UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING & INFORMATICS

Open Source Spelling Checker for Kîmîîrû Language

BY

Anondo, Timothy Kimathi

P56/P/7851/2004

Supervisor

Mr. E. K. Miriti

2013

Submitted in partial fulfillment of the requirements of the Master of Science in Information Systems

DECLARATION

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

Signed:_____

Name:

Reg. No:

Date:_____

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

Signed:_____

Name:

Date:_____

ABSTRACT

Computational Linguistics has been of extensive research interest in Europe, America, South Africa, and other parts of the world. However, very few Human Language Technology Projects have existed in Kenya, particularly for Bantu Languages because Kiswahili and English are the dominant languages. Therefore there is a need to develop tools to support electronic document preparation in resource-poor languages.

This work describes the development of 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.

The focus of this project is the creation of two major Hunspell files namely; the affix file (.aff) and the dictionary file (.dic). The affix file enables the creation of all the rules involved in deriving the Kîmîîrû nouns and the verbs from the root words (stems). All the stems plus the appended rules are stored in the dictionary file.

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%. This Functional system is aimed at being adopted in major open-source products such as Open Office, Mozilla products such as ThunderBird and FireFox, Google Chrome.

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my Project Supervisor Mr Evans Miriti as well as members of the project panel who gave me the golden opportunity to do this wonderful system on the Kîmîîrû language. I am really thankful to them.

Secondly, I would also like to thank my parents, especially my dear Dad, Jeremiah Anondo who, not only provided the primary sources of my corpus but, also proof read the word list. I thank Nancy Gakii who manually annotated the word list and all other friends who helped me a lot in finishing this project within the limited time.

I developed this project, not only for the award of marks, but to also increase my knowledge.

THANKS AGAIN TO ALL WHO HELPED ME.

TABLE OF CONTENTS

DECLARATION ..... *ii*
ABSTRACT ..... *iii*
ACKNOWLEDGEMENT ..... *iv*
LIST OF FIGURES ..... *viii*
LIST OF TABLES ..... *ix*
CHAPTER 1 : INTRODUCTION ***1***
1.1 BACKGROUND *1*
1.2 STATEMENT OF THE PROBLEM *2*
1.3 PROJECT JUSTIFICATION *3*
1.4 PURPOSE OF THE PROJECT *4*
1.5 SPECIFIC OBJECTIVES *4*
1.6 SCOPE AND LIMITATION *4*
CHAPTER 2 : LITERATURE REVIEW ***5***
2.1 KÎMÎÎRÛ LANGUAGE *5*
2.1.1 KÎMÎÎRÛ ALPHABET 6
2.2 KÎMÎÎRÛ ORTHOGRAPHY *8*
2.2.1 KÎMÎÎRÛ TONES 9
2.2.2 KÎMÎÎRÛ VOWELS 10
2.2.3 DOUBLING OF KÎMÎÎRÛ VOWELS 11
2.2.4 KÎMÎÎRÛ DIPTHONGS 11
2.2.5 KÎMÎÎRÛ SEMIVOWELS 11
2.3 KÎMÎÎRÛ MORPHOLOGY *12*
2.3.1 NOUN MORPHOLOGY 12
2.3.2 KÎMÎÎRÛ PRONOUNS 14
2.3.3 KÎMÎÎRÛ VERBS 15
2.4 KÎMÎÎRÛ LANGUAGE TECHNOLOGY *16*
2.5 RELATED WORKS *16*
2.5.1 WORD PROCESSOR FOR GĨKŨYŨ LANGUAGE WITH SPELL CHECK *16*

2.5.2 OPEN SOURCE SPELL-CHECKER FOR GĨKŨYŨ USING THE HUNSPELL LANGUAGE TOOLS 21
2.5.3OPEN SOURCE SPELL CHECKER FOR DHOLUO USING THE HUNSPELL LANGUAGE TOOLS 26
2.5.4 OPEN SOURCE KIPSIGIS SPELL CHECKER AND LANGUAGE TOOL 26
2.5.5 OPEN SOURCE SPELLCHECKER FOR LUHYA-LULOGOLI 27
2.5.6 SPELL CHECKERS 28
2.5.7 HUNSPELL LANGUAGE TOOLS 34
CHAPTER 3 : METHODOLOGY ***36***
3.1 ANALYSIS *36*
3.1.1 SCOPE ANALYSIS 36
3.1.2 PROBLEM ANALYSIS 37
3.1.3 DECISION ANALYSIS 37
3.2 SYSTEM DESIGN *37*
3.3 MAJOR COMPONENTS OF THE DEVELOPED SPELL CHECKER SYSTEM *38*
3.3.1 GRAPHICAL USER INTERFACE (GUI) 38
3.3.2 SPELL CHECKER 39
3.3.3 WORDLIST 39
3.3.4 PERSONAL DICTIONARY 40
3.4 CORPUS COLLECTION *41*
3.5 IMPLEMENTATION OF THE KÎMÎÎRÛ SPELLCHECKER *41*
3.6 HUNSPELL LANGUAGE SPECIFIC SETUP FOR KÎMÎÎRÛ *42*
3.6.1 SUGGESTIONS COMPONENT 43
3.6.2 AFFIXATION COMPONENT 45
3.6.3 NOUN COMPONENT 46
3.6.4 VERB COMPONENT 48
CHAPTER 4 : TESTING AND RESULTS ***52***
4.1 TESTING *52*
4.2 EVALUATION *57*

4.2.1 MAJOR CHALLENGES.....59
CHAPTER 5 : DISCUSSION.....*60*
5.1 OVERVIEW.....*60*
5.2 ACHIEVEMENTS.....*60*
5.3 LIMITATIONS.....*61*
5.4 RECOMMENDATIONS.....*61*
5.5 CONCLUSION.....*62*

APPENDICES.....*63*
APPENDIX 1.....*63*
REFERENCES.....*63*

LIST OF FIGURES

Figure 1 : Spell checking operation in Gĩkũyũ Language cited in Muriithi, 2008. 18
Figure 2 : Spell checking operation in Kîmîîrû Language.....40
Figure 3 : Word found in the Main and/or Personal Dictionary. 52
Figure 4 : Word not found in Main and/or Personal Dictionary.....53
Figure 5 : Word found through Affix removal 54
Figure 6 : Word not found in Main and/or Personal Dictionary using test corpus.....55
Figure 7 : Word not found in Main and/or Personal Dictionary using test corpus.....56
Figure 8 : Kîmîîrû Spell Checker deployed in Open Office.org Writer 57

LIST OF TABLES

Table 1 : Mîîru clan (dialects) and their geographical regions cited in Ataya, 2012. 5

Table 2 : Kîmîîrû alphabet and pronunciation cited in Ataya, 2012. 6

Table 3 : Instances of old and new Kîmîîrû vowel graphs cited in Ataya, 2012 8

Table 4 : Old Kimeru problem of future negative commands and future affirmative statements cited in Ataya, 2012. 9

Table 5 : New Kîmîîrû with a high tone mark to indicate future negative commands and no mark to indicate future affirmative statements cited in Ataya, 2012. 10

Table 6 : Tone mark for 'if' or conditional clause cited in Ataya, 2012. 10

Table 7 : Kîmîîrû noun classes cited in Ataya, 2012 12

Table 8 : Kîmîîrû orthography set character count 44

Table 9 : Evaluation of test results based on four outcomes.....58

CHAPTER 1 : INTRODUCTION

1.1 BACKGROUND

Computational Linguistics has been of extensive research interest in Europe, America, South Africa, and other parts of the world. However, very few Human Language Technology Projects have existed in Kenya, particularly for Bantu Languages because Kiswahili and English are the dominant languages (Wagacha 2007; DePauw 2007). This means that most of all textual communication is in the two languages. Therefore there is a need to develop tools to support electronic document preparation in other resource-poor languages.

According to Wikipedia, a spell checker is a computer application that identifies possible misspellings in a text by referring to the accepted spellings in a database. Most spellcheckers function as part of a larger program, such as a word processor or search engine. Having a spell checker is not quite a replacement for a real human editor, but it can help users catch basic spelling mistakes so that their communications are more presentable. The earliest spell check programs simply scanned documents for words they didn't recognize, and alerted users to the fact that something was spelled incorrectly, without providing any suggestions for alternate spellings. These systems were eventually replaced by spell checkers which generated a list of possible words for the user to select from, to replace the misspelled word. Recently, research has focused on developing algorithms which are capable of recognizing a misspelled word, even if the word itself is in the vocabulary, based on the context of the surrounding words. This mitigates the detrimental effect of enlarging dictionaries, allowing more words to be recognized.

Kîmîîrû, as a Kenyan Bantu language, can be classified as a resource scarce language (RSL) with respect to language technology resources, tools and applications (Wagacha 2010; De Pauw, 2010). Again, Kîmîîrû orthography contains two diacritically marked characters (î and û) that require extra keystrokes to generate, a situation which often makes users opt for the diacritically unmarked equivalents, resulting in non-standard

Kîmîîrû texts. These extra characters also pose a challenge for automated corpus collection methods, such as those using optical character recognition (OCR).

There is an urgent need for language technology support for the Kîmîîrû language due to its current commercialization as evidenced in the emergence of Broadcast and Print Media. Currently, the language boasts of six FM stations and, the Kîmîîrû Bible (2010) which utilizes the two diacritically marked characters (**î** and **û**), both spoken and written. This will encourage the growth and usage of the language. This developed system will make a positive contribution to this need, because it will enhance creation of correctly spelled Kîmîîrû texts.

1.2 STATEMENT OF THE PROBLEM

In the modern world, there is need for tools that cater for various language speakers due to the current climate of multilingual localisation, internationalism of software, the spread of information through the World Wide Web and other resources, and the general recognition that speakers of other languages other than English have an equal right to access this information explosion. Indeed, Dhonnchadha et al., (2003) asserts that languages must endeavour to keep up with and avail of language technology advances if they are to prosper in the modern world.

So-called 'resource poor' languages are largely ignored in the development of Information and Communication Technologies. This may be mainly due to lack of comprehensive documentation, lack of interest from developers, lack of political will and lack of funds to finance such an enterprise (Wagacha 2010; De Pauw 2010). The majority of these languages are from the least developed countries as presented by Berment (2004).

Kîmîîrû language is one such example of a resource poor language as some of the readily available documentation is the Holy Bible (1964, 2010), hymnals (2000, 2007) Kimeru proverbs (1995), and children story books. Interestingly, a book about Meru foods

entitled “Mantu ja Biakuria” was written by Agnes R. Fraser, a member of the United Missions, in 1963.

As stated earlier, Kîmîîrû orthography contains two diacritically marked characters (î and û) that require extra keystrokes to generate, a situation which often makes users opt for the diacritically unmarked equivalents, resulting in non-standard Kîmîîrû texts. Thus, A Kîmîîrû spell checking system will enhance creation of correctly spelled Kîmîîrû texts.

1.3 PROJECT JUSTIFICATION

- The development of the system was one way of formalizing linguistic knowledge, and thus can be considered as a form of documentation for poorly investigated languages.

- Academically, the spelling checker may be of use to Kîmîîrû and related language authors, editors and publishers when producing books and journals in the language.

- Prevention of Culture Breakdown because languages represent the culture and diversity of different people around the world. Unavailability of language resources eventually leads to extinction. Failing to salvage the language will lead to extinction of the Kîmîîrû culture and consequently the people.

- Enhanced Testing: Due to the morphological complexity of under-resourced languages, the Kîmîîrû language provided a good ground for testing of the Hunspell Language Toolkit.

- Spur economical development: The Government of Kenya through the pillars of Vision 2030 has identified computer technologies as the main engine for development. Thus, there was an urgent need to develop software in an indigenous language like Kîmîîrû.

- Reduction of the language technology divide created between the languages of the developed nations and those of the less developed. According to Dhonnchadha et al., (2003), “languages must endeavour to keep up with and avail of language technology advances if they are to prosper in the modern world”

1.4 PURPOSE OF THE PROJECT

The overall aim of this project was to develop an open source spelling checker for Kîmîîrû language using the Hunspell Language Tools.

1.5 SPECIFIC OBJECTIVES

- Understand the fundamental principles and morphological composition of Kîmîîrû nouns and verbs.
- Analyze how verb and noun prefixes and suffixes can be appended to stems in order to produce all the possible Kîmîîrû words.
- Design algorithms on how prefixes and suffixes can be appended to stems.
- Implement the rules in the affix file.
- Generate verbs and nouns in the dictionary file.

1.6 SCOPE AND LIMITATION

The system should be able to perform a morphological analysis of Kîmîîrû language highlighting nouns and verbs derivation and also provide a suggestion component used to generate probable suggestions for a misspelled word. The system will neither include a grammar checking facility nor a thesaurus. The Spellchecker will also be limited in scope to focus on the Kiimenti dialect and hence will not cover any of the eight (8) other sub-groups in the larger Mîîrû community. This is because Kiimenti is the dominant dialect with over 500,000 speakers. Again, it is the same dialect which has attained standardization through the development of literature for teaching, and learning the vernacular of the same and thus recognized as an official dialect of the Mîîru people.

CHAPTER 2 : LITERATURE REVIEW

2.1 KÎMÎÎRÛ LANGUAGE

According to Ethnologue, the Kîmîîrû language is spoken by roughly 1.3 million people in the Meru district around Mt. Kenya, in the central Kenya highlands. Linguistically, it is a Bantu language of the Niger-Congo family of languages. The language has nine main dialects that are located within a geographical continuum as depicted in Table 1. And, by comparison, Kîmîîrû has overall roughly 60% similarity to other important languages in the region which are Kikuyu, Kiembu, and Kikamba. They share a certain amount of common vocabulary across the especially in basic items like, “cook”, “farm”, “see”, “tree”, “class of domestic animals” among many others.

