggestion prediction component that are closer to the misspelled word.
iii. Ranking of corrections: ordering of suggested corrections in decreasing order of their likelihood for being actual intended word.

2.3 Types of Spelling Errors

Spelling errors are common in processing a text in a document made by human. Kukich [6] classified error types into two classes: typographic error type and cognitive error types. The other famous studies in the area of spelling error types are researches done by Damerau [26] and Peterson [29].Error words can be grouped in to two error types such as non-word error and real-word error. Non-word error further classified as typographic error, cognitive error and phonetic error types [30].

2.3.1 Typographic Error Types

In typographic error the writer knows the correct spelling but simply makes an incorrect combination of characters. Typographic errors happen when correct spelling of the word is known but typed incorrectly. They are errors that are related to keyboard miss-punches (example: the-teh). According to Hema and Sunitha [30] typographic errors fall into one of the following categories:

i. Substitution Error: it happened due to single character is substituted by another letter.
ii. Deletion Error: it happened due to one character is deleted from the word.
iii. Insertion Error: this is due to the insertion of an extra character into a word.
iv. Transposition Error: happen when two characters in a word is transposed.
v. Run-on Error: it happened when a space is missing between two words
vi. Split word Error: It happened due to an extra space between two words.

A study by Damerau [26] shows that 80% of the misspelled words in English are non-word errors and are caused by single error misspellings such as one missing letter; one wrong letter; transposition of two adjacent letters.

Another study by Peterson [29] found 360 errors in computer copy of Webster's Seventh Collegiate Dictionary which had escaped detection since the files was keyboard. They also found 155 errors made by college students when they retyped the word division list of the U.S. Government Printing Office. From the experiment done by Peterson [29] shows that transposition error is 4(2.6%), insertion error is 29 (18.7%), deletion error is 49 (31.6%), substitution error is 62 (40.0%) and total error of 144 (92.9%) using government printing office data source and in case of using Webster's Seventh data source transposition error 47(13.1%), insertion error 73(20.3%), deletion error 124(34.4%), substitution error is 97(94.7%) and total error is 341(94.7%). From these result, they concluded that

substitution error is the most common type of error to occur. Based on Kukich [6] study, 58% of all substitution errors involved adjacent typewriter keys. A study by Damerau [26] shows that the four class of error occurs due to misreading, hitting a key twice and letting the eye move faster than the hand.

2.3.2 Cognitive Error Types

Cognitive errors are caused by the writers' misconceptions (e.g. recieve-receive). The source of cognitive error is presumed to be a lack of knowledge on the part of the writer or typist how to spell the word [6]. Generally, the writer is unaware or does not know the actual spelling of the word and the writer tries to write it with probable spelling guessing it with similar pronunciation of the letter that sounds somewhat similar to the actual word. The similar sounding or homophone letters are often substituted in this type of error.

2.3.3 Phonetic Error Types

Phonetic Errors are a result of substituting a phonetically equivalent sequence of characters (e.g. Separate-Separete). According to Kukich [6], phonetic errors are special class of cognitive errors in which the writer substitute a phonetically correct but orthographically incorrect sequence of letters for the intended word. It frequently impossible to assign a single category to a given error. For example, recieve necessarily a cognitive error or typographic transposition error. Fortunately, it is often necessary to categorize errors in order to develop a useful spelling correction because many correction techniques handle typographic and cognitive misspelling well. Kukich [6] reported in their study that 38% of the spelling errors are generated by the subjects were incorrect despite phonetically plausible.

2.4 Techniques of Spell Checker

There are two common activities that any spell checker can perform. These are spelling error detection that flags out misspelled word in a text and spelling error correction provides suggestion for misspelled word.

2.4.1 Spelling Error Detection

A word is a valid word if it has meaning in specific language else it is an error word in that language. There are two techniques for error detection. These are N-gram analysis and dictionary lookup. The error detection process usually consists of checking to see if an input word is a valid index or dictionary word. Efficient techniques have been devised for

detecting such types of errors. Spellcheckers rely mostly on dictionary lookup and text recognition systems rely on n-gram techniques [31, 32, 33].

N Gram Analysis

N-grams are n-letter sub sequences of words or strings where n usually is one, two or three. One letter n-gram is called unigram or monogram; two letter n-grams are referred to as bi-grams and three letter n-grams as trigrams. N-gram analysis is a method to find incorrectly spelled words in text and used for non-word errors. Instead of comparing each entire word in a text to a dictionary, just n-grams are controlled. The n-gram algorithm was developed as one of the benefits is that it allows strings that have differing prefixes to match and the algorithm is also tolerant of misspellings. Each string that is involved in the comparison process is split up into sets of adjacent n-grams. A check is done by using an n-dimensional matrix where real n-gram frequencies are stored. In general, n-gram detection technique work by examining each n-gram is an input string and looking it up in a precompiled table of n-gram statistics to determine either its existence or its frequency of words or strings that are found to contain nonexistence or highly infrequent n-grams are identified as either misspellings.

The major advantage of n-gram algorithm is that it requires no knowledge of the language that is used with, so it is language independent or it is a neutral string matching algorithm. Using n-grams to calculate, for example the similarity between two strings is achieved by discovering the number of unique n-grams that they share and then calculating a similarity coefficient, which is the number of the n-grams in common divided by the total number of n-grams in the two words [31, 32, 33].

Dictionary Lookup

Dictionary lookup technique is strait forward. It checks every word of input text for its presence in dictionary. If that word present in the dictionary, then it was accepted as correct word, otherwise it put into the list of error words. The most common technique for gaining fast access to a dictionary is the use of a Hash Table. To lookup an input string, one simply computes its hash addresses 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. Hash tables main advantage is their random-access nature that eliminated the large number of comparisons needed to search the

dictionary. The main disadvantage of hash table is that the need to develop a clever hash function that avoids collisions [31, 32, 33].

Hashing

Hashing is a good technique for implementation in keyed tables. It is a technique used for storing and retrieving information as fast as possible. It also used in performing optimal searches and retrievals because it increases speed, betters ease of transfer, get better retrieval, optimizes searching of data, reduces overhead. One of the major advantage of hashing is to optimize disk access and packing density. Hashing is to reduce disk space and access time by inserting and retrieving a record form the table in only on seeks [34, 35]. Hashing based searching algorithm (HBSA) indicates each array elements with a particular index value and index value with the function in the form of M mod N where M is the index value and N is the user defined number. HBSA maps the hash value with index value and then define the particular address to the elements. Therefore, this increases the searching speed of the algorithm, the complexity of the Hash base searching technique is log(N).

Binary Searching Algorithm

The strategy of Binary Search Algorithm (BSA) is to check the middle of the elements in the list. If the key value is less than the middle value then it searches to the left side till 0. This process continues till N, if the key value is greater than the middle value. The complexity of BSA is log (N) as stated by Rautaray and Kumar [35].

2.4.2 Spelling Error Correction

Two types of spelling error correction: interactive and automatic. In interactive, the spellchecker can suggest more than on correction for each misspelled word and the user decide to select for replacement. In case of automatic correction, the spellchecker has to decide on the one best correction and the error is automatically replaced with it [3].

Spelling correction is a process of detecting and providing suggestions for misspelled words in a text. As Mishra and Kaur [33] stated, in computing, spell checker is an application program that flags words in a document that may not be spelled correctly. Two types of spelling error correction: isolated word error correction and context-based error correction. In isolated word error correction, misspelled word can be analyzed in isolation without giving consideration to its context. Corrections are based on only misspelled

words itself. In case of context-based error correction, correction can be done with consideration of the context of the error word Context based technique mainly used for correcting real word errors. Real word errors are words which have correct meaning in the dictionary but contextual error in a given text. Most of the early research works in the area of spell checking was more focused on isolated word error correction.

Edit Distance Technique

According to Masek and Pateson [36] edit distance between two characters is the minimum cost of a sequence of editing operations which transforms one string into the other. Using edit distance algorithm, it is possible to perform deleting, inserting and replacing one symbol a time with possibly different costs for each of these operations. There are many distance measures [37, 38]. Those are: Hamming distance, Levenshtein distance, Damerau distance and Jaro-Winkler distance. Hamming distance is a metric between two strings of equal length. Levenshtein distance is a mathematical measure for calculating the difference between two strings. Levenshtein assumes that the transmission of binary information usually considers a channel model in which 0→1 and 1→0 substitutions are admitted. Equation (1) shows that how Levenshtein's distance metric is recursively defined in which distance measures are insertion, deletion and substitution done. Damereu distance is an extension of the Levenshtein distance by adding new operation of transition. Damereu distance [36] is a well-known metrics for making spelling corrections through string-to-string comparison. Jaro-Winkler distance has been designed and is best suited for short strings. The score is 0 with no similarity and 1 with an exact match as shown in the Equation (2).

$$dist_{a,b}(i,j) = \begin{cases} 0 & ,i = j = 0 \\ i & ,j = 0 \ and \ i > 0 \\ j & ,i = 0 \ and \ j > 0 \\ min \begin{cases} dist_{a,b}(i-1,j) + 1 \\ dist_{a,b}(i,j-1) + 1 \\ dist_{a,b}(i-1,j-1) + 1_{(ai \neq b)} \end{cases} & erwise \end{cases} \quad \ldots (1)$$

Where $dist_{a,b}(i,j)$ stand for distance between a and b with i, j matrix and min for minimum.

$$dj = \begin{cases} 0 & if \ m = 0 \\ \frac{1}{3}\left(\frac{m}{|s1|} + \frac{m}{|s2|} + m - \frac{t}{m}\right) & otherwise \end{cases} \quad \ldots\ldots\ldots\ldots\ldots (2)$$

Where dj stand for jar distance, s1 is for string1 and s2 is for string2, m is number of matching characters and t is half the number of transposition.

N-Gram Technique

N-Gram technique can be used in two ways, either without a dictionary or together with dictionary. N-gram can in the form of trigram, bigram and unigram have been used in variety of ways in text recognition and spelling correction techniques. Without a dictionary, n-grams are employed to find in which positions the misspelled word error occurs. If there is a unique way to change the misspelled word so that it contains only valid n-grams which is taken as the correction. The performance of this technique is limited. 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 [36, 32, 33].

Rule-Based Technique

According to Kukich [6], rule-based technique are algorithms that attempts to represent knowledge of common spelling error patterns in the form of rules for transforming misspellings into valid words. The candidate generation process consists of applying all applicable rules to a misspelled word and retaining every valid dictionary word that results and ranking is done by assigning a numerical score to each candidate based on a predefined estimate of the probability of having made the particular error that the invoked rule corrected.

Phonetic Matching Technique

Phonetic matching is a technique used to obtain words based on sound they produce. Different algorithms are proposed to solve the problem of phonetic matching. Soundex, metaphone, DMetaphone, phonex, caverphone and NYSIIS algorithms are the common phonetic matching techniques [39]. Soundex algorithm is the first algorithm developed for phonetic matching. It is used to assign indexes to words based on the sound they produce, as pronounced in English. Soundex code for a name consists of a letter followed by three numerical digits. The letter is the first letter of the name and the digits encode the remaining consonants. Similar sounding consonants can share the same digits. For example, the letters a, e, h, i, o, u, w, y are encoded to number 0 and all zeros are removed from the string in Soundex algorithm. Letter b, f, p, v are each encoded as number 1 where

c, g, j, k, q, s, x, z are each encoded as number 2 and d, t encode as number 3 and l encoded as number 4, m, n encoded as number 5 and lastly r encoded as number 6. Using Soundex algorithm, in case of English names, both 'Robert' and 'Rupert' return the same string code value 'R163' while 'Rubin yields 'R150'. Soundex algorithm have the following few disadvantages such as its dependency on the first letter, failure of detection of silent consonants. So, Soundex algorithm is only used in applications with high false positives and false negatives can be tolerated. Metaphone algorithm considers set of letters as an alternative to letter by letter encoding to identify the phonetic variations and inconsistencies in words. This algorithm initially performs transformations using diphthongs such as changing MB to B if at the end of the word, SCH to K and drop all the vowels in the encoded word. It shows that the phonetic sound of vowels combined with the consonants is considered instead of individual consonant or vowel sounds. Metaphone code length varies from 4-letter code to 12-letter code as this improves precision [39]. DMetaphone (Double Metaphone) algorithm is a sound indexing algorithm that groups letters by different pronunciations. The double methaphone also produces character code like methaphone. The main difference between this algorithms is that double metaphone produces secondary key along with a primary encoded word to identify the most common native pronunciation. The algorithm retains only the first vowel sound to same character 'A' while all other vowel sounds are dropped. Later few other transformations are done on the remaining letters based on the letters present in the successor index and predecessor index.

The Caverphone algorithm has several sequential transformations based on the characteristics of the word. Initially, all the letters are converted to lowercase and then it removes 'e' at the end of the word. The encoding uses the numbers '2' and '3' to encode few phonetic sounds. Then after these numbers are again converted into alphabetic phonetic encoding. Unlike Soundex, the vowel sound is retained either as 'A' if at the beginning, or as '3' elsewhere. The obtained word is followed by few other transformations such as 'y' is converted to 'Y3' upon appearance at the start of the word, s, t, p, k, f, m, n are converted into uppercase if present as group of consecutive letters, 'r' present at the end of the word is converted to 3. Later, '2' are removed and '3' is converted to 'A' at the end of the word. After all transformations, the encoded word is truncated to ten letters and is appended by '1', if necessary. NYSIIS (New York State Identification and Intelligence System) phonetic algorithm is a computational algorithm which is

designed for American names. The algorithm transforms beginning and ending of words and then followed by set of rules to encode the remaining letters. The vowels are retained by converting all of them to 'A' and there are some special cases where 'AY' is converted to 'Y', removing 'A' if they are existing at the end of encoded. Phonex is an approach where words are pre-processed before encoding. Encoding transformations are done at the beginning, ending and at the middle of the word during pre-processing using phonex technique [39].

Neural Network Technique

According Randhawa and Saroa [31], neural network method performs well on small dictionaries. Back propagation algorithm is one of the most common used in neural network. It consists of 3 layers: input, hidden and output layer. In this input information is represented by on-off pattern A=1 indicates that node is turned on and A=0 indicates node is turned off. In their spell-checking applications, a misspelling represented as binary n-gram vector may be taken as input and output pattern might be vector of m elements means number of words in the lexicon.

Noisy Channel Model technique

In probabilistic system, it is possible to find argmax P(w/s). Applying Bayes' rule and dropping the constant denominator produce unnormalized posterior as given in Eq. (3) [38].

$$\textbf{argmax}_w \, P(s \mid w) * P(w) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

The general architecture of noisy channel model is look like as shown in Figure 2.1.

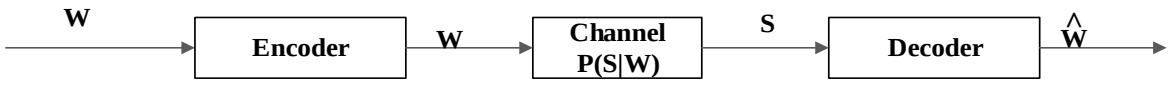

Figure 2.1: *General Architecture of Noisy Channel Model*

Brill and Moore [38] proposed a noisy channel model for spelling correction, with two components: the source model P(w) and the channel model P(s|w). Their model assumes that natural language text is generated in the following way: first the user chooses a word to output, according to probability distribution P(w). Then the user attempts to output the

word w, but the noisy channel induces the person to output string instead, according to the distribution P(s|w). For example, under typical circumstances we would expect P (the|the) to be very high, P (teh|the) to be relatively high and P (hippopotamus/the) to be extremely low. They demonstrated their noisy channel model without a language model and they get the result of 52% reduction in spelling corrections error rate compared to Damerau-Levenshtein distance technique [38]. With a language model, their model gives a 74% reduction in error. Words with highest probability is considered as correct match of the misspelled word.

