



University of Nairobi

School of Computing and Informatics

Master of Science Information Systems

**Open Source Spell Checker For
Bukusu Dialect**

By

Joel Bwayo

ID: P56/7348/2006

Project Supervisor: Dr. W. Ng'ang'a

University of NAIROBI Library



0439210 6

Declaration

This project as presented in this report is my original work and has not been submitted to any other university.

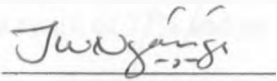
Signed: 

Name: Joel Bwayo

ID: P56/7348/2006

Date: 31/05/2011

This project has been submitted in fulfillment of the requirements of Master of Science Information Systems of University of Nairobi with my approval as supervisor.

Signed: 

Name: Dr. Wanjiku Ng'ang'a

Date: 08.06.2011

Abstract

This project describes the development of an open source spellchecker for Bukusu language using Hunspell language tools. It examines Bukusu morphology, highlighting the inflection of various parts of speech in the Bukusu language including verbs, nouns and pronouns.

The project focuses on the definition two major Hunspell files in the realization of the spell checker, i.e. the affix file (.aff) and the dictionary file (.dic). The dictionary file contains the Bukusu lexicon, whereas the affix file contains stemming rules that facilitate word suggestion using character prevalence and replacement procedures.

The functional system is aimed at being adopted in major open-source products such as OpenOffice, Google Chrome, Firefox and Thunderbird. The open-source platform used in the development of the spell checker will also allow for future improvements on the system.

The developed spell checker realizes an acceptable performance with a precision of 79.3%, a recall of 77% and an accuracy of 68.9%

Acknowledgement

I owe my deepest gratitude to my supervisor, Dr. Wanjiku Ng'ang'a, whose guidance and support throughout the course of the project enabled me to develop a better understanding of the subject.

I also offer my regards and blessings to my parents and wife who relentlessly encouraged and supported me during the entire duration of the project.

Joel Bwayo

TABLE OF CONTENTS

LIST OF FIGURES	6
LIST OF TABLES	7

CHAPTER 1

1. INTRODUCTION	8
1.1. FRAMEWORK OF CURRENT RESEARCH.....	8
1.2. PROBLEM STATEMENT.....	9
1.3. JUSTIFICATION.....	9
1.4. PROJECT OBJECTIVES.....	9
1.5. SCOPE.....	9

CHAPTER 2

2. LITERATURE REVIEW	10
2.1. THE LUHYA COMMUNITY	10
2.2. LUHYA CLANS	10
2.3. LUHYA DIALECTS	11
2.4. BUKUSU CLAN.....	11
2.5. BUKUSU ORTHOGRAPHY.....	11
2.6. RELATED WORKS.....	15
2.7. SPELLCHECKERS	15

CHAPTER 3

3. METHODOLOGY.....	20
3.1 ANALYSIS.....	20
3.2. SYSTEM DESIGN	21
3.3. CORPUS COLLECTION.....	22
3.4. IMPLEMENTATION OF THE SPELL CHECKER.....	23

CHAPTER 4

4. TESTING AND RESULTS	30
4.1. TESTING	30
4.2. EVALUATION	32

CHAPTER 5

5. CONCLUSION.....	34
5.1 OVERVIEW.....	34
5.2 LIMITATIONS.....	34
5.3 RECOMMENDATIONS.....	35
5.4 CONCLUSION.....	35

REFERENCES.....	37
-----------------	----

LIST OF FIGURES

FIGURE 3.2: System Design.....	23
FIGURE 4.1: Bukusu Spell Checker deployed in Ubuntu CLI.....	29
FIGURE 4.2: Bukusu Spell Checker deployed in Open Office.org Writer.....	30

LIST OF TABLES

TABLE 2.2: Luhya tribes and their dialects	10
TABLE 2.3: Vocabulary comparison between Luhya dialects	10
TABLE 2.5a: Noun classes	13
TABLE 2.5b: Bukusu pronouns	13
TABLE 2.5c: Bukusu pronouns in corresponding noun classes	13
TABLE 2.5d: Verb conjugation “Khusoka”	15
TABLE 2.5e: Verb conjugation “Khukula”	15
TABLE 2.5f: Verb conjugation “Khukhwiicha”	15
TABLE 3.4: Bukusu Orthography set character count	25
TABLE 4.0: Evaluation results on developer’s test set	32
TABLE 4.1: Evaluation results on Mr. Mutonyi’s test set	32

Chapter 1

1. Introduction

1.1 Framework of Current Research

Much of a peoples' cultural and intellectual heritage, is contained within and expressed through the local language. Recent research by various organizations has established a link between the localization of ICTs and the preservation of indigenous language and culture. Localization, described as the adaptation of ICT to the language and culture where it is used, allows that cultural pressure to be reduced eliminated or even reversed (ANLoc Conference 2006). One of the leading organizations involved in this research, ANLoc (The African Network for Localization) has carried out several projects across Africa with a mission to empower Africans to participate in the digital age by eliminating language barriers primarily through the localization of ICT. One of their more successful projects has been the development of localized spellchecking systems in assorted indigenous African languages. The Bukusu spellchecker will therefore be an indirect extension of the research currently being carried out by ANLoc and similar organizations.

1.1.1 Language and ICT

Over millennia the transfer of cultural knowledge has customarily taken place in written and oral form. Within traditional African communities this knowledge has been passed from generation to generation mainly through the latter method and for this reason language has played a pivotal role in this process.

In the past three or so decades ICT has allowed documentation and sharing of knowledge to happen more easily and accurately. In the absence of localized ICTs however, the process of documenting indigenous knowledge has to happen via the influence of a foreign language. The foreign influence tends to impact negatively on the accuracy and integrity of the indigenous knowledge as some key details are often omitted or rephrased due to lack of direct translations. This factor has acted as an impediment to those intending to preserve this knowledge via the use of ICT and has resulted in limited availability of digital resources for a majority of indigenous communities.

The current trend of, high dependence on digital information by the young (5-25 years) urban population, means that they have an inherent disadvantage when attempting to access digital resources pertaining to their indigenous cultures. The scarcity of these language-specific resources has led to a waning interest by this segment of the population in their cultures, leading to the steady erosion of these cultures over time.

A peripheral effect of the absence of localized ICTs, is the slow uptake of ICT among the rural population thus hindering their access to information.

The Bukusu spellchecking system will provide a technological platform that will act as an incentive for increased accurate documentation of cultural resources through the use of digital academic texts, blogs, novels, poems and short stories. In addition it will bridge the prevailing digital divide that exists among the rural population.

1.2 Problem statement

The lack of a spellchecking system for the Bukusu language has contributed to the inadequate documentation of indigenous knowledge thereby playing a role in the reduced uptake of the language among the current and upcoming urban generation.

1.3 Project Objectives

The main objective of this project is to develop a tool that will be used to spell check text written in the Bukusu dialect, and this will eventually facilitate accurate documentation of more Bukusu linguistic and cultural resources. The proposed tool will be implemented as a feature of major Open Source products such as Google Chrome, Firefox, Thunderbird, and OpenOffice.

Secondary objectives include:

- i. Deployment of a suggestion mechanism within the spell checker via implementation of various Natural Language Processing (NLP) algorithms
- ii. Development of a Wordlist containing correctly spelt Bukusu words.