The following table depicts the Mîîru dialects and the geographical distribution on the nine (9) clans.

Table 1 : Mîîru clan (dialects) and their geographical regions cited in Ataya, 2012.

	Mîîru clan (dialects)	**Geographical Region**
1	Kîîgembe	Meru North
2	Gîtigania (Gîtiania)	Meru North
3	Kiimenti	Meru Central
4	Kîîgoji	Meru Central
5	Kîmîîtîîne	Meru South
6	Kîmwîmbî	Meru South
7	Kîmûthambi	Meru South
8	Gîchuka	Meru South
9	Kîtharaka	Meru South

2.1.1 KÎMÎÎRÛ ALPHABET

The following table depicts the full Kîmîîrû Alphabet and respective pronunciation.

Table 2 : Kîmîîrû alphabet and pronunciation cited in Ataya, 2012.

Alphabetical Letter	Pronunciation
A	aa
B	mbii
C	cii
E	ee
G	njii
Î	îî
I	ii
J	njei
K	kei
L	eelo
M	eemu
MP	eemu na pii
N	eeni
ND	eeni na ndii
O	oo
R	aara
S	eesi
T	tii
U	uu
Û	ûû
W	ndabiriû
Y	waî

The English consonants **f, q, s, v, x** and **z** are not found in the Kîmîîrû language. Moreover if a word is written in any of these letters, then the word is borrowed. Examples are:

a) Sabatû >> Sabbath
b) Zakaria >> Zakariah

The words start with the letters not found in the Kîmîîrû alphabet. The final syllable will always end with a vowel.

Like in other Bantu languages some consonants are paired because they cannot stand alone. For example the consonants **'h'** and **'d'** do not normally appear alone except in borrowed proper names but are preceded by other consonants in the initial position. Examples are:

a) ndawa >> medicine
b) thaani >>plate
c) chai >> tea

In the above example **'d'** is preceded by **'n'** while **'h'** is preceded by **'t'** and **'c'**.

Kîmîîrû words can also undergo full or partial reduplication, depending on the number of syllables in a word. Words with two syllables or less undergo full reduplication, while words with more than two syllables undergo partial reduplication, with only the first two syllables being reduplicated. Examples are:

a) mmaamia >> make me sleep.
b) nnaania >>make me feel cherished.
c) baiîribaîiri >>two by two (people).
d) biîrîbiîrî >>two by two (things).
e) ûmûnene >> big(person).

Kîmîîrû is highly inflectional, with a phonological structure based on c-v (consonant followed by verb) system. It is tonal because it is more spoken than written hence introducing ambiguity. It is also agglutinative because words are formed from a battery of affixes.

2.2 KÎMÎÎRÛ ORTHOGRAPHY

Ataya (2012) contended that orthography is the writing and spelling system of a language. Reading and spelling can be hampered by or made easier by the orthography chosen to write documents in that language. He further argued that languages differ in their sound and spelling systems thus creating the important realization that it is not often the case that a writing system reflects exactly the spoken message or that all the sounds in a language have a graph that represents them.

Previously only five vowels were used in the Kîmîîrû (Latin-based) orthography; **a, e, i, o, u**. Because some spoken Kîmîîrû language sounds were not represented by those vowels, reading translated text was difficult. Target users (Mîîru) have to read what they know, not what they see written. Solution to the inadequate vowel sounds was solved by the addition of two vowels; **î, û**. The use of diacritic marks to provide differences of vowel graphs that represent more than one sound is a common practice in creating new orthographies (Ashby 2005; Maidment 2005). Thus the new Kîmîîrû Orthography contains seven vowel graphs, namely: **a, e, i, o, u, î, û**. Below is an illustration of the different sounds by the Old and New Kîmîîrû vowel graphs.

Table 3 : Instances of old and new Kîmîîrû vowel graphs cited in Ataya, 2012

	Old Orthography	**New Orthography**
1	Kuura (to rain)	Kûûra (to get lost)
2	Mukuru (blunt machete)	Mûkûrû (old man)
3	Matu (clouds)	Matû (ears)
4	Kiuru (nest)	Kîûrû (temple)
5	Kuuma (come out of)	Kûûma (to dry)

When words are marked clearly with diacritic marks, for instance, in the word kuura, ambiguity is removed. *Kuura* here would mean *to rain* or to *get lost* depending on context or the first meaning coming into the user's mind. After providing a diacritic mark on the vowel **'u'** as in the word *kûûra* (to get lost), the word is restricted to one meaning only.
Like all other Bantu languages Kîmîîrû is highly tonal with every syllable having its own tone which should be written or somehow represented in written form.

2.2.1 KÎMÎÎRÛ TONES

Ataya (2012) asserted that assigning diacritic marks to existing vowel graphs to produce new semantic value alone is not enough to deal with the Kîmîîrû language orthographical problems. This is occasioned by the theoretical advancement that every syllable in the Kîmîîrû language should have a tonal mark. Achieving such a literal feat would alter the look of the text making it unappealing to the user's eye. However lack of tonal marks creates confusion as illustrated in the following table.

Table 4 : Old Kimeru problem of future negative commands and future affirmative statements cited in Ataya, 2012.

Old Kimeru	Future Negative Command	Future Affirmative
	1st meaning	2nd Meaning
Ukoragana	You shall not kill	You shall kill
Ukathungira	You shall not commit adultery	You shall commit adultery
Ukaiya	You shall not steal	You shall steal

The problem of different semantic values can be resolved by using a tone mark over the future negative commands while the future affirmative remains unmarked.

Table 5 : New Kîmîîrû with a high tone mark to indicate future negative commands and no mark to indicate future affirmative statements cited in Ataya, 2012.

New Kîmîîrû	1st Meaning	New Kîmîîrû	2nd Meaning
Ûk**ó**oragana	You shall not kill	Ûkooragana	You shall kill
Ûk**à**thûûngira	You shall not commit adultery	Ûkathûûngira	You shall commit adultery
Ûk**à**iya	You shall not steal	Ûkaiya	You shall steal

The use of a dieresis mark (¨) is also utilized in the new Kîmîîrû to represent the 'if' or conditional clause tone mark.

Table 6 : Tone mark for 'if' or conditional clause cited in Ataya, 2012.

New Kîmîîrû 'if' clause mark	Meaning
Ûköoragana	If you kill
Ûkäthûûngira	If you commit adultery
Ûkäiya	If you steal

Like all other Bantu languages Kîmîîrû is a tonal language and ideally, every syllable has its own tone which should be written or somehow represented in written form.

2.2.2 KÎMÎÎRÛ VOWELS

a	antû >> people
e	eekûrû >> women
i	iraatû >> shoes
o	nyomba >> house
u	yuku >> book
î	îtu >> cloud
û	ûta >> bow

2.2.3 DOUBLING OF KÎMÎÎRÛ VOWELS

In Kîmîîrû language some vowels automatically double. Examples are:

a) aakûire (the first ‘a’ refers to the third person singular pronoun he/she or him/her while the second ‘a’ indicates past tense. The inclusion of the verb root ‘kuire’ is translated as he/she died).

b) ageeta (the initial ‘a’ is for the personal pronominal form, the ‘ga’ indicates the future tense. The verb root is ‘îta’ whose inclusion translates to he/she will go).

c) biaawe (is a possessive word meaning belonging to him/her)

2.2.4 KÎMÎÎRÛ DIPTHONGS

Dipthongs are a new combination of vowel sounds created when vowels are sounded together. Examples are:

a) a + a = aa (ara+anda = araanda >> he/she planted)
b) a + e = ae or ee (ara + ena = araena or arena >>he/she breathed)
c) a + o = ao or oo (ara + ona = araona or aroona >>he/she saw)

2.2.5 KÎMÎÎRÛ SEMIVOWELS

The vowels ‘î’ and ‘û’ tend to become semi – vowels when preceding others and change when followed by another vowel. ‘w’ and ‘y’ form vowel sounds especially when certain vowel sounds coalesce. Examples are:

a) w – û + aawe becomes waawe >> his/hers(person)
b) y – î + uku becomes yuku >> book

2.3 KÎMÎÎRÛ MORPHOLOGY

2.3.1 NOUN MORPHOLOGY

Morphology is concerned with the internal structure of words. Inflectional morphology of bantu languages is encoded in nouns and verbs (Mutonyi, 2000).

Kîmîîrû noun consists of a stem and a prefix whereby the stem may have two syllables. Ataya (2012), identifies nine classes of Kîmîîrû nouns. These classes are distinguished by the prefixes affixed to a noun stem. The stem may have or may not have their origin in a verb root. He further points out that there are inflections inherent in the noun, verb, adverb and adjective which have no genders or cases but possess number distinctions.

Table 7 : Kîmîîrû noun classes cited in Ataya, 2012

1. 'MU'-'A' CLASS': Class of the human beings	
SINGULAR	**PLURAL**
muntû – person	antû - persons
mûkûrû – man	akûrû - men
mûka – woman	aka – women
2. 'MU'-'MI' CLASS: Class of inanimate things	
mûti – tree	mîtî – trees
mûtaratare - strawberry	mîtaratare – strawberries
mwari – door	mîari – doors
3. 'KI', 'GÎ'-'I', 'BI' CLASS: Class of non personal things which can be seen or handled	
kîara – finger	biara - fingers
kîeni - field	bieni – fields
gîkwa – yam	ikwa – yams
4. 'RÛ'-'N', 'M' CLASS: Class of things, inanimate things, abstract things and concrete things	
rûgoji – horn	ngoji – horns

rûguma – wound	ngûma – wounds
rûbungûro – key	mbungûro – keys
5. 'I', 'RI'-'MA' CLASS: Singular in this class start with î or r and plural start with m	
îgita – period	magiita - periods
riitho - eye	meetho – eyes
rîîtwa - name	marîîtwa – names
6. 'N' CLASS: Class where the word in singular is same as in plural	
nyomba – house	nyomba – houses
nyongû - pot	nyongû - pots
nkoro - heart	nkoro – hearts
7. 'KA', 'GA'-'TÛ', 'TWA' CLASS: any noun from any class can be inflected to fit into this class. It compares the size of the object.	
kanyua - mouth	tûnyua - mouths
kaana – child	twana – children
kagitûjû – rabbit	tûgitûjû – rabbits
8. 'U', 'W' – 'MO', 'MA' CLASS: All abstract and non-abstract nouns are grouped. Plurals are formed by prefixing into the singular noun root.	
ûtombo - brain	mootombo – brains
ûûme - wisdom	maûûme – wisdom
ûntû - thing	mantû – things
9. 'KU', 'GÛ', -'MA', 'MO' CLASS: a collection of concrete nouns	
kûgûrû - leg	magûrû – legs
guoko - hand	mooko - hands
gûtû – ear	matû - ears

2.3.2 KÎMÎÎRÛ PRONOUNS

The Kîmîîrû pronoun has number and person where number refers to either singular or plural, while person refers to the first, second or third person. The exception of this rule is the demonstrative and relative pronouns which have a number and follow the noun classes.

Examples are:

2.3.2.1 PERSONAL PRONOUNS

a) 1st Person – Ni >>I.
b) 2nd Person – Gwe >>You.
c) 3rd Person – We >>He/She.

2.3.2.2 EMPHATIC PRONOUNS

a) 1st Person – Ûûni >>I, Myself.
b) 2nd Person – Ûûgwe >>You, yourself.
c) 3rd Person – We, wengwa >>He, himself, her, herself.

2.3.2.3 POSSESSIVE PRONOUNS

a) 1st Person – Gîaakwa >>Mine.
b) 2nd Person – Gîaaku >>Yours.
c) 3rd Person – Yaakîo >>Its.

2.3.2.4 REFLEXIVE PRONOUNS

a) 1st Person – ningwa >>myself.
b) 2nd Person – gwengwa >>yourself.
c) 3rd Person – wengwa >>himself/herself.

2.3.2.5 DEMONSTRATIVE PRONOUNS

a) Ûûjû/ûû >>This (person).
b) Jûjû/jûû >> This(thing).
c) Gîkî/kîî >>This(big).

2.3.2.6 RELATIVE PRONOUNS

a) Ûrîa >> who.
b) Îria >> which, that.
c) Rîrîa >> which, that.

2.3.3 KÎMÎÎRÛ VERBS

The Kîmîîrû verb is mostly expressed by the imperative and the infinitive and is marked by the particle **'kû'** or '**gwa**' or '**gwî**'. The particles can be described as the equivalent of the English infinitive 'to'. Examples are:

a) gwata >> catch, hold.
b) îta >>go.
c) kûrîma >>to cultivate/weed.

2.3.3.1 TRANSITIVE AND INTRANSITIVE VERBS

Transitive verbs are distinguished from intransitive verbs because the former has a direct object while the latter does not possess the same. Examples are:

2.3.3.1.1 TRANSITIVITY

a) Ga nkûgera >>I throw/am throwing.
b) Nkûjûkia >>I have taken.

2.3.3.1.2 INTRANSITIVITY

a) Ni nkûthûgaania >>I am thinking.
b) Riûa nî rîkwara >>The sun is shining.

2.3.3.2 CAUSATION

Infinitive Verbs in Kîmîîrû often end in **'-a-**'. Addition of the affix 'i' or 'ithi' before the final 'a' brings out the idea of causation. Examples are:

a) -îta >>go -ît**ithia** >>make to go.
b) -menya >>know -meny**ithia** >>make to know.