2.5 Kafi Noonoo Language

2.5.1 Writing System

Kafi Noonoo language uses Latin script for writing system. The language has 22 consonant and 5 vowels. Form these, 6 consonants are both short and long consonants and five consonants are borrowed from English and Amharic languages. Totally there are 32 letters including borrowed letters [10, 12]. Figure 2.2 shows Kofi Noonoo letters in capital form.

A, B, C, D, E, F, G, I, J, K, L, M, N, O, P, Q, R, S, T, U, W, X, Y, Z, CH, PH, SH

Figure 2.2: *Kafi Noono Letters in Capital Letter Form*

Table 2.1 describes that example of Kafi Noonoo consonants, single consonants, doubled consonants and borrowed letters with an example.

Table 2.1: *Consonants and Borrowed Letter in Kafi-noonoo language*

Consonants	Single Consonat	Example	Doubled Consonant	Example	Borrowed Letter
b, c, f, g, h, j, l, p, r, s, t, v, w, x, y, z, ch, sh, ts, zh	b	ibo	bb	yubbo	v, z, ny, ts, zh
	d	wodo	dd	goddo	
	g	gogo	gg	daggo	
	m	qomo	mm	nummo	
	n	dano	nn	funnoo	
	ph	gapho	pph	kuppho	

In Kafi Noonoo language, there are 6 consonants that are not doubled in any case in the writing system. These letters are called naturally short (weak consonants) in the language. We cannot double these consonants in any case in the writing system [40]. Such consonants are h, r, w, f, y, sh as shown in Table 2.2 with example.

Table 2.2: *Naturally Short Consonants in Kafi Noonoo Language*

Naturally Short Consonants (Halla koxee Bireena'o)	Example (Shaahiyoo)
f	ee<u>f</u>o
h	aa<u>h</u>o
r	dii<u>r</u>o
w	kawo
y	ke<u>y</u>o
sh	a<u>sh</u>o

The language has also five short and long vowels. In order to show long sound, vowels are doubled and in case of short sound vowels are written as a single [40] as shown in Table 2.3 with examples.

Table 2.3: *Sample Example of the Five Short and Long Vowels*

Short Vowels	Example	Long Vowels	example
a	baro	aa	baaro
e	deco	ee	deeco
i	kimo	ii	kiino
o	gommo	oo	goomo
u	busho	uu	buucho

In Kafi Noonoo Language, long consonants and vowels are obtained by doubling the corresponding short consonants and vowels respectively. These creates difference meaning in case of both cases. For example, the word *deco* means date where as *deeco* means axe. In other cases, two the same words give different meaning based on the way they read [10]. For example, take the following words:

bago (sheep) *koroo* (saddle) *kemo* (sell)

bago (hitting) koroo (burning by fire) *kemo* (buy)

The letter "ch" in Kafi Noonoo have special behavior, if there no is vowels after the letter "ch" it looks like weak consonant. For example, *yaach* (tomorrow), *baach* (only), *booch* (for them) in these given words no vowels after the letter "ch" and it reads as weak consonant. The other consonant "p" cannot be weak in any case which is strong consonant by nature. Consonants like "j, l, p, r and ch "cannot come at first to form a word in Kafi Noonoo language. That means there is no words that begins with these consonants in the language. In other case, if there is no vowel after consonants "m, n" then the other consonants that come after "m, n" cannot be doubled. Because consonants that come after them always reads as strong sound [12, 40]. For example, *damba* (above), *shimbo* (title), *genjoo* (long), *shendoo* (hold) in these given example we cannot double the consonants "b, j, d" because they come after the consonants "m, n".

2.5.2 Kafi Noonoo Morphology

In Linguistics, morphology is a branch of linguistics that studies about the formation of words in a given language. Morphological operations provide two basic functions in word processing. The first one is that the creation of new word (new lexicon). For example in English, 'buyer' is created from a word 'buy' by morphological operation. The second one

is that spelling out the appropriate form of a lexeme in a particular syntactic context. For example in English, the word 'bottle factory' is created from two different words as a single word to represent a single entity instead of 'a factory for production of bottles'. Thus, morphology provides a means for extending the set of words of a language in a systematic way [12, 41, 42].

Types of Morphemes

A morpheme, the morphological building blocks of words, is the minimal linguistic units with a lexical or grammatical meaning. One or more morphemes combined to form a word that has a meaning. Like English and other languages, Kafi Noonoo morphemes are classified into two parts: free morphemes/lexical morphemes and bound morphemes/affix'. Free morphemes are morphemes that can appear as word by itself. These morphemes cannot need to be attached to other words to give meaning. In other way, bound morphemes are morphemes that cannot appear as a word by itself. These morphemes need to be attached to other morphemes to give meaning. For example, when we consider the word *qaminittino* (shortness), *qamino* (short) is free morpheme and *–ittino* is bound morpheme. In this given example the hyphen (-), preceding this morpheme, indicates that it requires another morpheme to appear before it in the word [12].

Free Morphemes

Free morphemes in Kafi Noonoo are grouped into two classes. These are lexical morphemes and functional morphemes'. Lexical morphemes are free morphemes that can standalone by their own having a meaning. For example, take the word *asho* (human) its meaning never changed in a sentence if we use as subject or object. In other way, functional morphemes are words that have little lexical meaning or have ambiguous meaning, but instead serve to express grammatical relationships with other words within a sentence. Sample example of functional morphemes for Kafi Noonoo language are *waan, bi, bo, no, koni.* They are closed class words. Functional words are lexically unproductive and are generally invariable inform. For example, *waan* **(**against), *gubb* (behind), *hini* (this) are some of functional words in Kafi Noonoo language [12]. A word changes meaning in different positions.

Bound Morphemes

Bound morphemes are grouped into two: bound root and affix. Bound roots do not occur in isolation and they require meaning only in combination with other morphemes. They play great role to the meaning of the word. Words like *daambe, daambu, daambot, daamisho, daammiyo, daammiye, daamaache, daammimo* and *daamichi* have bound root *daam-* in Kafi Noonoo language. Affixes are bound morphemes that are attached to the other morphemes. Affixes may appear at the beginning of a word, at the middle of a word, at the end of a word or both at the beginning and at the end of a word. So, based on where they appear, Kafi Noonoo affixes are classified into three class: prefix, infix and suffix. Prefixes are affixes that are attached to word at the begging of a word (occur at the left side of a word). For example, in the word *ogoogo* (biggest) the affix *og-* appear at the beginning of a word *oogo* (big). But, in Kafi Noonoo, there is no more known prefixes, only few prefixes exists as stated by Simegn Tekle [12]. Infixes are affixes that are inserted inside other word (occur at the middle of a word). For example, in a word *wochaachinnoo* (again and again) the affix *-chaa-* appear at the middle of the word *wochinnoo* (again). In Kafi Noonoo, infixes occur rarely. Suffixes are affixes that are attached at the end of a word (appear at right side of a word). For example, in a word *tawaan* **(**towards me) the affix *–waan* appear at the end of the word *ta***.** In Kafi Noonoo, there a number of suffixes exist at the end of the root word which means words in the language are rich suffix. Affixes are also classified into two classes based on grammatical context and word class they change. These are derivational morphemes and inflectional morphemes.

Derivational morphemes

Derivational morphemes are added to morphemes to create new separate words. They change the both the meaning and class of the word. For example, *kooroo* (to write) is verb word class, so we create *koorecho* (writer) agent noun for male by adding affix *-echo* and *koorechi* (writer) agent noun for female by adding affix *–echi.* The most known derivational morphemes in Kafi Noonoo are *-echo, -echi,-ittino, -iyo, -inno, -cho, -innee, -ero* [12].

Inflectional Morphemes

Unlike, derivational morphemes, inflectional morphemes do not create separate new words and do not change word class. They modify the word in which they occur in order to

indicate grammatical property of like plurality, tense, time, gender, direction and so on. Examples of word inflection is given below in Table 2.5 taken from [40].

Table 2.4: *Kafi Noonoo Word Inflection*

Word	Meaning	Affix	Inflected word	Meaning	Purpose of affix
bechoo	flower	-eena'o	becheena'o	flowers	plurality
shemmo	read	-ete	shemmete	he read	tense
Boongi	Bonga	-waan	Boongiwaan	to Bonga	direction
bushoo	boy	-ee	Bushee	girl	gender
hanaach	today	-ich	Hanaachich	for today	time

In the above given example in Table 2.5, adding affix to a root word cannot change word class rather it creates the word for different purpose. Some of the common inflectional morphemes in Kafi Noonoo language are *-e, -ee, -i, -aa, -n, -na, ich,-che, -echee, -shi, -ete, -naa, -ite, -cha-, -shi-, ich, -echee-, -eena'o, -eena'one, -eena'oche, -waan, -ichomon, -chon,-iyee*. For example, consider the word *mixo* (wood), by adding suffix *-i* to the word *mixo* will produce *mixi* (small pieces of wood). Generally, most of the Kafi Noonoo inflectional morphemes and derivational morphemes are suffixes. The language is more productive in suffixes than prefixes and infixes as described by Simegnih Tekle [12].

2.5.3 Kafi Noonoo Word Formation

Kafi Noonoo words are formed from the roots by means of different morphological operations like affixation, compounding and reduplication. Words in any language have not the same task. Some words express an action (verbs), others express a thing (noun) and others join one word to another to create new word which express something. So, when we want to create a sentence, we use these different classes of words which have different jobs.

Based on [12] Kafi Noonoo has five mainly known word classes such as noun, adjective, adverb, verb and preposition. Among these word classes, some of them are classified into other sub classes.

Kafi Noonoo Word Inflection

Inflection is a morphological marking of properties on a lexeme resulting in a number of forms for that lexeme, a set of grammatical words. Word inflection cannot change the class of the word, but it creates various forms of the same word. Inflection morphemes generally, do not change basic meaning of the words and part of speech of the words [1, 12, 41]. For instance, in Kafi Noonoo, *giisho, giishi, giisheche gishiisho, gishiishe* are all adjectives.

Noun Inflection: Noun in Kafi Noonoo like in other languages used to represents or identify name of place, person, things or objects, ideas. For example, name of person: *Fiqiri* (Fikru), *Aaxxoogi* (Atoge), *Mangashi* (Mengesha), name of place: *Addisaabi* (Addis Ababa), *Boongi* (Bonga), *Bixxi* (Bitta) and name of object: *xaqqo* (stone), *aaco* (water), qoreddoo (cloth). Noun in Kafi Noonoo grouped into four sub classes such as common noun, abstract noun, collective noun and proper noun. Common noun is a noun that is not the name of the one particular person or thing. It is a word used to name general items rather than specific ones. Common nouns are everywhere and we can use them all the time. Examples of common nouns in Kafi Noonoo language: *mixo* (wood), *kexo* (house),*aaco* (water), *doyee kexo* (school), *haasho* (fish), *xaqqo*(stone). Abstract noun is a type of noun that denote an experiences, ideas, qualities, feeling and state rather than a concrete object. We cannot see or touch abstract noun because these nouns cannot exist as material objects. For example, *shalligoo* (idea), *emiroo* (happiness), *shuno* (love), *ariyoo* (knowledge), *shixo* (hate). Collective noun is a group of noun that denotes a group of people, things or animals. Noun in the collective class is can be used in either the singular or plural form depending on the context of the sentence. For example, *maccoo* (people), *maachoo* (female), *indoo* (mothers) are some examples for collective nouns in Kafi Noonoo language. Proper Noun is a name used for individual person, place or organization. A proper name is specific name for particular person, place or thing. In Kafi Noonoo, like English, the first letter of the proper noun is capitalized. For example, *Boongi* (Bonga) name of place, *Qocechi* (Kochech) name of person, *Yuuniversitii* (University) name of organization in Kafi Noonoo language. Kafi Noonoo nouns are inflected in case of gender, number and case. Inflectional suffixes are added to original root noun word to form various form of noun of the same class.

Case: Kafi Noonoo nouns are inflected to indicate instrumental, objective and nominative case. Suffixes like *-n, -ch, -choche, -chommon* are case indicate for nominative case in

Kafi Noonoo language. For example, *Qocechi Aaxxoogi**ch** aaco immetan* (Kocheche gave water to Atoge).

Gender: Kafi Noonoo has two grammatical genders (Feminine and Masculine) like other Afro-Asiatic languages. In Kafi Noonoo nouns end with *-ee* or *-i* indicates feminine gender and other end with *-oo* or *-o* indicates masculine gender. Table 2.5 shows noun inflection for the case of gender.

Table 2.5: *Noun Inflection for Gender*

FEMININE GENDER		**MEANING**	**MASCULINE GENDER**	**MEANING**
Bushee		girl	Bushoo	Boy
Biceree		mule (F)	Biceroo	Mule(M)
Maxi		bee (F)	Maxo	Bee(M)
Caayi		porcupine (F)	Caayo	Porcupine(M)

In Kafi Noonoo, there is noun inflection for inanimate things for augmentation and diminutiveness. For instance, the noun *mixi* (small pieces of wood) is the inflected word of noun *mixo* (wood) and *aaci* (little water) is the inflected word of noun *aaco* (water). These kinds of inflections are done by adding suffix *-i* to the root word.

Number: Like English plural marker - (e)/s, there is the common plural marker suffix *-eena'o* and *-ina'o* in Kafi Noonoo language. This suffix is added for the inflection of regular noun for marking of plural noun. For example, for noun inflection for the case of number plural marking.

Noun	**Meaning**	**affix**	**Plural noun**	**Meaning**
Asho	human	-eena'o	asheena'o	people
Bechoo	flower	-eena'o	becheena'o	flowers
Kexo	house	-ina'o	kexina'o	houses
Cooxo	animal	-ina'o	cooxina'o	animals

In Kafi Noonoo, there is also irregular number inflection of noun for marking plural noun. For instance, *indee* (mother) is singular noun, *indoo* (mothers) is plural noun.

Verb Inflection: Kafi Noonoo, like other Afro-asiatic languages, follow the grammatical rule of subject-object-verb (SOV) agreement for formulating a sentence. Verb is a word that express the action done by the subject on the object. For example, *mamo* (eat), *uyo* (drink), (*hamo)* 'go', *koto* (sit), *woco* (run), *deewo* (bring) are some verbs in kafi Noonoo which express an action on the subject. Verbs can be inflected for many cases in Kafi Noonoo language. The inflection of verb can occur for gender, tense, number and other cases. For example:

- *Ababi doyee kexooch hammite* (Abebe went to school).
- *Bushiisheena'o doyee kexooch hammiteete* (children went to school).
- *Aaxxoogi doyee kexooch hammetan* (Axoge went to school)
- *Bushiisho doyee kexooch kaameeloona hammibeeteete* (children went to school by bus).

In the given example, in line 1, verb *hammite* is inflected for the case of gender which is masculine and is paste tense for the case of tense. In given example, line 2, verb *hammiteete* is inflected for the case of number which is plural form indicator and tense is past tense. In given example, line 3, verb *hammetan* is inflected for the case of gender which is feminine and paste tense for the tense indicator. In given example, last line, the verb *hammibeeteete* is inflected for number which is plural form indicator and present tense for tense indicator. *Hammite, hammiteete, hammetan* and *hammibeeteete* inflected words which are generated from a root word *hamo* 'went'. The inflected words for a verb *daamo* (take) is depicted in Annex B.

In other ways, in Kafi Noonoo, verbs have different properties based on their sound of consonants. Some naturally weak sound consonants can be converted to the same behavior strong sound consonant in case of verb inflection. For example, 'y' changes into 'j' as in *gaayo => gaajjiyee*, *bayo => bajjiye*, *yooyo => yoojjiye* and 'y' changes to 'ch' as in *goyo =>gochiye*, *koyo=>kochiye*.

