DEVELOPMENT OF MODELS FOR NON-WORD ERROR DETECTION AND CORRECTION SYSTEM FOR KISWAHILI LANGUAGE

Mulingi Yono

REG N0: 2020-01-00855

Dr Robert Ssali Balagadde

Supervisor

a project submitted to the faculty of information technology in fulfillment of the requirements for the award of the degree of master of information system of kampala international university

NOVEMBER 2023

DECLARATION

I **Mulingi Yono** declare that I am the author of this project and that any assistance I received in its preparation is fully acknowledged and disclosed in the report. I have also acknowledged any sources from which I used data, ideas or words, either quoted directly or paraphrased. I also certify that this research report was prepared by me specifically for the partial fulfillment for the degree of Masters in Information System at Kampala International University.

Signature: …… Date: ……/……/……

Name:……

Reg No: ……...

APPROVAL

Having gone through the content and structure of this work, I therefore certify that the work; "Development of Models for Non-Word Error Detection and Correction System for Kiswahili Language" is original and has been done under my close supervision and is now ready for submission for examination, with my approval.

Signature: ……

Dr. Balagadde Robert Ssali

Supervisor

Date: ……

ACKNOWLEDGEMENTS

I wish to acknowledge with gratitude, the extensive support I received in preparation of this proposal. First and foremost, my sincere acknowledgement and gratitude goes to the God the Almighty who provided me with wisdom and general good health during the course of this work.

Secondly, I appreciate the support, kind direction, mentoring, academic advice and constant encouragement of the academic supervisor,

Thirdly, my heartfelt appreciation goes to my family: my Dad, Deograthias Bashibirhana Mulingi; my Mom, Alphonsine Bahati Bukasa; my fiancé, Lina Bisimwa; my brothers: Ronan Mudekereza, Jean de la grace Bashi, Guy Audin Bashi; my sisters: Cito Sungura, Rolande Cinama, Leslie Bashi, Jennifer Nshombo, Sylvia Imani; my nieces: Aurora and Auriel, for their constant encouragement.

Fourthly, I am obliged to all Management Information Systems (MIS) facilitators and peers in class, for their peer-to-peer assistance, inspiration, encouragement and cooperation.

Finally, in special way I like to record my profound and deep appreciation to my friends Peter; Allen; my team members from RevarTech Ltd: Adrien Katamya, Ghislain Dunia, Joas madragule, and Herver Bashige; for their full support to reach this far.

ABSTRACT

The development of effective spellcheckers for Kiswahili, such as the JAMBO SPELL CHECKER (2004), has marked significant progress; however, a noticeable research gap persists in the field of non-word error detection and correction systems tailored specifically for the Kiswahili language. This study addresses this gap by proposing and implementing enhanced models, SwaDetect and SwaCorrect, designed to adeptly detect and rectify non-word errors in Kiswahili. With a focus on addressing limitations in scope, speed, and accuracy prevalent in existing solutions, our research aims to pioneer a more comprehensive and efficient system for non-word error detection and correction in Kiswahili. Notably, experimental results demonstrate SwaDetect's exceptional accuracy of 99% in Kiswahili word detection, operating at a processing speed of 65 Hz (65 words per second), while SwaCorrect proficiently corrects erroneous words with an average accuracy of 82% for Edit Distance One (ED1) through ED3, maintaining an overall correction speed of 58 Hz (58 words per second). Our study encapsulates the development of models crucial for advancing the accuracy and speed of non-word error detection and correction systems dedicated to the Kiswahili language.

Table of Contents

ABBREVIATIONS AND ACRONYMS

ANC: **American National Corpus**

BERT: **Bidirectional Encoder Representations from Transformers**

BNC: **British National Corpus**

CCL : **Correction candidates list**

CCL: **Correction Candidate List**

CCs: **Correct Candidates**

DLD : **Damerau Levenshtein Distance**

DLT: **Dictionary Lookup Technique**

DSRM: **Design Science Research Methodology**

ECM: **Error Correction module**

ED: **Edit Distance**

EDM: **Error Detection Mechanism**

FN: **False Negatives**

FP: **False Positives**

HCS: **Helsinki Corpus of Swahili**

JC: **Jaccard Coefficient**

KIU : **Kampala International University**

MIS: **Masters in Information System**

ML: **Machine Learning**

NLP: **Natural Language Processing**

OAU: **Organization of African Unity**

SALAMA: **Swahili Language Manager**

SDG: **Sustainable Development Goals**

TDM: **Type Detection Module**

TN: **True Negatives**

TP: **True positives**

UN: **United Nation**

USA: **United Stade of America**

LIST OF TABLES

LIST OF FIGURES

1 CHAPTER ONE: INTRODUCTION

The impact of information technology tools on human life is undeniable, as they greatly simplify and accelerate task completion while reducing manual efforts. One area where this impact is particularly notable is Natural Language Processing (Adeyanju&Adewole, 2013). NLP, sometimes referred as Computational Linguistics in the Arts domain, is a subfield of Artificial Intelligence, which in turn is a subfield of Computer Science. NLP focuses on enabling computers to read, understand, and extract meaning from Human or Natural Languages. Over the years, one of the aspects in which NLP has been involved is Spelling Error Detection and Correction, which dates to the 1960s. Now, spelling and lexical checking components are integrated into many applications, such as word processing software, email and web browsers, among others, making them a vital writing aiding tool. Spell checkers are computer applications that identify misspellings in text by using dictionary lookup technique, whose dictionary or lexicon comprises of accepted words in the language in question. Numerous commercial and non-commercial spell checkers and correctors, including Microsoft Spell Checker, Unix Spell, GNU's Spell, as well as A Spell and their variants, have been developed, however, there is hardly any such advanced tool or system for the Bantu Languages, such as Kiswahili, which is a widely spoken language in East Africa by hundred and thirsty million (130,000) people over the world. The available spell checker for Kiswahili (JAMBO SPELL CHECKER,2004) are associated with low speed, and low accuracies and therefore, the need for an improved version, and hence, the advent of this research works.

This research work aligns well with the United Nation (UN) Sustainable Development Goals (SDG). SDG 4.6 stipulates that: “by 2030, all youth and a substantial proportion of adults, both men and women, achieve literacy and numeracy”. This research work aims at enhancing the accuracy of written Kiswahili text through the use of the proposed writing aid, which in term can be used in promoting literacy and thereby support the realization of the aforementioned SDG.

Another SDG is 9.c, which aims at significantly increasing access to information and communications technology and striving to provide universal and affordable access to the Internet in least developed countries. The output of this research work is an artifact which can

be used to process information by correcting non-word errors in Kiswahili text. This research work is part of the effort to computerize or make the computer to comprehend or understand our local natural languages.

1.1 Problem Statement

In an ideal situation, Kiswahili language writers would create accurate and error-free documents. However, due to human error, inaccurately written documents in Kiswahili document archives remain a significant challenge, posing a risk to the language's global recognition. Existing solutions like the JAMBO SPELL CHECKER (2004) have limited scope, speed, and accuracy due to their erroneous dictionary, which contains misspelled words (like "waliyofundis" instead of "waliyofundisha", "waliwasimami" instead of "waliwasimamia", among other). The dictionary was created using news papers from the internet, which source contain misspelled words and named entities - for instance, Chiligati" and "Ishengoma," - that should not be part of the dictionary.The need for accurate and error-free documents in kiswahili cannot be overlooked, as this would improve communication, education, and future research works in the language.

1.2 Research Objectives

1.2.1 Main Objective

The aim of this research is to create an automated non-word error detection and correction system that can accurately identify and correct non-word errors in kiswahili text.

1.2.2 *Specific Objectives*

1) Study the orthography of Kiswahili language.
2) Review the literature on the techniques used in non-word error detection and correction systems as well as similar systems.
3) Design a novel non-word error detection and correction system for Kiswahili language.
4) Develop the detection and correction system.
5) Conduct performance evaluations of the new non-word error detection and correction system

1.3 Research Questions

1) What are the structural and linguistic peculiarities of the Kiswahili language that need to be considered for developing a non-word error detection and correction spellchecker?
2) What are the different techniques used in non-word error detection and correction systems, and how do they compare to similar systems?
3) What techniques and algorithms can be used to design and develop an efficient non-word error detection and correction spellchecker for Kiswahili language?
4) How can the newly created spellchecking system be implemented and integrated with other software and applications?
5) How effective and accurate is the new spellchecking system in identifying and correcting non-word errors in Kiswahili language texts, and how does it compare to existing spell checkers?

1.4 Significance of the Study

Technology is a rapidly evolving field, and the need to improve existing computer tools to enhance communication among people is crucial. As the number of kiswahili language users continues to grow, there is an increasing demand for applications for processing Kiswahili language.

This study aims to contribute significantly to:

1. The field of computational linguistics by developing an NLP System for kiswahili which can detect and correct non-word errors in Kiswahili text. The NLP System could be used as a model for developing similar systems for other under-resourced languages.

2. The development of other NLP tools such as grammar checkers, language translators, question-answering systems, kiswahili language compilers, optical character recognition systems, and more (Oyieke, 2020).

3. Improving the quality of kiswahili texts by using the proposed system as a writing aid which can automatically detect and correct non-word errors in text.

4. Supporting non-native speakers in their bid to learn kiswahili through writing correct kiswahili text. This issue aligns with Uganda's Vision 2040, which prioritizes the development of a knowledgeable society that can communicate effectively in diverse languages (National Planning Authority, 2013).

1.5 Scope of the Study

1.5.1 Geographical Scope:

The study will primarily focus on Kiswahili language text written in East Africa, specifically in countries where Kiswahili is widely spoken, such as Tanzania, Kenya, Uganda, Rwanda, Burundi, and the Democratic Republic of the Congo. However, the developed spellchecker can potentially be used for Kiswahili text written anywhere in the world.

1.5.2 Content Scope:

The study will focus on non-word error detection and correction for Kiswahili language text. Non-word errors refer to spelling mistakes where the misspelled word does not exist in the language's lexicon (Balagadde & Premchand, 2016 b, Jiang, Yang, & Rio 2019). The study will utilize the edit dictionary lookup and Jaccard Coefficient technique to identify and correct non-word errors in Kiswahili language text.

1.5.3 Time Scope:

The study will be conducted within a period of 12 months from the date of approval. The development and implementation of the spellchecker will be done in phases, and each phase will have specific milestones and deadlines to ensure timely completion of the study. The evaluation of the spellchecker's performance will also be conducted within this time frame.

1.6 Dissertation Structure

The dissertation consists of five chapters.

Chapter 1 provides an introduction to the study, including the problem statement, research objectives, research questions, scope, and limitations.

Chapter 2 examines the literature review. Various techniques used in non-word error detection and correction systems as well as spell checkers for other languages have been surveyed. In addition, various theories and concepts on spell checkers and related research work on different languages and associated components, such as corpora have been presented and discussed.

Chapter 3 presents Research Methodology which is the blue print for conducting this research work. The major methodology used in this research work is the "Research Science Methodology", since an artifact is the main output of this research.

Chapter 4 presents background of kiswahili language since this the application domain of the proposed system.

Chapter 5 entitled "System Design Implementation and Evaluation" presents the process of developing the design system and implementing it. The final system is evaluated through experimentation and results obtained are analysed and discussed.

Chapter 6 entitled "Conclusion and Future Work" presents conclusion emanating from the evidence adduced in research work as well as suggestions for future works.

2 CHAPTER TWO: LITTERATURE REVIEW

2.1 Definitions of Terms

2.1.1 Linguistics Concepts

Linguistics is a field of study concerned with language research and development. As this research focuses on language spell checking, it is important to borrow and use some concepts from linguistics. Morphology is one of the most important concepts in linguistics, which refers to the study of the internal structure of words and the systematic form to meaning correspondence of words. Morphology deals with the ways in which words (lexemes) are formed and with spelling the appropriate form of a lexeme in a particular syntactic context. Some words can be split into smaller units that can also have their own meanings. Hence, a term called morpheme is defined as the minimal linguistic unit with a lexical or grammatical meaning. For example, in the word "Unakula", we have the prefix “U” that represents the second person singular (you), “Na” is an infix that indicates the present tense, and “kula” is the root word, and it means "to eat." Kiswahili is an agglutinative language which means that its words can be formed by a combination of different morphemes, where these morphemes are not modified in spelling or phonetics prior to their use in any word, (Prosz´eky and Kis, 1999).

2.1.2 Text Corpus

An electronic text corpus, which is a computer-readable format of a large collection of language data in written form, is a valuable tool for obtaining statistical information about a language (Manning and Schutze, 1999). Corpora, which are bodies of text that consist of several collections of texts, are available for almost every language. Some of the most well-known corpora include the Helsinki Corpus of Kiswahili (HCS) (Hurskainen, 2004), the American National Corpus (ANC) for American English (Smith.j.A,2005), the British National Corpus (BNC) for British English (Smith, E. R. 2010), among others.

The HCS is a result of the SALAMA (Kiswahili Language Manager) project initiated by Hurskainen, (Hurskainen, 2004), and it currently contains over twelve million standard Kiswahili words. The system uses a two-level morphology approach to collect terminologies

from the internet and lists of terminology coined by the National Kiswahili Council of Tanzania. These corpora are essential for developing Natural Language Processing (NLP) applications.