2.3.3.3 APPLICATION

The suffix ‘îra’ or ‘era’ added to the verb root gives the idea of an application. Examples are:

a) -îta >>go. -îtîra >> go for.
b) -aka >> build. -akîra >> build for.

2.3.3.4 RECIPROCATION

The suffix ‘eeni’ or ‘ieni’ is added to a root verb. Examples are:

a) -ona >>see . -onaneeni >>see each other.
b) -teethia >>help. -teethanieni >>help each other.

2.3.3.5 PASSIVE FORM

The consonant ‘w’ or ‘u’ is inserted between the stem and the final ‘a’. Examples are:

a) -ona >>see -onwa >>be seen
b) -onia >>show -onua >>be shown

2.4 KÎMÎÎRÛ LANGUAGE TECHNOLOGY

At the time of developing this project there did not exist any language technology research efforts on Kîmîîrû. This is the first effort towards development of a spellchecker in Kîmîîrû language.

2.5 RELATED WORKS

2.5.1 WORD PROCESSOR FOR GĨKŨYŨ LANGUAGE WITH SPELL CHECK.

This prototype system was developed by Brian Kingori Muriithi (2008), a former MSc. Information Systems student of the University of Nairobi.

The main aim of this project was to develop a tool that could be used to spell check text written in the Gĩkũyũ language. Due to the unique orthography of the language, this tool was implemented as a feature of a basic word processor. As an extension to the project, the interface of the word processor was also in Gĩkũyũ. A secondary aim was to develop the application in such a way that it was platform independent and could be used on any system that supported the Unicode character set.

Each word in a given text was searched for in a dictionary created from a compiled list of valid words. Words that were not found were highlighted and if considered to be incorrectly spelled, sensible alternatives were offered, from which the user chose a replacement. On selection of a replacement, the system automatically altered the text. If no suggestions were available, the user of the system had the option of either permanently adding the word to the dictionary or ignoring it.

The system utilized the Rapid Application Development technique and the spell checking component operated at the user's request, checking an entire document at once, notifying the user when an error was encountered. The developer selected Rapid Application Development (RAD) techniques because it emphasized extensive user involvement in the rapid and evolutionary construction of working prototypes of a system to accelerate the system development process. RAD is also called a spiral approach because one repeatedly spirals through the phases to construct a system in various degrees of completeness and complexity.

Due to the unique orthography of the language, this tool was implemented as a feature of a basis word processor. As an extension to the project, the interface of the word processor was also in Gĩkũyũ. The application was developed in such a way that it was platform independent and could be used on any system that supported the Unicode character set. The dependent spell checking method using a dictionary-look-up program used the Levenshtein distance algorithm to generate suggestions for replacement.

The inclusion of shortcut keys to enable the typing of characters, used in the language but not found on the standard keyboard, reduced the number of spelling errors committed by users typing in Gĩkũyũ, making text easier to read and understand. With respect to ease of use, it was found that the standard keyboard does not cater for the additional vowels, 'ũ' and 'ĩ' used in Gĩkũyũ. This led the developer to add shortcut keys to enable easy access to these vowels to the user. These keys were as follows:

- Ctrl-Q = ĩ ;
- Alt-Q = Ĩ ;
- Ctrl-Z = ũ ;
- Alt-Z = Ũ;

These key combinations were selected for the following reasons:

- The letters ‘Q’ and ‘Z’ do not exist in the Gĩkũyũ language, therefore, these assignments, other than others, may serve to mitigate confusion.

- They were easy to remember and additionally they provided viable suggestions for phrases and terms that may be used in localizing software to this language.

2.5.1.2 OPERATION OF THE PROTOTYPE SYSTEM

This particular spell checking system operated at the user's request, checking an entire document at once, notifying the user when an error was encountered. The figure below illustrated its functions.

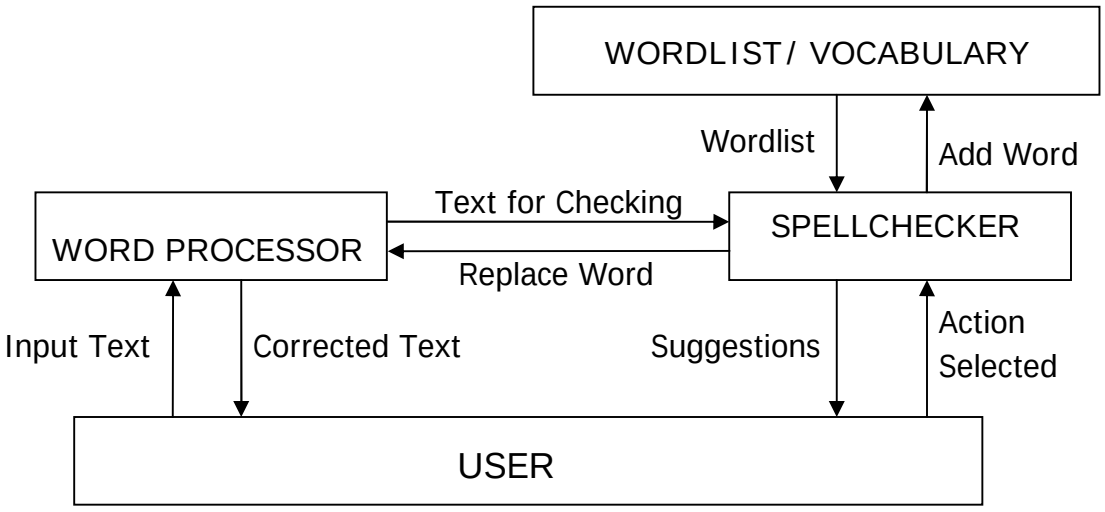

Figure 1 : Spell checking operation in Gĩkũyũ Language cited in Muriithi, 2008.

2.5.1.3 DESIGN OF THE DICTIONARY/ VOCABULARY

The dependent spell checking method used a dictionary-look-up program which at that time was the most accurate approach. This involved the maintenance of a word list of correctly spelled words. The spell checker simply checked to see if the words in the text

file being checked appeared in this word list. Any word it failed to find was then tagged and sensible alternatives offered. This approach was selected for the following two main reasons:

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

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

2.5.1.4 WORD LIST

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. This wordlist ideally provided as accurate a representation of the language as possible.

The size of the wordlist was very important; a word list which is too small would lack vital words that are in everyday use in the language. In practice, however, an optimal size has to be a limited one. A larger number than the optimum would lead to misspellings being skipped as they would be mistaken for other words.

Sources for the words used to populate the wordlist were:

- Gĩkũyũ – English Dictionary, edited by T.G. Benson: The developer had to manually input words from this text.

- Maitũ nĩ ma Itũ (Our Mother is our Truth): A website dedicated to the development of the Gĩkũyũ language by Gatua wa Mbũgwa.

- Gĩkũyũ kĩa Mũgĩkũyũ: A website dedicated to the development of the Gĩkũyũ language through stories, poems and song by Charuthi Ng'ang'a Wairia.

- Kĩrĩra Kĩa Ũgikũyũ: A book by Mathew Njoroge Kabetũ.

- Kaguraru na Waithĩra: A book by Mathew Njoroge Kabetũ.

- A text file provided by the developer's supervisor, Prof. P.W. Waiganjo. This required cleaning of the data in order to extract valid Gĩkũyũ words.

Due to the format used (UTF-8), in order to read in the wordlist for the creation of the dictionary, the developer had to use an InputStreamReader enclosed within a BufferedReader, thus allowing for its specification. Likewise, in order to add a word to the dictionary, he had to use an OutputStreamWriter within a Buffered writer.

2.5.1.5 COMPILATION OF THE WORDLIST

Some of the texts used were downloaded from the internet and stored as text files (.txt extension). All the work was then done in notepad. The texts were read through one by one and numbers, English words or anything else that was judged inappropriate for the wordlist was deleted. The words were then arranged in a list and sorted alphabetically. Duplicate words were removed and all remaining words were combined to form a single wordlist.

The books were first scanned and Optical Character Recognition (OCR) software was used to make the text available for editing.

The outcome was a single wordlist containing over 13,000 unique words, stored in alphabetical order, on separate lines, separated only by the carriage return character.

2.5.1.6 DICTIONARY STRUCTURE

Hash Tables were chosen by the developer because not only were they fast, efficient and accurate, the structure suited the suggestion mechanism (Levenshtein Distance) used. They were also efficient in storage space making it ideal for use in situations where memory was limited.

The developed system was a fairly robust basic work processor, providing most of the functionality found in other word processors commonly in use. Based on tests carried out, the accuracy of the spellchecker was calculated to be 67.4%.

2.5.2 OPEN SOURCE SPELL-CHECKER FOR GĨKŨYŨ USING THE HUNSPELL LANGUAGE TOOLS

The development of this spellchecker was based on previous works on a Gĩkũyũ dictionary-based system, incorporated in a text editor (Chege, 2007).

Gĩkũyũ can be classified as a resource scarce language with respect to language technology resources, tools and applications. This situation can be attributed to different factors:

First, Kiswahili and English are the dominant languages in Kenya, meaning that most of all textual communication is in these languages. Second, Gĩkũyũ orthography contains two diacritically marked characters (ĩ and ũ) that require extra keystrokes to generate, a situation which often makes users opt for the diacritically unmarked equivalents, resulting in non-standard Gĩkũyũ texts. These extra characters also pose a challenge for automated corpus collection methods, such as those using optical character recognition (OCR).

However, despite such an unfavorable backdrop, Gĩkũyũ, together with a few other Kenyan languages, are steadily becoming commercial languages as evidenced by their increased

use in broadcast media, publishing and advertising. These developments pointed to a need for the development of an open source spell-checker for Gĩkũyũ using the Hunspell Toolkit. The closest effort towards Gĩkũyũ spell checking is a dictionary-based system, incorporated in a Gĩkũyũ text editor (Chege, 2007). This system worked well, but for only a limited number of words, as contained in the dictionary. This spellchecker overcame this limitation by using a rule-based approach for determining the correct spelling of any Gĩkũyũ word.

2.5.2.1 NOUN MORPHOLOGY

Gĩkũyũ nouns can be grouped into two categories, namely, derived and underived nouns. Underived nouns consist of named entities, while derived nouns can be formed in one of

two ways: by affixing diminutive, augmentative or collective prefixes to an underived noun, or through verbal nominalization.

The following examples illustrate these processes:

- nyũmba → kĩ-nyũmba (a big house)
- imondo → tũ-mondo (many nice handbags)
- thaaka → mũ-thaak-i (player)
- hooya → i-ho-ero (Place of prayer)
- getha → i-geth-a (harvesting occasion)
- thooma → ga-thom-i (the small one who reads)

Membership to a noun class is determined by a concord system with agreement enforced on other sentence components, such as adjectives and verbs. All Gĩkũyũ nouns, derived or underived, can also be optionally affixed with the locative suffix -inĩ, which changes the meaning from a referential entity to a location, as shown in the following examples:

- metha-inĩ (on the table)
- mũtĩ- inĩ (on the tree)

2.5.2.2 VERB MORPHOLOGY

A typical Gĩkũyũ verb consists of a combination of zero or more dependent morphemes, a mandatory dependent morpheme and a mandatory final vowel. The simplest verb consists of a verb root and a final vowel. These are usually commands or directives. Subjunctive verb formations, i.e. commands, can optionally take a plural marker -i or –ni. Examples include:

- ma-thaak-e (so that they play)
- in-a-i (sing)
- nĩ-ci-mũ-hat-agĩr-a (they usually sweep for him/her)
- reh-e-ni (bring)

Gĩkũyũ verbs can also undergo full or partial reduplication, depending on the number of syllables in a word. Words with two syllables or less undergo full reduplication, while words with more than two syllables undergo partial reduplication, with only the first two syllables being reduplicated.

Examples are:

- negena → nega-negena (make noise a little more)
- tiga → tiga-tiga (leave a little more)

Gĩkũyũ verbs are also affected by consonantal and vowel phonemics. Meinhof's Law involves consonants b, c, r, t,g, k being replaced with NC composites in verbs obeying first person singular, noun classes 1, 8 and 9 concord systems. Dahl's Law is a consonantal mutation that involves the cause sound k appearing before trigger voiceless sounds c,k,t, being replaced with its equivalent voiced sound g. Examples include:

- rathima → ndathima
- uma →nyumia
- kũ-theka →gũtheka
- kĩ-ka-thira → gĩgathira

Vowel mutation includes vowel lengthening before prenasalized stops and vowel assimilation when some vowel combinations appear in the neighborhood of each other (Mugane, 1997).

2.5.2.3 CORPUS COLLECTION

The primary development corpus was from a collection of a set of 19,000 words from previous works on Gĩkũyũ at the School of Computing and Informatics, University of Nairobi (Wagacha et al., 2006a; Wagacha et al., 2006b; De Pauw et al., 2007; De Pauw and Wagacha, 2007).

Though the corpus had a bias on religious material it also included poems, short stories, novels and Internet sources. The corpus was pre-processed manually to eliminate non-Gĩkũyũ words, and to correct diacritics, where necessary. The corpus was then manually annotated where words were categorized into corresponding parts of speech, in line with Hunspell's defined continuation classes. Perl scripts were used for generic annotation and marking. The test corpus was acquired from two sources: a popular Gĩkũyũ blog "Maitũ nĩ ma itũ (Our Mother is our truth)" was chosen as it contains diacritically-marked texts on a variety of contemporary topics and Ngũgĩ wa Thiong'o's novel "Mũrogi wa Kagogo", which was not diacritically marked and represented how a normal user would type on a standard keyboard.