Adverb: An adverb is a word that modifies or quantifies a verbs, adjectives and other adverbs. Like other Afro-Asiatic languages, Kafi Noonoo adverbs modifies a verb that express location, time, and manner. Some examples of Adverbs in Kafi Noonoo are: *hanaach* (today), *afaafino* (quickly), *aabich* (where), *aaf* (front), *bullaabo* (always). There are many suffixes attached to these adverbs to generate other adverbs. Some of the suffixes

that are added to adverbs for inflection purpose are: *-a, -aa, -ee, -na, -ch, -che, -icho, -ich, -ichona, -ichi, -ichish, -ichishona*

Preposition: Prepositions, in Kafi Noonoo language, are words that come after a noun or pronoun to express location, source, destination and relation to another element. Some prepositions used in Kafi Noonoo are: *aaf, gubb, dech, damba, toommo* and suffixes like *-aa, -o, -ooche, -ich, -ichee, -ch, -iwaan, -oona, -eexi* are some of the suffixes attached to the preposition to inflect words.

Kafi Noonoo Word Derivation

Unlike inflectional morpheme, derivational morphemes are morphemes that are the combination of a word root with a grammatical morpheme, usually results in generating new meaning words with different part of speech.

Noun Derivation: In Kafi Noonoo language, noun can be derivate to another noun (abstract noun) by adding suffix like **-ittino** as in the given example. The following example shows abstract nouns that are derived from base nouns.

Base noun	**Meaning**	**Astract Noun**	**Meaning**
bushoo	child	bushittino	childhood
taatoo	king	taatittino	kingdom
indee	mother	indittino	motherhood
nihoo	father	nihittino	fatherhood
asho	human	ashittino	humanhood

Adjective to Noun Derivation

Adjectives are words that precede the noun or pronoun to describe, quantify, or identify the word they precede. For example: *cello* (red), *oogoo* (big), *nooco* (sweet), *beello* (clever), *gamino* (wide) *qamino* (short) are some examples of adjectives in Kafi Noonoo language. By addition of suffix like *-ittino*, adjectives can be converted to noun. For example:

Adjective (Ashigo)	Meaning	Noun	Meaning
ceello	red	ceellittino	redness
oogoo	big	oogittino	bigness
nooco	sweet	noocittino	sweetness
beello	clever	beellittino	cleverness
gamino	wide	gaminittino	wideness
qamino	short	qaminittino	shortness

Verb to Noun Derivation

In kafi Noonoo verbs are changed to noun by adding suffixes like *-echo, -echi* to verbs. The suffix *-echo* converts the verb to noun that indicate masculine gender and in contrast, the suffix *-echi* converts the verb to noun that indicate feminine gender. For example:

Verb	Meaning	noun (M)	Noun (F)	Meaning
kooroo	to write	koorecho	koorechi	writer
woco	to run	woccecho	woccechi	runner
goyo	to pluagh	gochecho	gochechi	farmer
shuuno	to work	shuunecho	shuunechi	worker
duubo	to sing	duubbecho	duubbechi	singer
doyo	to learn	doyecho	doyechi	learner
kaayo	to play	kaachecho	kaachechi	player

Compounding of words

Compounding is a process of creating new words by grouping existing words. The individual words have different dictionary meaning and may have the same or different part of speech. After compounding, these different words with different meaning will represent single object having single meaning. In Kafi Noonoo, compound words are written by using hyphen, or separating by space or as a single word without spacing the two word. The following example is based on [12, 43] that shows how different word are combined to form new word which is compound.

Compounding Nouns

In Kafi Noonoo, two nouns are combine to form single noun that represent a single object and noun compounding is a productive process in the language. Noun- noun compound is very productive compounding in Kafi Noonoo language.For example:

Noun	Meaning	Noun	Meaning	Compound Noun	Meaning
maahoo	lion	achoo	body	maahee-achoo	very expensive
halloo	nature	gaanno	resource	hallee-gaanno	natural rsource
haroo	morning	xumo	night	haree-xumo	bad condition
kosho	food	nihoo	father	kochi-nihoo	step father

Adjective -Noun compounding

Nouns can be formed by combining adjectives and nouns that derive new word of class type noun. For instance:

Adjective	Meaning	Noun	Meaning	Compound Noun	Meaning
aa'o	black	madoo	board	aa'i madoo	blackboard
oogo	big	shambetoo	weekend	oogi shambetoo	Sunday

Verb-Noun Compounding

Verbs can be combined with nouns and produce single compound nouns having a single meaning. For example:

Verb	Meaning	Noun	Meaning	Compound Noun	Meaning
ariyoo	to know	budo	context	ariyee budo	source of knowledge
gutino	to kneel	aafo	eye	gutine aafo	ball of knee

Adverb Compounding

.Adverb compounding can be formed by combining adjective with noun and verb with adjectives. Adverb compounding is not productive alike noun and verb in Kafi Noonoo language.

Adjective-Noun Compounding

In Kafi Noonoo, Nouns can be combined with adjectives and produce adverb. For example:

Adjective	Meaning	Noun	Meaning	Compound Adverb	Meaning
bulla	all	aabo	sun	bullaabo	always

Verb-Adjective Compounding

In Kafi Noonoo, verbs can be combined with adjectives and produce adverb. For example:

Verb	Meaning	Adjective	Meaning	Compound Adverb	Meaning
beemo	live	genjoo	long	beemi genjoo	long live
qitoo	to die	wodoo	healthy	qita-woda	absolutely

2.6 Summary

In this chapter, some of the previous studies are reviewed in details. The approaches and requirements like dictionary based approaches and N-gram approaches for developing spell checker are addressed.

The other issue discussed in this chapter was types of spelling errors. Two main class of spelling error types. These are non-word error type and real-word error type. In case of non-word error type, the misspelled word has no meaning in that specific language and in case of real word error type, the misspelled word has meaning but syntactically and semantically incorrect in that specific language. The most common non-word spelling error that can occur during typing are typographic error types, cognitive types and phonetic error types.

Techniques that can be applied on spell checker to develop spell checker for detecting and correcting errors that are discussed in this chapter are N-gram analysis and dictionary look up for error detecting an error, edit distance, N-gram, rule-based and noisy channel model techniques for correcting errors.

Finally, linguistic structure of Kafi Noonoo language like morphology and word formation (inflection, derivation and compounding) are described in detail.

Chapter Three: Related Works

3.1 Introduction

In this chapter, we discuss some related work done in the area of spell checking that incorporate the techniques and approaches discussed in Chapter Two. We review the works that are important for better under sting of the concepts in the area of spell checker previously done. Section 3.2 discusses about different spell checkers developed for English language with the approaches and techniques used. Section 3.3 addresses the spell checkers developed for Arabic language. The next Section 3.4 spell checkers for Indian languages are discussed in details. Spell checker for Malay language is discussed in Section 3.5 in details. Section 3.6 presents spell checker developed for Ingala Language and finally in Section 3.7 the spell checker for Afaan Oromo is discussed. Various studies related to the current research work are reviewed with the approaches and techniques used in spell checker development for detecting and correcting error words. Finally, shows the knowledge gaps between the reviewed work and this work is identified.

3.2 Spell checker for English Language

Bhaire *et al.* [4] developed spell checker application that autosuggest the misspelled words for window-based application. They develop such application by using data structure tree for storing dictionary and edit distance algorithm for spelling correction which provides the advantage of autosuggest. The spell checker includes the features like multi word suggestion to enhance the performance. But, the system does not offer any alternative spelling word from the list if the error is at the starting of the word and if the user-written word contains too many errors then it may not offer any alternative spelling suggestion. Draw back of this system is that since their approach is dictionary based approach which needs to store all words (inflectional, derivational and compounded words) of the language in dictionary. This leads to consume space and there is a probability of missing more common inflectional, derivational and compounded words of the language.

Seth and Mieczyslaw [44] developed an intelligent spell checker that adapts to the user's mistakes using a new model in software architecture (self-adaptive software). The architecture is designed upon the structure of an adaptive controller which uses the feedback mechanism to induce adaptability of the system. They used five knowledge sources to develop a spell checker. The input to the system is a text file containing a list of words. The goal of the system is to recognize erroneous words and then to attempt to

correct the identified erroneous words and present suggestions to the user on what word should be used instead. The suggestions presented to the user are based on the mistakes made most often in the document. The application, Smart Spell-Checking System (SSCS), has been implemented as an instance of the Adaptive Software Architecture. It consists of three domains (Input, Error and Evaluation) and nine KSs. Both the detection and correction of erroneous words are implemented within the KSs. The detection of erroneous words is implemented using a dictionary and a user defined dictionary. The user defined dictionary contains words that the user wishes to be considered, in addition to standard dictionary entries. The following Knowledge Sources were used for correcting erroneous words: left-right character shifter, Character doubler, end character appender, character remover and subsequent character switcher. A larger number of KSs would produce more valid suggestions to the system and consequently would result in better coverage of spelling errors. They compare their system against non-adaptive system and it performed much better and picked the right corrected word in more than 80% of the cases.

This system also uses only dictionary of two types such as normal dictionary and user defined dictionary where detection of erroneous word can be done. No consideration of the morphology of the language.

Amorim and Zampieri [7] attempted to develop an effective spell-checking method using clustering algorithms. They come with a novel approach to spell checking using dictionary clustering and their goal was to reduce the number of times distances have to be calculated when finding target words for misspellings. The method is unsupervised and combines the application of anomalous pattern initialization and partition around medoids (PAM). This spell checker is also done based dictionary clustering for dictionary look up and not consider the morphology of the language.

Kevin Atkinson [45] developed spell checker for English language called GNU Aspell. GNU Aspell is a free and open source spell checker designed to eventually replace Ispell. It can be used either be as a library or an independent spell checker. Its main feature is that it does a superior job of suggesting possible replacements for a misspelled word than just about any other spell checker out there for the English Language. Unlike Ispell. Aspell can easily check documents in UTF-8 without having to use a special dictionary. It also supports for using multiple dictionaries at once. This spell checker also does not consider the morphological structure of the language and others.

3.3 Spell checker for Arabic Language

Hamza *et al.* [15] developed a spell-checking system for Arabic language independently from the vocabulary by introducing morphological analysis in the spell-checking process. Their aim is to use a dictionary of small size that represents Arabic language stems to correct spelling errors instead of using a large dictionary. They compare their method with Levesthein method by considering three indicators: the correction average time, the rate of rectified words and the size of each system in the lexicon. They used 170000 words as lexicon size for Levenshtein method and N words lexicon size for their system. Based on these three indicators, they get best result when they compare their system with the Levesthein method as in the following. Based on their approach when they evaluate the system gives result 85% for addition operation, 81% for deletion operation, 86% for permutation operation with average time (erroneous word) 0.10 ms whereas evaluating with Levenshtein method produce the result of 44% for addition operation, 41% for deletion operation , 46% for permutation operation with average time(erroneous word) 0.19 ms.

Another research conducted for Arabic spell checker was by Shaalan *et al.* [16] who proposed towards automatic spell checking for Arabic. They developed a tool capable of recognizing and suggesting corrections of misspelled input for common spelling errors. The system was composed of Arabic morphological analyzer, lexicon, spell checker and spell corrector. They limit their system to detect and correct spelling errors to isolated words. The tool developed tries to add missed character, replace incorrect character, remove incorrect character and add a space to split words. The developed tool is useful for automating the proofreading of the human typed Arabic texts.

Mourad Mars [46], developed a robust spell checker for Arabic text based on dictionary lookup and morphological analyzer. The system detects error in Arabic text and correct the detected errors in the text. To create a dictionary, corpus of years 2000/2005 used which was downloaded from MultiUN.a. They used MADAMIRA for morphological analyzer. MADAMIRA is a system developed mainly for morphological analysis and disambiguation of Arabic text. Their experimental result shows that their system is better in performance than previous spell checkers. Their system gives the result of 95.1% correction ranked first, 72.87% precision, 65.34% recall and 68.90% F1-Measure whereas when evaluated with MS Word 2013 produces 93.74% correction ranked first, 62.63% precision , 53.92% recall and 58.38 F1-Measure. For the case of evaluating with AyaSpell,

results of 93.53% correction ranked first, 69.65% precision, 63.32% recall and 66.33 F1-Measure are obtained. So, from this result of comparison it is concluded that their spell checker performs better than tha of MS Word 2013 and AyaSpell.

3.4 Spell checker for Indian Language

Aggrawal [18] proposed a web-based application called Hindi Editor with Spell Checker that allows the user to type directly in Hindi on Internet using their normal keyboard and get spell checking done and save the file. It checks each word automatically while typing and checking can be done in a dictionary and if not found in it then in a corpus. It also has facility of adding words in the dictionary (from corpus or new words). Hindi is the official language of India which consist 11 vowels and 33 consonants. Hindi is also the third most spoken language in the world [19]. Kaur and Singh [19], designed and implemented HINSPELL-Hindi Spell Checker using Hybrid approach. HINSPELL is a web-based spell checking and correcting application for Hindi language. HINSPELL only deals with non-word errors. The main features of HINSPELL are large correct database and user interactive. Two different applications are designed in HINSPELL. One is dictionary creation tool, executed once to create the own dictionary and second is a spell checker for Hindi language and it is implemented in c# language. At start, user gives the input Hindi text and the system detect the errors by looking up for that particular word into the created Hindi dictionary and provides the correct suggestions for that misspelled word in the suggestion list. After that user can select the suggestion from the suggestion list and replace errors accordingly. The final output is a corrected text without any spelling mistakes. They used dictionary look up technique for detecting errors and Weightage algorithm, minimum edit distance and statistical machine translation techniques for error correction. The system detects approximately 83.2% of the errors and provides 77.9% of the correct suggestions for the misspelled words.

This spell checker was based on only dictionary look up method approach and uses large correct database without considering the morphology of the language.

Subash *et al.* [47] developed spell checker system that detects the error by using a dictionary lookup approach and N-gram based technique for error correction for Malayalam language. In dictionary method, it checks each word of input for its presence in the dictionary. If the word is present in the dictionary then it is a correct word else put into the list of error words. N-gram based technique corrects error by finding similarity between words and computing a similarity coefficient. They implemented it as two phases.

The first phase is error detection in which the system flags words in a document that not spelled correctly. Error detection is done through a dictionary lookup approach. The second phase is error correction which provides suggestion for misspelled words. Error correction is implemented through N-gram based technique. They test the system with test datasets of a dictionary contained 10000 words which are collected from Malayalam corpus. The accuracy is a parameter; it is used to evaluate the efficiency of the proposed N-gram analysis algorithm over a rule-based Malayalam spell checker. They got the accuracy which is based on the number of words stemmed correctly. This spell checkers is designed depending only on dictionary look up of the words stored in the dictionary of the language. For languages with rich and complex morphology this system misses to detect and correct inflected, derivated and compounded words. Murthy et al. [27], developed a non-word Kannada spell checker by using morphological analyzer and dictionary look up methods. Kannada is spoken as a first language by 38 million people and as second language by another 9 million people in Southern India, primarily in the state of Karnataka [48].All the root words of the language can be paced in a single dictionary. They use a morphological analyzer and a dictionary of only the root words. Initially the input word is searched in the root tree, if the input word doesn't exist in the dictionary as a root word, the algorithm will employ a morphological analyzer to strip all the suffixes and get the root word. The application makes use of tree data structure to build the dictionary and Levenshtein edit distance algorithm to find suggestions for misspelled word. Their result shows that the application better both in space and time efficiency. Upon finding the error, it suggests only those words that are similar to the erroneous word.