2.1.3 NLP (Natural Language Processing)

The ability of computers to learn, understand, and interpret natural languages is known as Natural Language Processing (NLP), (Collobert, R., Weston,2011).

In human communication, language is used as a means to convey information. The term language can be applied to either natural language or computer language. Natural languages are the languages spoken by humans, such as English, Japanese, French, and Kiswahili while, on the other hand, formal or computer languages have rigid structures defined by a set of rules, such as Java, C++, and Python. Natural languages are highly flexible, which means that they cannot be characterized as a definitive set of sentences, and they are challenging to work with because their rules are not as precise as those of formal languages.

2.1.4 Spell Checker Understanding

The task of a spell checker is to identify incorrectly spelt words and suggest alternative words with correct spellings. A number of techniques and algorithms have been developed in this regards.

Kukich (Kukich, 1992) has classified spelling errors into two categories: lexical (Non-word) errors and grammatical errors. Lexical errors, also known as usage errors, occur when a word is misspelled or is not a valid orthographic form of the language. Grammatical errors, on the other hand, are morphosyntactic errors that involve the combination of words or their grammatical modification, such as errors in conjugation or declension. The spell-checking process can be divided into two stages: detection and correction for both lexical and grammatical errors.

2.2 Non-Word Error Detection

Non-word error detection is the first step in before suggestions are generated to the erroneous word. Many authors have conducted research in this area. (Hládek et al. 2020);

2.2.1 Dictionary Lookup Technique

Dictionary Lookup Technique involves comparing words against a dictionary to identify misspelled words. This approach assumes that correctly spelled words are captured in the dictionary. Non-existent words in the dictionary are considered potential errors and are flagged by the system. However, this approach may not be effective for detecting real word errors according to Kukich (Kukich,1992).

2.2.2 Bi-gram Matrix Technique

This method employs a non-positional bi-gram matrix, sized at 26 by 26, designed to capture the presence or absence of specific bi-grams. In this approach, each bi-gram is assigned a binary value: one if it is found within the language corpus under consideration, and zero if not. The input text is then scrutinized to verify the existence of its constituent bi-grams within the matrix. If any bi-gram is found to be absent, the word is marked as a non-word. This technique has demonstrated its effectiveness in identifying errors arising from Optical Character Recognition (OCR) processes, although it is generally less precise when it comes to detecting errors introduced by human authors.

2.2.3 Pattern Matching Technique

In this approach, predefined patterns or rules are used to identify non-word errors. These patterns can be based on common misspellings, typographical errors, or specific error patterns observed in the language.

The system compares each word in the text against the patterns and flags those that match as potential non-word errors.

Pattern matching can be effective for detecting errors that follow specific patterns, such as repeated letters, transposed letters, or common typographical mistakes. (G.W. Milligan and P.D. Isaac. 1980)

2.3 Non-Word Error Correction

In a spelling correction process, after detecting errors, the next stage is to generate words that could be the correct form of the misspelled word. According to Kukich (Kukich,1992), the correction process consists of two steps: the generation of words that are likely to be the correct spelling, and the ranking of those generated words. Here are a few ways to archive this.

2.3.1 Lexicographical distance

The lexicographical distance calculates the minimum number of inserting, deleting, transposition, and substituting letters necessary to transform one word into another (Vienney, 2004). The Levenstein distance is the most popular lexicographical distance, also known as the "edit distance." This metric calculates the edit distance by considering four string operations: substitution, insertion, transposition, and deletion (Levenstein, 1966). These operations are described below:

- **Insertion:** This algorithm corrects erroneous words with a missing character. The principle of this algorithm is to insert a letter of the alphabet at the position where the error occurred and check if the resulting word is correct before adding it to the list of candidate words. The process is repeated for all the letters of the alphabet.
- **Deletion:** This algorithm removes characters from a wrong word. The algorithm removes a character from the word and checks if the resulting word is correct before adding it to the suggestion list. The process is repeated for all the letters of the word.
- **Substitution:** The substitution algorithm replaces each letter of the word, one after the other, with a letter of the alphabet and checks if the resulting word is correct before adding it to the suggestion list. The same process is repeated for all the letters of the alphabet.
- **Transposition:** This algorithm changes the position of one character of the word by putting it in all the other positions, and each time, it checks if the resulting word is correct before adding it to the suggestion list. The process is repeated for all the letters of the word.

2.3.2 The Burkhard-Keller Tree (BK-Tree)

The Burkhard-Keller Tree, commonly referred to as the BK-Tree, stands as a tree-based data structure designed for locating near-matches to a given string query. This innovative data structure was first introduced by Walter A. Burkhard and Robert M. Keller in their seminal paper titled "Some Approaches to Best Match File Searching" (Burkhard and Keller, 1973).

Over the years, the BK-Tree has found widespread application in various text processing tasks, particularly in spell checking algorithms.

The BK-Tree algorithm leverages two fundamental concepts, namely the Levenshtein distance and the Triangular Inequality, to drastically reduce the time complexity of string-matching operations. This optimization has made it a key player in the realm of string comparison and retrieval.

One of the significant advantages of the BK-Tree lies in its role in implementing the auto-complete feature in numerous software and web applications. Its efficiency and ability to handle large datasets have contributed to its popularity in enhancing user experience and productivity.

In a Burkhard-Keller Tree, any node can be chosen as the root node, and it may possess zero or multiple sub-trees. Each node within the structure represents individual words from a dictionary. Importantly, the total number of nodes in the BK-Tree corresponds to the number of words inserted into the dictionary.

The edges connecting the nodes in the BK-Tree are assigned integer values based on the Levenshtein distance, which adheres to the following three criteria:

1. Exact Match: If the Levenshtein distance between two words, X and Y, is zero, then X is identical to Y.

2. Symmetry: The distance from X to Y is equal to the distance from Y to X, ensuring symmetry in the distance calculation.

3. Triangle Inequality: The path from one point to another should never be longer than any route that passes through any intermediary point. This principle enhances the efficiency of the BK-Tree in near-match retrieval.

2.3.3 Jaccard Coefficient Technique

The Jaccard coefficient, also known as the Jaccard similarity coefficient or Jaccard index, is a statistical measure used to assess the similarity between two sets. In the context of Natural Language Processing (NLP), the Jaccard coefficient technique can be applied to compare and quantify the similarity between sets of words or documents. Here's a brief description of the technique: The Jaccard coefficient technique is useful in various NLP applications, such as text

clustering, information retrieval, and document similarity analysis. It allows for a quantitative assessment of the similarity between sets of words or documents based on the shared elements and the totally unique elements present.

a) Set Representation

The Jaccard coefficient technique represents each set of words or documents as a collection of unique elements, disregarding the order or frequency of occurrence. For example, given two sets of words, each set is represented as a collection of distinct words contained within it.

b) Calculation of Intersection and Union

The Jaccard coefficient calculates the similarity between sets by comparing the intersection and union of the elements within the sets. The intersection represents the common elements shared by both sets, while the union represents the total unique elements present in both sets.

c) Jaccard Coefficient Calculation:

The Jaccard coefficient is computed as the ratio of the size of the intersection to the size of the union of the two sets. It is calculated using equation 2.1:

$$J(A, B) = |A \cap B| / |A \cup B| \quad (2.1)$$

where A and B are the sets being compared.

d) Interpretation of Jaccard Coefficient

The resulting Jaccard coefficient ranges from 0 to 1, with 0 indicating no similarity and 1 indicating complete similarity. A higher Jaccard coefficient implies a greater degree of similarity between the sets being compared.

2.3.4 Alpha-code

To identify the most suitable replacements for an unknown word, most spell checkers rely on a method known as alpha-coding, which involves encoding the word into a specific string representation. This alpha-code is essentially a string of characters formed by arranging all the

letters of the word in alphabetical order. The process of alpha-coding varies based on different computation methods.

Ndiaye and Faltin (2003) proposed a method that involves sorting the consonants followed by the vowels in alphabetical order to create the alpha-code for a word. This results in a unique alpha-code for each word, and the set of words associated with a particular alpha-code is referred to as the class of that alpha-code. The correction system maintains a set of alpha-codes corresponding to the lexicon, and for each alpha-code, it stores the class of words associated with it.

Pollock and Zamora (1984) introduced a model that associates each word in the dictionary with its alpha-code, which consists of the consonants making up the word. This method necessitates the use of two dictionaries: one for the words themselves and another for their corresponding alpha-codes. Correction is achieved by comparing the alpha-codes of candidate words with the alpha-code of the erroneous word, making it effective for correcting permutation errors.

When a misspelled word is encountered, the correction process begins by establishing the alpha-code of the incorrect word. If this alpha-code is found in the dictionary, the system retains the words associated with that alpha-code as potential candidates for correction. In cases where the alpha-code is not found in the dictionary, the system iteratively modifies the alpha-code by adding one or two characters to it and checks if the newly generated alpha-code exists in the lexicon. This process is repeated by both adding and removing characters from the alpha-code.

The resulting candidate words are then subjected to a proximity calculation with the incorrectly spelled word, enabling the system to rank these potential corrections. It's worth noting that while each word has a unique alpha-code, multiple words may share the same alpha-code. This feature allows a spell checker to search for words that share a similar alpha-code with the unknown word, enhancing its ability to identify suitable replacements.

2.3.5 Machine Learning Approaches for Non-Word Error Correction

Machine learning approaches offer the advantage of learning from data and adapting to various error patterns. These models can capture complex linguistic patterns and context-specific information, resulting in accurate non-word error correction. Here is an example;

Unsupervised Learning: refers to using machine learning techniques to automatically identify and correct errors in text without the need for labeled training data that explicitly specifies the errors and their corrections. This approach relies on the model to learn patterns and anomalies in the text data on its own, making it a valuable tool for addressing a wide range of errors beyond just misspelled words.

- Start by collecting a large corpus of text data that may contain various types of errors, such as grammatical mistakes, punctuation errors, stylistic inconsistencies, and more.
- Convert the raw text data into a numerical representation suitable for machine learning. Common techniques include tokenization (splitting text into words or phrases), vectorization (converting words or phrases into numerical vectors), and extracting linguistic features (e.g., part-of-speech tags, syntactic structures).
- Apply unsupervised clustering algorithms, such as k-means clustering, hierarchical clustering, or DBSCAN, to group similar text segments together. The goal is to create clusters of text that share common characteristics, which may include similar types of errors. (R.C. de Amorim. 2012.)
- Once errors are detected within a cluster, apply correction strategies to fix them. Correction strategies could be rule-based (e.g., replacing misplaced punctuation marks), NLP-based (e.g., suggesting alternative phrases using language models), or a combination of both.

2.4 Related Works

In the field of natural language and computational linguistics, several studies have been conducted by researchers across different regions. One area of focus has been the automation of spell-checking tools, with various applications available on the web, text editors, word processors, and other internet platforms.

2.4.1 Kîmîîrû language

For instance, Anondo, Timothy Kimathi (Anondo,2013) propose an open-source spellchecker for Kîmîîrû language using the Hunspell language tools which examines the morphological analysis of Kîmîîrû language, highlighting nouns and verbs derivation and also provides a

suggestion component used to generate probable suggestions for a misspelled word. He used dictionary look up technique for detection, and according to him this approach was selected for the following two main reasons:

a) This method was vastly more accurate than those based on independent spell-checking methods

b) In the future the application would be extended to include a grammar checking tool. This would require not only storage of a word list but also part-of-speech information for each word.

The developed spellchecker is the first spellchecker for Kîmîîrû language and it can correctly classify Kîmîîrû words with an accuracy rate of 80%, precision rate of 100% and a recall rate of 78%.

2.4.2 Gĩkũyũ Language

The dictionary was created from a wordlist compiled and saved as a text document in UTF-8 format. This format was necessary as the Gĩkũyũ language contains vowels that use the tilde as a diacritic. They used Hunspell language tool to generate CCL, as indicated in his research paper. (Ng'ang'a, WKamau, C,2010). The suggestions component was used to generate probable suggestions for a misspelled word. It was implemented in the affix file. Hunspell used two sections in the affix file when generating suggestions for misspelled words. The first is the TRY command. This listed the language's orthography set in order of frequency. A more frequently used character has more weight during suggestions. The second command used in the suggestion component is the REPLACE command.

When tested on a test corpus, the spell checker attains a precision of 82%, recall of 84% and an accuracy of 75%.

2.4.3 Lulogoli language

Another relevant study was conducted by Aseyo and John Orege (Aseyo&John, 2011), who developed a spell checker for the Lulogoli language using the Hunspell spell checker engine. Their work involved constructing a word list based on word roots found in hymn books, story

books, and spoken language. They addressed the recognition of morphologically complex words, common in Lulogoli due to its agglutinative nature, by leveraging an affix file built on pre-segmented word derivations from the corpus.

2.4.4 Luganda Language

In the context of Luganda, Robert Balagadde and P. Premchand (2016) published work on non-word error detection and correction. Their system utilized a detection module employing the dictionary lookup technique. They also employed the Jaccard Coefficient (JC) as it showed good correlation with the Damerau Levenshtein Distance (DLD) and had a computationally efficient linear algorithm. The Jaccard Coefficient was used for the selection and ranking of the Correction Candidate List (CCL).