2.5.2.4 HUNSPELL LANGUAGE SETUP FOR GĨKŨYŨ

The spellchecker was implemented using the concept of continuation classes, where a word is represented as a composition of one or more morphemes.

To handle Gĩkũyũ diacritics, it was important to set character support to Unicode (UTF-8). In addition, since Gĩkũyũ verbs generated many affix rules, Flag is set to a number so as to handle the numerous affix rules. The Gĩkũyũ alphabet includes the apostrophe and hyphen, as in ng'ombe and iria-inĩ, and the orthography set was therefore extended with these characters. This was important as it helped Hunspell determine word stops. Since Gĩkũyũ has more than one level of prefixes and suffixes, support for complex prefixes, as well as circumfixation, had to be enabled in Hunspell.

2.5.2.5 SUGGESTIONS COMPONENT

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 TRY command is shown below:
TRY ei˜ıanrtocduu˜gmhbykw’jNRTCGDMHBEAUU˜ YOII˜KWJ

The second command used in the suggestion component is the REPLACE command. The command listed the most commonly misspelled n-grams and their replacements. The major n-grams are a result of influence from different dialects, foreign languages and also by differences in spoken versus written Gĩkũyũ. Examples of Gĩkũyũ replace suggestions included:

REP 35
REP s c
REP sh c
REP sh ch
REP c ch
REP f b
REP v b
REP l r

The developed Gĩkũyũ spellchecker and “suggester” engine was incorporated into OpenOffice Writer and evaluated using the test corpus. The spell checker had a fairly representative Gĩkũyũ lexicon, 19,000 words, and achieved an acceptable realization of a Gĩkũyũ spellchecker. When tested on a test corpus, the spell checker attained a precision of 82%, recall of 84% and an accuracy of 75%.

Given that the developed Gĩkũyũ spellchecker and the Hunspell tools are open source, the spell checking function developed could be adopted in major open-source products such as Mozilla and OpenOffice products.

2.5.3 OPEN SOURCE SPELL CHECKER FOR DHOLUO USING THE HUNSPELL LANGUAGE TOOLS

This Spell Checker was developed by Agola Joshua Otieno (2010) a former MSc. Computer Science Student of University of Nairobi.

The Hunspell tool was used to develop an open source Spellchecker in Dholuo and to create two files namely the affix and the dictionary files. The affix file enabled the creation of all the rules involved in deriving the nouns and the verbs from the root words. All the root words (stems) plus the appended rules were stored in the dictionary file.

This spellchecker was the first for Dholuo and the Hunspell tool was used to develop an open source Spellchecker in the language.

2.5.3.1 HUNSPELL LANGUAGE SPECIFIC SETUP

Character support was set to Unicode (UTF-8). This was necessary to handle diacritics. Flag was set to a number (Flag Num) because Dholuo verbs generated many affix rules. Due to the presence of extra characters; the hyphen and the apostrophe, the orthography set was extended with these characters (WORDCHARS - '). Support for complex prefixes, as well as circumfixation, had to be enabled in Hunspell (COMPLEX PREFIXES; CIRCUMFIX 001).

The suggestions component was implemented in the affix file and the commands TRY and REPLACE were used to generate suggestions for misspelled words.

The developed Dholuo spellchecker was incorporated into the OpenOffice Writer then tested against a test corpus. The system developed achieved an acceptable representation of Dholuo morphology. It correctly classified Dholuo words with an accuracy rate of 0.814, precision rate of 0.917, recall rate of 0.800 and coverage of 0.820.

2.5.4 OPEN SOURCE KIPSIGIS SPELL CHECKER AND LANGUAGE TOOL

This Spell Checker was developed by Ronoh Wycliffe (2011) a former MSc. Computer Science Student of University of Nairobi.

This project entailed the morphology of the various parts of speech of the Kipsigis language, the development of corpus preparation and analysis tool using Jython and a description of the development of an open source Kipsigis spell checker using the Hunspell language tools. Kipsigis is considered to be a resource scarce language whose print and digital usage is low. The development of the spelling recognition system required a lot of manual effort on corpus preparation and analysis. Therefore, this project described the development of a tool that helped to automate and thus speed up this procedure.

Hunspell requires two files to define the language that it is spell checking, the first file is an affix file that defines the meaning of flags, the second file is a dictionary file which contains words alongside flags.

The spell checker tested on four data sets ranging from 460 to 540 words achieved an average accuracy rate of 96%, an average precision rate of 100%, an average recall rate of 95% and an average coverage rate of 94%. The spell checker developed sought to be adopted by leading open source systems such as Open Office, Mozilla and Google chrome.

2.5.5 OPEN SOURCE SPELLCHECKER FOR LUHYA-LULOGOLI

This Spell Checker was developed by Aseyo, John Orege (2011) a former MSc. Computer Science Student of University of Nairobi.

This project entailed the development of spell checking software and described the process of constructing a spell checker for the Lulogoli language and its implementation for the Hunspell spell checker engine. The word list was an adaptation of word roots coming from Hymn books, story books and spoken language. Recognition of morphologically complex words, which are common in Lulogoli due to its agglutinative nature, was made possible by the affix file which had been built based on ready made morpheme segmentation of word derivations appearing in the corpus. Rules derived in

the affix file were improved by semantic classification of all involved roots, for which a system had been created based on corpus analysis in combination with knowledge on the capability of each affix to accept roots from different semantic classes.

The developed spellchecker for Lulogoli language using Hunspell language tools was composed of 13,943 root words and more than 600 affix rules that were used to generate words in the order of more than 100,000 Lulogoli words. Results obtained in applying the developed spellchecker in OpenOffice Writer exhibited a fairly acceptable performance that had practical use in spell checking documents in the Lulogoli but were used with high acceptability for Lutiriki, Lunyole, Lukisa, Luwanga, Luisukha and Luidakho. This tool can also be used for correction and collation of more language corpus for Lulogoli thus bridging the digital divide between Lulogoli and other developed languages. The resulting spellchecker was a working proof of concept, to be further improved and integrated in the Free open source software.

2.5.6 SPELL CHECKERS

2.5.6.1 DEFINITION

A spell checker or spelling checker in computing terms is defined as a design feature or a software program designed to verify the spelling of words in a document, helping a user to ensure correct spelling. A spell checker may be implemented as a stand-alone application capable of operating on a block of text; however, spelling checkers are more often implemented as a feature of a larger document-related application e.g. a word processor or email client, electronic dictionary or search engine.

2.5.6.2 HISTORY OF SPELL CHECKERS

The area of spell checker programs has been researched since 1957. The first spell checker application became commercially available in 1971. This program was developed by Ralph Gorin at Stanford University in the United States. SPELL was originally developed for the DEC-10 and revised versions of this program were still in use more than twenty years later (McGettigan, 1997).

2.5.6.3 SPELL CHECKER OPERATION

The first spelling checkers were "verifiers" instead of "correctors," offering no suggestions for incorrectly spelled words. This was helpful for typographical errors but it was not so helpful for logical or phonetic errors. The challenge faced by developers was the difficulty in offering useful suggestions for misspelled words. This requires reducing words to a skeletal form and applying pattern-matching algorithms. An important feature of the latest word processor programs is that they have this capability.

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 spelling checker customarily consists of two parts:

a) A set of routines for scanning text and extracting words.
b) A wordlist which is the vocabulary and often referred to as a dictionary against which the words found in the text are compared.

The scanning routines sometimes include language-dependent algorithms for handling morphology. Even for a lightly inflected language like English, word extraction routines will need to handle such phenomena as contractions and possessives. It is unclear whether morphological analysis provides a significant benefit.

The wordlist might simply be a list of words, or it might also contain additional information, such as hyphenation points or lexical and grammatical attributes. As an adjunct to these two components, the program's user interface will allow users to approve suggested replacements and modify the program's operation.

One exception to the above paradigm is spelling checkers which use solely statistics, such as n-grams. 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, such as the “see also” entries of encyclopedias.

Recently, research has focused on developing algorithms which are capable of recognizing a misspelled word, even if the word itself is in the vocabulary, based on the context of the surrounding words. Not only does this allow non-sensical errors to be caught, but it mitigates the detrimental effect of enlarging dictionaries.

2.5.6.4 DESIGN

A basic spell checker carries out the following processes:

a) It scans the text and extracts the words contained in it.
b) It then compares each word with a known list of correctly spelled words (i.e. a dictionary). This might contain just a list of words, or it might also contain additional information, such as hyphenation points or lexical and grammatical attributes.
c) An additional step is a language-dependent algorithm for handling morphology. Even for a lightly inflected language like English, the spell-checker will need to consider different forms of the same word, such as plurals, verbal forms, contractions, and possessives. For many other languages, such as those featuring agglutination and more complex declension and conjugation, this part of the process is more complicated.

2.5.6.5 SPELL CHECKING METHODS

Spell checking methods can be divided into two broad categories;

a) Independent Spell Checking Methods
b) Dependent Spell Checking Methods

2.5.6.5.1 INDEPENDENT SPELL CHECKING METHODS

Independent spell checking methods do not utilize a wordlist or vocabulary. Instead they use statistical means to detect misspelled words, thus the term independent.

2.5.6.5.1.1 TOKEN LISTS

This type of method identifies all the distinct words in a file of text and stores the frequency (number of occurrences) of each word. Words with lower frequencies are identified as being potentially misspelled.

2.5.6.5.1.2 DI-GRAMS AND TRI-GRAMS

This method extends the token list concept by using a large corpus of text from the desired language. The frequency of all two-letter pairs (di-grams) or three-letter triplets (tri-grams) 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 erroneous. This technique was very popular in the development of spell checking programs.

2.5.6.5.2 DEPENDENT SPELL CHECKING METHODS

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

2.5.6.5.2.1 DICTIONARY LOOK-UP

Here, 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 not found.

2.5.6.5.2.2 ADVANTAGES AND DISADVANTAGES OF THE ABOVE METHODS

The main advantage of the independent spell checking methods is related to storage space. The fact that no wordlist/vocabulary is maintained means that less space is required for this functionality to be available. This was particularly important in earlier spell checkers as they were used on machines where memory was limited.

Author ignorance exposes the main disadvantage in the use of the Token List method. Misspellings due to author ignorance could go undetected, as this type of error will probably occur consistently and therefore have a high frequency. For example, a user may believe that the correct spelling of the word 'friend' is 'freind'. As the author will always spell the word incorrectly, the spell checker will not tag this as an error. This further implies that the accuracy of a spell checking system developed using this method is questionable.

Another weakness with this method is that rarely used words, by their description will have a low frequency and thus could be highlighted as being potentially incorrect. Since there is no reference besides the actual text being checked, it follows that larger texts will produce more accurate frequency lists and therefore more accurate error detection.

The Di-gram and Tri-gram method proves to be much more accurate, as the peculiarity measurements are collected from a single large corpus of text from the desired language. This eliminates the problem of consistent misspellings by an author going undetected. The popularity of dependent spell checking methods grew in tandem with the increase in available computer memory. While increasing accuracy, they are not infallible. The accuracy of a spell checking method based on a dictionary-look-up program is directly related to the accuracy of the dictionary; it must be both valid and contemporary. The actual size of the dictionary can also cause problems of a grammatical nature. Up to 20% of errors can go undetected because they match other words in the dictionary (Mitton, 1987).

Even with an increase in accuracy, human proof reading is still necessary as the 'Holy Grail' of spell checking systems is yet to be achieved.

2.5.6.6 SUGGESTION ALGORITHMS

2.5.6.6.1 USING NEAR-MISS STRATEGY TO FIND SUGGESTIONS

The first algorithm implemented by a SpellChecker for building a suggestion list is a near miss strategy. It was developed by Geoff Kuenning for ISpell, and makes an assumption

that the word is not necessarily misspelled, but rather mistyped. The misspelled word is changed by altering a letter, deleting or adding it, inserting a blank space, or interchanging two adjacent letters. If these steps result in a word contained in the dictionary, then an estimate of how far from the original word is computed. To measure the proximity of words, the modified Levenshtein distance notion is used.

2.5.6.6.2 USING PHONETIC COMPARISON TO FIND SUGGESTIONS

The phonetic suggestion algorithm takes into account the pronunciation of a word. The **S**pellChecker component utilizes the implementation of the Double Metaphone search algorithm. Two phonetic codes (primary and secondary) are calculated for each word. The calculation rules are different for different languages. They are based on the set of pronunciation rules for that language.

Then, the phonetic strategy compares the phonetic code of the misspelled word to all the words in the word list. If the phonetic codes match, then the word is added to the suggestion list.

2.5.6.6.3 SUGGESTION RANKING

After the list of suggestions is composed, it should be ordered so that the user doesn't have to scroll through it, searching for a perfect match. The implemented solution makes use of the Levenshtein algorithm to calculate the word distance. This distance becomes a parameter for list ordering. Additional assumptions on the nature of a spelling error may help modify the algorithm.

The user makes his/her choice from the list of suggestions. The misspelled word can be replaced with a word from the suggestion list, ignored, or edited by the user. The last possibility indicates a spell checker miss, and provides an option for appending the corrected word to an auxiliary user dictionary.

2.5.7 HUNSPELL LANGUAGE TOOLS

Developing a spell checker requires a method of determining the set of valid words in a given language, against which the words to be checked are compared. Therefore a spell checker customarily consists of two parts:

- A set of routines for scanning text and extracting words. These scanning routines must include language dependent algorithms for handling morphology, in this case for the Kîmîîrû language.
- An algorithm for comparing the extracted words against a known list of correctly spelled words.