1.4 Justification

The development of the proposed system can be validated by the following reasons:

- i. There is currently no existence of a spell checker for the Bukusu dialect, therefore it is difficult to document the rich culture of the community.
- ii. The proposed system aims to facilitate the formalization of linguistic knowledge thereby enabling increased uptake of the language in various learning institutions.
- iii. The growing number of Luhya speakers (approximately 5.3 Million) within Kenya and in the Diaspora means that there is a substantial growing interest in the language and its preservation.
- iv. The proposed system will assist authors, bloggers, poets and copywriters to publish their works in local dialect thus increasing the awareness about the Bukusu language and culture among other local communities.
- v. The deployment of the proposed spellchecker within open source email clients such as Thunderbird email client will facilitate accurate correspondence in local dialect.

1.5 Scope

This system will be limited to providing a spellchecking component with a suggestion mechanism, however it will not include a grammar checking facility and a thesaurus function.

The proposed spellchecker will also be limited in scope to focus on the Bukusu dialect and hence will not cover any of the 15 other sub-groups in the larger Luhya community.

Chapter 2

2. Literature review

2.1 The Luhya community

The Luhya are Kenya's second largest ethnic community accounting for 14% (5.3 Million) of the country's total population. The Luhya principally occupy the Western province of Kenya however, substantial populations have settled in the Kitale area of the Rift Valley Province. A small population of the community numbering about 50000 is also found in neighboring Uganda.

2.2 Luhya clans

The Luhya community is composed of 16 major distinct clans (tribes). These clans include Bukusu, Idakho, Isukha, Kabras, Khayo, Kisa, Marachi, Maragoli, Marama, Nyala, Nyole, Samia, Tachoni, Tiriki, Tsotso, and Wanga.

The table below depicts the Luhya dialects and the geographical distribution on the 16 clans.

Table 2.2: Luhya tribes and their dialects

Luhya Tribe	Luhya dialect	Region
Bukusu	Lubukusu	Bungoma (Kenya)
Idakho	Luidakho	Kakamega (Kenya)
Isukha	Luisukha	Kakamega (Kenya)
Kabras	Lukabarasi	Kakamega (Kenya)
Khayo	Olukhayo	Busia (Kenya)
Kisa	Olushisa	Butere, Mumias (Kenya)
Logoli (Maragoli)	Lulogooli	Maragoli, Vihiga (Kenya)
Marachi	Olumarachi	Busia (Kenya)
Marama	Olumarama	Butere, Mumias (Kenaya)
Nyala	Lunyala	Busia (Kenya)
Nyole	Ugandan Nyole Kenyan Nyole	Tororo (Uganda) Vihiga (Kenya)
Samia	Lusamia	Busia, Kakamega (Kenya), Tororo (Uganda)
Tachoni	Lutachoni	Lugari, Malava (Kenya)
Tiriki	Lutirichi	Vihiga (Kenya)
Tsotso	Olutso	Kakamega (Kenya)
Wanga	Oluwanga	Butere, Mumias (Kenya)

2.3 The Luhya dialects

Each of the 16 Luhya clans speaks a dialect distinctly different from the others in vocabulary, though some are mutually intelligible.

The Idakho, Isukha, and Tiriki speak essentially the same dialect. These are largely intelligible with Bukusu, Logoli, Nyala, and Kenyan Nyole. Other varieties with high degrees of mutual intelligibility are Ugandan Nyole, Samia, Wanga, Marama, Kisa, Marachi, Khayo, Tachoni and Kabarasa. Bukusu is also intelligible with Masaba, which is not considered Luhya.

There are similarities as well as differences in vocabulary between the Luhya dialects. In some cases the noun stems are entirely dissimilar and this is illustrated in the following table.

Table 2.3: Vocabulary comparison between Luhya dialects

English	Bukusu	Kisa	Maragoli	Nyole	Wanga
I (me)	Ese	Eshie	Nzi	Ise	Esie
Words	Kámakhu.a	Amakhuwa	Makuva	amang'ana,	amakhuwa
Chair	Síisala	eshifumbi	Ndivi	Indebe	Eshisala
Head	Kúmurwe	Omurwe	Mutwi	Omurwe	Om'rwe
Money	Líípéesa	Amapesa	Mang'ondo	Amang'ondo, Am'mondo, Etsilupia	Amapesa, Irupia

2.4 The Bukusu clan

The Bukusu are currently classified as the largest of the 16 clans, making up 17% of the entire Luhya population. The Bukusu live mainly in Bungoma district, which borders Uganda to the west, Busia to the southwest, Trans Nzioa to the east, and Kakamega district of Kenya to the southeast. Across the border in Uganda live the Masaba and the Gisu, both closely related to the Bukusu by a shared language and a common culture. Inter-marriage between the Bukusu and these Ugandan tribes is very common and is, in fact, encouraged by the respective communities. As a result, many Bukusu have close relatives amongst the Gisu and Masaaba, and vice versa.

2.5 Bukusu Language

2.5.1 Tones

Bukusu is a tonal language with words marked using high and low tones. In the present transcription system, long vowels are marked as a sequence of two vowels, e.g., (aa) whereas, a syllable boundary between two vowels is marked by ' '. High tones are marked with an acute accent, however low tones remain unmarked (Marlo 2008 p.3). The high tone occurs frequently in the Bukusu vocabulary, however there can only be a maximum of two instances of this high tone in a given word. This differs from the

English system of stressed vowels. Another variance is that, Bukusu tone mainly correlates to pitch and less to amplitude and has no connection to vowel length.

The Bukusu-English dictionary referenced in this project contains diacritic markings for high tones only, leaving low tones unmarked. This transcription convention is cascaded down to the design of the spellchecker, and therefore unmarked low tones are not flagged by the system as being incorrect.

Example:

Chíinyenyi >> Vegetables

Chíinyenyi >> Food eaten as a relish with a basic starch dish (meat, fish etc)

Ēekholo >> Clan

Eekholo >> ShriII cries of alarm

2.5.2 Morphology

According to Wikipedia, linguistic morphology refers to the identification, analysis and description of the structure of morphemes and other units of meaning in a language like words, affixes, and parts of speech and intonation/stress, implied context.

Most of the inflectional morphology of Bantu languages is encoded in nouns and verbs. It is commonly assumed that patterns established for nouns also apply to adjectives, unless the contrary is explicitly stated, as adjectives tend to copy the prefix structure of their head nouns. (Mutonyi 2000 p. 4)

2.5.2.1 Nouns

Bukusu nouns can be classified into two categories i.e. un-derived and derived nouns. The former are composed of named entities whereas the latter are formed through the use of prefixes of diminutive, augmentative or collective on an un-derived noun.

Example:

Omusoreri: <boy/young man>

Basoreri: <boys>

Khasoreri: <little boys>

Noun Classes

There are 6 documented noun classes in the Bukusu language i.e. Li-/Kama-, Kumu-/Kimi-, Si-/Bi-, E-/Chi-, Lu-/Chin-, Om-/Ba-.

Each noun class is identified by its singular and plural class prefixes added to the noun stems.

Table 2.5a: Noun classes

Noun class	Singular	Plural
Li-/Kama-	Litore <banana>	Kamatore
Kumu-/Kimi-	Kumusala <tree>	Kimisala
Si-/Bi-	Siifuba <chest>	Biifuba
E-/Chi-	Enyanga <day>	Chinyanga
Lu-/Chin-	Luliimi <tongue>	Chiinimi
Om-/Ba-	Omundu <person>	Babandu

When a class is modified by one or more adjective, it takes the class prefix of the governing noun. The same rule applies to nouns that are not overtly marked with class prefixes e.g. 'Maji' <Mother>.

1. The Om-/Ba- contains all nouns that identify different sorts of human beings.
2. The li-/Kama- class contains two deverbal nouns i.e. 'Khukhwira' <kill> and 'Khufwa' <death>.
3. The lu-/Chin- contains a noun related to a verb, 'Khukana' <to tell>.

2.5.2.2 Pronouns

Subject pronouns in Bukusu are usually optional and are mainly used for emphasis.