Here are additional examples of studies on non-word error detection and correction systems for different languages:

2.4.5 Amharic language:

Tessema, M., & Abebe, B. T. (Tessema&Abebe,2014) developed a spell-checking system for Amharic that utilized a statistical approach based on n-gram language models. They focused on detecting and correcting non-word errors in Amharic text by considering the context and frequency of words in a given corpus.

2.4.6 Hindi language

Kumar, A., & Singhal, S. (Kumar&Singhal,2016) proposed a spell-checking system for Hindi that combined rule-based and statistical approaches. They used linguistic rules specific to the Hindi language for non-word error detection and employed a statistical model based on the Jaccard coefficient to rank and suggest corrections for misspelled words.

2.4.7 Chinese language

Li, Y., & Li, X. (Li, Y. & Li, X,2011) developed a spell-checking system for Chinese that incorporated a language model and phonetic similarity measures. They focused on detecting

and correcting non-word errors by considering the phonetic similarity and contextual information of Chinese characters in a given corpus.

These examples demonstrate a variety of approaches used in different languages to detect and correct non-word errors. Each study employed specific techniques and methodologies tailored to the characteristics of the target language.

2.4.8 Kiswahili spell checker

The existing Kiswahili spell checker was released in December 2004. The spell checker used a dictionary consisting of 67900 Kiswahili words and was developed using MySpell, (JAMBO SPELL CHECKER, 2004)

However, it used different papers from the internet and some which contains misspelled words to create it dictionary and that was before the greatest work on Kiswahili corpus named Helsinki Corpus of Kiswahili (Helnsinki Corpus) published in 2006.

The wordlist has misspelled or incomplete words, for example, the word “waliyofundis” should be “waliyofundisha” and “waliwasimami” should be “waliwasimamia” (JAMBO SPELL CHECKER, 2004). Moreover, the wordlist contains names of people such as “Chiligati” and “Ishengoma”, which were not supposed to be in it, because only selected and well-known names of people, cities and town are allowed to be in the wordlist.

Table 2.1: Sample of words used in the dictionary of the existing Kiswahili spell checker.

Sample of words used in the Dictionary of the existing kiswahili Spell Checker

67900	Aliitaja	Bonde
Ababa	Allah	Buki
Abdallah	Azori	Buleani
Abiramu	Baali	Bungoma
Abiudi	Babuloni	Bute kataa waliwasimami waliwaunga waliwavuruga waliyokataa waliyofundisunga
Adamu	Babulonia	
Adi	Badar	
Aha	Bahari	
Afikir	Bajini	
Afya	Bernike	
Ahero	Biblia	
Akichangia	Bidaya	
Alidhaniwa	Blasto	

2.5 Research Gap.

Despite existing efforts to develop spellcheckers for Kiswahili, such as the JAMBO SPELL CHECKER (2004), there remains a significant research gap in the field of non-word error detection and correction for the Kiswahili language. The existing solutions suffer from limitations in terms of scope, speed, and accuracy, primarily due to their dictionary compilation method and inadequate coverage of non-word errors.

The JAMBO SPELL CHECKER, for instance, relies on a dictionary that includes misspelled and incomplete words obtained from internet sources. This approach introduces errors and inconsistencies into the spell-checking process. The dictionary contains misspelled words, names of individuals, and other non-linguistic elements, which are not appropriate for accurate non-word error detection and correction. As a result, the effectiveness of the spell checker in detecting and correcting non-word errors is compromised.

To bridge this research gap, this study proposes the development of an improved non-word error detection and correction system specifically designed for the Kiswahili language. The proposed system will leverage advanced algorithms and unsupervised machine learning techniques to enhance the accuracy and efficiency of non-word error detection and correction in Kiswahili documents.

By addressing this research gap, the proposed system aims to improve the quality of written texts in Kiswahili, a widely spoken Bantu language in East Africa. The development of an

effective non-word error detection and correction system will contribute to the global recognition and acceptance of Kiswahili as a language of communication, education, and research. Furthermore, this proposed system lays the foundation for the future development of more advanced language technologies, including grammar checkers, paraphrasing systems, and machine translation systems tailored for Kiswahili.

In summary, the research gap identified in the literature review is the lack of a comprehensive and accurate non-word error detection and correction system for the Kiswahili language. The proposed system aims to fill this gap by utilizing advanced algorithms and unsupervised machine learning techniques, ultimately improving the quality of written texts and promoting the global recognition of Kiswahili as a language.

3 CHAPTER FOUR: KISWAHILI AUTHOGRAPY

Kiswahili is a Bantu language originally spoken along the East African Coast from Southern Somalia to the Northern part of Mozambique and has existed for more than 1,000 years. It's generally spoken by more than 130 million people all over the world, The language has a rich history and has evolved over time, influenced by various cultures and languages. Kiswahili orthography is based on the Latin script and consists of 29 letters, including five vowels and 24 consonants.

Table 3.1: Table showing Kiswahili Alphabet Consisting of 24 symbols

KISWAHILI APLHABET	
VOWELS	Aa, Ee, Ii, Oo, Uu
CONSONANTS	Bb, CHch , Dd, Ff, Gg, Hh, Jj, Kk, Ll, Mm, Nn, Pp, Rr, Ss, Tt, Vv, Ww, Yy, Zz

The objective of this chapter is to conduct a thorough examination of Kiswahili orthography and its distinctive characteristics, which constitute a crucial foundation for crafting an efficient error detection and correction system. Within this chapter, we delve into various facets of Kiswahili orthography, encompassing spelling rules, while also addressing the complexities inherent to Kiswahili orthography and proposing strategies to overcome these challenges in the development of our error correction and detection system.

Kiswahili, classified as a Bantu language, is part of the broader Niger-Congo language family. It shares this linguistic heritage with languages such as Buganda in Uganda, Sotho in Lesotho, Zulu in South Africa, and Kikuyu in Kenya, among others. Although these languages are not mutually intelligible, they trace their roots back to a common ancestral language and exhibit fundamental similarities in vocabulary, word formation processes, and sentence structure.

In essence, this chapter endeavors to provide a comprehensive grasp of Kiswahili orthography, elucidating its significance and its potential to elevate the precision of written Kiswahili through the development of a non-word error correction and detection system. For instance, consider the Kiswahili term "mtu," signifying "person," and its plural form, "watu," signifying "people."

This chapter underscores the parallels in singular and plural word forms, as depicted in Table 3.1.

Table 3.2: Examples of how nouns are pluralized in some Bantu languages

Bantu Language	Country	Singular	Plural
Baganda	Uganda	*omuntu*	*abantu*
Sotho	Lesotho	*motho*	*batho*
Zulu	South Africa	*umuntu*	*abantu*
Kikuyu	Kenya	*muntu*	*abato*

Due to the dynamic exchange between local communities and foreign influences, Kiswahili has seamlessly incorporated loanwords from a multitude of languages, including Arabic, English, German, Portuguese, Persian, and Hindi. It is estimated that foreign vocabulary constitutes approximately 30% of the Kiswahili lexicon, with Arabic contributing the majority. For instance, the term "Kiswahili" finds its origins in the Arabic word "Sahel," signifying "coast."

Numerous factors have contributed to the rapid proliferation of Kiswahili, both within Africa and beyond. These factors encompass media, trade, educational systems, and the notable efforts of influential leaders such as Presidents Nyerere of Tanzania, Jomo Kenyatta of Kenya, and Prime Minister Milton Obote of Uganda, who championed Kiswahili during the struggle for independence (uhuru). President Nyerere, notably, elevated Kiswahili by adopting it as the medium of instruction in primary schools.

Currently, Kiswahili serves as the primary language in Tanzania, Kenya, and the Democratic Republic of Congo, with several other countries employing it as a second language. These nations include Uganda, Zambia, Mozambique, Malawi, Rwanda, Burundi, Somalia, and the Comoro Islands. Notably, Kiswahili enjoys international recognition, with the African Union, formerly known as the Organization of African Unity (OAU), designating it as one of Africa's official languages.

Furthermore, Kiswahili has gained global visibility through major radio networks such as the BBC, Voice of America, Radio South Africa, Deutschewelle (Germany), Radio Cairo, Radio Japan, Radio Beijing, All India Radio, and Radio Moscow International. Several universities and colleges across Europe, Asia, North America, and other parts of Africa offer Kiswahili programs. The language has also made appearances in North American films like "Hotel Rwanda," "The Last King of Scotland," "The Lion King," and "Darwin's Nightmare."

Kiswahili's reach extends to a wealth of internationally renowned songs, including "Hakuna Matata" (No Worries), "Malaika Nakupenda Malaika" (Angel, I Love You, Angel), and "Jambo Bwana" (Hello Mister). Moreover, some English songs, like Lionel Ritchie's "All Night Long" and Michael Jackson's "Liberian Girl," incorporate Kiswahili phrases.

Additionally, there exists a multitude of websites dedicated to Kiswahili grammar, culture, history, and current affairs. The most ambitious initiatives to promote Kiswahili have originated in the United States, notably the world-renowned Kamusi Project, managed by Yale University's Kiswahili Department. Noteworthy tech giants Google and Microsoft have also introduced Kiswahili-language internet search engines, making Kiswahili accessible on a global scale.

3.1 Kiswahili Word Categories

Kiswahili word categories (Parts of speech) sometimes called word classes, in computational linguistic also known as lexical categories. It refers to the group of words which are characterized by their semantic content. Words can be classified using various criteria. Traditionally, words in Kiswahili are classified into eight classes; Noun, Pronoun, Verb, Adverb, Adjective, Conjunction, Interjection and Preposition. *Refer table 3.2*

Table 3.3: Kiswahili Word Categories and corresponding examples

Class	Translation(English)	Example
Nomino	Noun	mtu (person), mbili (two)
Viwakilishi	Pronoun	mimi (I), wao(they)
Vitenzi	Verb	kula (eat), lala (sleep)
Vielezi	Adverb	shuleni (school), asubuhi (morning)
Vivumishi	Adjective	nzuri (good), mrefu (tall)
Viunganishi	Conjuction	na (and), lakini (but)
Vihisishi	Interjection	lo! (lol!), ah! (ah!)
Vihusishi	Preposition	mbele ya (in front of)

Different languages differ in syntax and morphology. For example: In English, Adjective (ADJ) comes before a noun (N), while Kiwahili, like many other Bantu languages has fairly fixed base word order. In Kiswahili Adjective follows a noun as shown below

- Mtoto/N mzuri/ADJ (Kiswahili) -> beautiful/ADJ baby/N (English)

3.2 Noun Classes (Ngeli za nomino)

Most languages around the world, with English as a notable exception, employ a system of categorizing nouns into distinct groups known as noun classes. In the case of Kiswahili, the historical classification of nouns into these classes was a reflection of how East African communities perceived the world around them. For instance, humans were grouped into one class, animals into another, sharp or elongated objects into yet another, and so forth.

However, over time, the East African region experienced increasing interactions with the outside world, primarily through trade expansion and later, colonialism. These interactions led Kiswahili to assimilate new vocabulary from languages like English, German, Portuguese,

Arabic, Hindi, and more. The influx of foreign terms into Kiswahili necessitated the reevaluation and expansion of the noun class system. Presently, Kiswahili employs a classification system comprising eight distinct noun classes.

Listed below are the names of the noun classes and a brief description of what they contain.

1. M-/WA- class contains human beings.
   Eg: mtu – person watu – people mganga – doctor, shaman waganga – doctors, shamans mgeni – visitor, guest wageni – visitors, guests
2. M-/MI- class contains trees, plants, etc.
   Eg: mti – tree miti – trees mmea – plant mimea – plants
3. JI-/MA- class contains fruits, parts of plants, etc. It also contains mass nouns and collectives.
   Eg: jibu – answer majibu – answers jina – name majina – names jimbo – province majimbo – provinces
4. KI-/VI- class contains objects useful to humans and artifacts, etc.
   Eg: kiatu – shoe viatu – shoes kiazi – potato viazi – potatoes kidonge – kijiko – spoon vijiko – spoons
5. N- class contains words borrowed from other languages, names of animals and relationship nouns, etc. Due to the large number of borrowed words in kiswahili, the N- class is the largest of all the noun classes. Nouns in the N- class are identical in both their singular and plural forms and therefore do not have singular or plural prefixes.
   - ***Nouns borrowed from other languages***:
     For example: baiskeli – bicycle(s) meza – table(s) barabara – road(s) barafu – ice kahawa – coffee
   - ***Names of animals:***
     For example: simba – lion(s) twiga – giraffe(s) pundamilia – zebra(s) paka – cat(s) mbwa – dog(s)

- ***Relationship nouns:***

  For exaample: baba – father(s); mama – mother(s); dada – sister(s); kaka – brother(s); bibi – grandmother(s); shangazi – paternal aunt(s); rafiki – friend(s); ndugu – relative(s);

  There are some nouns in the N- class which do not belong to any of the categories listed above, for instance:

  - ndizi – banana(s); nyumba – house(s); chumvi – salt; chupa – bottle(s); simu – telephone(s); mboga – vegetable(s); takataka – garbage

6. U- class contains household objects, names of countries, abstract nouns and qualities. Most nouns belonging to this class are abstract nouns, uncountable nouns and names of some countries. Almost all nouns in this class have the letter U- as a prefix when in singular form however a few nouns begin with the letter W- in the singular form. Here are a few categories of nouns which belong to the U- Class:

   - ***Abstract nouns:***

     utata – complication, ufalme – kingship uzee – old age utoto – childhood Uislamu – Islam Ukristo – Christianity ujana – youthfulness uvivu – laziness umoja – unity

   - ***Uncountable nouns:***

     ugali – corn meal porridge ubongo – brain matter uboho – bone marrow wali – rice umeme – electricity

   - ***Names of countries or regions:***

     Ulaya – Europe Uchina – China Unguja – Zanzibar Uingereza – Great Britain Uhindi – India

7. PA- class contains locatives.

   For example: kabati – cupboard kabatini – in the cupboard nyumba – house

8. KU- class contains verbal nouns.

   In Kiswahili, the KU-Class serves a specific purpose—it is exclusively used with verbs to create infinitives or gerunds. In English, an infinitive is typically constructed using the preposition "to" followed by the verb itself (e.g., "to go," "to eat," "to work," and so forth). An infinitive can function as a verb complement within a sentence (referred to as a verbal

infinitive) or as the subject of a sentence (known as a verbal noun). On the other hand, a gerund is a verb form that concludes with -ing. It carries the same meaning as a present tense participle and can similarly serve as the subject of a sentence (also termed a verbal noun). Below, you'll find some illustrative examples:

Infinitive - Verbal Infinitive: Ninapenda kusoma. – I like to read. **Gerund - Present tense participle:** Ninapenda kusoma. – I like reading **Infinitive - Verbal Noun:** Kusoma ni kuzuri. – To read is good.