Therefore this system utilized the Hunspell tools (N´emeth, 2010), which facilitated the definition of the valid words in a language, as well as the likely suggestions.

Hunspell is a spell checker and morphological analyzer library and program designed for languages with rich morphology and complex word compounding or character encoding. It was originally designed for the Hungarian language, based on MySpell and is backward compatible with MySpell dictionaries.

Hunspell requires two files to define the way a language is being spell checked: a dictionary file containing words and applicable flags, and an affix file that specifies how these flags will control spell checking. An optional file is the personal dictionary file.

2.5.7.1 MAIN FEATURES OF HUNSPELL

The Main features of Hunspell spell checker and morphological analyzer are;

- Unicode support where affix rules work only with the first 65535 Unicode characters.
- Morphological analysis in custom item and arrangement style and stemming.
- Contains a maximum of 65535 affix classes and twofold affix stripping for agglutinative languages, like Azeri, Basque, Estonian, Finnish, Hungarian and Turkish.
- Support complex compoundings in languages like Hungarian and German.

- Support language specific features (for example, special casing of Azeri and Turkish dotted i, or German sharp s.
- Handle conditional affixes, circumfixes, fogemorphemes, forbidden words, pseudoroots and homonyms.
- Free software of the varieties of LGPL, GPL, MPL tri-license.

CHAPTER 3 : METHODOLOGY

3.1 ANALYSIS

The methodology utilized to develop this system was the Structured System Analysis and Design (SSAD). SSADM is an integrated set of standards and guides for the analysis and design of computer systems consisting of:

- Structural standards which define the structure of a development project in the form of explicitly defined tasks, with clearly defined interfaces between them, and clearly defined tangible products.

- Technique guides which provide a set of proven usable techniques and tools, and detailed rules and guidelines on when and how to use them.

- Documentation standards which provide the means of recording the products of development activity at a detailed level.

This methodology was adopted mainly for the following reasons:

- It is a linear sequential model where all requirements are clearly understood and stated upfront.
- It provided a systematic step by step approach in development of the spellchecker.
- Economic feasibility: hardware and software resources are readily available.
- Technological feasibility: current software technology will easily support the implementation of the system.
- Operational feasibility: intended users will have basic computer knowledge.

3.1.1 SCOPE ANALYSIS

The system should be able to perform a morphological analysis of Kîmîîrû language highlighting nouns and verbs derivation and also provide a suggestion component used to generate probable suggestions for a misspelled word. The system did not include a grammar checking facility or a thesaraus. The Spellchecker was also limited in scope to

focus on the Kiimenti dialect and hence did not cover any of the eight (8) other sub-groups in the larger Mîîrû community.

3.1.2 PROBLEM ANALYSIS

Development of this Kîmîîrû spellchecker system was necessitated by

- Lack of a Kîmîîrû spell checking system in both proprietary and open source platforms.
- Kîmîîrû's classification as a resource scarce language (RSL).
- Scanty digital documentation on Kîmîîrû language.

3.1.3 DECISION ANALYSIS

The developer prescribed the following recommended course of actions during the development of the spellchecker:

- Inclusion of diacritic marks for high tones within the wordlist.
- Omission of diacritic marks for low and falling tones.
- Bias in inclusion of religious material (Bible verses) for populating the test corpus.
- Exclusion of CorpusCatcher tool during corpus collection because of little digital documentation on Kîmîîrû language.

3.2 SYSTEM DESIGN

According to Paggio and Music (1998), valid words are words that are part of the language, or which are sanctioned by the language system, in contrast to invalid words, which are not part of the lexicon or language system. When the spelling checker claims a word is invalid, it is flagging that word, while accepting a word means treating it as valid. Accordingly, a flag is an indication that a word has been tagged as invalid (regardless if the word really was invalid or not). Suggestions are alternative valid words that are offered to the user to replace a flagged word with. Therefore the goal of a spelling checker is to flag all invalid words and to accept all valid words. If this is the case, all invalid words are correctly flagged.

Therefore a user to this system will have three choices:

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

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

3.3 MAJOR COMPONENTS OF THE DEVELOPED SPELL CHECKER SYSTEM

3.3.1 GRAPHICAL USER INTERFACE (GUI)

A GUI is a type of user interface which allows people to interact with a computer and/or computer-controlled devices that employ graphical icons, visual indicators or special graphical elements (Galitz, 2010).

An effective GUI includes text, labels or text navigation to clearly present the information and actions available to a user. The actions are usually performed through direct manipulation of the graphical elements. The functionality of an application can be programmed perfectly, but if the GUI is hard to interpret or annoying to use, then the program ultimately will be a failure and the end user will likely choose something easier or more convenient. Creating a simple, easy to use GUI is vital to the success of a software project.

The process for developing useful GUIs includes several key steps including:

- Creating a GUI Design and Development Plan.
- Understanding the Key Elements in GUI Design.
- Modeling System Interactions.
- Designing Screen Layout and User Interaction.
- Effectively Presenting the Data.
- Reviewing and Verifying the Design

In regards to ease of learning the developer decided to adopt the windows ‘look and feel’ as this is the most widely encountered by users and would present the most “natural” interface.

3.3.2 SPELL CHECKER

A spell checker or spelling checker in computing terms is defined as a design feature or a software program designed to verify the spelling of words in a document, helping a user to ensure correct spelling. A spell checker may be implemented as a stand-alone application capable of operating on a block of text; however, spelling checkers are more often implemented as a feature of a larger document-related application e.g. a word processor or email client, electronic dictionary or search engine.

3.3.3 WORDLIST

The maintained wordlist contains a comprehensive collection of approximately 4000 correctly spelt words from the Kîmîîrû vocabulary. The same wordlist also contains flags that correspond with stemming rules defined within the affix file. This flags are used to determine the type of affixation required for a given word.

The following figure illustrates the spellchecking operation:

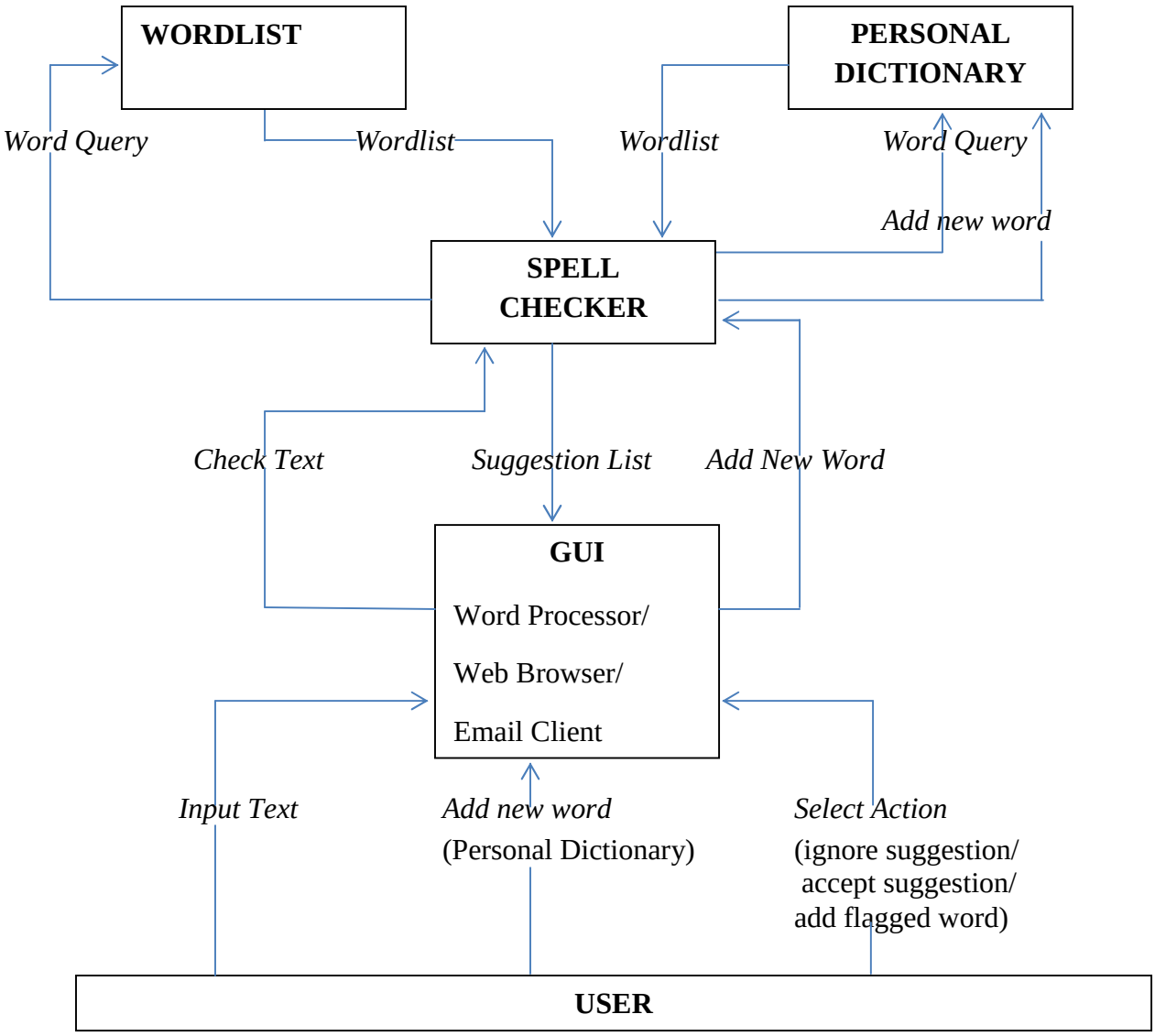

Figure 2 : Spell checking operation in Kîmîîrû Language

3.3.4 PERSONAL DICTIONARY

A personal dictionary is a component that stores words not captured in the primary word list and yet are perceived to be correct by the user. Hence, new words are appended in the personal dictionary after a user opts to include a word that has been flagged as erroneous by the system.

3.4 CORPUS COLLECTION

The primary source of Kîmîîrû words were obtained from bible verses of the Kîmîîrû bible (2010) and Internet Resources, specifically a podcast of 100 Kîmîîrû words by Wahome Kaburu (2005). The bible was first scanned and Optical Character Recognition (OCR) software was used to make the text available for editing. The word list was then manually cleaned to get rid of non - Kîmîîrû words and also diacritic restoration was performed where possible. The outcome was a single wordlist containing approximately 4,000 unique words. By this time, the Hunspell affix file was almost fully developed and it was used for applying rules to the wordlist in the Hunspell dictionary file.

3.5 IMPLEMENTATION OF THE KÎMÎÎRÛ SPELLCHECKER

The system was fully developed using Hunspell language tools. Cygwin which is a Linux-like environment for Windows was used as the main development and deployment platform because a UNIX/ LINUX environment is mandatory for deploying hunspell tools.

Cygwin consists of a DLL (cygwin1.dll), which acts as an emulation layer providing substantial POSIX (Portable Operating System Interface), system call functionality, and a collection of tools, which provide a Linux look and feel.

Two primary files were defined using Hunspell tools during the development phase; The Affix file defines the meaning of special flags in the dictionary. An affix file (*.aff) may contain a lot of optional attributes. For example, **SET** is used for setting the character encodings of affixes and dictionary files. **TRY** sets the change characters for suggestions. **REP** sets a replacement table for multiple character corrections in suggestion mode. **PFX** and **SFX** defines prefix and suffix classes named with affix flags. (N´emeth, 2010).

A dictionary file (*.dic) contains a list of words, one per line. The first line of the dictionaries (except personal dictionaries) contains the approximate word count (for optimal hash memory size). Each word may optionally be followed by a slash ("/") and one or more flags, which represents affixes or special attributes. Dictionary words can

contain also slashes with the "" syntax. Default flag format is a single (usually alphabetic) character. In a Hunspell dictionary file, there are also optional fields separated by tabulators or spaces.

Hunspell tools are used to satisfy user requirements and are executed within the Cygwin bash shell and test restricted to checking one input string at a time.

After development phase the system was deployed in an Open Source Word processing application called OpenOffice.org Writer 3.4.1. This provided an interface from which end users could input blocks of text that were to be checked for spelling errors. At this stage the Test corpus was extracted from printed religious literature, specifically bible verses from the Kîmîîrû bible (2010) and the Kimeru bible (1964). These verses were not previously used in the Training corpus.

3.6 HUNSPELL LANGUAGE SPECIFIC SETUP FOR KÎMÎÎRÛ

As with all Bantu languages, Kîmîîrû exhibits an SVO word order, agglutinative verb structure, and is highly tonal. In addition, circumfixation is prevalent in Kîmîîrû where a certain suffix can only co-occur with a given prefix or set of prefixes. Therefore to handle the Kîmîîrû diacritics it is important to set character support to Unicode (UTF-8). The Kîmîîrû alphabet includes the apostrophe and hyphen as in *Ing'anagia* (I get satisfied/enough) and *O-jûmwe* (just one) and therefore the orthography set is extended with these characters. The importance of this cannot be overemphasized because it helps Hunspell determine word stops. Also Kîmîîrû supports a battery of affixes at different levels hence support for complex prefixes as well as circumfixation has to be enabled in Hunspell.

Once downloaded and installed the following optional commands (attributes) in Hunspell were enabled to support the Kîmîîrû language

- SET *UTF-8*

  This attribute is enabled to handle the Kîmîîrû diacritic marks.

- FLAG *NUM*

  The ‘num’ sets the decimal number flag type. Decimal flags numbered from 1 to 65000, and in flag, fields are separated by commas. For Kîmîîrû the Prefixes were allocated the range 1-199 while Suffixes started from 201 onwards.