Table 2.5b: Bukusu Pronouns

	Singular	Plural
1 st Person	Ese	Efwe
2 nd Person	Ewe	Enywe
3 rd Person	Nije	Nibo

A possessive pronoun follows a possessed noun and must agree with it in class.

Table 2.5c: Bukusu Pronouns in corresponding noun classes

Noun class	Li-/Kama-		Kumu-/Kimi-		Si-/Bi-		E-/Chi-		Lu-/Chin-		Om-/Ba-	
	Sg.	Pl.	Sg.	Pl.	Sg.	Pl.	Sg.	Pl.	Sg.	Pl.	Sg.	Pl.
1P. Sg. Or	Ijange	Kange	Kw-	Kj-	Sj- Bj-	j- Chi-	Lw-	Ch-	W-	B-		
	Ijase	Kase										
2P. Sg.	Ijooo	Kooo	Kw-	Kj-	Sj- Bj-	j- Chi-	Lw-	Ch-	W-	B-		
3P. Sg.	Ijewe	Kewe	Kw-	Kj-	Sj- Bj-	j- Chi-	Lw-	Ch-	W-	B-		
1P. Pl.	Ijefwe	Kefwe	Kw-	Kj-	Sj- Bj-	j- Chi-	Lw-	Ch-	W-	B-		
2P. Pl.	Ijenjwe	Kenjwe	Kw-	Kj-	Sj- Bj-	j- Chi-	Lw-	Ch-	W-	B-		
3P. Pl.	Ijabwe	Kabwe	Kw-	Kj-	Sj- Bj-	j- Chi-	Lw-	Ch-	W-	B-		

2.5.2.3 Verbs

All Bukusu verbs are marked with a prefix or suffix or a combination of both indicating the tense. All verbs marked with a prefix morpheme preceding any tense prefix identify the verb subject.

Verb Affixation

Verb-marker morphemes

There are two main sets of subject marker morphemes in the bukusu language:

1. /en- o- a- khu- mu- ba- li- ka-/
2. /na- wa- a- khwa- mwa- ba- ija- ka-/

The remaining verb marker morphemes are aspectual and tense markers.

/li- khe- kho- kha- -nga makhu- -ile la- -e ne no na/

Affixation for Bukusu verbs is done using two main rules:

Rule 1: Subject verb agreement

Bukusu verbs are affixed depending on the subject.

Example:

- a) Omwaana **acha** >> the child went
- b) Babaana **bacha** >> the children went

The subject in the examples above changed from singular in (a) to plural in (b) and subsequently the prefix was adjusted to correspond.

Rule 2: Object verb agreement

Verb prefixes are affixed depending on the object.

Example:

- Omuundu **kabaapa** >> Someone beat them (relates to human beings)
Omuundu **kachiipa** >> Someone beat them (relates to animals or inanimate objects)

Verb Conjugation Classes

Majority of Bukusu verbs fall into two tonal conjugation classes i.e. 'Khusoka' <to swim> and 'Khukula' <to buy>. Other verbs with stems starting with Khw e.g. 'Khukhwiicha' <to come> and 'Khukhwiira' form a third conjugation class.

Table 2.5d: Verb conjugation “Khusoka”

Khusoka	
Ese esoka (I swim)	Efwe khusoka (We swim)
Ewe osoka (You swim)	Enywe musoka (you swim pl.)
Nije asoka (He/She swims)	Nibo basoka (They swim)

Table 2.5e: Verb conjugation “Khukula”

Khukula	
Ese engula (I buy)	Efwe khukula (We buy)
Ewe okula (You buy)	Enywe mukula (You buy pl.)
Nije akula (He/She buys)	Nibo bakula (They buy)

Table 2.5f: Verb conjugation “Khukhwiicha”

Khukhwiicha	
Ese niicha (I come)	Efwe khwiicha (We come)
Ewe wiicha (You come)	Enywe mwiicha (You come pl.)
Nije kecha (He/She comes)	Nibo becha (They come)

2.6 Related Works

Former students as well as current faculty from the School of Computing at the University of Nairobi have carried out a number of computational linguistics-based efforts.

Chege et al. [2] developed an open source spell checker for Gikuyu using Hunspell language tools. Munyao [5] developed a word processor with spell checking functions for Akamba. Muriithi [6] developed a dictionary-based word processor with a spell checker for Gikuyu. Otieno [8] developed an open source Luo spell checker using Hunspell tools. Chege [1] developed a dictionary-based based text editor and spell checker for Gikuyu with diacritic support.

In addition, several other natural language processing (NLP) efforts have been carried out at the same institution. These include text-to-speech systems, morphological analysis and machine translation for Gikuyu. (Chege 2009 p.2)

2.7 Spell checkers

2.7.1 Definition

A spell checker, in computing terms, refers to an application program that flags words in a document that may not be spelled correctly. Spell checkers are sometimes implemented as stand-alone applications capable of operating on a block of text, however recent trends have seen them deployed as part of larger applications, such as, web browsers, email clients, electronic dictionaries, word processors and search engines.

2.7.2 History of Spellcheckers

First Generation Spell Checkers

The First Spell checker was developed in the 1970's by a team of linguists from George Town University on behalf of IBM Corporation. This system, like many others to follow it in this decade was primarily deployed on Mainframe computers as standalone application.

Second Generation Spell Checkers

The 1980's marked the entry of Spell checkers among the Personal Computers initially on TRS-80 computers and subsequently IBM PCs. Some of the major developers in this era included Maria Mariani, Soft-Art, Microlytics, Proximity, Circle Noetics. They were primarily responsible for spurring the uptake of the standalone spell checkers in PCs, Unix systems, Macintosh machines.

Developers of popular word processors such as WordPerfect and WordStar were among the first to integrate spell checkers in their packages in the Mid 1980s effectively ending the dominance of standalone spell checkers.

Third Generation Spell Checkers

Recent trends have seen the extension of spell checking components beyond word processors to implementation in; Web browsers Mail clients, Blogs and mobile phone applications. Spell checkers have also become increasingly sophisticated incorporating grammar checking and thesaurus functions.

2.7.3 Spell Checker operation

Simple spell checkers operate on individual words by comparing each of them against the contents of a wordlist, possibly performing stemming on the word. If the word is not found it is considered to be an error, and an attempt may be made to suggest a word that was likely to have been intended.

2.7.4 Design

A spelling checker is programmed on how to evaluate the distance between a misspelled word and the words in its vocabulary. Words whose evaluated distance is the smallest are offered as candidates for replacement.

A spell checker usually consists of two components:

- i. A set of routines for scanning text and extracting words
- ii. An algorithm for comparing the extracted words against a known list of correctly spelled words (i.e., the wordlist).

The scanning routines sometimes include language-dependent algorithms for handling morphology. The wordlist might contain just a list of words, or it might also contain additional information, such as hyphenation points or lexical and grammatical attributes.

Exceptions to the paradigm depicted above are spell checkers which generate suggestions based solely on statistical information, for instance using n-grams, as described below. This approach usually requires a lot of effort to obtain sufficient statistical information and may require a lot more runtime storage. These methods are not currently in general use. In some cases spell checkers use a fixed list of misspellings and suggestions for those misspellings. This less flexible approach is often used in paper-based correction methods.

2.7.5 Spell checking methods

Many different methods are used in designing spell checkers. They can be classified into two broad categories, independent and dependent spell checking methods.

a) Independent Spell Checking Methods

Independent spell checking methods do not use a wordlist or vocabulary; instead they use statistical means to detect misspelled words, hence the term independent. They come in two main varieties:

Token Lists

This approach involves identifying all the distinct words in a file of text and storing the frequency of each word. Words with lower frequencies are identified as being potentially misspelled.