Gerund - Verbal Noun: Kusoma ni kuzuri. – Reading is good.

3.3 Pronouns (Kiswahili)

- **Personal pronoun (Viwakilishi vya nafsi)**

  Personal pronouns are the same whether used as subject or object. However, they have different forms depending on the person referred to. Table 3.3 shows the list of personal pronouns.

Table 3.4: Personal Pronouns in singular and Plurial

	Singular	Plural
1st person (Nafsi ya kwanza)	Mimi (I, me)	Sisi (we, us)
2nd person (Nafsi ya pili)	Wewe (You)	Ninyi (you)
3rd person (Nafsi ya tatu)	Yeye (him, her, she, he)	Wao (they, them)

The Pronoun *it, they and them* are not expressed by personal pronouns in Kiswahili when they do not refer to people. Instead demonstrative pronouns are used. Table 3.4 shows demonstrative pronouns and other types of pronouns.

Table 3.5: Types of Pronouns and examples

Pronouns	Examples
Interrogative pronoun (Viwakilishi viulizi)	**Nani** Examples: **Nani** atakuja kesho? (as subject) Ulimwona **nani**? (as direct object) Alimpa **nani** barua? (as indirect object) Ulisafiri na **nani**? (as object of a preposition) **Akina nani (Plural)** always take class 2 noun agreement. Examples: Akina **nani** watakuja kesho? (as subject)
Reflexive pronoun (Viwakilishi virejesh i)	–ji- Example: *Uli**ji**saida (You helped yourself)*
Demonstrative pronoun (viwakilishi vioneshi)	**Huyu, Yule, wale, hao…** Example: *Hawa ni watoto wangu. Wale ni wa dada yangu.*
Possessive pronoun (Viwakilishi vimilikishi)	**Changu, wangu, yangu…** Example: Kitabu hiki ni changu.

Relative pronouns (Viwakilishi virejeshi)	**Amba- ye-, o-, ko-, po-…..** Examples: Nina rafiki **ambaye** anaishi Zanzibar Nina rafiki anayeishi Zanzibar. Nina rafiki aishiye huko Zanzibar.(I have friend who lives in Zanzibar)
Indefinite pronoun	**-ote, -ingi, -ingine, -o –ote, baadhi ya, mojawapo, kadhaa** Example: Vitu **vyote** vipo (There is everything)

3.4 Verb (Kitenzi)

Kiswahili verb has a complex morphology. Generally, Kiswahili verbs consist of subject agreement marker, tense marker, object marker, stem, suffixes and final vowel (ignoring negative markers, relative markers and voice morphology).

- ***Kiswahili verb structure***

  The Kiswahili verb consists of a subject marker, a tense marker and a verb stem.

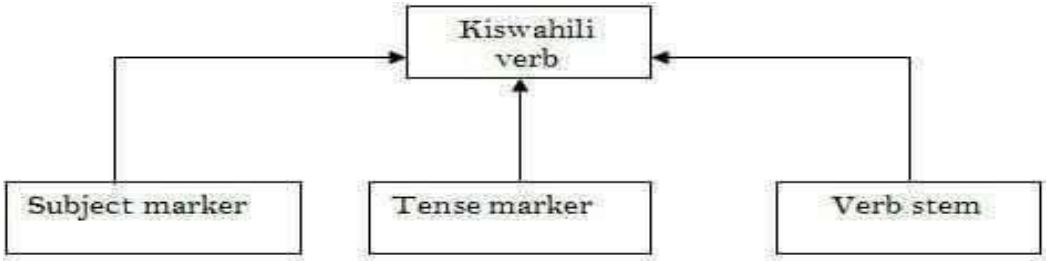

Figure 3.1: Structure of Kiswahili Verb

For instance, the word "Ninalala" can be split up into three parts fitting the above category.

- Ni-na-lala I am sleeping
- Ni - is the subject marker for "I"
- na - is the tense marker indicating "present tense" lala - is the verb stem for "sleep"

The verb in Kiswahili is marked by a prefix which identifies the subject. The following are some of the various prefixes.

- ni - I - First person singular
- u - you - Second person singular
- a - she/he - Third person singular
- tu - we - First person plural
- m - you - Second person plural
- wa - they - Third person plural

In referring to the "present tense" the tense marker "na" is used Thus: Ninalala, unalala, analala, tunalala, mnalala, wanalala

In referring to the "past tense" the tense marker "li" is used.
Thus: nililala, ulilala, alilala, tulilala, mlilala, walilala. The future tense marker is "ta" e.g., nitalala, tutalala, etc.

3.5 Adjectives

In Kiswahili, any word, phrase, or clause employed to modify or elucidate a noun is categorized as an adjective. The realm of adjectives in Kiswahili unfolds into distinct categories, each governed by its unique set of rules:

1. **Agreement Adjectives (Vowel-Initial Stem):** These adjectives necessitate agreement, with the agreement prefix affixed to an adjective stem that commences with a vowel.

2. **Agreement Adjectives (Consonant-Initial Stem):** Similar to the previous category, these adjectives call for agreement, but the agreement prefix is linked to an adjective stem that initiates with a consonant.

3. **Exceptional Adjectives:** These adjectives also require agreement but deviate from the conventional agreement patterns of other adjectives.

4. **Invariable Adjectives:** Originating from Arabic roots, these adjectives maintain their form and do not undergo agreement adjustments when paired with the nouns they modify.

5. **Compound Adjectives:** These adjectives are formed by combining nouns, verbs, or other words in Kiswahili, coupled with the associative marker -a.

These diverse adjective categories within Kiswahili encapsulate the intricacies of noun modification and elucidation in the language.

3.5.1 Vowel Stem Adjectives

Adjectives that must take agreement fall into two categories, adjectives that start with vowels and adjectives that start with consonants.

Table 3.5 shows the agreement taken by adjective stems that begin with either a, e, i, o, or u, the letter y is not considered a vowel in Kiswahili.

Table 3.6: Vowel Stem Adjectives

		-angavu	-ekundu	-ingine	-ororo	-unganifu
Sing.	M	mwangavu	mwekundu	mwingine	mwororo	munganifu
plur	W	waangavu	wekundu	wengine	waororo	waunganifu
Sing	N	nyangavu	nyekundu	nyingine	nyororo	nyunganifu
Plur	N	nyangavu	nyekundu	nyingine	nyororo	nyunganifu

3.5.2 Consonant Stem Adjectives

Table 3.7: Consonant Stem Adjectives

	Noun Class	***-zuri***	***-baya***
Singular	M-	mzuri	mbaya
Plural	WA-	*wazuri*	*wabaya*
Singular	M-	*mzuri*	*mbaya*
Plural	MI-	*mizuri*	*mibaya*

Singular	M-	*mzuri*	*mbaya*
Plural	MA-	*mazuri*	*mabaya*
Singular	KI-	*kizuri*	*kibaya*
Plural	VI-	*vizuri*	*vibaya*
Singular	N-	*nzuri*	
Plural	N-	*nzuri*	

In general adjectives with consonant stems follow Table 4.6, however when consonant stem adjectives must agree with nouns from the NClass then the following rules apply.

1. When a consonant stem adjective beginning with the lettersd, g, j, y, or z is used to describe a noun from the N- Class then the prefix *n-* is necessary.
2. When a consonant stem adjective beginning with the letter bis used to describe a noun from the N- Class then the prefix *m-* must be used.
3. All other consonant stem adjectives do not take agreementwhen describing nouns from the N- Class.

3.5.3 Exception Adjectives

Four commonly employed adjectives in Kiswahili exhibit a distinct pattern of agreement, differing from the standard agreement rules explained earlier. These adjectives are:

- ote (meaning "all" or "whole")
- o ote (meaning "any")
- enye (meaning "having")
- enyewe (meaning "-self")

Notably, with the exception of the M-/WA- noun class, these adjectives adhere to an agreement pattern that aligns with the associative marker -a, making them relatively straightforward to remember.

Table 3.7 illustrates the noun class agreement for each noun class when used with the adjectives -ote, -o ote, -enye, and -enyewe. It's essential to be mindful of word order when incorporating the adjectives -ote or -o ote into sentences alongside other adjectives. These adjectives primarily convey quantity and, therefore, typically appear as the final adjectives, unless an adjective functions as a predicate, in which case the predicate adjective must always occupy the last position.

Table 3.8: Exception Adjectives

Noun Class	***-ote***
C- Singular V- Plural	*chote* *vyote*
W- Singular Z- Plural	*wote* *zote*

- *Chukua vitabu vikubwa vyekundu vyote.* – Take all the big red books.
- *Vitabu vikubwa vyote ni vyekundu.* – All the big books are red.
- *Chukua vitabu vikubwa vyekundu vyo vyote.* – Take any of the big red books.
- *Vitabu vikubwa vyo vyote ni vyekundu.* – Any of the big books are red.

3.5.4 Compound Adjectives

Compound adjectives in Kiswahili are crafted by combining nouns, verbs, and various other words. The process of forming compound adjectives consistently involves the placement of the associative marker (-a of Association) before the word undergoing transformation into a compound adjective. Moreover, it's crucial to ensure that these compound adjectives agree in accordance with the noun they describe, reflecting a harmonious relationship between the elements. The associative marker (-a of Association) serves as the bridge connecting two words, signifying a meaningful association between them. Here are some illustrative examples of compound adjectives:

- maji ya moto:hot water

- mwili wa baridi: cold body
- mahali pa hatari: dangerous place
- sehemu ya siri: confidential location

3.6 Adverbs

An adverb, while frequently employed to enhance a verb, possesses the versatility to modify not only verbs but also adjectives, other adverbs, or even entire phrases within a sentence. Essentially, an adverb sheds light on the manner, timing, or location of an action performed by a person or object. Adverbs can be categorized into five primary types: Adverbs of Manner, Place, Time, Frequency, and Degree. Within this chapter, our exploration will delve into these various categories of adverbs and elucidate their formation.

3.7 Prepositions and Conjunctions

A preposition describes a relationship between words in a sentence and it shows time, space, and logical relationship. A conjunction is a word that connects words, phrases and clauses. There are two kinds of conjunctions: coordinating conjunctions and subordinating conjunctions. A coordinating conjunction links words, phrases and independent clauses in a sentence. However, a subordinating conjunction connects independent clause(s) and dependant clause(s). Some words can be either a preposition or conjunction depending on the context.

3.7.1 Prepositions

Within Kiswahili, certain words naturally function as prepositions, while others take on the role of prepositions through the incorporation of the associative marker (-a of Association) or related phrases. Here are examples of common Kiswahili words that inherently serve as prepositions:

- mpaka – until, as far as, up to
- kutoka, toka, tokea – from, out of
- hata – even, until
- bila – without
- kama – as, if, like
- tangu – since
- kisha – then, and then hadi – until, as far as, up to

3.7.2 Conjunctions

In Kiswahili, conjunctions take two distinct forms: coordinating conjunctions and subordinating conjunctions. Here are some examples of each:

Coordinating Conjunctions:

- na - meaning "and" or "also"
- pia - signifying "also" or "too"
- tena - conveying "again," "furthermore," or "besides"

Examples:

1. Alileta mkate na mchuzi. (He/She brought bread and curry.)

2. Nyumba yake ya kupanga ni chafu, juu ya hayo iko mbali sana.(His/her rental house is dirty, and furthermore, it is very far.)

Subordinating Conjunctions:

- Kwamba - meaning "that" or "because"
- Ninafikiri kwamba atafika. (I think that he will arrive.)
- Ili - signifying "so that" or "in order to"
- Nimeenda dukani ili nipate chakula. (I went to the store so that I could get some food.)