- COMPLEXPREFIXES

  This attribute is enabled to support a battery of Kîmîîrû affixes at different levels.

- NEEDAFFIX *001*

  This flag signs virtual stems in the dictionary. Only affixed forms of these words will be accepted by Hunspell hence this command will facilitate circumfixation for Kîmîîrû language.

- CIRCUMFIX *002*

  Circumfixation is prevalent in Kîmîîrû language where a certain suffix can only co-occur with a given prefix or set of prefixes hence when this command is enabled it would support the same.

- WORDCHARS

  The Kîmîîrû alphabet includes the apostrophe and hyphen therefore the orthography set is extended with these characters.

3.6.1 SUGGESTIONS COMPONENT

In Hunspell the suggestions component is used to generate probable suggestions for a misspelled word. It is implemented in the affix file that employs both independent and dependent spellchecking methods; dictionary look up and n-grams respectively. Dictionary look-up is done by Levenshtein Distance. Levenshtein distance is a measure of the dissimilarity between two strings, refered to 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. The greater the Levenshtein distance, the more different the strings are.

For example:

- If s is "test" and t is "test", then LD(s,t) = 0, because no transformations are needed. The strings are already identical.
- If s is "test" and t is "tent", then LD(s,t) = 1, because one substitution (change "s" to "n") is sufficient to transform s into t.

The Levenshtein distance algorithm has previously been used in:

- Spell checking
- Speech recognition
- DNA analysis
- Plagiarism detection

Hunspell uses two sections in the affix file when generating suggestions for misspelled words. The first is the **TRY** command. This lists the language's orthography set in order of frequency. 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 8 : Kîmîîrû orthography set character count

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
1817	322	126	118	527	1	770	329	1007	257	851	9	663	1056	664	19	0	1131	21	818	642	4

w	x	y	z	û	î	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
476	0	206	3	580	815	194	128	53	2	16	0	151	2	164	95	362	1	344	439	25	1

Q	R	S	T	U	V	W	X	Y	Z	Û	Î
0	140	11	114	39	4	35	0	46	0	99	137

Therefore the resultant output for TRY command is as shown below;

TRY < arniktîgomûuewhbjycdsplvzf'NKMAITGRÎBÛJCYUWOESVDHLP>

The Second command used in the suggestion component is the REPLACE (REP) command which is a fixed list of the most common misspelled n-grams and their replacements within the Kîmîîrû vocabulary. Major n-grams are a result of influence

from different dialects, foreign languages and also by differences in written from spoken Kîmîîrû word. N-grams are derived from 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. The remaining n-grams were a representation of the disparity between written and spoken Kîmîîrû words e.g. *c* - written, *ch* – spoken.

Examples of Kîmîîrû replace suggestions include:

REP 47
REP s c
REP sh c
REP sh ch
REP ch c
REP c ch
REP f b
REP v b
REP l r
REP d nd
REP g ng
REP ng g
REP ng ng'

3.6.2 AFFIXATION COMPONENT

An affix is either a prefix or a suffix attached to root words to make other words. Affix classes were defined with an arbitrary number affix rules. Affix classes are signed with affix flags. The first line of an affix class definition is the header. The fields of an affix class header are:

- Option name (**PFX** or **SFX**).
- Flag (name of the affix class); could be a character or a decimal number.

- Cross product (permission to combine prefixes and suffixes). Possible values are either **Y** (yes) or **N** (no).
- Line count of the following rules or the sequence of entries needed to properly store the affix information.

Accepted example would be: PFX 1 Y 6

The Fields of affix rules will contain the following

- Option name (**PFX** or **SFX**).
- Flag (name of the affix class).
- Stripping characters from beginning (at prefix rules) or end (at suffix rules) of the word.
- Affix (optionally with flags of continuation classes, separated by a slash)
- Conditions to be met before an affix can be applied.
- Zero stripping or affix is indicated by zero. Zero condition is indicated by dot. Condition is a simplified, regular expression-like pattern, which must be met before the affix can be applied.

Accepted example would be: SFX 1 y ied [^aeiou]y or SFX 1 0 ed [^ey]

3.6.3 NOUN COMPONENT

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

PFX 1 Y 6

PFX 1 mu a mu

PFX 1 mû a mû

PFX 1 mw a mw

PFX 1 mû û mû

PFX 1 mw e mw[e]

PFX 1 mû n mû[t]

PFX 2 Y 8

PFX 2 mû mî mû

PFX 2 mu mi mu

PFX 2 mw mî mw

PFX 2 mw mi mw

PFX 2 mû kî mû[gr]

PFX 2 mû ma mû[gr]

PFX 2 mûî mî mûî

PFX 2 mwî mî mwî

#Inside the Dictionary File

mûcoore/1

mûcûnkû/1

mûgambo/1,2,203

mûgate/2,203

mûgeni/1,2

mûgiro/1,2,203

Accepted words with this example: *acoore, mîgambo, ageni, kîgeni*

Derived nouns are formed through circumfixation. The CIRCUMFIX and NEEDAFFIX are used to enforce circumfixation. Examples of circumfixation include:

PFX 18 Y 9

PFX 18 0 mw/002 [e]

PFX 18 0 gûko/002 [o]

PFX 18 0 ico/002 [o]

PFX 18 0 îtîo/002 [o]

PFX 18 0 mû/002 [r]

PFX 18 o jwono/002 [o]

PFX 18 o aro/002 [o]

PFX 18 o wo/002 [o]

PFX 18 o yo/002 [o]

SFX 210 Y 5

SFX 210 ia io/18,002

SFX 210 a e/18,002

SFX 210 0 gwa/18,002

SFX 210 a ia/18,002

SFX 210 ia i/18,002

#Inside the Dictionary File

ora/5,6,12,13,18,201,202,203,204,205,210

rîîthia/5,6,12,18,201,203,204,210

Accepted words with this example *: jwonore, aroragwa, mûrîîthi*

3.6.4 VERB COMPONENT

The Kîmîîrû verb is mostly expressed by the imperative and the infinitive and is marked by the particle **'kû'** or '**gwa**' or '**gwî**'. The particles can be described as the equivalent of the English infinitive 'to'. Infinitive verbs often end in **–a** while irregular ones end in **–ya** (Ataya, 2012).

The Kîmîîrû spellchecker is implemented using the concept of continuation classes, where a word is represented as a composition of one or more morphemes as shown below:

Foc + Subj + Neg + Cond + Tense + Obj + Redup + Verb + DvbExt + Asp + FVwl

The continuation classes for verbs cater for the focus marker, concord subject, negation, conditional, tense, object classes, deverbal extensions, aspectual markers and final vowels. From the foregoing, verbs can have up to seven prefixes and four suffixes but Hunspell only supports a maximum of three prefixes and three suffixes. To overcome this hurdle all prefixes were combined into one complex prefix. The singular complex prefix approach that was adopted is as shown below in the following verb component definition.

#Class of nonpersonal things, size-kî-,-gî,-i-,-bi-,

PFX 4 Y 7

PFX 4 0 ara [magkc]

PFX 4 0 a [enmgkcrtû][^t]

PFX 5 Y 7

PFX 5 0 kû [ûaecgriîkbmnot][^tgm]

PFX 5 0 tû [mrgcknotû][^û]

PFX 6 Y 8

PFX 6 0 ba [uegacîimknrtû] [^fqsvxzt]

PFX 6 0 bû [emgnoabkrîtuû] [^tfnqsvxzgk]

#Transitive verbs,-gan-,-nkû-

PFX 12 Y 11

PFX 12 0 ya [abcgkmrtû]

PFX 12 0 nkû [egajtimnoîrû]

#Verb inflections,causative form,-ia-,-ithia-

SFX 201 Y 9

SFX 201 a ithia [bmnktygdru][^rk]a

#Verb applicative form,-îra-,-era-,

SFX 202 Y 9

SFX 202 a îra [bhmdntkrgjyu][^n]a

SFX 202 a îrwa [tgrkmdwbn]a

#Verb reciprocative,locative form,-eeni-,-ieni-

SFX 203 Y 12

SFX 203 a eeni [mkndgwryhjbut]a

SFX 203 0 ga [ymudtnbjrgiîehk]a

#Verb passive form,-wa-,-ua-

SFX 204 a wa [kndtgmrhbyuj]a

SFX 204 0 gwa [ghrîkmnbudjyteu][^d]a

In the Dictionary File

gûra/4,5,6,12,16,201,202,203,204

kworota/4,5,6,12,201,202,203,204

Accepted words with this example: *agûra, kûgûra, bagûra, nkûgûra, gûrithia, gûrîrwa, gûrîra, gûreeni, gûraga, gûrwa, gûragwa, akworota, tûkworota, bakworota, yakworota, kworotia, kworotîra, kworotaga, kworotwa.*

Hence, the derivation of new verbs in Kîmîîrû language occurs when a case extension is suffixed to an existing verb base. The first method involves suffixing a thematic extension after the verb base or stem and before the verbs final vowel, while the second method occurs when a verbalizing extension attaches to a noun.

Thematic extensions mark special relationships between verbs and their subject or object noun phrases and include causative, applicative, reciprocative and passive forms.

Causative Form

SFX 201 Y 2

SFX 201 a ithia [bmnktygdru][^rk]a

SFX 201 a ia [bdmnkgyrt][^r]a

In the Dictionary File

îta (go)/201

ona (see)/201

Accepted words with this example: *îtithia(*make to go*), onia(*make to see*).*

Applicative Form

SFX 202 Y 2

SFX 202 a îra [bhmdntkrgjyu][^n]a

SFX 202 a era [kntygmrb]a

In the Dictionary File

îta (go)/202

ona (see)/202

Accepted words with this example*: îtîra(*go for*), onera(*see for*).*

#Reciprocative Form

SFX 203 Y 2

SFX 203 a eeni [mkndgwryhjbuti]a

In the Dictionary File

ona (see)/203

teethania (help)/203

Accepted words with this example: *oneeni (see each other), tethanieeni (help each other).*

#Passive form

SFX 204 Y 2

SFX 204 a wa [kndtgmrhbyuj]a

SFX 204 ia ua [khnrgy]ia

In the Dictionary File

ona (see)/204

oria (heal)/204

Accepted words with this example: *onwa (*be seen*), orua(*be healed*)*

CHAPTER 4 : TESTING AND RESULTS

4.1 TESTING

Testing of the Kîmîîrû spellchecker was done in two phases. The First phase involved the use of the Hunspell tools operated from the Cygwin Bash Shell to establish the various user requirements of the spell checker and this was performed iteratively. In this phase testing was limited to one input string at a time.

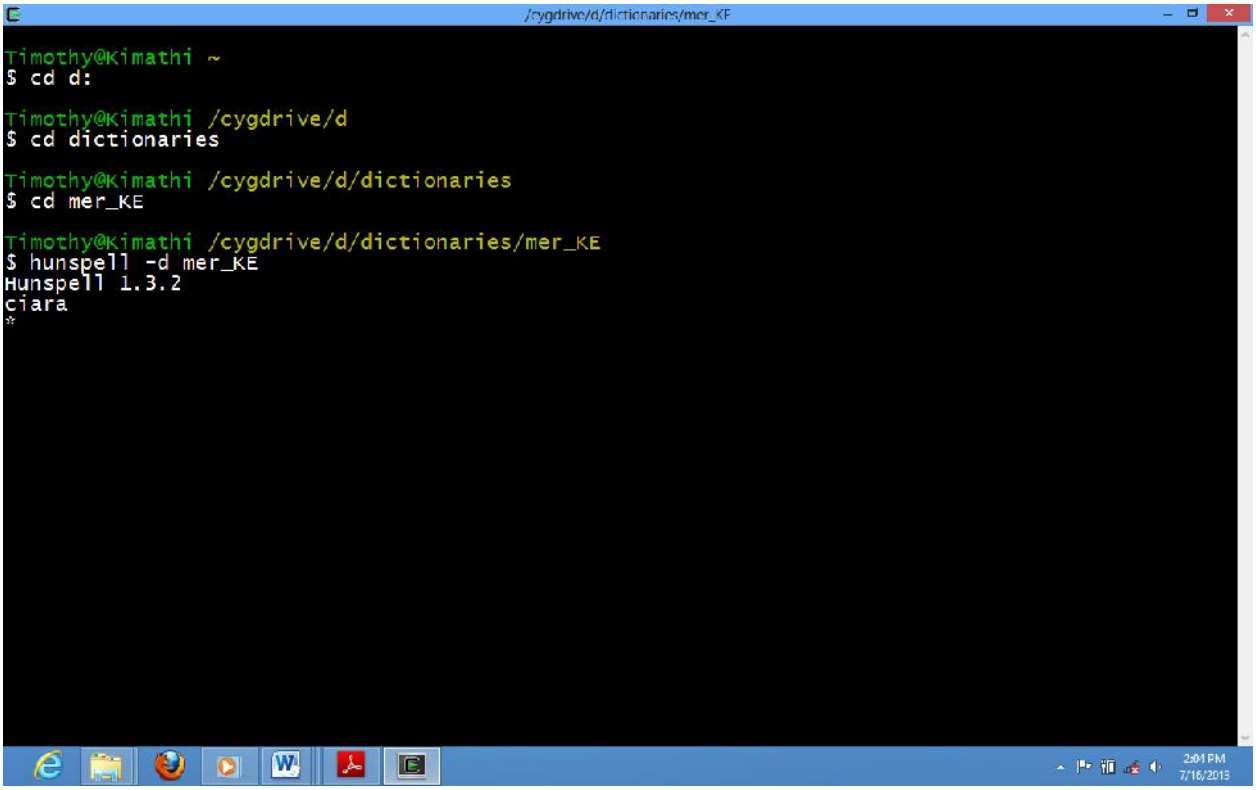

Figure 3 : Word found in the Main and/or Personal Dictionary.