***n*-grams (Di-grams and Tri-grams)**

This method extends the token list concept. Using a large corpus of text from the desired language, the frequency of all two-letter pairs (di-grams) or three-letter triplets (trigrams) is calculated. A peculiarity measurement is then given to each word in the text file being spell checked based on the frequency of the di-grams or tri-grams found in the word. Words with a high peculiarity measurement are tagged as being potentially wrong.

b) Dependent Spell Checking Methods

Dependent spell checking methods involve the use of a vocabulary or wordlist. This method increases the accuracy of spell checking systems immensely, in comparison to those designed based on independent spell-checking methods.

Dictionary Look-Up

As the name suggests, a wordlist of correctly spelled words is maintained by the system. The spell checker simply runs through this list to see if the words in the text file being checked appear. It then tags as incorrect any words that are not found.

2.7.6 Suggestion Algorithms

Near Miss Strategy

The near miss strategy is a fairly simple way to generate suggestions. Near miss takes the approach that the user didn't necessarily misspell the word but rather they mistyped it. Two words are considered near if they can be made identical by inserting a blank space, interchanging two adjacent letters, changing one letter, deleting one letter or adding one letter. If a valid word is generated using these techniques, then it is added to the suggestion list. The near miss strategy however, doesn't provide the best list of suggestions when a word is truly misspelled.

Phonetic Strategy

To understand the phonetic strategy, phonetic code needs to be defined. A phonetic code is a rough approximation of how the word sounds. Each character in the code represents a sound. It is important to note that the phonetic code does not indicate how to pronounce the word; it is merely a representation of how it sounds.

The phonetic strategy is executed by comparing the phonetic code of a misspelled word to all the words in the word list. If the phonetic codes match, then the word is added to the suggestion list.

The process above becomes more complicated when affix compression is introduced. An affix compressed wordlist only contains base words i.e. words without prefixes or suffixes. The phonetic code of the misspelled word can't be simply compared to the word list because the misspelled word may or may not be a base word. To solve this issue, all affix rules that pass the conditions of the rule from the misspelled word are removed and the resulting string is added to a possible base word list. Another important note is that the possible base word list is not a list of real words. It is only a list of strings that can be made by removing the affix rules from the misspelled word.

After compiling a list of possible base words from the misspelled word, a phonetic code is generated on them and a comparison is done on those codes with the list of base words. If one of the codes matches the base word code, that word is added to the list of suggestions. Because all the affix keys are removed and only the base words are compared, an expanded base word could result in the correct word. Applying all the affix rules to the matching base word expands it, thus deriving a list of all the possible words. That list is then appended to the suggestion list.

The Edit Distance algorithm

After generating a list of suggestions, they need to be ranked according to the likelihood of their correctness. The Edit Distance Algorithm is then applied to the list to generate accurate ranks. The Edit Distance Algorithm is defined as; the smallest number of insertions, deletions, and substitutions required changing one string into another. The word with the smallest Edit Distance is ranked first on the suggestion list. In some spell checkers the Edit Distance Algorithm implementation has one slight modification in that it adds an extra edit distance if the first character and last character don't match. The

rational behind this is that users generally tend to get the first character and last character correct when trying to spell a word.

2.7.7 Hunspell Language Tools

Hunspell is a set of open source tools and utilities used for development and testing of spellcheckers and morphological analyzers. The main goal of Hunspell and its predecessors is to compress a language's lexicon into a manageable size. Hunspell is an enhancement of its predecessors Ispell and MySpell. Hunspell is the official spellchecker being used in OpenOffice and Mozilla products.

Hunspell was built to provide important functionalities not available in MySpell. It was built to support languages with rich morphology, including complex prefixes and compounding. Hunspell is built to support up to three prefixes and suffixes each. In addition, it also supports the use of circumfixes. Circumfixation dictates that a certain suffix can only be used together with a given prefix.

Hunspell requires two files to define the language that it is spellchecking. The first file is a dictionary containing words for the language, and the second is an "affix" file that contains stemming rules for the words contained in the dictionary.

A dictionary file (*.dic) contains a list of words, one per line. The first line of the dictionaries (except personal dictionaries) contains the word count. Each word may be optionally followed by a slash ("/") accompanied with one or more flags, which represents affixes or special attributes. Default flag format is a single (usually alphabetic) character. In a Hunspell dictionary file, there is also an optional morphological field separated by tabulator.

Key Features

- i. Extended support for language peculiarities; Unicode character encoding, compounding and complex morphology.
- ii. Improved suggestion using n-gram similarity, rule and dictionary based pronunciation data.
- iii. Morphological analysis, stemming and generation.
- iv. Hunspell is based on MySpell therefore also works with MySpell dictionaries.
- v. C++ library under GPL/LGPL/MPL tri-license.

Chapter 3

3. Methodology

3.1 Analysis

The proposed system will be developed using the Rapid Application Development (RAD) approach. This technique emphasizes on, extensive user involvement in the rapid and evolutionary construction of working prototypes of a system to accelerate the system development process. RAD is sometimes called a spiral approach because the developer repeatedly spirals through the phases to construct a system in various degrees of completeness and complexity. (Maurer & Martel 2002)

Rapid Application Development (RAD) was selected as the methodology of choice mainly for the following reasons:

- a. To reduce development time as prototyping accelerates requirements analysis and system design.
- b. Prototypes are interactive models that can be seen and experienced by the user.
- c. Errors and omissions tend to be detected earlier in prototypes than in system models.
- d. The iterative approach of the methodology enables inevitable changes in the system to be factored in during system development.
- e. Testing and training are carried out during the entire prototyping process resulting in quicker commissioning of the system.
- f. User feedback can lead to more innovative solutions.

3.1.1 Requirements gathering

The developed system aims at being deployed on a pre-existing system and therefore a majority of the requirements had already been captured. The supplementary requirements perceived to be key to this project include:

1. The inclusion of Tonal marking (diacritics) within the wordlist.
2. The inclusion of a personal dictionary.

3.1.2 Scope analysis

Various factors such as, technical resources and time constraints contributed to the containment of the project scope by the developer. The resultant scope was as follows:

1. Development of a spell checking system for Bukusu dialect.
2. Exclusion of thesaurus functions.
3. Exclusion of grammar checking facilities.

3.1.3 Problem analysis

The inception of the Bukusu spell checker project was necessitated by three main fundamental shortcomings:

1. The lack of a Bukusu spell checking system on either proprietary or open source platforms.
2. Bukusu is a resource scarce language.
3. There is little digital documentation on Bukusu language.

3.1.4 Decision analysis

During the course of the planning and implementation phases, the developer arrived at the following key decisions.

1. The inclusion of diacritics markings for high tones within the wordlist.
2. The omission of diacritic markings for low and falling tones.
3. The exclusion of CorpusCatcher tool during corpus collection.
4. Bias in inclusion of religious material (Bible verses & Hymns) for populating the test corpus.

3.2 Proposed System Design

The developed spell checking system operates at the user's request, checking an entire block of text at once, or alternatively checking individual words as input by the user. Any errors encountered by the system are flagged and conveyed to the user. A suggestion list containing suitable replacements to the incorrectly spelt word(s) is then presented to the user for action. The user will subsequently have three choices:

- i. Ignore the suggested words
- ii. Replace the incorrectly spelt word with an option from the suggestion list.
- iii. Add a new (flagged) word to the personal dictionary.

After selecting the action the necessary changes, if any, will be effected to the document.