These conjunctions play distinct roles in connecting words, phrases, or clauses in Kiswahili sentences, adding depth and complexity to the language's expression.

4 CHAPTER THREE METHODOLOGY

4.1 Overview

The methodology employed in this study encompasses distinct phases, comprising data collection and preparation, system design and development, and evaluation. Each of these phases encompasses a series of purpose-driven activities aimed at fulfilling the study's objectives.

To address the challenge at hand, this research adopted the Design Science Research Methodology, a problem-solving approach that places a strong emphasis on empirical testing and iterative refinement of the proposed solution. This methodology is particularly well-suited for the development of systems that exhibit both effectiveness and efficiency in addressing complex problems.

In summary, the primary goal of this study was to create a non-word error correction and detection system for the Kiswahili language, utilizing the Design Science Research Methodology.

4.1.1 The Design Science Methodology

The Design Science Methodology (DSRM) is a research methodology that combines the scientific method and design thinking principles to develop innovative solutions to complex problems. DSRM is particularly suitable for developing software systems because it involves a cyclical process of problem identification, requirements gathering, design, implementation, and testing. DSRM encourages a human-centered approach to design, which focuses on the needs and requirements of the end-users.

The rationale of choosing DSRM in this study is that it offers a systematic and iterative process for developing the error correction and detection system for Kiswahili. As a non-word error correction and detection system, it requires a comprehensive and iterative approach that addresses the unique challenges posed by the Kiswahili language.

The DSRM will be used in this study to develop the error correction and detection system for Kiswahili. The methodology will be applied in an iterative and cyclical process consisting of the following steps:

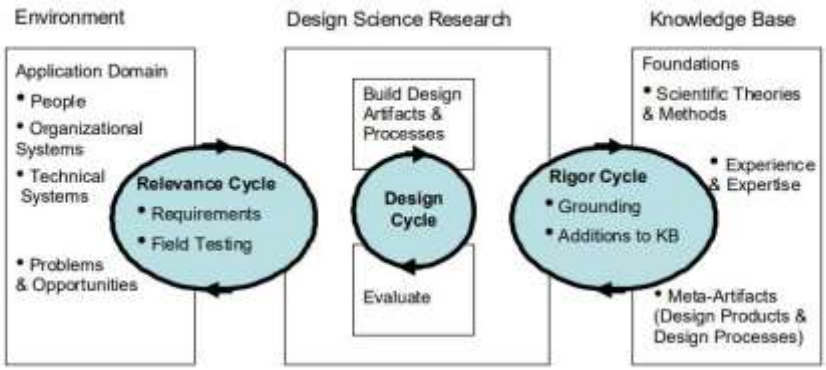

Figure 4.1: A Three Cycle View of Design Science Research Methodology. Alan R. Hevner Information Systems and Decision Sciences, University of South Florida, USA.

"Three Cycle View of Design Science Research" provides a framework for conducting and evaluating design science research in the field of information systems. This view consists of three interrelated cycles that guide the iterative and incremental process of creating and validating innovative artifacts.

1. **Relevance Cycle:** The first cycle, known as the Relevance Cycle, focuses on the identification of real-world problems and the formulation of design goals. Researchers engage with stakeholders to understand their needs and challenges, and then define the objectives for the artifact to address those needs. This cycle emphasizes the importance of relevance and the alignment of the artifact with the practical requirements of the intended users and context.
2. **Rigor Cycle:** The second cycle, called the Rigor Cycle, emphasizes the rigorous development and evaluation of the artifact. Researchers apply rigorous scientific methods and principles to design and construct the artifact. This cycle involves iterative prototyping, testing, and refinement of the artifact to ensure its quality, effectiveness, and

fitness for the intended purpose. Rigor is maintained through systematic evaluation and validation processes, such as expert reviews, usability testing, and controlled experiments.

3. **Design Cycle:** The third cycle, known as the Design Cycle, focuses on the actual design and creation of the artifact. Researchers apply relevant theories, frameworks, and methodologies to guide the design process. This cycle involves making informed decisions regarding the artifact's architecture, components, functionalities, and user interfaces. The design decisions are based on the knowledge and insights gained from the Relevance and Rigor Cycles.

These three cycles are interconnected and iterative, meaning that the findings and insights from one cycle inform and guide the subsequent cycles. The cycles also provide opportunities for feedback and learning, allowing researchers to refine and improve the artifact at each stage. The ultimate goal of the Three Cycle View is to produce artifacts that not only address real-world problems but also contribute to scientific knowledge and theory.

By following this three-cycle approach, researchers can systematically create and evaluate innovative artifacts in the field of information systems. The framework ensures that the research is both relevant to practical needs and rigorous in its scientific foundations, leading to the advancement of knowledge and the development of valuable solutions for real-world problems.

Overall, the Science Design Research Methodology will provide a structured and iterative approach to the development of the error correction and detection system for Kiswahili. It will enable the research team to address the unique challenges posed by the Kiswahili language and develop an effective and user-centered system.

4.2 Conceptual Framework

Centering on the realm of non-word error detection, SwaSpell, an intelligent system was created and engineered to identify and rectify spelling errors within Kiswahili text. SwaSpell comprises two distinct subsystems: SwaDetect, responsible for detecting non-word errors using dictionary look-up techniques, and SwaCorrect, which employs the Jaccard coefficient to meticulously select and rank candidates within the correction list.

Diagrammatically, Figure 4.2 and Figure 4.3 show the conceptual framework, an overview or a bird's eye view of the proposed system (SwaSpell).

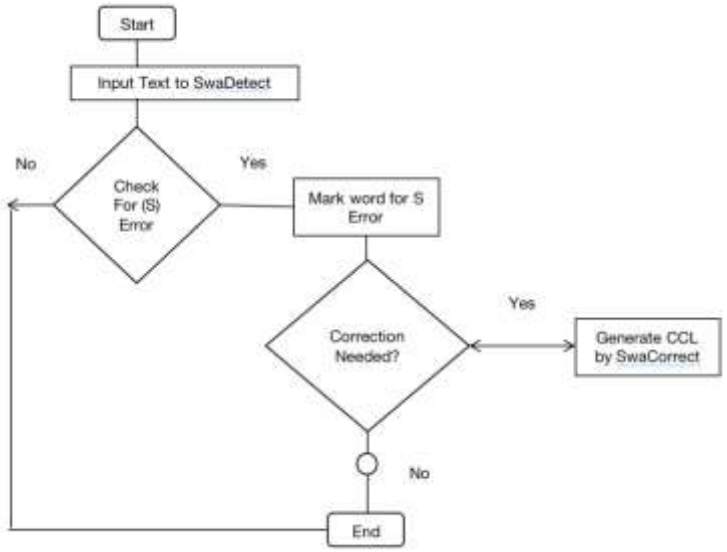

Figure 4.2: Conception framework as a flow chart for Interactive Detection - Correction System

```
Start
    Input Text to SwaDetect
        If Spell Error
            Mark word as Non Word Error
                If Correction Needed
                        Generate CCL by SwaCorrect
                Else
                        Display Word Spell Error
            Else
Stop
```

Figure 4.2 Conceptual framework in pseudo code For Interactive Detection - Correction System

4.3 New Spellchecker Implementation

The prototype is implemented using the Python programming language, chosen for its strong relevance to the application domain. Python has gained significant traction among researchers and developers for creating spellcheckers in various languages. For instance, Balagadde and Premchand (2016) utilized Python to develop a non-word error detection and correction spellchecker for Luganda, while Anondo pioneered the creation of the first spellchecker for the Kîmîîrû language (Anondo, 2013). Python's suitability is further underscored by its object-oriented nature , supported by built-in data types like lists and dictionaries.

4.3.1 Developer Requirements

1. Hardware Requirements

    - Processor specifications: minimum Intel Core i5 since this research work would be dealing with large datasets or heavy algorithms.
    - RAM: a minimum of 4 GB of RAM is recommended
    - Storage capacity: At least 128GB of free storage space is recommended.

2. Software Requirements

    - Integrated Development Environment (IDE): Utilization of a suitable code editor or IDE tailored to the programming languages and technologies utilized in spellchecker development, for instance, in this project Visual Studio Code, and IntelliJ IDEA were used.
    - Programming Languages: Python, Java, C++, or JavaScript are programming languages which could be used however in this project Python was used.

4.3.2 User Requirements

1. Hardware Requirements

    A standard computer or laptop equipped with a minimum of 1.6 GHz or faster dual-core processor, memory of 4 GB and storage capacity of 1 GB

2. Software Requirements

- Operating System: Compatibility with the user's hardware, encompassing options like Windows, macOS, or Linux for desktop, and iOS or Android for mobile.
- Web Browsers: Support for modern web browsers such as Chrome, Firefox, Safari, or Edge.
- Spellchecker Software: Compatibility with the user's environment, be it applications like Microsoft Word, web browser extensions, or standalone spellchecker apps.

4.4 System Evaluation

System evaluation stands as a pivotal stage in the development of any software system, including error correction and detection systems. These critical steps ensure the system's effectiveness and reliability in accurately identifying and rectifying spelling errors.

The evaluation of the system's performance relies on a multitude of metrics, encompassing Accuracy, Precision, Recall, as well as Speed of detection and correction.

In the context of a spellchecker, accuracy refers to the spellchecker's ability to correctly identify and suggest corrections for misspelled words in a given text. It measures how often the spellchecker provides the right suggestions for fixing spelling errors. Essentially, accuracy in a spellchecker context indicates the proportion of correctly suggested corrections out of the total corrections made by the spellchecker, expressed as a percentage. Higher accuracy means that the spellchecker is more reliable in offering accurate spelling suggestions.

Precision measures the accuracy of the spellchecker's correction suggestions, quantifying the proportion of correctly suggested corrections. Recall, on the other hand, assesses the spellchecker's ability to identify and suggest corrections for all misspelled words in the text, representing the proportion of detected errors.

The speed of detection refers to how quickly a spell checker identifies misspelled words or errors in a given text. It measures the efficiency and swiftness with which the spell checker can locate

and flag potential spelling errors without necessarily suggesting corrections. A faster speed of detection means the spell checker can promptly identify errors in the text.

The speed of correction pertains to how quickly the spell checker can not only identify errors but also provide accurate suggestions or corrections for those errors. It gauges the efficiency of the spell checker in either identifying and rectifying misspelled words or other linguistic errors. A faster speed of correction indicates that the spell checker not only detects errors promptly but also offers appropriate fixes with minimal delay.

The experimentation employs a confusion matrix, allowing the identification of True Positives (TP), False Negatives (FN), False Positives (FP), and True Negatives (TN), which serves as the basis for calculating Accuracy, Precision, and Recall using the formulae 4.1, 4.2 and 4.3 respectively, where:

- **True positives (TP):** valid words and which is recognized by the spelling checker as correct word. (Say it as: Truly positive
- **True negatives (TN):** invalid word that the spellchecker recognized as incorrect word (Truly negative)
- **False positives (FP):** the system predicts the word to be correct, yet it is incorrect (falsely positive)
- **False negatives (FN):** correct word that the spellchecker recognized as incorrect word. (Falsely negative); the system predicts the word to be negative yet is correct

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP} + \mathrm{TN}} \quad (4.1)$$

$$Precision = \frac{\mathrm{TP}}{TP + FP} \quad (4.2)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (4.3)$$

Measuring Detection Speed in SwaDetect employs two key metrics to gauge the speed of detection. The initial metric, denoted as FD and evaluated using Equation 4.4, quantifies the number of words that can be examined in a single second. FD, directly influenced by the

computer's processing power, exhibits a proportional relationship with detection speed. In other words, a higher FD value signifies a faster detection speed.

$$FD = Tot / T \quad (4.4)$$

Where:

- Tot stands for the total number of tokens.
- T signifies the time taken to process these tokens.

FC, as assessed by Equation 4.5, serves as a gauge of SwaCorrect's rapidity in producing Corrected Candidate Lists (CCL). FC is directly influenced by the computer's processing capabilities and quantifies how many words can be inspected within a single second.

$$FC = N / T \quad (4.5)$$

Where:

- N represents the total number of tokens.
- T denotes the time required to process these tokens.

5 CHAPTER FIVE: SYSTEM DESIGN, IMPLEMENTATION AND EVALUATION

5.1 Introduction

In essence, this chapter offers a comprehensive exploration of the design, implementation, and evaluation facets of the proposed Kiswahili system. It provides valuable insights into the practical development and deployment of a solution aimed at rectifying non-word errors within written Kiswahili text.

In details:

- The first section furnishes an encompassing view of the system's design and architecture, shedding light on its constituent components and their respective functionalities.
- Following that, the subsequent section elucidates the system's implementation, encompassing details regarding the employed programming languages, tools, datasets, and resources.
- Lastly, the chapter concludes with the presentation of the system's evaluation, delineating the methodology and metrics harnessed to gauge its performance.

5.2 System Design

The conceptualization of the Kiswahili system was rooted in the foundational principles of NLP and Machine Learning (ML). The design journey traversed several critical stages, encompassing data collection, meticulous pre-processing, and the seamless orchestration of system integration. An overarching emphasis was placed on sculpting the system to be inherently modular, effortlessly scalable, and intuitively user-friendly.