For each input line, a single line is written to the standard output for each word checked for spelling on the line. If the word is found in the main dictionary, or personal dictionary, then the line contains only a '*'. In this case the word *ciara* (give birth) was found in the main dictionary.

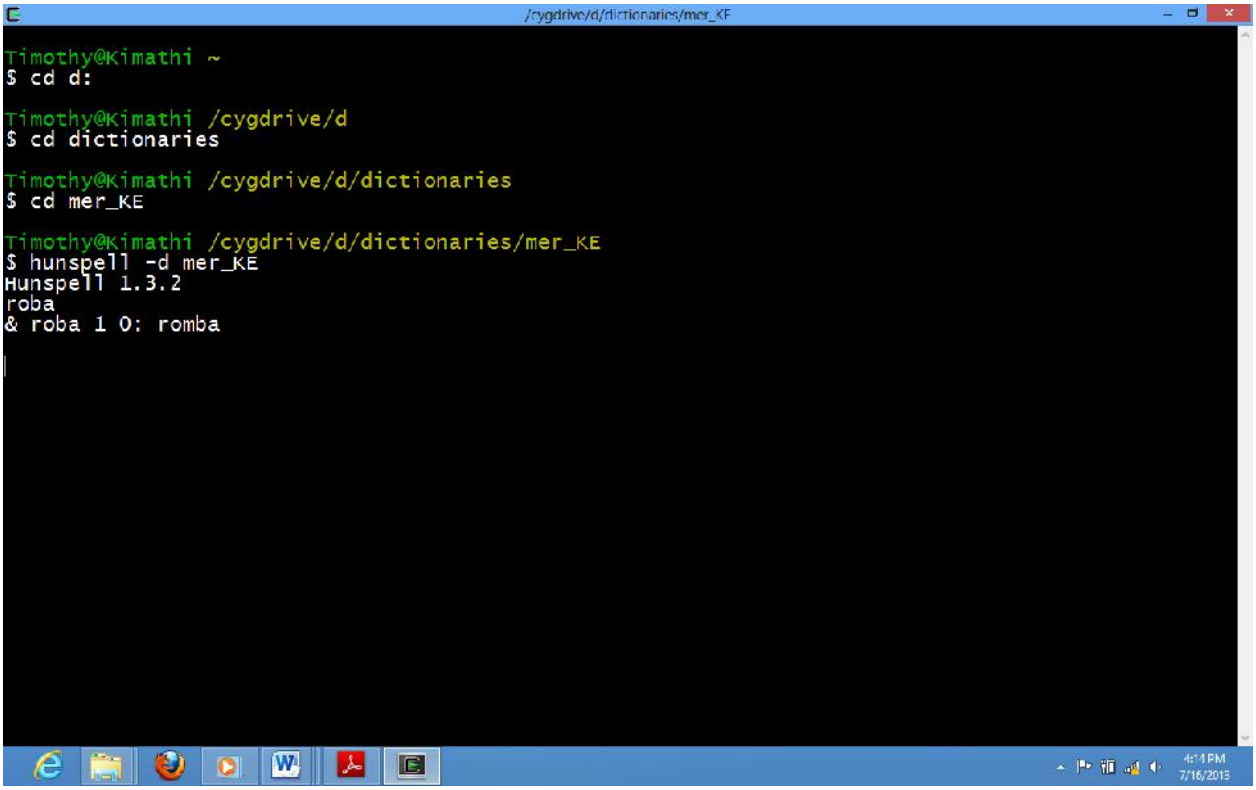

Figure 4 : Word not found in Main and/or Personal Dictionary

If the word is not in the dictionary, but there are near misses, then the line contains the symbol **'&'**, a space, the misspelled word, a space, the number of near misses, the number of characters between the beginning of the line and the beginning of the misspelled word, a colon, another space, and a list of the near misses separated by commas and spaces.

In this case the misspelled word is *roba*, the number of near misses are one (1), the character offset is zero (0) and the correct spelling is *romba (*pray).

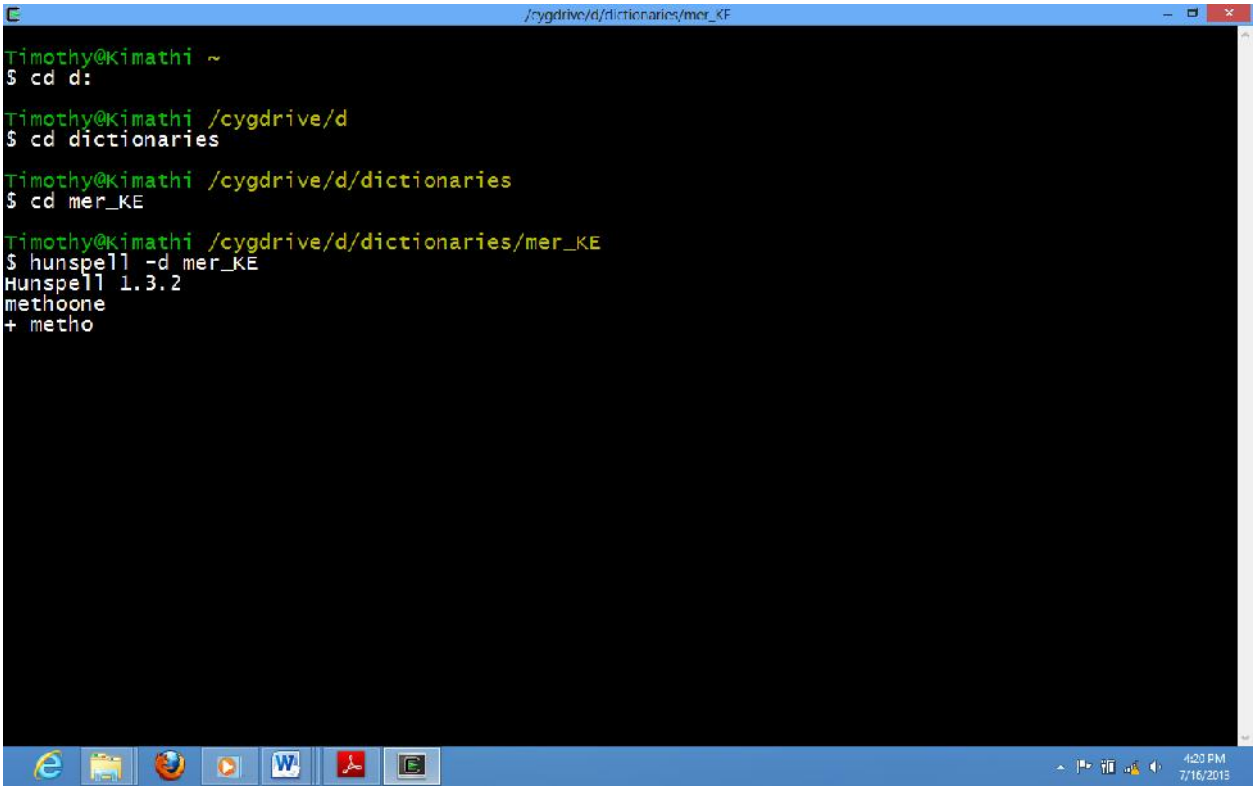

Figure 5 : Word found through Affix removal

If the word was found through affix removal, then the line contains a **'+'**, a space, and the root word. In this case *metho* (eyes) is the root word of *methoone* (in my eyes).

Finally, if the word does not appear in the dictionary, and there are no near misses, then the line contains a '#', a space, the misspelled word, a space, and the character offset from the beginning of the line. Each sentence of text input is terminated with an additional blank line, indicating that Hunspell has completed processing the input line.

These output lines can be summarized as follows:

- OK: *
- Root: + <root>
- Compound: –
- Miss: & <original> <count> <offset>: <miss>, <miss>, ...
- None: # <original> <offset>

Below are two illustrations depicting the use of words from the test corpus. The words were derived from the Kimeru bible (1964).

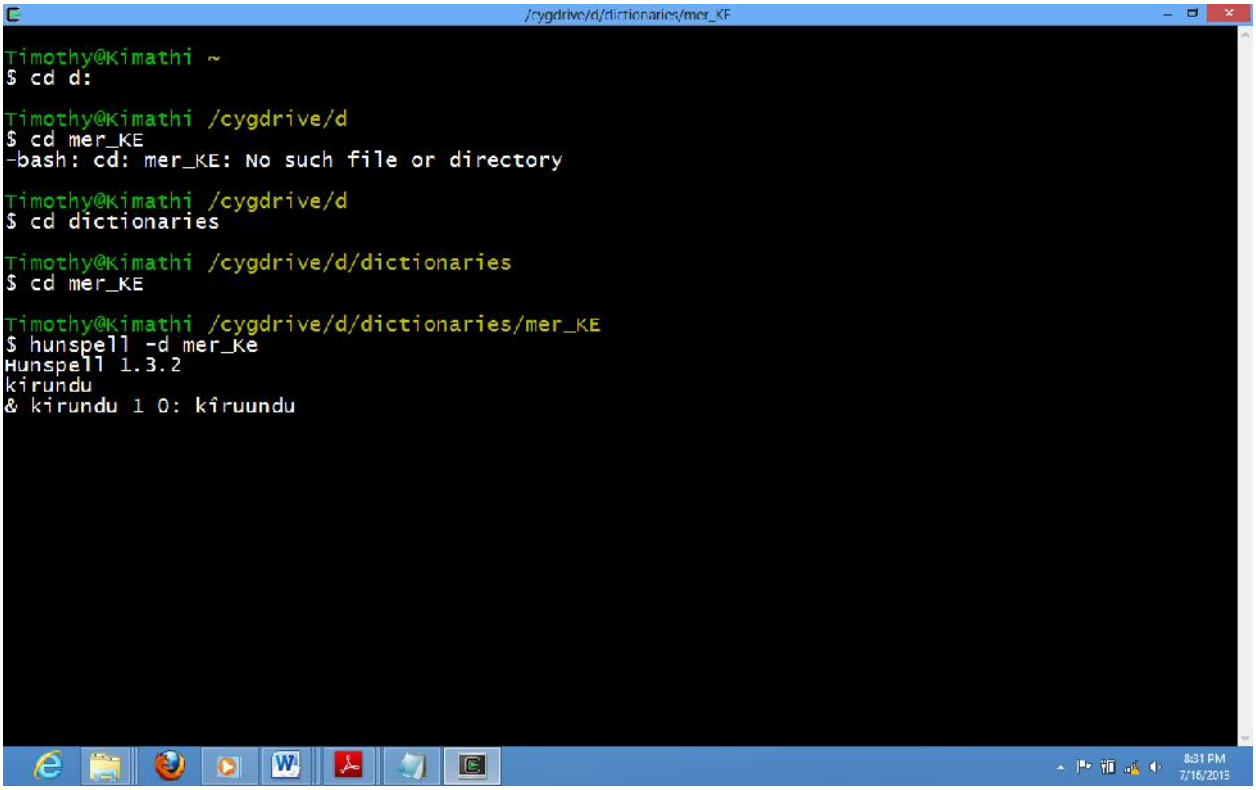

Figure 6 : Word not found in Main and/or Personal Dictionary using test corpus

In this illustration the misspelled word *kirundu* is corrected to *kîruundu* (Holy Ghost) complete with diacritic restoration and doubling of the vowel ***'u'***.

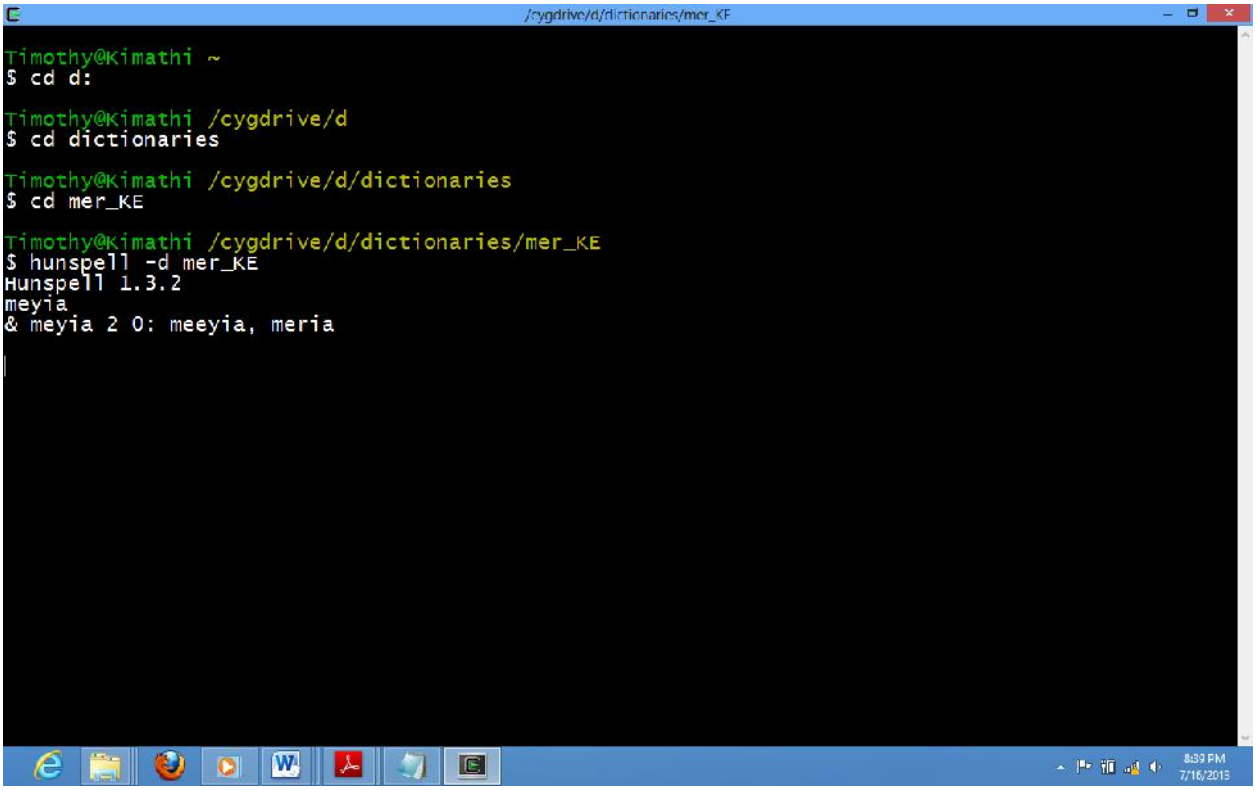

Figure 7 : Word not found in Main and/or Personal Dictionary using test corpus

In this instance the misspelled word *meyia* has two misses. The first miss *meeyia* (sins) gives the correct spelling while the second miss is a suggestion for replacement of the character '*y*' with '*r*'.