3.5 Spell Checker for Malay Language

Basri *et al.* [49] developed automatic spell checker for Malay blog. The previous spell checker for the Malay language uses a dictionary that contains pair of commonly misspelled word and its correctly spelled word in detecting and correcting misspelled word. However, this kind of spell checker can only misspelled words that exist in the existing dictionary, otherwise it requires user interaction to correct it manually. This approach works well if the spell checker is standalone system but it is not really an effective system when the spell checker is part of another Information Retrieval (IR) application such as document retrieval for weblog. There will be always new misspelled words created along with increasing number of weblog. Because of this, they proposed new approach that detects and automatically corrects misspelled words in Malay without

any interaction from the user. In their approach, there are five modules that the system should go through in detecting and correcting misspelled words. These are:

- **Tokenization***:* All the texts extracted from a collection of weblog articles are tokenized and form a list of unique words.
- **Elimination of symbol:** all the symbols except alphabet and word combination with number are eliminated such as '?', '!', '#', '$', '%', '^', '*', '~'.
- **English Word Elimination:** all the English words exist in the word list are filtered out, because their proposed spell checker was designed for Malay weblogs and English words are considered as misspelled words.
- **Stemming Process:** is the process of removing affixed word to its root word by following the morphological rule of the language. The purpose of stemming process is to identify either the word is affixed word or not.
- **Misspelled Identification:** one of the causes of a misspelled word is typo word. Typo word is a word that exists In Malay language but the word is spelled based on their pronunciation. These words are not recognized by any systems because they are not spelled according to standard Malay spelling rules.

In order to evaluate their approach, they used the Malay weblogs written by Malaysian as their tests datasets. In their experiment only 11 out of 23 weblogs categories are used since the other 12 categories are written in English. These weblogs were chosen because these weblogs are considered as the most visited weblogs in Malaysia. After eliminating all symbols from the datasets, they managed to get 11896 words. From these words, there are 3046 distinct words obtained. After eliminating the English words, there are 2675 words left. They apply the spell checker for these 2675 distinct words and 2453 words are successfully identified and corrected. That means only 232 words or 8.67% words are failed to be detected and corrected by using their proposed approach. Still this spell checking system is implemented using dictionary only, morphology of the language if not considered for inflectional, derivational and compounded words of the language.

3.6 Spell Checker for Igala Language

Igala is a language of the ethnic group located at the eastern flank of the confluence of rivers Niger and Benue. They are the 9th largest linguistics group in Nigeria [50]. Ayegba *et al* [50] modeled a language processor for Igala which can detect and correct non word or typographic errors and called their system as igSpellChecker. They used the approach of lexicon look up method for determining the validity of words and generate correct alternatives for misspelled words using similarity key technique. Full form of

lexicon technique was adopted which lists all words and their derivatives in the dictionary. The system is tested with the data which is unedited news article printed out and manually spell check by Igala native speakers. The data set was found to contain 1876 valid words and 422 misspelled words. From this sample, 1782 out of 1876 words were correctly recognized as valid words by the spell checker while all 422 misspelled words were flagged as misspelled. The quality or effectiveness of the igSpellChecker system is evaluated using the evaluation metrics such as lexical recall, error recall, and precision and suggestion adequacy. The evaluation result show that the result obtained is 94.9% lexical recall, 100% error recall, 72.5% precision and 81.2% suggestion adequacy. Storing all the lexicon form with their derivate in dictionary is steal difficult that leads to the problem of consuming time, insufficient memory space, missing some new words and the likes. Full form of lexicon approach is also have a limitation of missing of storing all derivatives of the words of the language and time consuming to prepare all the derivatives and space consuming to store all the derivatives.

3.7 Spell Checker for Afaan Oromo Langauge

Gaddisa Olani [23] developed Afaan Oromo Spell Checker (AOSC). The system was designed based on a dictionary look-up with morphological rule based (i.e., Morphology based spell checker). The system was evaluated using with datasets of different size. The system is evaluated with a total number of 1811 words, from these 1732 were valid Afaan Oromo words and 1535 were accepted as valid words and 197 are flagged as invalid words by the spell checker system. Based on the evaluation metrics (lexical recall, error recall and precision), they got the result of 88.62% lexical recall, 100% error recall and 28.62 precision. The author experiment shows that the lexicon size and rules in the knowledge base play a great role to recognize the valid input word, flag the invalid word and generate correct suggestions for wrongly typed word. Large lexicon size and having good rule in the knowledge base will improve the performance and quality of the spell checker system.

3.8 Summary

In this chapter, we discussed different spell checker systems developed for various languages such as English, Spanish, Turkish, Arabic, Asian languages, Afaan Oromo. We addressed the approaches and techniques used for each spell checker systems with respective language. Most of the spell checkers discussed are based on dictionary look up approach except that was developed for Afaan Oromoo and Asian spell checkers. Some of

the spell checker developed was didn't consider the morphology of the language. Due to variations in morphology of the language, various approaches are used to develop spell checker systems for respective languages. We have learnt that spell checkers developed for other languages cannot be directly applied to Kafi Noonoo language due to its morphological complexity. Accordingly, it is understood that development of Kafi Noonoo spell checker system is required further investigation. Thus, in this work we aim to apply morphology based approaches to develop spell checker for Kafi Noonoo language.

Table 3.1: *Summary of Related Work*

work	Approach	Strength	Limitations
Spell checker for English words	• Dictionary based approach	• multi word suggestion to enhance the performance	• the system does not offer any alternative spelling word from the list if the error is at the starting of the word • for window-based application
Effective spell-checking for English words	• Dictionary Clustering approach	• reduce the number of times distances have to be calculated when finding target words for misspellings	• Morphology of the language is not considered
Spell-checking system for Arabic language	• Dictionary based approach • Morphology Based approach	• use a dictionary of small size	• Only morphology of Arabic word was considered

HINSPELL-Hindi Spell Checker	• Hybrid approach	• facility of adding words in the dictionary (from corpus or new words) • large correct database and user interactive	• Morphology is not considered and instead uses large database
A non-word Kannada spell checker	• morphological analyzer • dictionary lookup method	• employ an algorithm to strip affixes form root words • build a dictionary as a data structure	• Only for Kanana words and morphology of language I considered
Automatic spell checker for Malay blog	• Dictionary lookup approach	• automatically corrects, without user interaction	• Only for Malay words and morphology of language
Igala Language Spell Checker :igSpellChecker	• Lexicon lookup approach	• Prepare derivatives of a word and store in the dictionary	• Preparing all derivatives of a word is still not enough • Need morphological analysis to be considered
Afaan Oromo Spell Checker	• Dictionary lookup • Morphology based	• Lexicon preparation • Rule preparation	• Depends on Afaan Oromo recourses

Chapter Four: Design of Kafi Noonoo Spell Checker

4.1 Introduction

In this Chapter, we discussed the design and implementation of Kafi Noonoo spell checker. The proposed system architecture is shown in Figure 4.1 which have four major components such as tokenization, error detection, word suggestion, and error correction. The other part of this architecture is morphological features and root dictionary which are used as a backend by the spell checker system. All these components are discussed in detail with approaches and techniques to implement them in this Chapter.

4.2 System Architecture

The proposed Kafi Noonoo spell checker system has four major components. These are tokenization, error detection, word generation and error correction components. Tokenization is responsible for splitting a text into standard lexicon or block of text can be split into individual words. Error detection component can detect whether the input word is correct or incorrect based on root dictionary and morphological features. Word suggestion component list all possible suggestion words where the user will select the correct word from the list. Once the suggestion list has been generated, each suggested list is ranked in increasing order based on their edit distance score. Error correction component correct erroneous word based on the users selected word from the given suggestion lists. Root dictionary is the backend component of the spell checker system that stores root words of the Kafi Noonoo language. Morphological features is a backend of the spell checker that stores morphological features such as affixes and rules to attach each affix to the root word to generate a word through morphological process of Kafi Noonoo language. Each components of KNSC with their purpose and algorithms are discussed in details in this section. The general architecture of Kafi Noonoo spell checker is provided in the Figure 4.1.

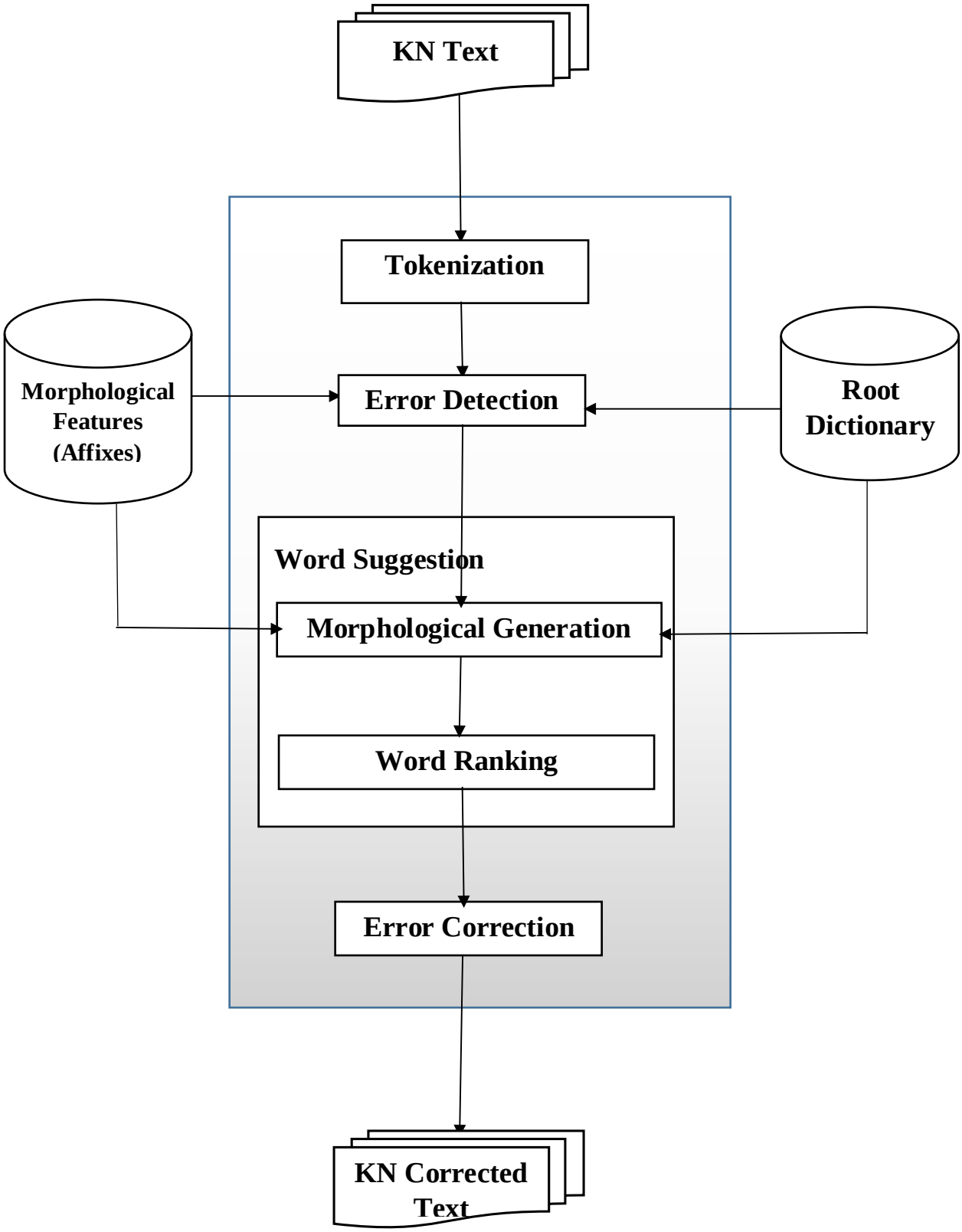

Figure 4.1: *The General Architecture of Kafi Noonoo Spell Checker*

4.3 Tokenization

Tokenization is one of the preprocessing task. Tokenization is a way to split block of text into individual tokens. So, these tokens may be paragraphs, phrases, sentences, punctuation marks or individual words. In our case for spell check, we consider tokenization at word level because our system is developed to check for the correctness of a word for Kafi Noonoo language at word level. The text is broken with the help of some boundary delimiter such as white space and punctuation marks. If punctuation marks come at the end of a word, they are removed from that word and recognized as individual token. Like English language, in Kafi Noonoo language words are separated or ended by blank space and punctuation marks [48]. Punctuation marks that are used at the end of words are period (.), comma (,), exclamation mark (!), question mark (?) and semicolon (:). White space is used as boundary delimiter between words. A word after the space becomes a token if a space is encountered. So, we consider blank space for tokenization in our case of text processing. Algorithm for tokenization of words shown below in Algorithm 4.1.

```
Input: Words/File

     For each word in file containing space

     Split each word by space

     Store each word in temporary dictionary

          If punctuation mark exist at end of a word

          Remove it

          Store each punctuation mark as individual tokens

          End if

     End for

Output: Tokenized words (Tokens)
```

Algorithm 4.1: *Algorithm for Tokenization Process*

For the case of stop word removal, Kafi Noonoo does not exhibit stop words and even if happen they if stop word exists it is possible to store these stop words in dictionary for the case of spell checking.

4.4 Error Detection

Error detection component detects whether the user input word is correct word or erroneous word. Error detection can be detecting a single word given by the user or checking through a large document (collection of words). If the error is not detected nothing to do except accepting the word as correct. But, if an error is detected it passes the error word to word suggestion where generation and ranking of word for an error word is done. If the input word is inflected, derivated or compound and part of the language doesn't recognized as error by the spell checker system rather employ morphological analysis to split the word into root word and affixes. Then after, root word is checked in root dictionary and affixes are searched in morphological features dictionary. If both root word and affixes exist in the respective dictionary and if rules applied to each affixes satisfy to match with the root word, the system recognize the word as correct else recognize as error word. In order to consider inflected, derivated and compounded words as correct, the identified affix and word class must much, otherwise these words are recognized as error word. In this work, error detection can be done through dictionary lookup and morphological analysis. Algorithm for error detection with the help of morphological analysis is depicted in Algorithm 4.2.

```
Input: tokenized word from tokenization component
     For each tokenized input word
     Check the input word in the dictionary
     If the input word is present in the dictionary
     Return correct word
     If the input word is not present in the dictionary
     Return incorrect word
     Else
     If a word is inflected, derivated or compounded
     Split each word to root word and affix word
     Check for the existence of root word in root dictionary
     and affixes in morphological features dictionary
          If both root word and affix exist
          Check for matching
          If match found
          Return correct word
          Else
          Return error word
       End if
     End for
Output: correct/incorrect word
```

Algorithm 4.2: *Algorithm for Error Detection*

4.4.1 Morphological Features

An affix file defines the meaning of special flags in the dictionary. An affix file ((*.aff) may contains a lot of optional attributes. These attributes may be SET for setting the character encodings of an affix and dictionary file, TRY is used to set the change characters for suggestions and PFX and SFX are used to define prefix and suffix classes named with affix flags respectively.

Affix rules optionally follow each word in the dictionary file separated from root word by using slash ("/"). This rule tells us what affixes can apply to the root word. Affix rule look like a string of letters or numbers that after each root word. Affix class format:

<option_name><rule_letter_identifier><Combineable_flag><number_of_rule_lines >

Where option name is either PFX for prefix or SFX for suffix, rule_letter_identifier contains a rule that can be attached and applied to the root word, combineable_flag is optional and you may ignore it normally since it is Y (yes) or N (no) depending on whether it can be combined with other rules or not, number_of_rule_lines this can lists different possibilities for how this rule applies in different situations and have the following format:

<option_name><rule_letter_identifier><No_letters_delete ><letter_added ><conditon>

Where No_letters_delete is the number of letter stripped from the word (the flag is 0 if there is no character to strip) and letter_added is a letter that added after stripping some letter form the word and condition is an expression to be fulfilled what to add . For example:

SFX A Y 2

SFX A o eena'o [lg]o

In this example, A defines that suffix -*eena'o* can be added to a word if the last character of the word is *o* and preceding with consonants *l, g* this with vowel character *o* is removed to satisfy the condition provided. This rule can be applied to a noun words like *bago* (sheep) to generate *bageena'o* **(**sheeps) and *kaameello* (car) to generate *kaameelleena'o* (cars). Sample affix rules for Kafi Noonoo language is given in Annex D. For morphological rule and lexicon creation our knowledge base is based on the discussion provided in [10, 12, 13, 40, 43].