Underpinning the developmental framework of SwaSpell, a novel system, are two pivotal sub-systems: the Non-word Error Detection Sub-system, aptly christened SwaDetect, and the Non-word Error Correction Sub-system, known as SwaCorrect. These Subsystems serve as the cornerstones of the system, collectively driving the mission to deliver accurate non-word error detection and correction capabilities in Kiswahili.

5.3 System Architecture:

The system architecture is illustrated in Figure 5.1

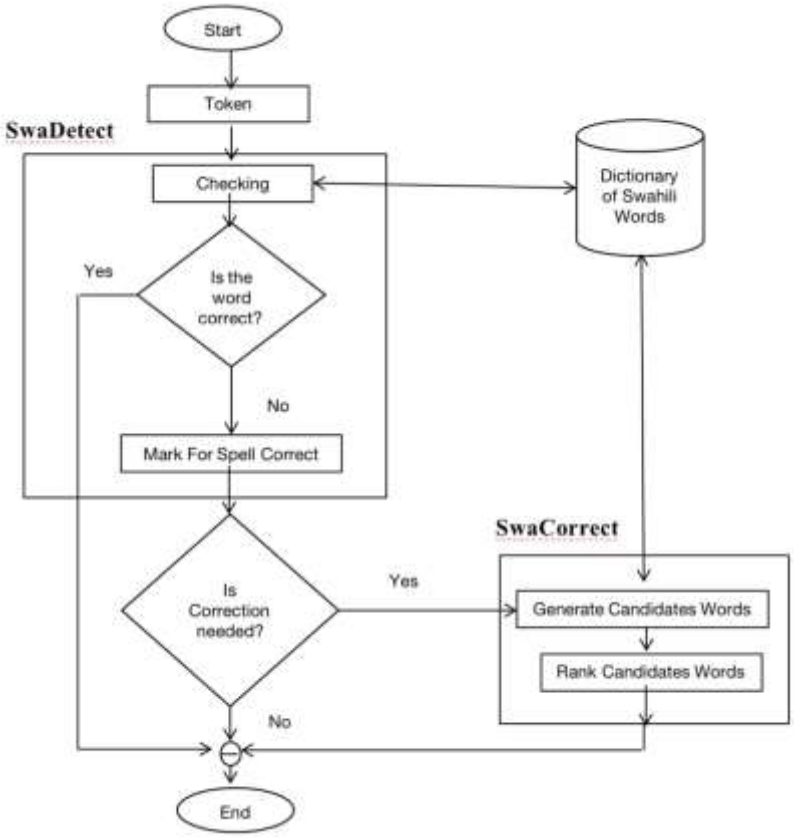

Figure 5.1: Architecture of SwaSpell

5.3.1 *Non-Word Error Detection Subsystem (SwaDetect)*

SwaDetect employs the Dictionary Lookup Technique (DLT), where each word within the input text undergoes a dictionary lookup process. In the event that a word is not found within the lexicon, it is categorically considered a non-word or incorrect and is subsequently flagged or added to the list of erroneous words. It's noteworthy that the available Open Source Spellchecker for Kiswahili has utilized various sources from the internet, including some with misspelled words, to construct its dictionary. Consequently, this spellchecker is susceptible to erroneous detection and correction. In our research, we have opted for a more robust corpus, specifically the acclaimed Kiswahili corpus (Helsinki, 2006). This corpus has been rigorously experimented upon and validated by Kiswahili linguistic professionals, rendering it the most reliable resource for our work.

SwaDetect, is invoked when a token is passed to the SwaSpell, whose primary function is to determine whether the token in question is a correct Swahili word or a Non-Swahili word, as per the dictionary lookup technique elucidated in Section 2.2.1; subsequently, it triggers one of the following measures:

i. If a token is found within the constituent dictionary or lexicon, it is classified as a Correct Swahili word.
ii. Conversely, if a token is absent from the constituent dictionary, it is categorized as a non-word. In this case, the word is marked or highlighted in red, awaiting correction.

In the second case, upon detecting an error, the SwaDetect invokes SwaCorrect if the end user requests for a correction.

5.3.2 *Non-Word Error Correction Subsystem (SwaCorrect)*

Upon identifying a misspelled word using SwaDetect, the subsequent phase involves the development of the Non-word Error Correction subsystem (SwaCorrect) whose primary objective is to furnish a Correction Candidates List (CCL) for the non-word errors detected by SwaDetect.

SwaCorrect is guided by two pivotal assumptions. The first assumption, grounded in research findings, suggests that a significant majority, ranging from 93% to 95%, of spelling errors can be attributed to an edit distance of one (ED1) from the intended target (Peterson, 1986). The second assumption stipulates that it is highly improbable for an end-user to commit an error on the first character, (Yannakoudakis, et. al.1983).

These assumptions streamlined the development of SwaCorrect in two critical ways. Firstly, they led to a reduction in the number of Correction Candidates (CCs), simplifying the process. Secondly, they mitigated the need for extensive processing power.

SwaCorrect's task of providing a list of Correction Candidates (CCL) was achieved by identifying strings at an edit distance of "n" from the misspelled word, employing the Minimum Edit Distance Technique.

Edit distance between two words essentially represents the sequence of editing operations required to transform one word into another. For example, transforming "jmbo" into "jambo" requires an edit distance of 1, as it involves inserting an "a" between "j" and "m."

As elucidated in the methodology, we implemented four types of edit distance techniques: Insert, Delete, Swap, and Replace.

- INSERT(add a letter) Example: "mt" => "mtu meaning person".
- DELETE (remove a letter) Example: "nyumba" => "nyuma", "yumba".
- SWAP (swap or interchange two adjacent letters) Example: "mama" => "amma".
- REPLACE (change one letter to another) Example: "wangu" => "yangu", "tangu".

However, due to the nature of how this technique generates candidate words, not every word in the candidate list is necessarily a valid word. Therefore, a filtration step is necessary to retain only those candidates that align with the vocabulary.

To enhance accuracy at this stage, we opted for the Jaccard Coefficient (JC), a similarity metric evaluated using Equation 2.1. JC was employed to select and rank the best correction candidates in descending order of JC value for presentation to the end user. It is worthwhile noting that JC is directly proportional to similarity measure that means the higher JC the higher is the Similarity measure. High similarity between two sets indicates a strong resemblance, while low similarity suggests a greater dissimilarity or distance.

Here are examples of non-word error correction using the Set-Based Jaccard Coefficient:

Example 1

let Set A = {J, M, B, O} and Set B = {J, A, M, B, O} then JC(A, B) = |{j, m, b, o}| / |{j, a, m, b, o}| = 4/5 = 0.8

Example 2

let Set A = {J, M, B, O} and Set B = { H, A, B, A, R, I } then JC(A, B) = |{j, m, b, o}| / |{ h, a, b, a, r, i }| = 1/10 = 0.1

In this context, {J, M, B, O} is more similar to {J, A, M, B, O} than {H, A, B, A, R, I}, as evidenced by a higher JC of 0.8 in comparison to 0.1.

It's crucial to note that correction candidates with an ED1 are automatically selected, as they typically exhibit the highest JC value. Our implementation of SwaCorrect is grounded on JC because of its computational efficiency, given its linear nature, compared to more computationally intensive metrics like Damerau-Levenshtein Distance (DLD), which is quadratic. Additionally, JC correlates well with DLD as demonstrated by Balagadde and Premchand (Balagadde and Premchand 2016 a) and Table 5.1, rendering it a suitable choice for our purposes.

Table 5.1: Relation between ED and JC

Case	Target	Erroneous Word	ED	JC
1	Jambo (hello)	Jjambo	1	0.8000
		Jammboo	2	0.666667
		jaambboo	3	0.571429
2	Kundi (Team)	Kkundi	1	0.666667
		Kunndii	2	0.571429
		kuunddii	3	0.500000

5.4 System Implementation

The implementation of the proposed system for Kiswahili followed a rigorous process to ensure that the system functions effectively and efficiently. This section discusses the programming languages, frameworks, and tools used, as well as gives an overview of the system's features and functionality.

The implementation process is divided into several stages, which included requirements gathering, design, development and test. During the requirements gathering stage, we worked closely with Kiswahili language experts to identify the common non-word errors that occur in written Kiswahili like the word Fedha (Money) which is mostly written as Feza due to the way it sounds. This stage is crucial in ensuring that the system is tailored to the specific needs of Kiswahili users.

After the requirements gathering stage, we proceeded to the design stage, where we developed a system architecture that could effectively detect and correct non-word errors in Kiswahili text. The design stage was followed by the development stage, where we used a combination of programming languages, frameworks, and tools to build the system as presented in Section 3.4.1.

Once the system was developed, we proceeded to the testing stage, where we carried out various tests to ensure that the system was functioning as expected. During the testing stage, we used a variety of test cases, including simulated non-word errors in Kiswahili text, to assess the system's accuracy and efficiency.

5.4.1 Programming Languages, Frameworks, and Tools Used

SwaSpell is developed using several programming languages, frameworks, and tools. These included Python programming language, Flask web framework, and MySQL database.

Python is chosen for its simplicity, readability, and versatility, which made it easy to develop the system's algorithms and integrate it with other tools and frameworks. Flask, on the other hand, was chosen for its lightweight nature and ease of use in developing web applications. It allowed us to develop a user-friendly web interface that Kiswahili users could easily access and use.

The MySQL database is used to store the system's data, including the Kiswahili lexicon, the non-word error patterns, and the corrections for each non-word error. It provided a robust and reliable platform for storing and retrieving data, ensuring that the system could access the necessary information quickly and efficiently.

5.4.2 Integrating backend code with GUI frontend

We have seamlessly merged the backend code with a Graphic User Interface (GUI) frontend to provide end-users with a straightforward way to use SwaSpell. Users can interact with our syatem through the GUI effortlessly.

To initiate the process, the end user simply enters a Kiswahili word into the input field and then clicks on the "Spellcheck Here" button. This action transmits the user's input text to SwaDetect, which proceeds to validate or reject the entered word. In the event of a rejection, the erroneous word is highlighted in red to indicate that it is an invalid word error.

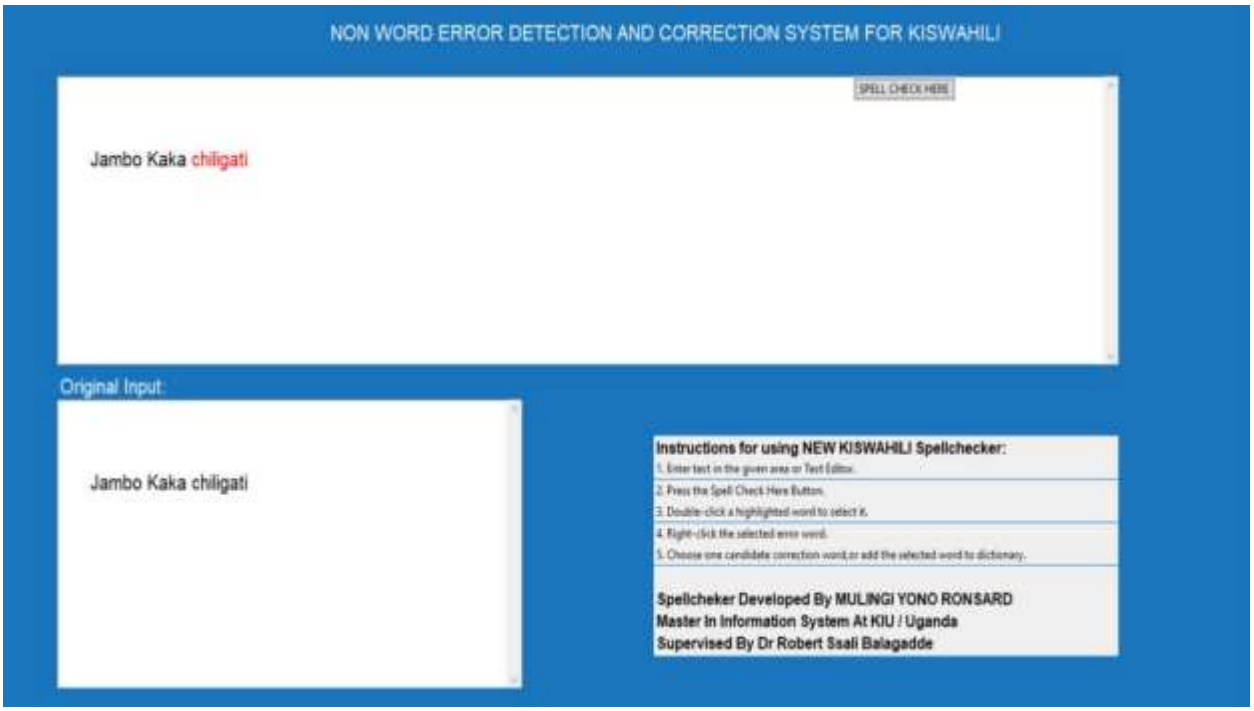

Figure 5.1: SwaSpell GUI

5.5 SwaDetect Evaluation

The evaluation process is initiated by passing as input to SwaDetect a dictionary containing correct Kiswahili words and erroneous words or non-words.