The major components of the developed spell checker system are:

3.2.1 Graphical User Interface (GUI)

The GUI is responsible for capturing the users input and conveying it to the spell checker for analysis. In this project the GUI of choice was the OpenOffice.org word processor also known as Writer. This interface allows the user to test individual words as well as blocks of text, for errors.

3.2.2 Spell checker

The spell checker mainly consists of two major components, i.e. a set of scanning routines and an algorithm that compares input text against a wordlist. When the spell checker receives input from the user interface it invokes a set of scanning routines. These routines are responsible for scanning the input text and extracting words. The algorithm then compares the extracted words against those contained in the wordlist. It then flags words that are not found in the wordlist as incorrect.

3.2.3 Wordlist

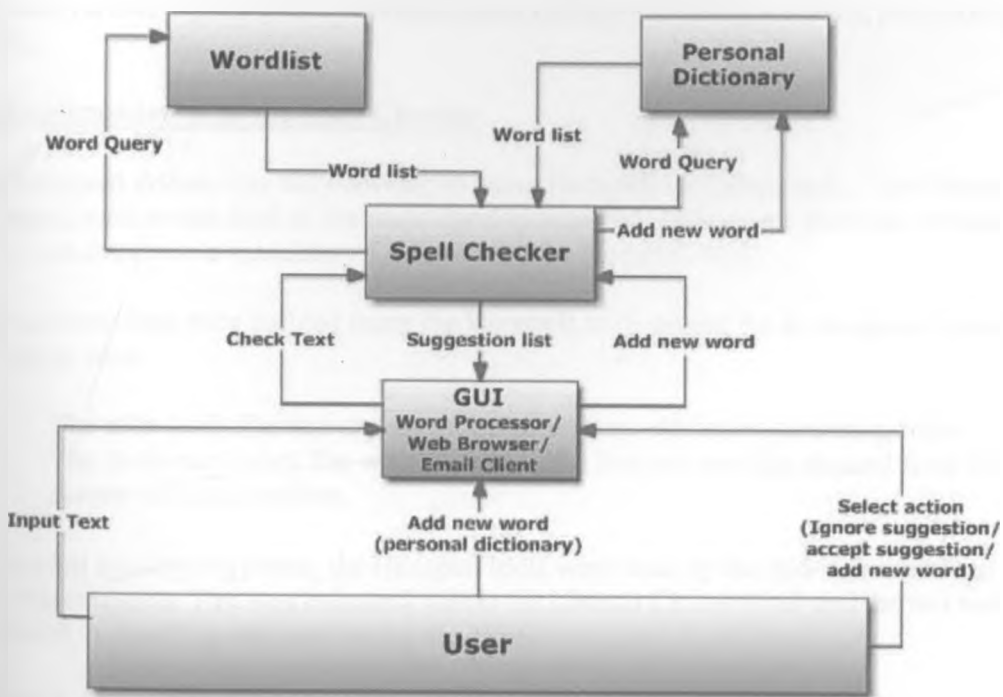
The wordlist contains a comprehensive collection of approximately 6200 correctly spelt words from the Bukusu vocabulary. The wordlist also contains flags that correspond with stemming rules defined within the affix file. The flags are used to determine the type of affixation required for a given word.

3.2.4 Personal Dictionary

This component stores words that are not captured in the primary wordlist yet are perceived to be correct by the user. New words are appended to the personal dictionary after the user opts to include a word that has been flagged as erroneous by the system.

Figure 3.2 below illustrates the spell checking operation.

Figure 3.2. System design



3.3 Corpus Collection

A Majority of the words contained in the developed wordlist are derived from a digital Bukusu-English dictionary, which comprises of approximately 6,200 words. The test corpus was extracted mainly from religious literature including, the Bukusu bible and hymnbooks. New words obtained from the test corpus were incorporated within the wordlist after completion of the testing phase.

3.3.1 The Bukusu-English Dictionary

The present Bukusu-English dictionary was available in a digital format and reflects revisions implemented as recently June 2008, by Aggrey Wasike. These revisions placed special emphasis on tone marking. The dictionary has since undergone light editing by Michael Marlo. As mentioned above, this resource was a major contributor of the overall corpus.

3.3.2 Printed Bukusu Literature

The test corpus was obtained from printed sources such as Bukusu Bibles and Hymnbooks using Optical Character Recognition (OCR) software. OCR is described as the mechanical or electronic translation of scanned images of handwritten, typewritten or printed text into machine-encoded text. For the purposes of this project the OCR of choice is Microsoft Office Document Imaging (MODI).

The pages from the Bible and hymnbooks were scanned and the resulting images were input into the OCR software. The extracted passages were then used as test data to evaluate various aspects of the developed spell checker including accuracy, precision and recall.

3.4 Implementation of the Spell Checker

The proposed system was fully developed using Hunspell Language tools. The Ubuntu operating system was used as the main development and deployment platform, because a UNIX environment is mandatory for deploying the Hunspell tools.

Two primary files were defined using the Hunspell tools during the development phase and these were:

- i. The affix (.aff) file that contains a list of Bukusu affixation/stemming rules.
- ii. The dictionary (.dic) file, which contains the Bukusu wordlist derived from the corpus collection efforts.

During the prototyping phase, the Hunspell tools were used by the end-user mainly to user requirements. This was executed within the Ubuntu CLI terminal and the test was restricted to checking one input string at a time.

After the development phase was complete, the system was deployed on an Open Source Word Processing application called OpenOffice.org Writer. This application provided the interface from which end-users could input blocks of text that were to be checked for spelling errors. The test corpus extracted from digital and printed literature was used for testing the system at various stages of development.

3.4.1 Hunspell Language-specific Setup

The Hunspell tools can be run on both Unix and Windows platforms, however the windows build has to be installed on Cygwin which a Windows utility that simulates a UNIX environment. Once downloaded and installed, Hunspell was set up for Bukusu

language. The character support was changed to UTF-8 to cater for the diacritics. The Flag was then changed to a number, because Bukusu verbs generate numerous affix rules.

SET UTF-8
FLAG num

The next step involved enabling complex prefixes as Bukusu has more than one level of prefixes and suffixes. Circumfixation is also declared by setting up FLAGS that are to be used during circumfixation

COMPLEXPREFIXES
NEEDAFFIX 001
CIRCUMFIX 002

3.4.2 Suggestions Component

In Hunspell, the suggestion component is implemented in the affix file. It employs both dependent and independent spellchecking methods i.e. dictionary look-up and *n*-grams respectively. The dictionary look-up is performed using the Levenshteins Distance algorithm. Levenshtein Distance (LD) is described as a measure of the dissimilarity between two strings, which can be defined as the source string (*s*) and the target string (*t*). The distance is the number of deletions, insertions, or substitutions required to transform *s* into *t*. In Hunspell 's' represents the user's text, whereas 't' represents the corresponding correct word in the wordlist. A smaller distance between 's' and 't', results in a higher rank on the suggestion list.

Hunspell uses two sections of the affix file to generate inputs for suitable character insertions and substitutions, i.e. the TRY command and the REPLACE (REP) command. The TRY command lists the language's orthography set and provides the LD algorithm with inputs for single character insertions and substitutions. A more frequently used character has more weight during suggestions. A script was used to rank characters within the set according to their frequency within the developed wordlist. The following table illustrates the frequency of each character.

Table 3.4: Bukusu Orthography set character count

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	
7778	2157	516	435	3220	423	670	4601	5100	140	6031	3542	2312	2809	2632	
p	r	s	t	u	v	w	y	z	á	é	í	ó	ú	ô	û
246	827	2211	668	6568	1	1343	1702	1	789	697	1358	416	3061	7	1

Below is the resultant output for the TRY command.