Morphological Analysis

Morphological analysis is a process to analyze a word forms into their roots and affixes. It will return root word along with its grammatical information depending on its word class. Creating a dictionary having all the words of the language is very difficult task considering the large variations of affix combination in the language. To overcome this problem, we use morphological features and dictionary of only the root words and dictionary of affixes that are going to be attached to the root word based on the provided rule. For example, only the root word *bushoo* is stored in the dictionary instead of storing all its inflections like *bushee, busheen, busheena, bushiisho, bushittino, bushoona, bushiisheena'o, bushiisheena'ona* etc. First the input word is searched in the root dictionary, if the input word does not exist in the root dictionary as a root word, then the algorithm will employ a morphological analysis to split all the affixes and get the root words. Even after the word stripped to root word and affixes but still the root word is not found in the root dictionary. Then it reports the input word as misspelled word and passes the error word to error correction module for suggestion and correcting the error word. As the knowledge of the researcher there is no morphological analyzer yet developed for Kafi Noonoo language. So, we use our algorithm which is depicted in Algorithm 4.3 to process the task of processing the text for morphological purpose. Algorithm 4.2 applied based on the rules stored in the database. Individual word is scanned from right to left for suffix stripping. If it finds a valid suffix, it strips the suffix from a word. Then the resulting root word is searched in the root dictionary. If it is found then the word is spelled correctly else the word is passed to error correction for further processing. All such pairs of valid suffix and their corresponding stripped part are stored in buffer for further processing. So suffix striping method is used to strip inflected form of words in to root words and suffixes .The general format of morphological analyzer is shown in Figure 4.2.

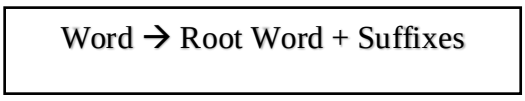

Figure 4.2: *The General Format of Morphological Analysis*

4.4.2 Root Dictionary

All the root words of the language can be placed in a single dictionary. This result in a huge dictionary of millions of words. A dictionary is a collection of unique words [27]. Our dictionary contains root words with affixes class attached to identify which affix is going to be attached to which root words. We designed a dictionary containing of more than 6000 root words without inflected, derivated and compounded words. The size of the developed dictionary is 101 KB. Root dictionary is used by error detection component to check for the existence of the root word and also used by morphological generation component to generate suggestion for the misspelled word. An affix is list of words that is going to be attached to the root word. The common affix in attached to root words are prefix, suffix and infix. For example, root words like *mamo* (eat), *uyo* (drink) can be stored in root dictionary and suffix words *-atan* like in words *maatan* (she ate) , *-ate* like in words *maate* (he ate) are stored in suffix morphological features. This created dictionary is used as database for the spell checker which spell checker can check for the existence of words in the dictionary. A powerful dictionary is required to develop spell checker for detecting and suggesting error words that is used as a reference. It is very difficult and laborious task to create a dictionary containing all the words of the language. It is time consuming and space shortage can happen in case of storing all the word forms in the dictionary. Because of Kafi Noonoo language is morphologically rich language that have different affix combination to generate different new valid word forms. For example, over 95 valid word forms can be generated from a single verb root *daamo* (take). The inflected word that are generated from a single verb *daamo* is presented in Annex B. Here we need to design two lexicon files for KNSC. The first one is root word dictionary file and the second one is morphological features which is affix file in our case. For creating both the dictionary file (*.dic) and affix file (*.aff), we adopted the format of Hunspell [51]. Hunspell[1] is Hungarian spell checker, morphological analyzer library and program designed for languages with rich morphology and complex word compounding or character encoding.

The dictionary file contains the list of word forms with information about morphological affix classes to combine with the roots and one or more flags which represent affixes or attributes. Each root words of the language are placed in the dictionary one word per line

[1] http://manpages.ubuntu.com/manpages/trusty/en/man4/hunspell.4.html

with information about affix to be combined with the root word. Root words are separated from class definitions associated with affixes by slash "/". General format for creating dictionary of words is:

<Root_word>/<Affix_Classes>

Root_word is a word that is stored in dictionary without any inflection, derivation and compounding. Affix_classs contains affixes and rules that can be applied to root word to inflect, derivate and compound a word. One or more affix classes can be attached to a single root word. For example:

- kaameello/A
- no
- bi

In this example, A indicates the affix class of the word *kaameello* (car) and based on the affix given, *kaameellona* (by car) is the possible derivation of the word *kaameello*. The words like *no* and *bi* have no affix class because these words cannot undergo affixation to generate another inflected word. Sample Kafi Noonoo dictionary file is provided in Annex E.

Root Word Classification

In order to create a meaningful spell checker system, it is necessary to identify the category of each root words of noun and verbs and the other word classes. The reason that word category is required is that all words cannot undergo the same affixation to produce meaningful word. All affix words cannot paired with the same root words. If so happen, there is a probability of generating meaningless word for misspelled word and recognizing misspelled word as a valid word. There is still a big challenge in grouping of a root word into different class of root word for each nouns and verbs because there is a little work done for morphological analysis and no annotated data for Kafi Noonoo language. Simegnih Tekle [12] tried to classify each nouns and verbs into some classes based on their morphological characteristics and which affix can match with which class of words. There are 3 class of nouns and 4 class of verbs total of 7 classes of root words are identified. For example, noun words with the last two syllable ending in short vowels *o* and having end consonants like *c, f, h, g, n, r, t, x* and **y** can take the same suffix for marking plural after deleting the last *o* vowel. Suffixes such as *-eena'o, -ina'o, -eena'one, -ina'one, -*

eena'och, -ina'och are some of the suffix for this class for plural marking. More details about the classification of root words for Kafi Noonoo language is depicted in Annex C.

4.5 Word Suggestion

Once an error has been detected by error detection component, it passes that erroneous word to word suggestion component for generating list of words as suggestion and ranking list of words generated as suggestions. After the process of error detection is completed, the next step is providing suggestion for the misspelled word that is detected as misspelled word by error detection component. Once the user entered incorrect word, the spell checker can provide a list of words as suggestion for misspelled word where the user can select it. There may be the probability of no expected word in the list of suggestion if there is no root word and affix word. Using edit distance approaches to generate suggestion by replacing, transposing, removing, adding characters to make list of words as suggestion for misspelled words. First, a character based candidate generation can be generated with all possible candidates within an edit distance. This can be done by considering replacing each character with all possible characters in the alphabet, transposing each pair of adjacent characters and deleting each characters. Levenshtein edit distance (LED) is a measure of the similarity between two strings (S1, S2) where S1 is the source string and S2 is target string. The distance is the number of deletions, insertions, substitutions or transposition operations required to transform S1 to S2. The greater the edit distance the more different the strings are and less the edit distance the more similar the strings are (more details discussion on edit distance is provided in Chapter 2, Section 2.4.2.2).

4.5.1 Morphological Generation

Morphological generation is a process of generating final fully-inflected words from the root and a set of morphological properties. Morphological generation is the task of producing the appropriate inflected form of word in a given textual context and according to some morphological features. It concatenate root words from root dictionary and affixes from morphological features to form inflected word and provide these words to suggestion component for correcting the erroneous word. A morphological generation needs to be designed to tackle the different syntactic categories such as verb, noun, adjective, adverbs, pronouns, etc., since the addition of morphological constituents to each of these syntactic categories depends on different types of information. The identified suffixes are used along with the rules for the purpose of the morphological generation.

Pairs of morphemes obtained after suffix stripping is processed in word suggestion component to classify errors in one of the following classes.

i. Correct root word and correct suffix

In this case, for providing suggestion for misspelled word the root word is searched in root dictionary and suffix is searched in morphological features, if both the root word and suffix are found in the dictionary and the combination satisfies the rule then the word is considered as correct word and no further processing. For example, *bushiisho= bushoo* (root word) + *iisho* (suffix). In this example, *bushiishoo* (children) is a correct word because the root word *bushoo* (child) exist in the root dictionary and suffix word *-iishoo* plural marker for this word also exist in the morphological features and based on the rule given for the suffix class is satisfies the combination of the two morphemes. For such words no further processing is required by morphological generator.

ii. Incorrect root and correct suffix

If the root word is incorrect, it is difficult to determine the exact suffix that match with the root word. But, after we strip the word, we get a list of possible pairs of stripped root word and valid suffix, then the valid suffix shows a specific category of the possible root. After getting the possible valid suffix, then possible root word is checked for a word match and if a match found, the possible suggestion are generated for the misspelled word. For example, the word *assheena'o* is misspelled, after stripping the valid suffix *-eena'o* which is a valid suffix, we get *assho* which is invalid root word. So, possible root is searched for *assho* root word based on the rule provided for the suffix *-eena'o* is going to match with. Words like *asheena'o, aasheena'o, asheena'on* are provided for suggestion. User can choose the correct word they want to replace the misspelled word from the generated suggestion lists.

iii. Correct root and incorrect Suffix

For the case of misspelled word that is occur due to the miss combination of valid root and invalid suffix, the suffix is searched from right to left and if found invalid suffix, then again the word is scanned form left to right and the stripped into a root word and possible suffix. The process provides a list of pairs. Then, for each pair, the suffix is searched in the paradigm table corresponding to each root. The process of searching and checking continue until suffix match is found. If a suffix matching is found then the obtained suffix

is combined with the root word to provide suggestion for the misspelled word. For example, the word *ashob* is recognized as an error word, then after applying stripping we get valid root word *asho* and invalid suffix *-b*, so the suffix is searched that can match with the root word *asho* and if possible suffix are found, possible suggestion words are generated. So, words like *ashon, asho* are possible generated suggestions by the spell checker system. Then after the user can select the word they want to replace the misspelled word from the suggested lists.

iv. Incorrect root and incorrect Suffix

For the case of misspelled word that is occur due to the miss combination of invalid root and invalid suffix, the morphological analyzer tries to split the word into root and suffix, but the stripped root word and suffix does not exist in the provided class and the word is recognized as misspelled word. Some nearest word are displayed as suggestions for that misspelled word by taking nearest root word and suffix that match to this root word class. For example, the word *ashb* recognized as error word, after stripping we get invalid root word *ash* and invalid suffix *-b*. So, words like *ashe, asha, ashi, asho* are possible generated suggestions by the spell checker system. Replacing misspelled word is left for the user why because automatic correction is not applied in our spell checker this is because there is a probability of replacing the invalid word with a word of not the user choice.

After classifying error to one of error classes, the morphological generator will make possible correction for the erroneous word based on the category of the error. For example, for the case of valid root and invalid suffix, possible classes of root word is identified and to which root word can this suffix attached is determined and the valid root indicate the possible category of suffix. In similar case, for invalid root and valid suffix possible suffix class is identified and to which suffix class this invalid root can attached to give correct word is identified and the valid suffix will show the possible category of the root word. So, suffix joining method is used for building morphological generator. The general format of morphological generation is shown in Figure 4.3.

Figure 4.3: *The General Format of Morphological Generation*

As the knowledge of the researcher there is no morphological generator yet developed for Kafi Noonoo language. So, we use our algorithm which is depicted in Algorithm 4.3 to process the task of processing the text for morphological purpose.

```
 Input: morphemes from error detection (morphological features
        and root dictionary) component

        For each error word given from error detection

         Identify root word class and affix class of the error
            word

              If match found

                 Append affix to root word

                 Generate list of words

          End if

          Return list of words

     End for

Output: list of words for suggestion
```

Algorithm 4.3: *Algorithm for Morphological Generation*

4.5.2 Word Ranking

As the process of suggestion generation is completed, the next process is ranking the list of words generated to be displayed on the top of suggestion list. Sorting the suggestions according to the relevance of word is the most important part of spell checker, so that the most likely candidate is placed first. LED algorithm is very efficient for many language to rank suggestions, so far most the spelling checkers recommended this method for ranking suggestions [52]. So, we apply this method to our spell checker system to rank suggestions.

We adopted algorithm used by [23] with slit modifications as provided below in algorithm 4.4.

```
Input: list of words from morphological generation
     For each of word given from morphological generation
               Call Edit_daitance
               If two or more suggestions have the same edit
               distance score
                    Call character_distance
                    Rank suggestion list by character_distance
               Else
               Rank suggestion list by Edit_distance
               End if
     Order the suggested list in ascending order of their rank
     End for
Output: Top N ranked list words for error correction
```

Algorithm 4.4: *Algorithm for Ranking Suggested List*

Our spell checker system which generates list of words for misspelled word, the generated words are ranked according to edit distance score, but if the two strings have the same edit distance score (value), the system will use character distance to rank the strings. We evaluated our ranker manually with the help of linguistics who have the knowledge of the language.

4.2.4 Error Correction

Once the system detected an error word the next step is to provide list of suggestions with appropriate ranking. After all this process completed, correction of an error word is done. Correction can be done in two ways. The first one is that the system can automatically replace the error word with correct word based on the given criteria. The second one is that the user interaction. Once the suggestions are listed, user can select the intended word

from a given list of suggestions. In case of automatic correction, there is a probability of replacing error word with a word that is not user intended word which leads to another error. So, in this work we left the correction of error word to a user. A user can select a correct word or intended word from generated list of suggestions to replace the error word. This can be done by selecting the correct word and clicking on the replace function then after the word is replaced by the user selected word. So, automatic error correction is not advisable as it lead to further errors. User intervention would provide better performance in spell checking.

Chapter Five: Experiment

5.1 Introduction

The goal of experimental analysis is to judge the quality and effectiveness of the proposed system performance. In this chapter, we discuss the data collected for developing the prototype of the KNSC and to test the performance of the prototype, the tools used to develop the system, present the prototype of the developed system. Lastly, we present results and provide final discussions.

5.2 Data Collection

Kafi Noonoo does not have publicly available annotated corpus text for spell checker and any other NLP applications. In this work, we collect electronic text and hard copy documents form Linguistic (Kafi Noonoo Department) department of Bonga College of Teachers Education and other sources like media (Bonga FM) to build dictionary lexicon and to evaluate the performances of the spell checker analysis system. We use the dictionary of root words which is prepared by Abebe Asfaw [40] that contains root words more than 6000 Kafi Noonoo root words which we use to build lexicon dictionary. This dictionary lexicon is built and classified into different category of classes to combine with affixes and affixes are also built manually. We applied the affixes identified by the work of Simegnih Tekle [12]. For the purpose of evaluating the performance of the system we prepare a set of words that are collected from different data sources such as books, News and documents collected form department of Kafi Noonoo language from BCTE. We also communicate with the linguistics experts to understand the structure and characteristics of the language and preparation of the dictionary lexicon and affix dictionaries with rules. We collected different sentences from different sources like books and papers that produced a total number 4207 words and this words contain inflected, derivate and compounded words. After removing repeated words we get 2743 unique words. The data set contains 2566 correct words and 177 invalid words. So after applying our spell checker system to test the data, the system detected 2461 words as a valid word and 282 words are recognized as invalid words by the system. Among the 282 invalid words 105 words are due to the absence of the root words in the dictionary but these words are part of the language while all the 177 misspelled words identified as misspelled. The summary of the sample data was provided in Table 5.1.