Our objective is to find out how does SwaDetect react on each of these words, in other words, we wanted to find out if SwaDetect detects erroneous words as non-words errors and Correct word as correct and after that calculate its Accuracy.

For this purpose, we prepared a dataset containing 300 tokens, which includes 200 correct words plus 100 erroneous Kiswahili words with different Edit Distances.

In this experimentation we prepared a dataset containing 300 tokens of which 201were correct Kiswahili words and 99 were erroneous Kiswahili words. The dataset was passed on to SwaDetect and the results were captured in the confusion matrix shown in Figure 5.2, whose terminologies were defined in section 3.4. Accuracy, Precision, and Recall were evaluated in percentage using the formulae in Equations: 3.1, 3.2, and 3.3 respectively as shown below.

$$Accuracy = \frac{200+99}{200+0+1+99} \text{x}100 \quad = 99.7\ \%$$

$$Precision = \frac{200}{200+1} \text{x}100 \quad = 99.5\ \%$$

$$Recall = \frac{200}{0+200} \text{X}100 = 100.0\ \%$$

Table 5.2: Confusion matrix for capturing data for evaluating SwaDetect performance in terms of Accuracy, Precision and Recall

Confusion Matrix	
TP (True Positive) 200	FP (False Positive) 1
FN (False Negative) 0	TN (True Negative) 99

5.5.1 SwaDetect Speed Evaluation

We carried out two sets of experimentations. One involved three Datasets with the same size but different content, precisely 500 words each; and the other, with three datasets of varying sizes but same content. We determined the time taken by SwaDetect to process the data and captured the results in Tables 5.4 and 5.5 respectively. We used Equation 3.4 to evaluate the detection speed of SwaDetect (F_D).

Table 5.3: Summary of results for determination of SwaDetect speed with datasets of the same sizes but different content

	Dataset With Only Erroneous Words	Dataset with Only Correct Words	Dataset with Mix Words in a Ratio of 1:1
N	**500**	**500**	**500**
Ts	**0.0235**	**0.0235**	**0.0235**
F_D	**43 Hz.**	**43 Hz.**	**43 Hz.**

Table 5.4: Summary of results for determination of SwaDetect speed with datasets of varying sizes but same content (that is, erroneous and correct words mixed in a ratio of 1:1)

	N	Ts	F_D
Dataset1	**237**	**0.0111**	**90 Hz**
Dataset 2	**350**	**0.0164**	**61 Hz.**
Dataset 3	**500**	**0.0235**	**43 Hz.**
Average(D1-D3)	**1087**	**0,017**	**65 Hz**

Average Processing Speed (Hz) = (90 Hz + 60.77 Hz + 42.66 Hz) / 3 = 193.43 Hz / 3 ≈ 65 Hz

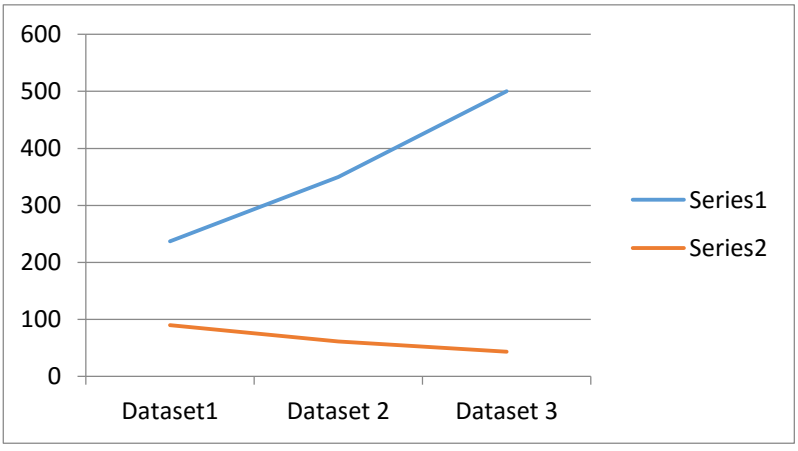

Figure 5.2: Graph showing the Relationship between Dataset Size and Detection Speed

F_D depends on the processing power of the computer. The specifications of computer used in the experimentation are: Intel® i5-8250 U Processor running 64 bit Windows 10 Ultimate.

5.6 SwaCorrect Evaluation

5.6.1 SwaCorrect Performance Evaluation

The evaluation process is initiated by preparing three dictionaries - namely Dic1 through Dic3 extracts of which are shown in Tables 5.5, 5.6, and 5.7 respectively - comprising of pairs of Target (Correct Kiswahili word) and its erroneous version.

Table 5.5: Extract of Dic1 containing erroneous words of ED1 from target (Correct word)

Case	Target	Erroneous Word
1	Jambo (hello)	Jjambo
2	Kundi (Team)	Kkundi
3	Unaotimiza (what you have done)	Umaotimiza
4	Atajifunga (to tie himself)	Atajjifunga
5	Jifanyieni (Do it yourself)	jifaniyieni
6	Jua (sun)	Juua
7	Kibao (Board)	Kibbao
9	Kula (to eat)	Kla

Table 5.6: Extract of Dic2 containing erroneous words of ED2 from target (Correct word)

Case	Target	Erroneous Word
1	Jambo (hello)	Jammboo
2	Kundi (Team)	Kunndii
3	Unaotimiza	Uunaottimiza
4	atajifunga	Atajiffunnga
6	jua	juwaa

8	habari	Pbary

Table 5.7: Extract of Dic3 containing erroneous words of ED3 from target (Correct word)

Case	Target	Erroneous Word
1	Jambo (hello)	jaambboo
2	Kundi (Team)	kuunddii
4	atajifunga	Atajjiiffunga
7	kibao	kiibbaoo
8	habari	pbari

The erroneous words are passed on to SwaCorrect to generate the Correction Candidate List (CCL) and if the target exists in the generated CCL then SwaCorrect has accurately provided a correction otherwise SwaCorrect has failed to provide a solution. A count on these entities is automatically captured and presented in Table 5.8. It is worthwhile noting that it is not possible to capture FN and TN and therefore they are assigned a value of zero. The reason behind this is that SwaCorrect is providing only the correct version of the invalid word. These two entities are relevant in detection problems. Accuracy (Ap) and Precision (PP) are evaluated using formulae in Equations: 3.1 and 3.2 respectively. After substitution, we observe that AP and PP are one and the same. The variation of how accuracy or precision varies with edit distance is visualized in Figure 5.3.

Table 5.8: Example of the Dictionary use for experimentation containing errors of ED1 from target (Correct word)

	Dic1 With Ed1	Dic2 With Ed2	Dic3 With Ed3	Average
Correct Prediction (TP)	112	83	32	N/A
Incorrect Prediction (FP)	2	17	18	N/A
Accuracy (A_P)	98	83	64	82

Precision (P_P)	98	83	64	82

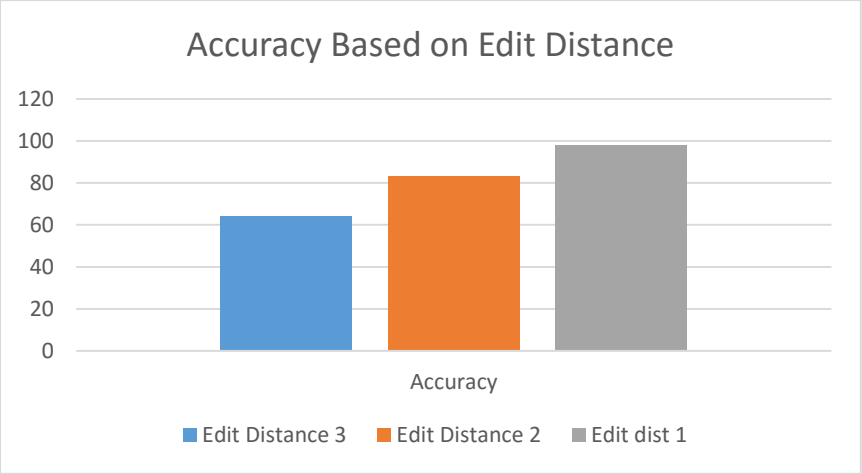

Figure 5.3: Bar chart showing how accuracy or precision varies with Edit Distance. The data for the visualization was captured from Table 5.8.

5.6.2 SwaCorrect Speed Evaluation

In reference to the experimentation, we prepared their dictionaries each with 100 tokens, namely Dic1 through Dic3 exacts of which are shown in Tables 5.5, 5.6, and 5.7 respectively, each item in each dictionary comprising of a pair of Target (Correct Kiswahili word) and it erroneous version. The erroneous words were passed on to SwaCorrect to generate the Correction Candidate List (CCL) and the time taken to generate CCLs for each dictionary was determined. The results of the experimentation were captured and presented in the Table 5.9. The correction speed of SwaCorrect (F_C) was evaluated using Equation 3.4.

Table 5.9: Summary of results for determination of SwaCorrect speed with Dictionaries of the same sizes but different content (that is, different Edit Distances)

	N	Ts	Fc (Hz)
Dic1 with ED1	100	0.0169	59
Dic2 with ED2	100	0.0172	58

Dic3 with ED3	100	0.0175	57
Average(ED1-ED3)	**300**	**0.0172**	**58**

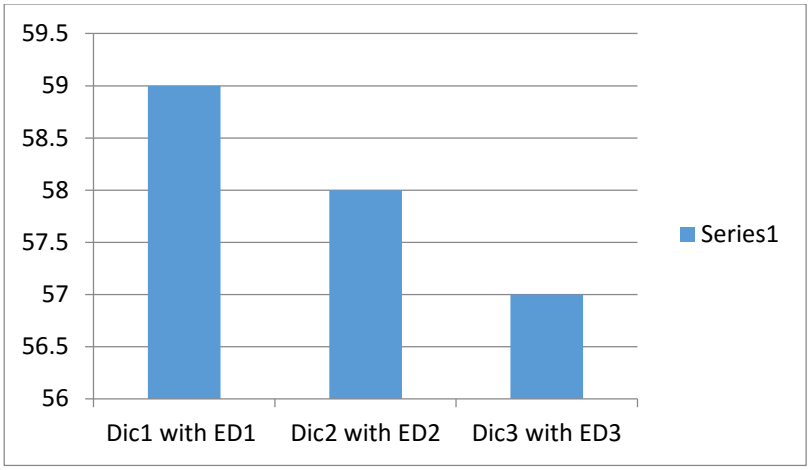

Figure 5.4: Bar chart showing how accuracy or precision varies with Edit Distance. The data for the visualization was captured from Table 5.9.

The computer utilized for the experiments was equipped with an Intel® i5-8250U Processor and operated on a 64-bit Windows 10 Ultimate Operating System.

5.7 Discussion of Results

Based on experimental findings, SwaDetect achieves an impressive accuracy rate of 99.7% this mean that it correctly detects and highlights 99.7% of the spelling errors in the text it examines.

The Precision of 99.5 for SwaDetect means that when it flags a word or phrase as a potential spelling error, it is correct 99.5% of the time, in other words, out of all the items the spellchecker identifies as errors, only 0.5% of them are actually not errors (false positives).

The Recall of 100% or detection means that it correctly identifies every single spelling error in the text without missing any. In other words, it captures and flags all spelling mistakes in the text, ensuring that nothing goes unnoticed.

In reference to Table 5.3 we observe that when the size of the Dataset is the same, Speed of Detection (F_D) is also the same. However, from Figure 5.2 we observe that size is inversely proportional to F_D. this concept is consonant with that deduced by Balagadde and Premchand, (Balagadde and Premchand, 2016 b)

The average F_D of 65Hz derived from Table 5.4 means that SwaDectect can detect errors in 65 words in one second which is good enough for interactive detection, given that the minimal speed of 10 Hz is required for interactive correction, (Peter Norvig, 2007).

In reference to Table 5.8, we observe that average Accuracy(A_P) or Precision (P_P) of SwaCorrect is 82% which means that it can successfully correct 82% of non-word errors that are ED1 through ED3 from their target (Correct word). We can also deduce that A_P is inversely proportional to ED that is when ED increases AP falls. This concept is visualized in Figure 5.3.

In reference to Table 5.9, we note that the average speed of correct (F_C) is 58 Hz which is far above 10 Hz that is the minimum requirement for interactive correction as stipulated by Norvig (Peter Norvig, 2007). This means that SwaCorrect can provide correction for 53 erroneous words per second. It is worthwhile noting from Figure 5.4 that FC falls when ED increases which emphasises the inverse proportionality between the two entities. This concept is consonant with that deduced by Balagadde and Premchand, (Balagadde and Premchand, 2016 a)

We can deduce that the F_D and F_C depend on the speed of the computer processor used. The faster the processor, the bigger F_D and F_C are.