TRY aukihleúnomsbyiwráégtcdföpjôúvzAUKIHLENOMSBYIWRGTCDFPJVZ '

The second command used in the suggestion component is the REPLACE (REP) command. The command is a fixed list of the most common misspelled *n*-grams and their replacements within the Bukusu vocabulary. The major *n*-grams are a result of influence

from different dialects, foreign languages and also by differences in written word from the spoken word. In this case, the *n*-grams were derived from two major sources, i.e. the test corpus and references from the spoken word. Majority of the *n*-grams extracted from the test corpus resulted from a consistent omission of diacritic markings e.g. use of 'e' in place of 'é'. The remaining *n*-grams were a representation of the disparity between written and spoken Bukusu words e.g. 'b' (written) > 'v' (spoken). Below is an excerpt from the affix file illustrating the REP command.

REP 25

REP a á

REP e é

REP i í

REP u u ú ú

REP o ó

REP o ô

REP u ũ

.....

REP v b

3.4.2 Noun Component

The noun component is implemented in two parts, namely the derived nouns and the un-derived nouns. Un-derived nouns have a class that consists of optional prefixes of diminutive, augmentative, locative and collective.

PFX	55	Y	2	
PFX	55	ómw	khákh	ómw
PFX	55	ómw	bíby	ómw
PFX	56	Y	2	
PFX	56	ómu	kháakh	ómu
PFX	56	ómu	bíi	ómu
PFX	94	Y	1	
PFX	94	ó	mú	ó
PFX	341	Y	1	
PFX	341	ó	á	ó

Inside Dictionary file

ómwaana/55,94,341

ómukhaana/56,341

Derived nouns are formed through circumfixation. The CIRCUMFIX is command used to enforce circumfixation. Example is shown:

PFX	343	Y	1		
PFX	343	sís	bíb	sís	/002
SFX	344	Y	1		
SFX	344	u	wiibyó	u	/343,002

Inside Dictionary file

sísyaangu/344

Expected output: bíbyaangwiibyó

3.4.2 Verb Component

In Bukusu, new words are derived from already existing ones by applying one of several affixation procedures. For instance, a new verb is derived when a case extension is suffixed to an existing verb base. Thus, *kul-a* 'buy' becomes *kul-il-a* 'buy for/with'.

3.4.2.1 Derivation of verbs

The derivation of "new" verbs in Bukusu is accomplished through two kinds of suffixation.

The first involves suffixing a "thematic extension" after the verb base or stem and before the verb's final vowel. The other procedure happens when a verbalizing extension attaches to a noun or adjective (Kanyoro 1993).

Thematic extensions mark special relationships between verbs and their subject or object noun phrases. These extensions include applied, causative, passive, reciprocal, stative and reversive.

#Applied

SFX	345	Y	2		
SFX	345	á	éla	á	
SFX	345	a	ela	a	

Example:

Khuukeenda/345 (To walk) >> Khuukeendela (Walk for/with)

Khúuteekhá/345 (To cook) >> Khúuteekhélá (cook for/with)

#Causative

SFX 210 Y 2
SFX 210 a ya a
SFX 210 e ye e

Example:

Khuulima/210 (To cultivate) >> Khuulimya (To cause to cultivate)

Bálime/210 >> Bálímye

#Passive

SFX 346 Y 4
SFX 346 ya iibwá ya
SFX 346 a eebwá a
SFX 346 a eebwá a
SFX 346 ya iibwá ya

Example:

Khúulya/346 (to eat) >> Khúuliibwá (to be eaten)

Khúura/346 (to place/put) >> Khúureebwá (to be put/placed)

#Reciprocal

SFX 282 Y 1
SFX 282 a ana a

Example:

Khuuloma/282 (to say) >> Khuulomana (to quarrel)

#Reversive

SFX 238 Y 2
SFX 238 á úlá á
SFX 238 a ula a ~

Example:

Khúufuungá/238 (to close) >> Khúufuungúlá (to unlock)

#Stative

SFX 250 Y 2
SFX 250 á ékhá á
SFX 250 a ekha a

Example:

Khüuremä/250 (to chop) >> Khüuremékhá (be choppable)

Chapter 4

4. Testing & Results

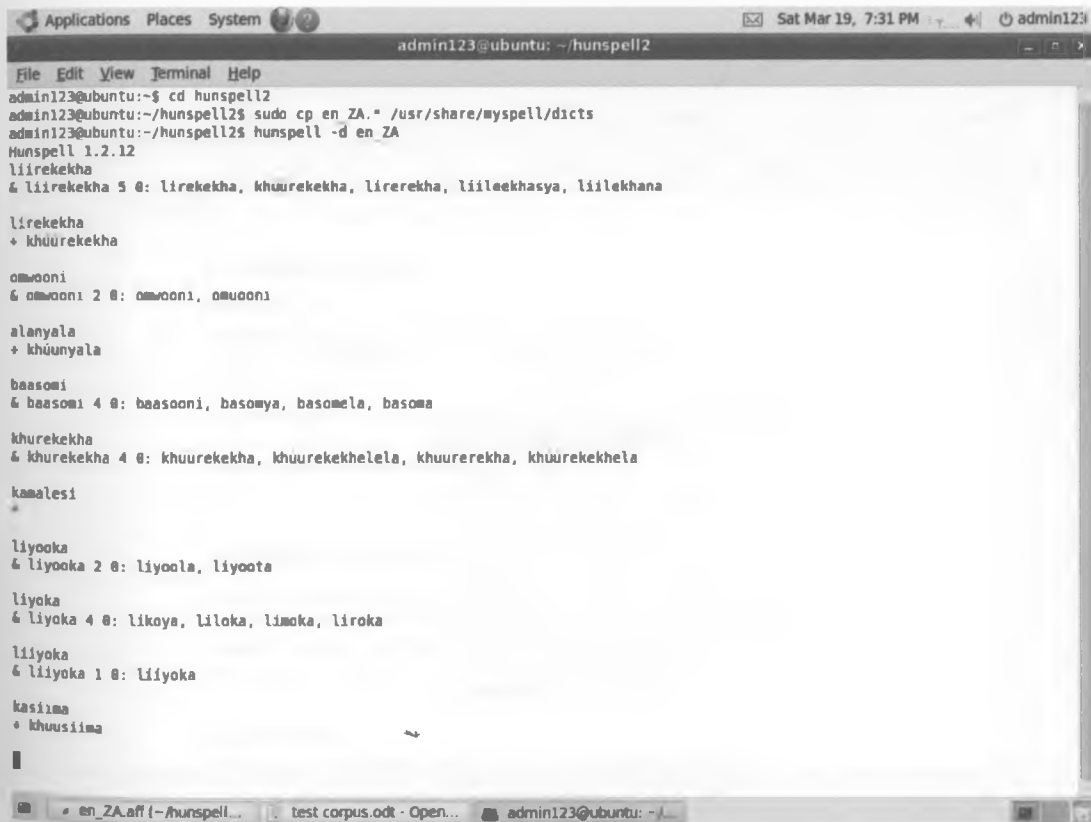
4.1 Testing

The Spell checker was tested in two main stages, firstly during the prototyping phase and secondly, after completion of the development.

The first phase of testing was carried out using the Hunspell tools operated from the Command Line Interface (CLI) of the Ubuntu operating system. This phase of testing was used to establish various user requirements of the spell checker, and was performed iteratively. This interface was limited to testing one input string at a time, and therefore would not suffice in testing blocks of text in the latter stages of the project.

Below is a screenshot of the Bukusu spell checker running within the Ubuntu CLI.