Table 5.1: *Summary of Sample Data*

Description	Value
Total no of words tested	2743
Total no of valid words	2461
Total no of invalid words	282

5.3 Development Environment and Tools

The prototype of Kafi Noonoo spell checker system is developed and tested on Toshiba laptop computer with Window 8 Enterprise operating system, Intel(R) Core i3 with 2.40 GHz processor speed, 4 GB RAM and 464 GB hard disk capacity. Some other software components used for developing KNSC are Microsoft Visual Studio, Notepad++ and Microsoft Office 2013 used for documentation.

Microsoft Visual studio 2010 Ultimate is used as development environment to create graphical user interface (GUI) which used for creating, debugging and deploying applications. **C#** [2] is used as programming language [25]. C# was developed as a language that would combine the best features of previously existing Web and Windows programming languages. C# is totally based on Object Oriented Programming (OOP). It is chosen because of the characteristics of its object oriented, simple and flexible, automatic memory management, cross platform interoperability. In this work, we used C# programming language to implement and execute the algorithms and methods discussed in Chapter 4.

Notepad++[3] is a free source code editor and notepad replacement that supports several languages and running in the MS windows environment. In this Thesis work, we used Notepad++ version 7.5.8 for designing a dictionary lexicon (both root word and affixes with rule) and writing codes.

5.4 Prototype of the System

The spell checking system which detects erroneous words and generate a set of correct words as a suggestion for erroneous word is implemented using C# programming language

[2] C Sharp (Microsoft programming language)

3 https://notepad-plus-plus.org/

in the environment of Microsoft Visual Studio 2010. The developed prototype system provides the functionalities of detecting error word, providing suggestion, replacing error word with correct word from the suggested list based on the user interest. Users can provide text in two ways, one by directly writing the text in the text area provided in GUI. Secondly, by providing a file text through open UI. Figure 5.1 shows the GUI of Kafi Noonoo Spell Checker system which is displayed for the user to enter the text to be spell checked which is the first way.

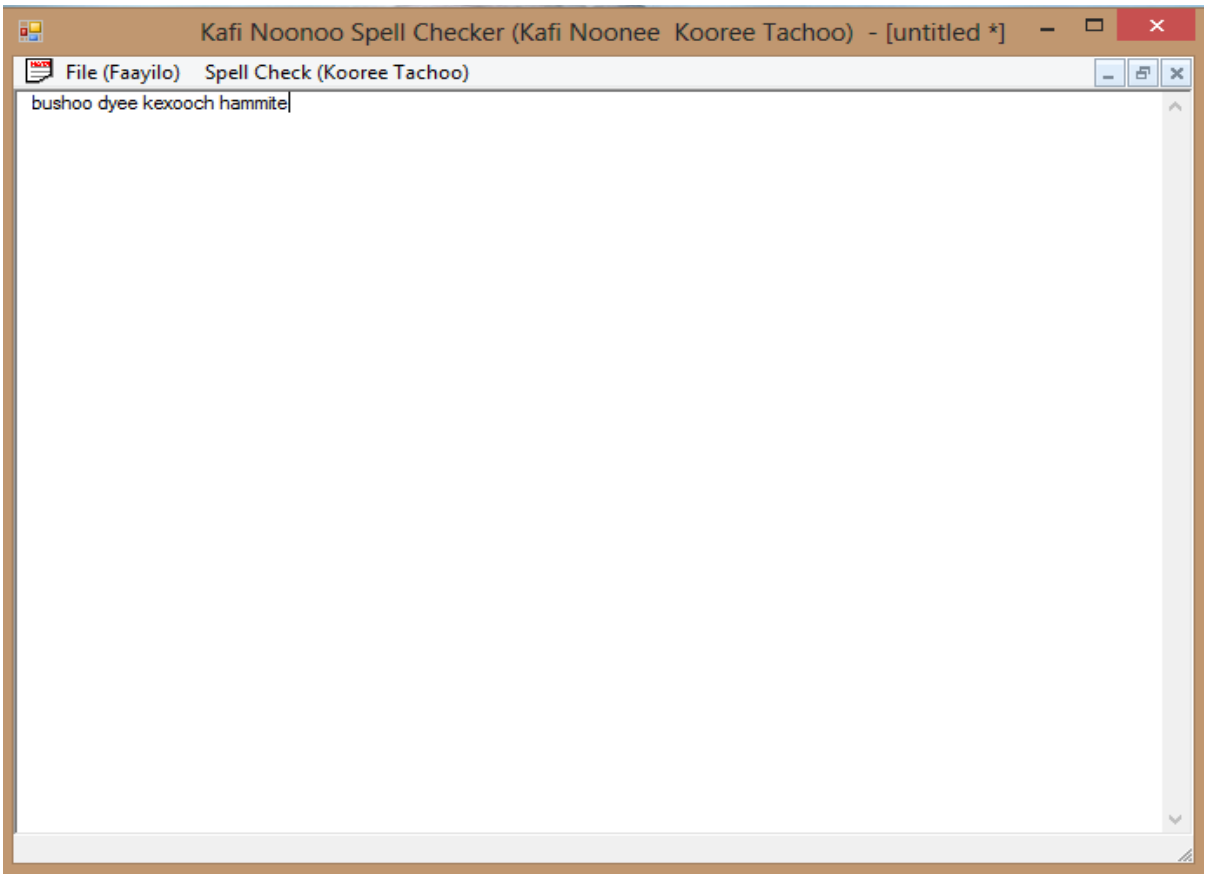

Figure 5.1: *Kafi Noonoo Spell Checker System User Interface*

If a user types a word and click on spell check button, then spell checker confirms the correctness of the word, if the word is erroneous word, it is marked with red color. Then after a list of correct candidate words are displayed for the replacement for the misspelled word and a user can choose one of the correct word for replacing the error word as shown in Figure 5.2.

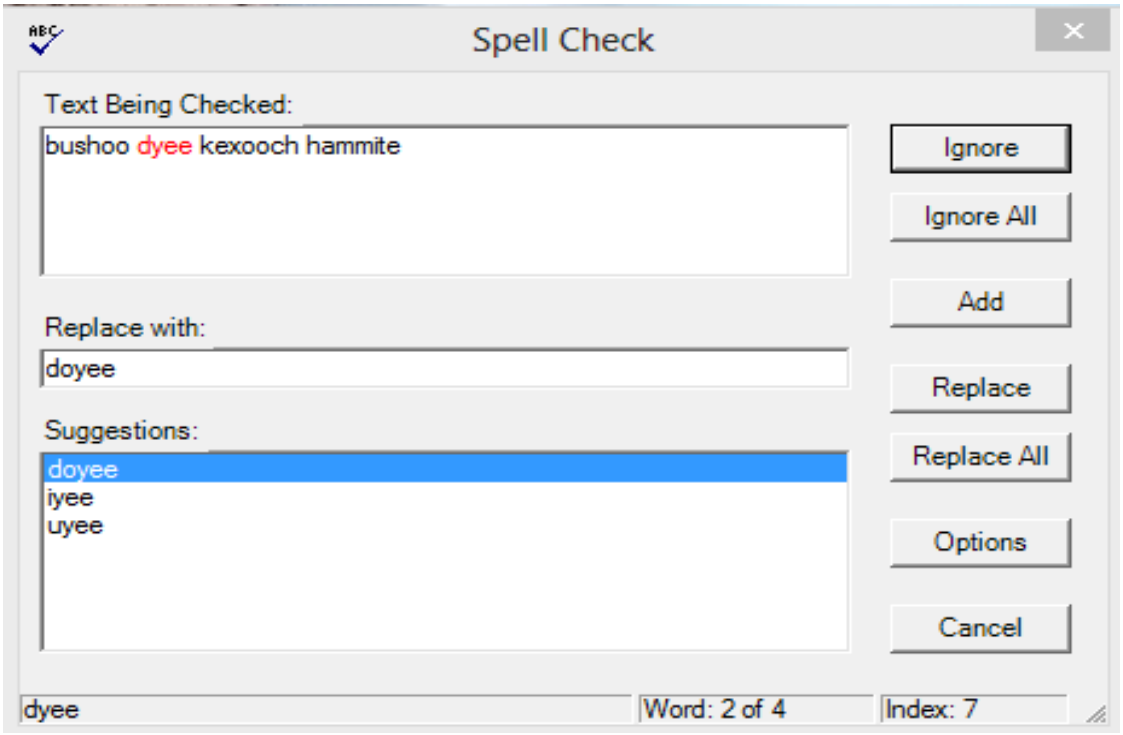

Figure 5.2: *Shows the System Display Misspelled Word with Red Color*

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

Ignore: this allows the user to leave the misspelled word as it is without correcting and jump to the next word to be checked for spelling error. If checked again that word will be recognized as again error.

Ignore all: this allow the user to leave the misspelled word as it without correcting and check next words one by one for invalid word. This invalid words are accept as valid word when the user allow leave them by ignoring all and next time when checked the system will recognize these words as valid word in that specific time of spell checking process.

Add: (optional) this function allows the user to add the word that is recognized invalid word to the temporary user dictionary provided for this purpose. This is not recommended because of new word addition can be done only by the language linguistics who have the knowledge of the correctness of the word to be accepted.

Replace: this allow the user to replace the misspelled word by choosing the correct words displayed as a replacement from suggestion list for that specific word.

Replace all: this allows the user to replace the misspelled word with the correct word chosen from the suggestion list. It replaces all such type of the same words in the document.

Options: is an optional which displays information about spell checker.

Cancel: this function allows the user to close the spell checker application.

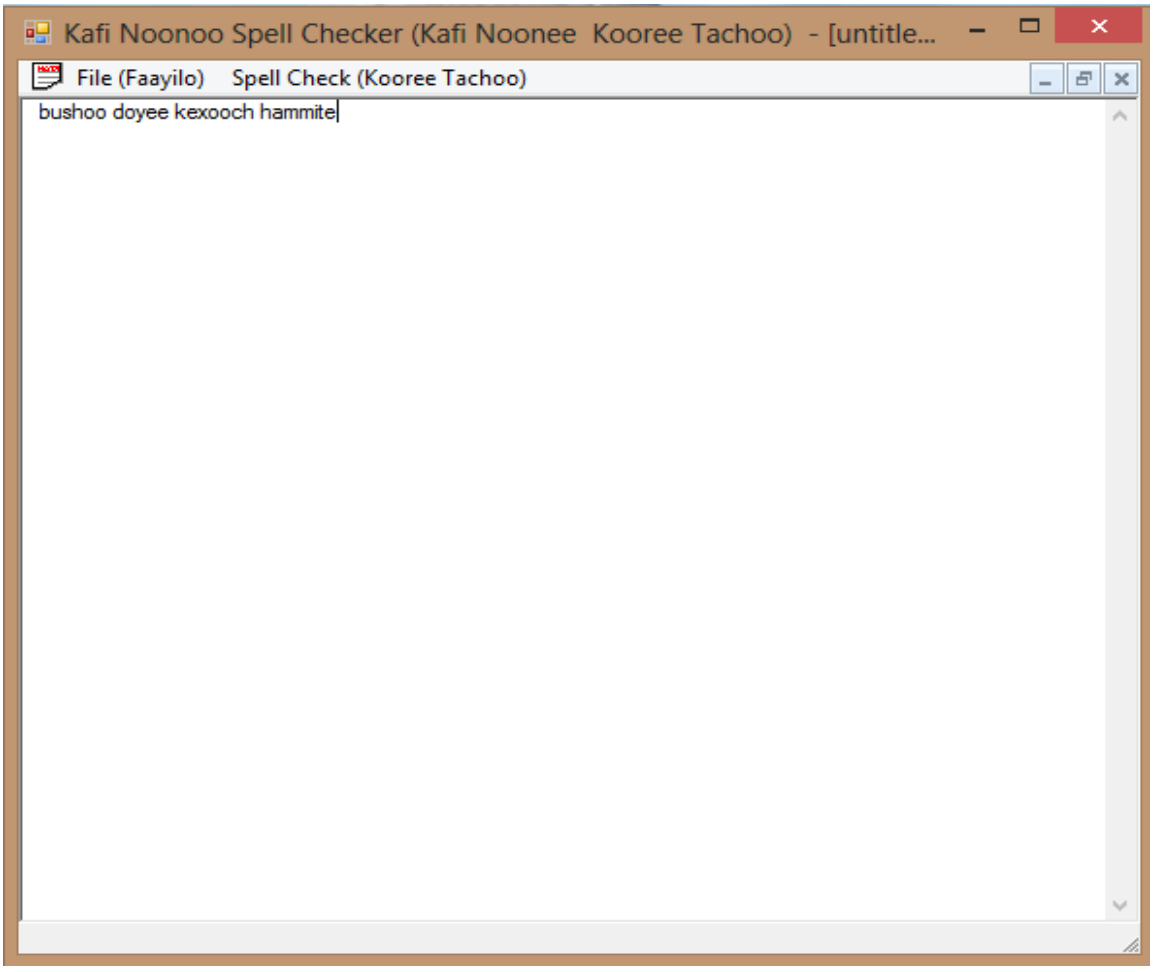

Figure 5.3: *Shows after Error Correction was Done*

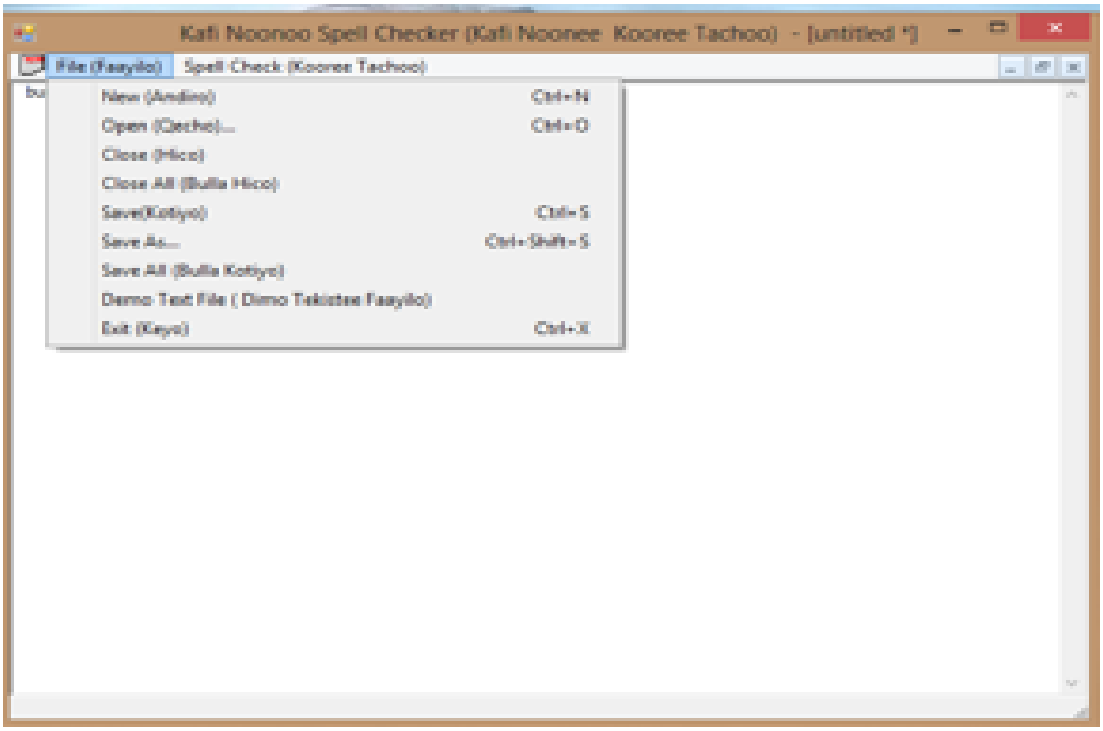

Figure 5.4: *System Allow User to Input File from Document*

The system allows the user to input text by opening from document using open button and after spell checking completed, it is possible to save the processed document as a user wish through save or save as button as shown in Figure 5.4. If a user close the system without saving the changes to document, the system prompt the user to save the processed document as shown in Figure 5.5.