5.8 Comparison with Other Systems

Table 5.10: SwaDetect compared to other Detection Systems

	Accuracy in percentage	Speed of Detection (HZ)	Detection Technique
SwaDetect	99.6	85	Direct DLT
Jambo Spell Checker (KILINUX, 2004)	Not Specified	Not Specified	Direct DLT
LugDetect (Balagadde and Premchand, 2016 b)	100	1474	Multi DLT
Aspell.31 (Atkinson, 2006) English Detector	93.1	137	DLT and other approaches
AKHAR (Punjabi Detector) (Kaur and K. Garg, 2014)	83.5%	Not Known	Direct DLT

Table 5.11: SwaCorrect compared to others Correction Systems

	Accuracy in percentage	Precision in percentage	Speed of Correction (HZ)	Correction Technique
SwaCorrect	85	85	58	Jaccard Coefficient Technique
Jambo Spell Checker (KILINUX, 2004)	Not Specified	Not Specified	Not Specified	Edit Distance Technique
Kîmîîrû_Spellchecker (Anondo, 2013)	80	100	Not Specified	Hunspell language tools
Gĩkũyũ_Spellchecker (Ng'ang'a and WKamau, 2010)	75	82	Not Specified	Hunspell language tools

LugCorrect (Balagadde and Premchand, 2016 a)	98	Not Specified	1365	Jaccard Coefficient Technique

In reference to Table 5.10, SwaDetect produced a better result than most of the detection systems considered in this research work namely, Jambo Spell Checker, Aspell.31, and AKHAR, apart from LugDetects which uses Multi DLT.

In reference to 5.11, SwaCorrect outperform all the corrector consider in this work apart from LugCorrect. The exceptional accuracy of LugCorrect can be attributed to the incorporation of a big Correct Candidate List (CCL) which has 10 candidates. The underlying rationale for this is that the likelihood of the target word being present in a CCL increases with the number of elements in the list. We plan to implement a similar approach in our future work.

6 CHAPTER SIX: CONCLUSION AND FUTURE WORK.

6.1 Conclusion

This research work has introduced models for addressing non-word error detection and correction in Kiswahili, a Bantu language, resulting in the development of SwaSpell comprising of SwaDetect and SwaCorrect.

Experimental results demonstrate that SwaDetect achieves a detection accuracy of 99.6% operating at an average speed of 65 Hz (words per second). This surpasses the minimal speed requirement of 10 Hz for interactive correction. It is noteworthy that the speed of detection is inversely proportional to the lexicon size.

In reference to SwaCorrect, experimental results indicate that it can correct erroneous words with edit distances of one (ED1) through three (ED3) from the target (correct version of the word) with an average accuracy (Av) of 85% operating at a processing speed of 58 Hz, equivalent to 58 words per second. Experimental results demonstrate that accuracy of correction is inversely proportional to edit distance.

6.2 Future work

SwaSpell's current scope is limited to non-word errors, however, this can be extended to encompass real word errors. Several approaches exist for addressing real-word errors, most of which employ statistical methods. For Kiswahili, the approach proposed by Mays et al. (Mays et al.), which utilizes word-trigram probabilities for detecting and correcting real word errors, is one option. Another approach, presented by Kernighan et al. (Kernighan et al), is a real-word error detector based on a noisy channel model. Additionally, Word Sense Disambiguation (WSD) algorithms can be employed to correct real word errors in context by leveraging sentence context to resolve semantic ambiguity.

Therefore, further investigation is needed to determine the most suitable approach for the highly inflected Kiswahili language among the mentioned alternatives. Moreover, exploring language and error models, such as statistical selection models for correction candidate selection, may enhance the performance of SwaCorrect.

REFERENCES

Adeyanju, O. A., & Adewole, K. S. (2013). Rule-based spell-checking system for Yoruba language. International Journal of Emerging Trends & Technology in Computer Science, 2(1), 92-96.

Amazigh, Spell Checker based on Naïve Bayes and N-Gram. Proceeding of Human Language Processing of Resource Scarce Languages Workshop, International Conference on Advanced Technologies and Humanitarian Sciences, July 25-26, 2019.

Anondo, Timothy K. (2013) , Open Source Spelling Checker for Kîmîîrû Language , 2013-11-26T06:35:00Z

Aseyo&John, (2011), Open source spellchecker for Luhy a-Lulogoli http://erepository.uonbi.ac.ke:8080/xmlui/handle/123456789/12562

Atkinson, K. , "Gnu Aspell 0.60.4", http://aspell.sourceforge.net/, 2006.

Balagadde Robert Ssali and Parvataneni Premchand, (2016 a). Non Word Error Correction for Luganda. Computation and Communication Technologies, Edited by Kumar, Senthil T. / Mathivanan, Bala, De Gruyter, e-ISBN: 9783110450101, PP: 429–448, Germany

Balagadde Robert Ssali and Parvataneni Premchand, (2016 b). Non word Error Detection for Luganda. Emerging Technologies in Engineering. Edited by Mahesh P. k, Su-Qun Cao, and Chanshyam Singh. McGraw Hill Education. ISBN - 13: 978-93-5260-317-6, PP: 1 – 17, India

Bird S., Klein E. and Loper E. (2009). Natural Language Processing with Python, O‟Reilly Media.

Brill E. (1995). Transformation-Based error driven learning and natural language processing: A case study on part-of-speech tagging‟ In Maryland, Proceedings of Computational linguistics Vol. 21.

Brill E. (2008). ‟Some advances in Transformation-Based Part of speech tagging‟, In Massachusetts, Proceedings of the National conference on artificial intelligence (AAAI-94).

Burkhard, W.A., Keller, R.M., 1973. Some approaches to best-match file searching. Commun. ACM 16 (4), 230–236. https://doi.org/10.1145/362003.362025.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research, 12*(Aug), 2493-2537.

Daelemans W. and Van den Bosch A. (2005). *Memory-based Language Processing*, Cambridge University press, Cambridge.

Damerau F J.; and Mays. E., (1989). "An Examination of Undetected Typing Errors", Information Process Management, 25, 6, 659-664.

Damerau F. J., (1964). "A technique for Computer Detection and Correction of Spelling Errors", Communication ACM, 7:171–176.

Damerau F. J., (1990). "Evaluating Computer-Generated Domain - Oriented Vocabularies", Information Process Management, 26, 6, 791– 801.

Editha D. Dimalen, Davis D. Dimalen (2006), An OpenOffice Spelling and Grammar CheckerAdd-in Using an Open Source External Engine as Resource Manager and Parser, Unpublished report, MSU- Iligan Institute of Technology, Iligan city.

Hassan, H. A., & Rashidi, T. H. (2012). A comprehensive study on spell-checking for Swahili text. Journal of Intelligent Information Systems, 39(1), 111-131.

Hinnebusch T. (1996), "What kind of language is Swahili?" Afrikanistische Arbeitspapiere, Vol. 1. No. 47, 73-95

Hládek, D., Staš, J., & Pleva, M. (2020). Survey of automatic spelling correction. Electronics (Switzerland), 9(10), 1–29. https://doi.org/10.3390/electronics9101670

Hurskainen (1985),Swahili Language Manager: A Storehouse for Developing Multiple Computational Applications/A:1026558915015

Hurskainen, A (2004), "HCS - Helsinki Corpus of Swahili. Compilers" *Institute for Asian and African Studies (University of Helsinki) and CSC*. No. 3, 363–397.

Jambo Spell Checker, (2004) Open Source Kiswahili Spell Checker (Sw-Tz)

James R. Curran and Raymond K. Wong (1998), *Formalisation of Transformationbased Learning,* Unpublished report, University of Sydney, New South Wales.

Jiang, Yang, & Rio (2019) "Spelling Correction of Non-Word Errors in Uyghur–Chinese Machine Translation.

Kaur J. and K. Garg, Hybrid Approach for Spell Checker and Grammar Checker for Punjabi, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 6, June 2014

Kernighan; Mark D.; Kenneth W. Church; and William A. Gale, "A Spelling Correction Program Based on a Noisy Channel Model", Proceedings of COLING 1990, 205-210, 1990.

KILINUX , (2004). Open Kiswahili Localization Project

Kukich K. (1992),Techniques for automatically correcting words in text. ACM Computing Surveys (CSUR) 24 (4), 377–439. https://doi.org/10.1145/ 146370.146380.

Kumar, A., & Singhal, S. (2016). Rule-based and statistical approaches for spell-checking in Hindi. In Proceedings of the 2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS) (pp. 1-6). IEEE.

Les premiers berbères, Entre Méditérranée, Tassili et Nili. EdisudIna-Yas, Aix-en-Provence-Alger. Hládek, D., Staš, J., Pleva, M., (2020). Survey of Automatic spelling correction. Electronics 9 (10), 1670. https://doi.org/10.3390/electronics9101670.

Levenshtein, V., (1966). Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10 (8) https://nymity.ch/sybilhunting/pdf/ Levenshtein1966a.pdf. Thierry

Li, Y., & Li, X. (2011). A phonetic-based approach for Chinese spell checking. In Proceedings of the 2011 International Conference on Computer Science and Service System (CSSS) (pp. 2254-2257). IEEE.

Mangu L. & E. Brill. (1997). "Automatic Rule Acquisition for Spelling Correction". In ICML: Proceedings of the 14th International Conference on Machine Learning.

Mangu L. and Brill E. (1997) , *Automatic Rule Acquisition for Spelling Correction, John Hopkins University*, Maryland.

Manning D. and Schutze H. (1999), *Foundations of Statistical Natural Language Processing*, The MIT press, London.

Manning, C., & Schütze, H. (1999). Foundations of Statistical Natural Language Processing. MIT Press.

Mays Eric; Fred J. Damerau; and Robert L. Mercer, "Context Based Spelling Correction", Information Processing and Management, 27(5):517–522, 1991.

Milligan G.W. and P.D. Isaac. 1980. The validationof four ultrametric clustering algorithms. PatternRecognition, 12(2):41–50

Moisan, Claude-Guy Quimper, Jonathan Gaudreault, and Sbastien Michaud (2014). A spell checker tailored to language learners. Int. J. Phytorem. 21 (1), 117–134. https://doi.org/10.1076/call.16.2.213.15881.

Msanjila P. (2005), "Problems of Writing in Kiswahili: A Case Study of Kiguru nyembe and Morogoro Secondary Schools in Tanzania", *Nordic Journal of African Studies,* Vol. 1, No. 14, 15-25

Ndiaye, M., Faltin, A.V., 2003. A spell checker tailored to language learners. Int. J. Phytorem. 21 (1), 117–134. https://doi.org/10.1076/call.16.2.213.15881.

Ng'ang'a, WKamau, C, (2010).Developing an Open source spell checker for Gikuyu

Peter Norvig (2007) "How to Write a Spelling Collector", an essay

Peterson, J.L., (1980). Computer programs for detecting and correcting spelling errors. Commun. ACM 23 (12), 676– 687.

Peterson, J L, (1986) "A Note on Undetected Typing Errors", Commun. ACM 29, 7 (July), 633-637.

Pollock, J.J., Zamora, A., (1984). Automatic spelling correction in scientific and scholarly text. Commun. ACM 27 (4), 358–368. https://doi.org/10.1145/ 358027.358048.

Prosz´eky and Kis, (1999). A word-grammar based morphological analyzer for agglutinative languages, Prosz´eky and Kis,

Proszéky, G., & Kis, G. (1999). Between Understanding and Translating: A Context-Sensitive Comprehension Tool. Author: Gábor Prószéky. Publisher: H-1126.

R.C. de Amorim. 2009. An adaptive spell checker based on ps3m: Improving the clusters of replacement words. Computer Recognition Systems 3,pages 519–526.

R.C. de Amorim. 2012. An empirical evaluation of different initializations on the number of k-means iterations. Lecture Notes in Computer Science,7629:15–26.

Rabat, Morocco. Damerau, F.J., (1964). A technique for computer detection and correction of spelling errors. Commun. ACM 7 (3), 171–176. https://doi.org/10.1145/363958.363994. Hachid, M., 2000.

Salifou, L., Naroua, H., (2014). Design of a Spell Corrector for Hausa Language. Int. J. Comput Linguistics (IJCL) 5 (2), 14–26 https://www.cscjournals.org/manuscript/Journals/IJCL/Volume5/Issue2/IJCL-56.pdf.

Shannon, (1951)N-gram Language Model/ https://web.stanford.edu/~jurafsky/slp3/3.pdf

Smith, E. R. (2010). The British National Corpus (BNC) as a Linguistic Resource for the Study of British English. British Linguistics Review, 25(3), 321-340.

Smith, J. A. (2005). The American National Corpus: A Comprehensive Resource for American English. *Linguistics Journal, 30*(2), 123-145.

Takeda, K., Nakamura, T., & Nakamura, Y. (2019). Science design methodology: a framework for designing and evaluating scientific research programs. Science and Engineering Ethics, 25(2), 487-504. doi: 10.1007/s11948-018-0027-1.

Tesha T. (2012), *Automating the classification of Swahili documents*, unpublished report, The University of Dodoma, Dodoma.

Tessema, M., & Abebe, B. T. (2014). Development of an Amharic spell-checking system using statistical and rule-based approaches. Information Technology Journal, 13(14), 2311-2320. Vol. I.

Thompson and Schleicher, (2001) Swahili Learners' Reference Grammar. African Language Learners' Reference Grammar Series, Thompson, Katrina Daly; Schleicher, Antonia Folarin ,ISBN-1-58684-115-7

Yannakoudakis, E.J. and Fawthrop, D., (1983)."The rules of spelling errors," Information Processing and Management, vol. 19, no. 2, pp. 87-99.