Figure 4.1 Bukusu Spell Checker deployed in Ubuntu CLI



```
Applications Places System Sat Mar 19, 7:31 PM admin123@ubuntu: ~/hunspell2
File Edit View Terminal Help
admin123@ubuntu:~$ cd hunspell2
admin123@ubuntu:~/hunspell2$ sudo cp en ZA." /usr/share/myspell/dicts
admin123@ubuntu:~/hunspell2$ hunspell -d en ZA
Hunspell 1.2.12
liirekekha
& liirekekha 5 0: lirekekha, khurekekha, lirerekha, liilaekhasya, liilekhana

lirekekha
+ khurekekha

omwooni
& omwooni 2 0: omwooni, omwooni

alanyala
+ khunyala

baasomi
& baasomi 4 0: baasooni, basomya, basomela, basoma

khurekekha
& khurekekha 4 0: khurekekha, khurekekhelela, khurerekha, khurekekhelela

kamalesi
+

liyooka
& liyooka 2 0: liyoola, liyoota

liyoka
& liyoka 4 0: likoya, liloka, limoka, liroka

liiyoka
& liiyoka 1 0: Liiyoka

kasiima
+ khusiima
```

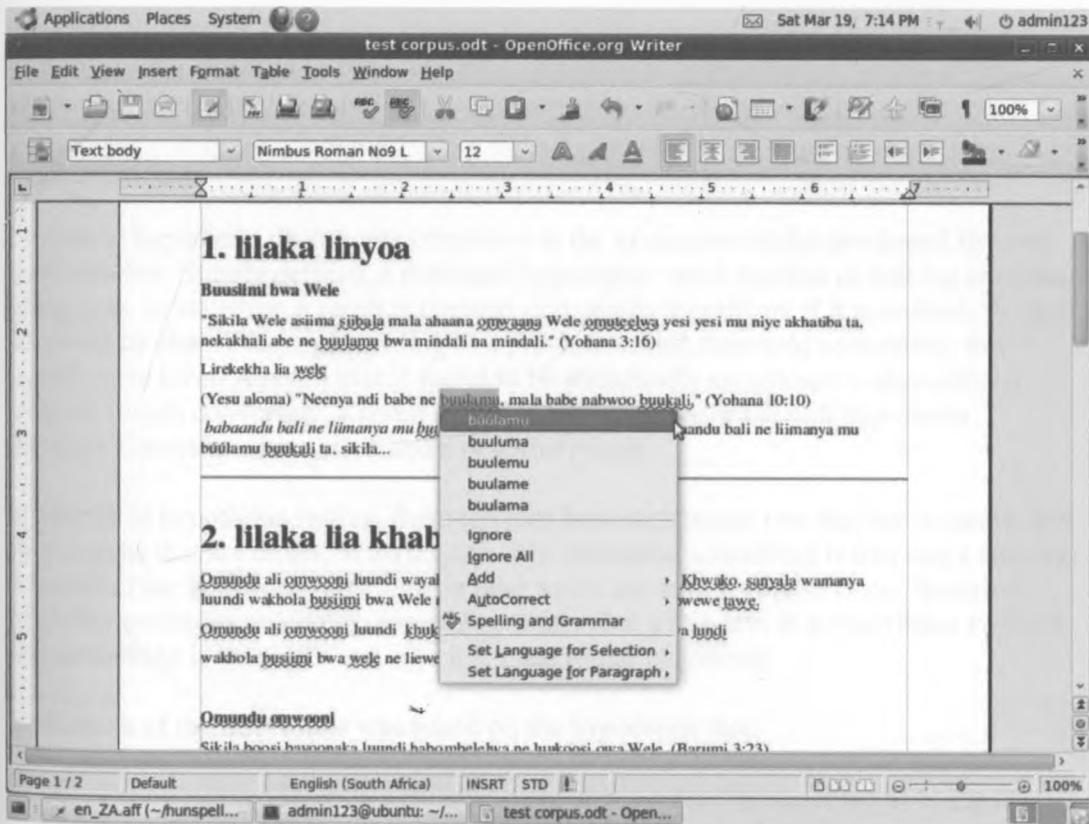
In the figure above, words marked with an “*” denote correctly spelled words that exist in the wordlist in the given form. Those marked with an “&” represent incorrectly spelled words. A suggestion list containing suitable replacements is subsequently generated and appended. Finally those words marked with “+” indicate correctly spelled words derived after employing affix rules to root words contained in the wordlist.

During the second phase of testing, the developed Bukusu spell checker was integrated into OpenOffice.org Writer and used to check blocks of text within the test corpus. The version of OpenOffice Writer that was used for testing was version 3.1. This enabled dictionaries to be added as extensions, unlike previous versions where the dictionary.lst file had to be edited. It is important to note however that, dictionaries can only be added as extensions to languages already contained in the OpenOffice.org approved language list. The major shortcoming of OpenOffice.org is that it does not allow for inclusion of a new language into its language list without formal vetting from its governing authorities.

To circumvent this challenge, the contents of the affix and dictionary files of an existing language, in this case English (SA), were replaced with those developed for the Bukusu language. This was a temporary measure, used mainly for testing and therefore the default settings were restored once testing was complete.

Below is a screenshot of the functional Bukusu Spell Checker deployed within the Open Office.org Writer Application.

Figure 4.2: Bukusu Spell Checker deployed in Open Office.org Writer



Initial testing in the second phase was carried out the developer, however further testing by an independent linguistic authority was later deemed necessary. This task was assigned to prominent Bukusu linguist Nasiombe Mutonyi, currently based in Virginia, U.S.A.

Below is an excerpt of the test corpus used by the developer in the illustration above:

1. Lilaka linyoa

Busime bwa Wele

"Sikila Wele asiima sibaia mala ahaana Omwana wewe omuteelwa, yeesi yeesi osuubila mu nnye akhatiba ta, nekakhali abe ne bulamu bwa mindali na mindali." (Yohana 3:16)

Lirekekha lia Wele

(Yesu alomaloma) "Necha ndi babe ne bulamu, mala babe nabwo bukali." (Yohana 10:10)

Sebali babandu basi bali ne limanya mu bulamu bukali ta, sikila...

2. Lilaka lia Khabili

Omundu ali omwoni Lundi wayahukhasibwa khukhwama khu Wele, khwako sanyala wmanya Lundi wakhola busiimi bwa wele ne lirekekha liewe khulwa bulamu bwewe tawe.

4.2 Evaluation

Statistical hypothesis testing was employed in the evaluation of the developed Bukusu spell checker. Simply defined, a statistical hypothesis test is method of making decisions using data. In statistics, a result is deemed statistically significant if it is unlikely to have occurred by chance alone, according to a pre-determined threshold probability, the significance level. A result that is found to be statistically significant is also called a *positive result*; conversely, a result that is not unlikely under the null hypothesis (Default/General position) is called a *negative result*.

In statistical hypothesis testing, there are four basic outcomes: two that are accurate, and two options that are errors. With the accurate outcomes, something is true and a test says it is true (True Positive); something is false and a test says it is false (True Negative). With the erroneous outcomes, something is false but a test says it is true (False Positive); and something is true but a test says it is false (False Negative).

Evaluation of the test results was based on the hypothesis that;

1. True Positive (TP) – Is a Bukusu word and is correctly classified as being one,
2. False Positive (FP) – Is not a Bukusu word and yet is classified as one,
3. True Negative (TN) – Is not a Bukusu word and is correctly classified as not being one,
4. False Negative (FN) – Is a Bukusu word yet is incorrectly classified as not being one.