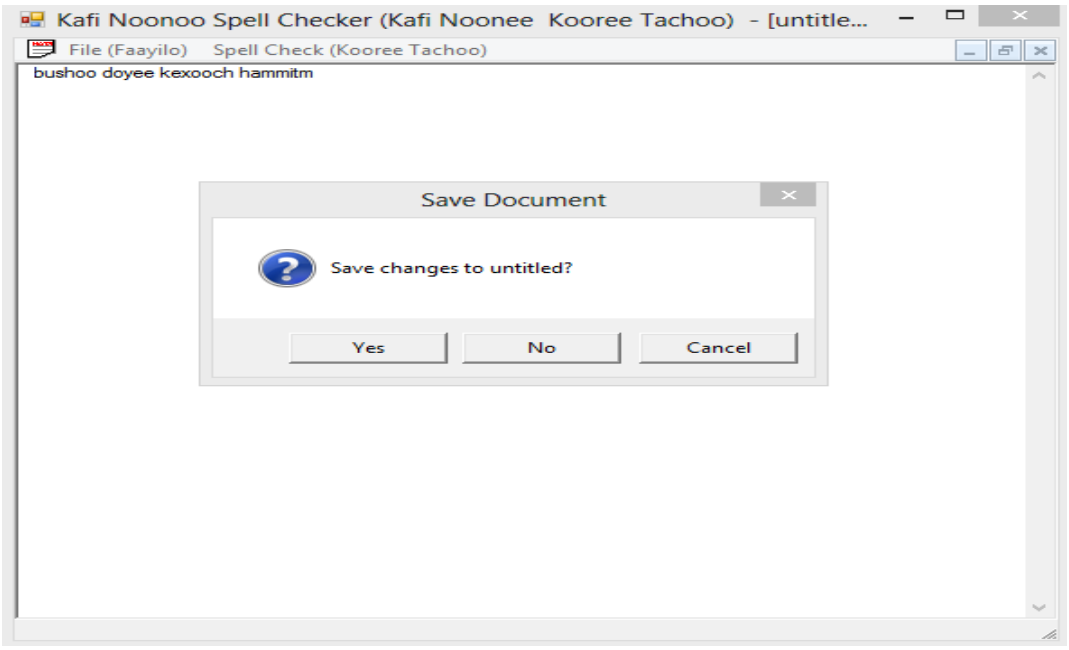

Figure 5.5: *System Request the User to Save the Processed Document*

5.5 Evaluation

5.5.1 Evaluation Procedures

The experiment conducted on the system in order to check the validity or the performance of the developed spell checker system for Kafi Noonoo language. The evaluation of the system is based on some metrics that measures the performance of the system. In this research, documents collected from different sources are used to check the validly of the system. The evaluation and test results are given in Section 5.5.2 in Table 5.1 and Table 5.3 respectively and the summary of the data used for the evaluation and testing is represented in Table 5.2.

5.5.2 Evaluation Metrics

The performance and quality of spell checker is measured using different metrics. According [5, 21, 53] the effectiveness of spell checker system is measured based on the three metrics namely precision, recall and F-measure which also used as metrics for the evaluation of the natural language processing tasks. Hamza *et al* [15] used the correction average time, the rate of rectified words and the size of each system in the lexicon to measure the performance of their work against Levenshtein distance system. In other work done [23] identified lexicon size, test set size, correction accuracy and error types as metrics for evaluating the performance of the spell checker system. Suzane [54] also used detection recall, correction recall and precision as evaluation metrics for measuring the performance of spell checker for their work. In this research we used lexical recall, error recall, and precision as evaluation metrics to measure the effectiveness of the selected approach for Kafi Noonoo Spell Checker System. The three most common used evaluation metrics for evaluating the performance of most spell checkers are lexical recall, error recall and precision. We used these three evaluation metrics to evaluate our spell checker system as we get recommendations and comments for linguistics who worked in Kafi Noonoo department in Bonga College of teacher's education.

Lexical recall is the number of valid words in the text that are recognized by the spell checker. It indicates the percentage of valid words correctly accepted by spell checker system.

Error recall is the number of invalid words in the text that are flagged as invalid by the spell checker system. It indicates the percentage of invalid word correctly flagged.

Precision is the ratio of the number of correctly flagged invalid words to the number of words flagged in the text. It is the number of correctly detected error word divided by the total number of detected words in the text by the spell checker. It is the percentage of correctly detected invalid words over all flags bye the spell checker. The general summary of the evaluation metrics is presented in Table 5.2.

Table 5.2: *Summary of Evaluation Metrics*

Metrics	**Description**
Lexical Recall	No of words recognized as valid/No of valid words
Error recall	No of misspelled words flagged/No of misspelled words
Precision	No of words correctly flagged/No of words flagged

5.5.3 Test Results

The main aim of this test is to evaluate the number of correctly accepted inflected and derivated words and correctly flagged out misspelled words by the spell checker system of Kafi Noonoo language. We also evaluate the performance of the spell checker system by using evaluation metrics which are represented in Table 5.2.The result obtained after applying the specified evaluation metrics that are discussed in Section 5.5.2 and computing the percentage is given in Table 5.3.

Table 5. 3: *Evaluation Results of the Experiment*

Metrics	**Result (%)**
Lexical Recall	95.91
Error recall	100
Precision	62.76

5.6 Discussion

From the experiment result as shown in Table 5.3, we obtain 95.91% lexical recall, 100% error recall and 62.76% precision. This shows that our system performs well with some limitations. We identified some of the reason that reduce the performance of the system with linguistics who evaluate our systems output. One of the identified reason is that lexical coverage. The dictionary used is not fully contain all the words of the language.

Words like name of person, place, scientific names are absent from dictionary used. For these reason some of correct words are flagged as invalid word. So including root words of these words in the root dictionary will increase the performance of the system. The second one is affix rule mismatching. There is a probability of applying affix rule to a word of not that affix class which will recognize meaningless words as correct. For example, in verb **class Y**, the suffix -*nee* works well like *ciino* + *nee* = *ciinnee*, but for the root word like *dano*+ *nee* = *dannee* it produce a word that is not part of the language. The other issue to be considered is the morphology of the language because Kafi Noonoo language is one of the richest language in morphology. As the knowledge of the researcher till now there is no morphological analyzer and morphological generator developed for Kafi Noonoo language. Full-fledged morphological analyzer and generator will increase the performance of the spell checker system

So, generally increasing the lexicon coverage of root dictionary and working on the well-organized classification of each root word with correct affix class and having full-fledged affix rule of the language will increase the performance of the system. The result obtained from the value of the evaluation metrics and suggestions from the linguistics is encouraging and satisfactory.

Chapter Six: Conclusion and Future Work

6.1 Conclusion

There are a number of research area in the field of natural language processing tasks for different language in the world. Among these areas, spell checker is one of the study area of NLP application. Spell checking is the task of detecting misspelled words in a document and correcting misspelled words with possible suggestion provided during spell checking process. Spell checker perform two main tasks such detection of error and correction of error.

Different techniques have been proposed to develop spell checker for different languages. Among these, N-gram analysis and dictionary lookup approaches are applied for error detection and edit distance, similarity key, neural network, N-gram, rule-based, probabilistic and noisy channel model are techniques used for error correction.

In developing spell checker for Kafi Noonoo language, this study applied both dictionary look up method and morphological analyzer (Morphology based spell checker) to detect error word and edit distance and rule based approach for error correction. These spell checker is designed and implemented to detect only non-word errors (typographic errors) in the language. Real word errors are not considered in these study which needs semantic analysis of the language and grammar of the language where we put it as a future work.

The developed KNSC can be used as standalone or part of other text processing applications. A user can provide a block of text either manually by typing in space provided in the text editor or by opening though a file. The block of text is divided into individual words by tokenizer and morphological analyzer will analyze each of words by using root and affix dictionaries. Here more than 6000 root words are provided in root dictionary to develop Kafi Noonoo spell checker.

Finally, evaluation and testing is carried out in order to prove the spell checker system using sample data of Kafi Noonoo langaue words randomly collected from papers and books. To evaluate the performance of the system for spell checking, we used 2743 unique words collected from different sources. To check the effectiveness of the system lexical recall, error recall and precision evaluation metrics were conducted. Based on these evaluation metrics we get promising result of 95.91% lexical recall, 100% error recall and 62.76% precision.

Generally we conclude that our morphology based spell checker can perform well still which needs improvement especially more work on morphology of the language.

6.2 Contribution of the Thesis

The main findings of this study is described as follows.

- We built a lexicon for error detection and correction for Kafi Noonoo language for spell checker
- We developed an algorithm that analyze a word morphologically for both error detection and generate suggestions.
- We developed a system that detects error words for Kafi Noonoo language
- We developed a system that provides a list of words a suggestion for correcting misspelled words for Kafi Noonoo language.
- Developed the first spell checker for Kafi Noonoo language that analyze, detects errors and provide list of suggestion words with ranking for error correction.

6.3 Future Work

There are a number of research areas in natural language processing tasks that can be done for local languages. Spell checking is one among these tasks. Spell checker is considered as a base for other NLP applications such part-of-speech tagger, grammar checker, machine translation, question-answering, text to speech synthesis, speech to text synthesis, anaphora resolution, text summarization, dialogue system and etc.

We would like to recommend the following key activities for the future work:

- We recommend that increasing the coverage of the lexicon size of the root word in the root dictionary will increase the performance of the spell checker.
- The proposed spell checker algorithm that is based on morphological analysis will need full-fledged morphological analyzer since in this work we apply our simple morphological analyzer due to there is no morphological analyzer developed yet for Kafi Noonoo language. However, developing an improved Kafi Noonoo language morphological analyzer will solve the issues of spell checker system.
- In this work, we proposed a spell checker that can detect and correct non-word errors of Kafi Noonoo language. One can extend it to work on spell checker that can detect and correct real-word errors by considering semantics and grammar of

the language to make spell checker system more interactive since Kafi Noonoo texts can exhibit real-word errors.

- Extending this work by testing and evaluating with a large corpus that includes different number data sources.
- One of the challenge we faced in this thesis was finding standard corpus of Kafi Noonoo language since the language is resource scarce. Therefore, future study may need to give attention to the development of standard Kafi Noonoo language corpus that can motivate the researchers to work on any NLP application not only on spell checker application and minimize time to collect corpus in order to evaluate their system performance by the researchers.
- For the case of suggesting words with more error distance can be need to be handled since this work considers only operations like addition, insertion, deletion, substitution and transposition of characters for providing suggestions.
- We developed a system as a demo in this work, so one can do as a project to develop full applicable Kafi Noonoo spell checker system that can be directly integrated with other NLP applications such as machine translation, text summarization, etc.

References

[1] Daniel Jurafsky and James H. Martin, Speech and Language Processing:An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition, June, 2007.

[2] Julia Hirschberg and Christopher D. Manning, Advances in Natural Language Processing, vol. 349, Department of Computer Science, Columbia University, New York, 2015.

[3] Tahira Naseem, "A Hybrid Approach for Urdu Spell Checking," Published Master's Thesis , National Univesity of Computer and Emerging Sciences, November, 2004.

[4] Jennifer Pedler, "Computer Correction of Real-Word Spelling Errors in Dyslexic Text," Published Doctorial Thesis, Birkbeck, London University, 2007.

[5] Vibhakti V. Bhaire, Ashiki A. Jadhay, Pradnya A. Pashte and Magdum P.G, "Spell Checker," *International Journal of Scientific and Research Publications,* vol. 5, no. 4, April 2015.

[6] Karen Kukich, "Techniques for Automatically Correcting Words in Text," *ACM Computing Surveys,* vol. 24, no. 4, December 1992.

[7] Renato Cordeiro de Amorim and Marcos Zampieri, "Effective Spell-Checking Methods Using Clustering Algorithms," in *Proceedings of Recent Advances in Natural Language Processing*, Hissar, Bulgaria, September, 2013.

[8] Aye Myat Mon and Thandar Thein, "Myanmar Spell Checker," *In Proceedings of International Journal of Science and Research (IJSR),* vol. 2, no. 1, January 2013.

[9] Tommi A. Pirinen, "Weighhted Finite-State Methods for Spell-Checking and Correction," Department of Modern Langauges, University of Helsinki, Finland, 2014.

[10] Zelalem Mekuria and Yaregal Assabie, "A Hybrid Approach to the Development of Part -of-Speech Tagger for Kafi Noonoo Text," in *Proceedings of the 15th*

International Conference on Intellegent Text Processing and Computational Linguistics(CICLing 2014), Nepal, 2014.

[11] Alejandro Gutman and Beatriz Avanzati , "Omotic Languages," 2013. [Online]. Available: http://www.languagesgulper.com/eng/Omotic.html. [Accessed 3 February 2018].

[12] Simegnih Tekle, "Kafi Noonee Doyee Indee Iiqqeena'o," Bonga, Kaffa, 2016.

[13] Girma Tesfaye, "Structural Description of the Noun Phrase of Kafi Noonoo," 2012.

[14] Aysm Solak and Kemal Of lazer, "Design and Implementation of a Spelling Checker for Turkish," Department of Computer Engineering and Information Science, Bilkent University, Bilkent,ANkara, Turkiye, 1993.

[15] Bakkali Hamza, Gueddah Hicham, Yousfi Abdellah and Belkasmi Mostafa, "For an Independent Spell-Checking System from the Arabic Language Vocabulary," *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 1, University of Mohammed V Souissi Rabat, Morocco, 2014.

[16] Khaled Shaalan, Amin Allam and AbdAllah Gomah, "Towards Automatic Spell Checking for Arabic," *International Jounal of Computer Applications* , vol. 53, no. 3, pp. 12-16, September, 2012.

[17] Jordi Atserias, Maria Fuentes, Rogelio Nazar and Irene Renau, "Spell-Checking in Spanish: The Case of Diacritic Accents," *Journal Article*, Barcelona, Spain, 2009.

[18] Ritu Aggrawal, "Hindi Editor with Spell Checker," Published Master Thesis in Computer Science, Vinayaka Mission University, September, 2007.

[19] Baljeet kaur and Harsharndeep Singh, "Design and Implementation of HiNSPELL-Hindi Spell Checker Uisng Hybrid Approach," *International Journal of Scientific Research and Management(IJSRM)*, vol. 3, no. 2, pp. 2058-2061, Baba Farid College of Engineering and Technology,2015.

[20] Adriane Boyd, "Pronunciation Modeling in Spelling Correction for Writers of English as Foreign Language," in *Proceedings of the NAACL HLT Student Research Workshop and Doctorial Consortium*, Boulder, Colorado, June, 2009.

[21] Daniel Naber and Betreuer, "A Rule-Based Style and Grammer Checker," Published Master Thesis, Diplomarbeit Technische Fakultat, Universitat Bielefeld, 2003.

[22] Johnny Bigert, "Automatic and Unsupervised Methods in Natural Language Processing," Published Doctorial Thesis, Stockholm, Sweden, 2005.

[23] Gaddisa Olani and Dida Midekso, "Design and Implementation of Morphology Based Spell Checker," *International Journal of Scientific and Technology Research*, vol. 3, no. 12, December, 2014.

[24] David Chappell, Introducing Visual Studio 2010, San Francisco, California, 2010.

[25] Mahesh Chand, Programming C# for Beginners, Garnet Valley PA, September, 2014.