Table 4.0: Evaluation results on developer's test set

Results	TP	FP	TN	FN	Total
No. of Instances	922	231	253	246	1652
Precision	Precision = $TP/(TP+FP) = 0.799$				
Recall	Recall = $TP/(TP+FN) = 0.789$				
Accuracy	Accuracy = $(TP+TN)/Total = 0.711$				

Table 4.1: Evaluation results on Mr. Mutonyi's test set

Results	TP	FP	TN	FN	Total
No. of Instances	254	69	55	84	462
Precision	Precision = $TP/(TP+FP) = 0.786$				
Recall	Recall = $TP/(TP+FN) = 0.751$				
Accuracy	Accuracy = $(TP+TN)/Total = 0.668$				

It was observed that, when spellchecking texts in which diacritics were omitted, many misspellings (True Negatives) are generated. When these misspellings involved one erroneous character, they were easily corrected using the suggestions generated. However, the suggestion component was unable to generate accurate suggestions, when the misspelling was a combination of a diacritic and two or more other characters.

The major contributor of unrecognized Bukusu words (False Negatives) are words that have not been included in the dictionary as well as proper nouns.

False positives occurred mainly due to incorrect placement of affix flags on dictionary words by the developer. This mainly resulted from the developer's limited familiarity with affixation rules for a segment of the vocabulary.

Previous spell checker developers [2] have cited over-generation of suggestions as a major challenge. This refers to the uncontrolled combination of prefixes leading to generation of numerous words that are not semantically correct in the given language's grammar. In this project over-generation was reduced through the use of separate definition of affix rules, as opposed to clustering several related rules together, as may have been the case in previous projects.

Chapter 5

5. Conclusion

5.1 Overview

The major focus of this project is, the development of a tool that can be used to spell check text written in the Bukusu dialect. The proposed system intended for deployment on a number of Open Source platforms such as word processors (OpenOffice.org Writer) and Web browsers (Mozilla & Google Chrome). The main objective of this tool is to provide a framework that will facilitate the documentation of more digital resources in the Bukusu language, hence enabling its preservation and future development.

5.2 Limitations

The limited online resources for the Bukusu language resulted in the restriction of the corpus sources, to printed texts obtained from the Bukusu dictionary, Bible and hymnbooks. The scarcity of online Bukusu resources rendered corpus collection tools such as, CorpusCatcher impractical. In order for the tool to perform optimally, it would require an input of Bukusu keywords. The tool would then use the input to query search engines for websites containing Bukusu text relating to the key words. The deficient online documentation on the Bukusu language meant that the tool was unable to yield any significant result from crawling the web.

As discussed in earlier chapters, Bukusu is a tonal language and it is therefore imperative to maintain its prescribed transcription structure. During the corpus collection process, the Optical Character Recognition software was unable to detect diacritic marks, thus resulting in erroneous output of several words. The situation was further compounded by the absence of a keyboard with diacritic inputs, when editing the output from OCR software. This technical limitation was the major contributor of True Negatives during the testing phase.

During the latter stages of the testing phase, the English-Bukusu dictionary was found to be deficient in Biblical vocabulary. The system subsequently generated numerous errors, because the bulk of the test corpus was derived from religious material.

The inability to add a new language to the OpenOffice.org language list also proved to be a major impediment. This effectively meant that dictionary extensions could not be created for the Bukusu language. The process undertaken to vet and incorporate a new language into the list is relatively slow, and thus would not have been a viable consideration given the time constraints of this project.

The limited access to linguistic expertise coupled with the developer's limited grasp of Bukusu orthography, meant that a portion of the documentation was subjective. The scarcity of published information relating to the language, led to high dependence on a single available source.

The lack of a baseline from related projects meant that, no data was available to compare and evaluate the performance of the developed system. The results obtained from this project will therefore form the baseline for related future research.

5.3 Recommendations

The methodology employed in this project, Rapid Application Development, used an iterative approach to extracting and incorporating user requirements within the system. Replicating the methodology to develop a spellchecker for other Luhya dialects is a viable option for future research. This will be facilitated by the availability of digital dictionaries for majority of the remaining Luhya dialects. The structure for Bukusu stemming rules is also applicable to some of the dialects defined as mutually intelligible in earlier chapters. As mentioned in the previous section, the results obtained from this project will serve as a baseline for related future projects. The result will be, a considerable reduction in development time.

The inclusion of the Bukusu language within the Open Office.org suite of software applications, will assist in increasing access to the developed system. Software developers will therefore be able to localize more systems to Bukusu with emphasis on the rural population, where the digital divide is perceived to be widest.

The development of a thesaurus and grammar-checking tool within for the Bukusu language will be a valuable extension to the newly developed spell checking system. As the uptake of the language increases, so will the demand for increased functionality by more experienced users e.g. linguists and authors

Bukusu professionals such as linguists, authors, musicians, copywriters and system developers should spearhead the formation of online repositories, archives, Bukusu blogs, chat-rooms and other social forums. This will lead to the creation, exchange and preservation of more Bukusu resources, thus leading to easier access to the information. The Open Source platform on which system is developed, will allow it to be deployed in a variety of online media, thereby enhancing the accuracy and integrity of the available resources.

The missing Biblical references should be considered for inclusion in the next edition of the Bukusu-English dictionary. This should also apply to the other Luhya-English dictionaries that may demonstrate similar deficiencies.

5.4 Conclusion

In this project, the developer reviewed the development of an open-source spellchecker for Bukusu language using Hunspell language tools. An average performance of the system was computed using the evaluation of both test results, from the developer and linguist. The developed system showed an acceptable performance of an accuracy of 68.9%, precision of 79.3% and recall of 77%, indicating that the system has practical use in spell checking documents.

As mentioned in earlier chapters, the functional Bukusu spellchecker will greatly assist in the localization of a variety of software applications. The perceived outcome is an increased uptake in the usage of the language across different media by members of the Bukusu community. The end result will be the accurate, unadulterated documentation of Bukusu culture and linguistic knowledge, subsequently ensuring its preservation for future generations.

References

1. Chege, K., 2007, 'Language independent Spellchecker: Gikuyu Word Processor', Unpublished 2nd year project, School of Computing & Informatics, University of Nairobi.
2. Chege, K., et al., 2009, 'Developing an Open Source Spellchecker for Gikuyu', Published School of Computing & Informatics, University of Nairobi.
3. Marlo, M., Sifuna, A., Wasike, A., 2008, *Bukusu-English Dictionary*, 4th edn National Science Foundation Graduate Research Fellowship.
4. Maurer, F. & Martel, S., 2002, *Extreme Programming: Rapid Development For Web-Based Applications*, IEEE Internet Computing, 6(1) pp 86-91.
5. Munyao, O., 2008, 'Akamba word processor with Spell checker', Published 4th Year Computer Science Project, School of Computing & Informatics, University of Nairobi.
6. Muriithi, B., 2007, 'Gikuyu word processor with a Spell checker', Published Master of Science Information Systems Project, School of Computing & Informatics, University of Nairobi.
7. Mutonyi, N., 2000, 'A Study of Bukusu phonology and Grammar', Published PhD dissertation in Linguistics, Ohio State University.
8. Otieno, J., 2009, 'Open-source Luo Spell checker using Hunspell tools', Published Master of Science Information Systems Project, School of Computing & Informatics, University of Nairobi.
9. The African Network For Localization, 2008, *Report on ANLoc Conference* [online] Available at:
<[http://www.africanlocalisation.net/sites/default/files/Report on ANLoc Conference.pdf](http://www.africanlocalisation.net/sites/default/files/Report%20on%20ANLoc%20Conference.pdf)> [Accessed 10 Oct 2010]
10. Wikipedia.org, 2006, *Linguistic Morphology* [online] Available at:
<[http://en.wikipedia.org/wiki/Morphology_\(linguistics\)](http://en.wikipedia.org/wiki/Morphology_(linguistics))> [Accessed 12 Dec 2010].