[26] Fred J. Damerau, "A Technique for Computer Detection and Correction of Spelling Errors," *Communications of the ACM,* vol. 7, no. 3, March 1964.

[27] Rajashekara Murthy, Vadiraj Madi, Sachin D and Ramakanth Kumar P, "A Non-Word Kannada Spell Checker Using Morphological Analyzer and Dictionary Lookup Method," *International Journal of Engineering Science and Emerging Technologies*, vol. 2, no. 2, pp. 43-52, June, 2012.

[28] Masaaki Nagata, "Context-Based Spelling Correction for Japanese OCR," in *The 16th International Conference on Computational Linguistics*, Coling, 1996.

[29] James L. Peterson, "A Note on Undetected Typing Errors," *Communication of the ACM,* vol. 29, no. 7, pp. 633-6637, July, 1986.

[30] Hema P. H and Sunitha C, "Spell Checker for Non-Word Error Detection: Survey," *International Journal of Advanced Research in Computer science and Software Engineering*, vol. 5, no. 3, March, 2015.

[31] Sumreet Kaur Randhawa and Charanjiv Singh Saroa, "Study of Spell Checking Techniques and available Spell Checkers in Regional Languages: A Survey," *International Journal for Technological Research in Engineering*, vol. 3, no. 2, November, 2014.

[32] Neha Gupta and Pratistha Mathur, "Spell Checking Techniques in NLP: A Survey," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 12, 2012.

[33] Ritika Mishra, Navjot Kaur, "A Survey of Spelling Error Detecion and Correction Techniques," *International Journal of Computer Trends and Technology*, vol. 4, no. 3, 2013.

[34] Mahima Sigh, "Choosing Best Hashing Strategies and Hash Functions," Published Master's Thesis, Master of Engineering in Software Engineering, Thapar University, June, 2009.

[35] Jyotirmayee Rautaray and Raghvendra Kumar, "Hash Based Searching Algorithm," *International Journal of Innovative Research in Science, Engineerign and Technology*, vol. 2, no. 2, February, 2013.

[36] Seung-Shik Kang, "Word Similarity Calculation by Using the Edit Distance Metrics With Constant Normalization," *Journal of Information Processing System*, vol. 11, no. 4, pp. 573-582, December 2015.

[37] William J. Masek and Michael S. Pateson, "Faster Algorithm Computing String Edit Distances," *Journal of Computer and System Sciecne*, 1980.

[38] Eric Brill and Robert C. Moore, "mproved Error Model for Noisy Channel Spelling Correction," in *Proceedings of the 38th Annual Meeting of the ACL*, Hong-Kong, 2000.

[39] Koneru, K., Pulla, V. and Varol, C., "Performance Evaluation of Phonetic Matching Algorithms on English Words and Street Names - Comparison and Correlation," in *In Proceedings of the 5th International Conference on Data Management Technologies and Applications (DATA)*, USA, 2016.

[40] Abebe Asfaw, Kafi Noonoo-Amharic-English Dictionary, Bonga, Kaffa, 2014.

[41] Jirka Hana, Introduction to Linguistics-Morphology, New York, October, 2011.

[42] Greet Booij, The Grammar of Words: An introduction to Linguistic Morphology, New York, USA: Oxford University Inc., 2005.

[43] Tilahun Gebretsadiki, Kafi Noonee Bekkaasho, Iidiyeemo, Keeme Kicoona Shaahiyee Yibbaatoona, Bonga, Kaffa, 2015.

[44] Deepak Seth and Mieczyslaw, "SSCS: A Smart Spell Checker System Implementation Using Adaptive Software Architecture," Northeastern University, Boston, USA, 2003.

[45] Kevin Atkinson, "GNU Aspell," 29 January 2017. [Online]. Available: http://aspell.net. [Accessed 09 June 2018].

[46] Mourad Mars, "Towards a Robust Spell Checker for Arabic text," in *International Conference on Computational Science and its Applications*, 2016.

[47] Neethu Subash, Panchami K S and Ambili T, "Automatic Error Detection and Correction in Malayalam," *International Journal of Science Technology and Engineering*, vol. 3, no. 02, August, 2016.

[48] Irene Thomson, "About World Languages(AWL)," 15 November 2016. [Online]. Available: http://aboutworldlanguages.com/kannada. [Accessed 09 June 2018].

[49] Surayaini Binti Basri, Rayner Alfred and Chin Kim On, "Automatic Spell Checker for Malay Blog," in *IEEE International Conference on Control System*, Computing and Engineering, Penang, Malaysia, November, 2012.

[50] Sani Felix Ayegba, Musa Ugbedeojo, Benson-Iyare Jessica Chinezie and Onoja Abu, "Igala Language Spell Checker," *Current Journal of Applied Science and Technology*, vol. 23, no. 1, August, 2017.

[51] V.Tron, P.Halacsy, P.Rebrus, A.Rung, P.Vajda and E.Simon, "Morphdb.hu: Hungarian Lixical Database and Morhphological Grammar," in *Proceedings of the*

5th International Conference on Language Resourses and Evaluation, Genoa, Italy, January, 2006.

[52] Rishin Haldar and Debajyoti Mukhopadhyay, "Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach," Web Intelligence and Distributed Computing Reseach Lab, India.

[53] Patrizia Paggio, "Validating the TEMAA Evaluation Methodology: a case Study on Danish Spelling Checkers," Cambridge University Press, 1995.

[54] Suzan Verberne, "Context-Sensitive Spell Checking Based on Word Tridram Probabilities," Published Masters Thesis, University of Nijmegen, 2002.

Annexes

Annex A: Kafi Noonoo Digits in Words

Haddoo	Numbers	digits
Hullo	Zero	0
Ikkoo	One	1
Guttoo	Two	2
Keemo	Three	3
Awuddoo	Four	4
Uuchoo	Five	5
Shiritoo	Six	6
Shabatoo	Seven	7
Shimito0	Eight	8
Yixiyoo	Nine	9
Aashiroo	Ten	10
Aaraa Ikko	Eleven	11
.	.	.
.	.	.
Hiyoo	Twenty	20
Shaasho	Thirty	30
Aabo	Forty	40
Aacho	Fifty	50
Shichoo	Sixty	60
Shaboo	Seventy	70
Shinnoo	Eighty	80
Yixanoo	Ninety	90
Ballo	Hundred	100
Humo	Thousand	1000

Annex B: Kafi Noonoo Verb Inflection for *daamo*(take)

daamo (take) verb inflection		
1. daamo	35. daammiyaachemina'o	68. daamaanbeetin
2. daamme	36. daammiyeemina'one	69. daamaanbeetote
3. daamiyeete	37. daammiyaachemina'one	70. daamaanbeete
4. daammehan	38. daammaataane	71. daamaanbeetete
5. daamiye	39. daammaanoone	72. daamaabeet
6. daammehin	40. daamaaneene	73. daamaabeeton
7. daammehon	41. daamiitoote	74. daamaabeetin
8. daammiiyo	42. daamite	75. daamiibeetote
9. daamo	43. daamatan	76. daamiibeete
10. daambon	44. daammiteete	77. daamaabeetan
11. daamb	45. daamaane	78. daamiibeetete
12. daambot	46. daamaano	79. daammiiyaach
13. daambe	47. daammetaane	80. daammiiyaachon
14. daambu	48. daammenoone	81. daammiiyaachin
15. daammaataane	49. daammiineene	82. daammiiyaachoote
16. daamaantaane	50. daammiitootee	83. daammiiyaache
17. daamaanoone	51. daammiite	84. daammiiyaachan
18. daammaneene	52. daammiitan	85. daammiiyaacheete
19. daamaanene	53. daammiitete	86. daammiiyaanbeet
20. daammiyoote	54. daammiiteete	87. daammiiyaanbeeton
21. daambeete	55. daammemina'one	88. daammiiyaanbeetin
22. daamiyooba	56. daamaach	89. daammiiyaanbeetote
23. daammemina'o	57. daamaachon	90. daammiiyaanbeete
24. daammiiyeemina'o	58. daamaachin	91. daammiiyaanbeetan
25. daammemina'oche	59. daamaachote	92. daammiiyaanbeetete
26. daammiiyemina'oche	60. daamaache	93. daamiyooba
27. daammemo	61. daamaachan	94. daamigaata
28. daammemi	62. daamaacheete	95. daammiibeegoora
29. daammemina'o	63. daamaanbeetan	
30. daammemina'one	64. daamaanbeetaane	
31. daammiyeemo	65. daamaanbeenoone	
32. daammiyeemi	66. daamaanbeet	
33. daammiyaachemo	67. daamaanbeeton	
34. daammiyaachemi		

Annex C: Sample Kafi Noonoo Root Word classification

Noun Classification

Class C: nouns ending with a single vowel **o.**

For example, nouns ending with a single vowel such as **asho, kexo, aabo, gato** can be grouped in this class and can take suffixes like **ich, eena'o, eena'one, eena'ochoommon, ochee ittino, ittinoch, iwaan, echo, icho, ichoommon.**

Class D: Nouns that ends with single vowel **o** and consonants like **x** such as **mixo, kexo,** maxo can be grouped in this class. In this class suffixes like **ina'o**, **ina'ooch**, **iwaan, yich, ina,**

Class F: nouns that ends with double **oo s**such as **baakkoo, gotoo, iboo, manoo** are noun words of this class and suffixes **chee, n, na, naa, aallo, eena'o, eena'oon**can be applied to this class of words.

Class G: Verbs that can end with a single **o** such **gamo, coodo, xeebo** are verbs of this class and suffixes such **as iye, e, ee, ii, iichoo, iyo, eteete,iteete, ito, ech, eyaano, iibeeto, eyaache, iti, ibon, ebon, eba, iba, ehe, eya, emmooch, eemmi, iqqi, iiree, iyemmo, iyemmi, immina'one o echina'one, itoye** can be applied to this class of verbs.

Class H: verbs that ends with double **o** such **daakkoo cuuqqoo, yaggoo, xefoo** are verbs that grouped in this class and suffixes such as **eebe, eyaanoone, echo, iyecho, ite, isho, eeta, eetoomon, itooch, iyeete, ee'i, iniye, emmo** can be applied to this class of verbs.

Class X: verbs that ends with single vowel **o** and preceding with consonants like **c, f, g, h, y**. For example, verbs such as **dico, ufo, hago, buho, keyo, shuu'o** are grouped in this class. Suffixes such as **jjiye, jjite, chiyeete, ppiye, ppihe, qqiye, qqiyeete, kkiye, kkiyoo** are suffixes applied to this class of verbs.

Class Y: verbs that ends with the consonant **n** vowel **o** .Verbs like **ciino, dano, gano** are of this class and can take suffixes like **niti, no, neton, niti, nna, nnee, nneecho**.

Annex D: Sample Kafi Noonoo Affix Rules in the Affix File

//class A words that ends with consonants

A Y 6

A 0 a [bcdfghmntxych]

A 0 aa [bcdfghmntxych]

A 0 e [bcdfghmntxych]

A 0 ee [bcdfghmntxyshch]

A 0 i [bcdfghmntxyshch]

A 0 ii [bcdfghmntxyshch]

//Class B words that ends with vowels (a, e,i,o,u)

B Y 8

B a e a

B a ee a

B a i a

B a ii a

B a u a

B a uu a

B e a e

B e aa e

//Class X verbs that ends with the last syllable consonants and vowels (yo, ho, fo ‘o, go)

X Y 11

X yo jjiye yo

X yo jjiyee yo

X yo jjite yo

X yo jjiteete yo

X yo jjiibeeteete yo

X yo chiyeete yo

X yo chiibeete yo

X fo ppiye fo

X fo ppiheete fo

X ho kkiye ho

X go kkiyoo go

Annex E: Dictionary File

Sample root words with their affix classes

aabaallo/BG	aabbeeno/BD	yuye/BD
aabboo/CF	aabdoomoo/CF	qomo/BGWV
aabichoo/CH	aabo/BD	baaxaano/BG
baaxaanoo/CF	baaxo/BG	baayo/BGX
baayoo/CH	babbero/BG	babberoo/CH
baccero/BG	baccoo/CH	bachiyoo/CH
bacho/BG	bachoo/CH	beemmo/BG
bacoo/CH	badd/B	baddilloo/CH
baddo/BG	beello/BG	beemo/BDW
beeno/BG	beenzino/BD	beeroo/CF
beeshoo/CF	beetto/BG	beexxoo/CH
beggeeshoo/CH	callo/BG	calloo/BD
cambixo/BG	camiyoo/CH	cammo/BD
camo/BGW	cando/BD	caneto/BG
cango/BGX	cano/BD	ca'o/BGX
capho/BD	cappelloo/CH	cappo/BG
caqiro/BD	caraabboo/CH	carcakko/BD
caroo/CF	cattelloo/CH	shawukko/BD
shaxoo/CF	shaxxoo/CH	shayo/BGX
sheboo/CH	shecco/BD	sheddo/BD
sheebo/BG	sheecho/BD	sheechoo/CH
sheefitto/BD	sheefo/BGX	sheefoo/CF
sheekkoo/CF	sheellero/BG	sheelloo/CF
sheemmo/BD	sheendoo/CH	sheeno/BD
shee'o/BGX	sheepphoo/CH	sheeroo/CF
yubboo/CF	yubo/BG	yucaato/BG
yucucco/BG	yuddo/BD	yudo/BG
yufuppo/BG	yuggo/BG	yugiro/BG
yugo/BGX	yukko/BG	yukoo/CH
yullo/BG	yumbaa'o/BD	yumoo/CH
yundo/BD	yunjoo/CF	yuppuppo/BD
yureto/BG	yuto/BD	Yuttimo/BD
yutto/BG	yuucaaco/BGV	yuucco/BD
yuuche/BD	yuuco/BGV	yuudo/BG
yuudoo/CF	yuufoo/CH	yuukko/BG
yuulloo/CH	yuummero/BD	yuumo/BGW
yuuniiformo/BD	yuunjoo/CH	yuunno/BG
yuunnoo/CH	yuurimo/BGW	yuuroo/CF

Annex F: Sample misspelled words detected during testing by spell checker

Misspelled word	Correct Word	Meaning
Gittoo	Guttoo	Two
Tunoo	Tuno	Happen
Tunooniye	Tunoniye	The happened one
Bireetoomon	Bireetoommon	As briefly explained
Kettoonna	Kettoona	Easily
Biritinoona	Birittinoona	As bright
Baribarittinooniye	Bari barittinooniye	As they are different
Worafe	Woraafe	Regional
Ogishoo	Oogishoo	Big with respect to small
Goyoo	Goyo	Plough
Ikkoonoomiyoo	Ikoonoomiyoo	Economy is
Nooonoona	Noonoona	By language
Uuchinoochee	Uuchinnoochee	From fifth
Abre	Bare	Different
Yweena'o	Yaweena'o	Methods
Shiijjiroochee	Shiijjeroochee	From generation
Beddaha	Beddaaha	Up to
Gettteebe	Getteebe	Was said
Kata	Kaata	Be quick
Watta	Wotta	Again
Naotoyee	Natooyee	After a year
Ammoonona	Amoonona	Anything
Qajjiii	Qajjii	Ignored
Tiijjigaata	Tiijjiigaata	If he picked up
Shunnetane	Shunnetaane	I like it
Xebee	Xebbee	Narrow
Safiro	Shafiro	
Aabchibona	Aabichibona	What can we do
Maaacooch	Maacooch	Inside stomach
Girttino	Girittino	Poverty
Kaameelles	Kaameellee	Car's
dubiyoonaa	dubbiyoonaa	By wereda

Declaration

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

Declared by:

Name: Fikru Tafesse Bekele

Signature: _____

Date: _____

Confirmed by advisor:

Name: Yaregal Assabie(PhD)

Signature: _____

Date: _____