The Second Phase of testing involved integrating the Kîmîîrû Spell Checker in the OpenOffice.org Writer version 3.4.1, and using it to check blocks of text within the test corpus. The OpenOffice.org Writer enables dictionaries to be added as extensions.

Below is a screenshot of the functional Kîmîîrû spellchecker deployed within the OpenOffice.org Writer Application.

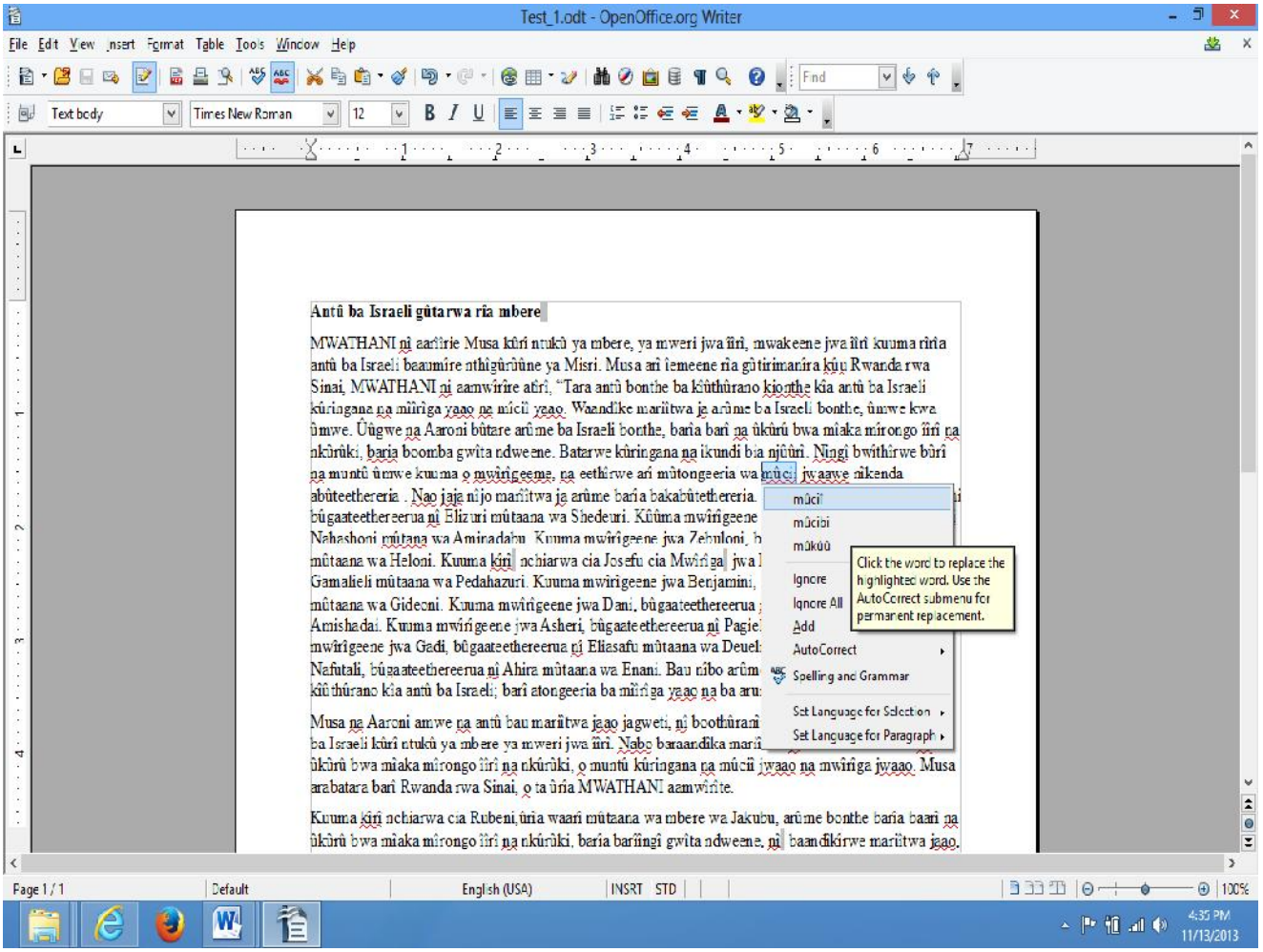

Figure 8 : Kîmîîrû Spell Checker deployed in Open Office.org Writer

4.2 EVALUATION

The developed Kîmîîrû spellchecker and suggester component was incorporated into OpenOffice Writer 3.4.1 and evaluated using the test corpus derived from two sources; the Kimeru bible (1964) and the Kîmîîrû bible (2010). The Kimeru bible (1964) lacks the two diacritically marked characters (**î and û**) in the Kîmîîrû orthography that require extra keystrokes to generate.

In this project, the evaluation of test results was based on the following four basic outcomes:

- True Positives (TP) represents those correctly spelled words that are recognized as such by the spell checker.
- False Positives (FP) will represent misspelled words that are not flagged as such.
- True Negatives (TN) will represent misspelled words that are flagged as such.
- False Negatives (FN) will represent correctly spelled Kîmîîrû words that are flagged as misspelled.

The results obtained are shown in the following table and they represent the evaluation metrics.

Table 9 : Evaluation of test results based on four outcomes

RESULTS	**TP**	**FP**	**TN**	**FN**	**TOTAL**
Coverage	1554	0	197	450	2201
Precision	TP/(TP + FP) = 100				
Recall	TP/(TP + FN) = 0.78				
Accuracy	(TP + TN)/TOTAL = 0.80				

Precision is a measure of the exactness of the spellchecker's responses. It basically confirms how much one should trust the spellchecker when it accepts a given word as correct.

Recall is a measure of the completeness of the spellchecker. It tells how much of the language the spellchecker covers, the lower the value the more likely it is that the spellchecker will complain about correct words.

Accuracy is derived from both precision and recall and is a general measure of the quality of the spellchecker.

4.2.1 MAJOR CHALLENGES

On analysis of the results some challenges were noted. Mainly, there was an abundant over-generation of suggestions. Previous spell checker developers have cited this phenomenon which refers to the uncontrolled combination of prefixes leading to generation of numerous words that are not semantically correct in a given language's grammar. Over-generation can be reduced through the use of separate definition of affix rules, as opposed to clustering several related rules together.

There was a high prevalence of False Negatives (FN). The test corpus relied on the Kimeru bible (1964) and the Kîmîîrû bible (2010). The former source lacks the two diacritically marked characters (**î and û**) in the Kîmîîrû orthography that require extra keystrokes to generate while the latter source's majority of nouns possessed doubling of vowels especially **'a'** and **'u'.** The different Kîmîîrû morphological compositions of the two bibles therefore led to correct words being flagged as incorrect. Another cause of this phenomenon was the absence of some word stems in the dictionary file, including but not limited to names of people and places. The accuracy of this spellchecker can be greatly improved by having a corpus containing substantial Kîmîîrû words to rival those of the Dholuo, Luhya-lulogoli and ofcourse the Gĩkũyû, just to mention a few.

During the spellchecking process there was a high generation of True Negatives (TN) especially when spellchecking diacritically marked texts. The suggestion component comfortably provided suggestions for the same and therefore the misspelled words were easy to correct. However the suggestion component did not cope suitably with situations where texts were missing the diacritic marks or a combination of a diacritic and one or more characters.

CHAPTER 5 : DISCUSSION

5.1 OVERVIEW

The overall aim of this project was to develop an Open Source Spellchecker for Kîmîîrû Language. The previous chapters in this report detailed the research, design and implementation of this system. This final chapter contains an evaluation of all aspects of the project; the achievements of the project, the limitations of the system and recommendations for future enhancements are herein discussed.

5.2 ACHIEVEMENTS

The achievements of this project evaluate the objectives set forth on the onset:

- This project has demonstrated the development of a Kîmîîrû spelling checker with an acceptable performance of an accuracy of 80%, a precision of 100% and a recall of 78%. Therefore this resulting spellchecker is a working proof of concept, to be further improved and integrated in the free open source software.
- Recognition of morphologically complex words, which are common in Kîmîîrû language due to its agglutinative nature, was made possible by the affix file.
- Rules derived in the affix file were improved by semantic classification of all involved roots. This was made possible by adopting the singular complex prefix approach.
- The developed spellchecker for Kîmîîrû language using Hunspell language tools was composed of 4000 root words and more than 400 affix rules that were used to generate nouns and verbs in the order of more than 40,000 Kîmîîrû words. This further enhanced the understanding of the fundamental principles and morphological composition of Kîmîîrû nouns and verbs.

5.3 LIMITATIONS

The limitations of the developed Open Source Kîmîîrû Spell checker are:

- The system is only able to perform nouns and verbs derivation neglecting pronouns, adverbs, adjectives and prepositions which also form the bulk of Kîmîîrû language.
- The system does not include a grammar checking facility nor a thesaurus.
- Word sense disambiguation is not achieved, meaning that semantic errors are not caught. This would need the inclusion of other complex tools that check not only the spelling of a word, but also its propriety in relation to context.
- The spellchecker's focus is only on the Kiimenti dialect and does not cover any of the eight (8) other sub-groups in the larger Mîîrû community.
- Tone marking of the words in the vocabulary is not present. This is one major limitation when reading and understanding text in the language since Kîmîîrû is a tonal language.

5.4 RECOMMENDATIONS

Recommendations for improvement and further work are closely related to the limitations outlined above. They include the following:

- The inclusion of grammar checking or context sensitive spellchecking component. Addition of graphemic information to the system will go a long way in achieving this.
- Tone marking of words in the vocabulary would greatly enhance ease of reading text written in the language.
- The inclusion of a wider scope of Kîmîîrû part of speech (POS) to increase the accuracy of creation of Kîmîîrû texts.
- Developing a spellchecker to cover the other eight (8) sub-groups of the larger Mîîrû community. This will involve standardization of the said dialects through the development of literature for teaching, and learning the vernacular of the same. The Kîtharaka dialect would be a good starting point because the dialect recently attained standardization through translation of the Bible.

5.5 CONCLUSION

The major focus of this important project was the development of an open source spellchecker for Kîmîîrû language using the Hunspell language tools. The results obtained in deploying the developed spellchecker in OpenOffice Writer version 3.4.1, have shown an acceptable performance with an accuracy of 80%, a precision of 100% and a recall of 78%. The developed spellchecker will be very useful in the collation and correction of additional Kîmîîrû texts during corpus compilation. While the provided dictionary comprising over 4,000 unique words may not have all words currently used in Kîmîîrû, it is presented in such a way that users of the system can add to it, enabling its growth and a more accurate documentation of the language.

As mentioned earlier the developed system may not only be of use in the development of similar systems for the other eight (8) Kîmîîrû subgroups, but to also other local languages that are closely related to Kîmîîrû; for instance, Kĩembu, which has a lexical similarity of 73% and Kĩkamba, with a similarity of 67%.

As Dhonnchadha et al., (2003) aptly put it “all languages must endeavour to keep up with and avail of language technology advances if they are to prosper in the modern world”.

APPENDICES

APPENDIX 1

REFERENCES

1. Agola, J. O., (2010). Open source spell checker for Dholuo using the Hunspell Language Tools. *MSc. University of Nairobi.*
2. Aseye, J. O., (2011). *Open source spellchecker for Luhya-Lulogoli.* MSc. University of Nairobi.
3. Ataya, J. K., (2012). *Advanced Kîmîîrû Grama*. QBS Publications.
4. Bański, P. and B. Wójtowicz., (2009). *A Repository of Free Lexical Resources for African Languages: The Project and the Method.* De Pauw, G. et al. (Eds.). 2009: 89-95.
5. Bwayo, J., (2011). *Open source spellchecker for Bukusu dialect.* MSc. University of Nairobi.
6. Chege, K.et al., (2010). *Developing an open source spell checker for Gîkûyû*. In Proceedings of the Second Workshop on African Language Technology (AfLAT, 2010).
7. De Pauw, G and Wagacha P.W., (2007). *Bootstrapping Morphological Analysis of Gikuyu Using Unsupervised Maximum Entropy Learning*. Proceedings Eighth INTERSPEECH Conference.
8. De Pauw, G. and Wagacha, P.W., (2007). *Bootstrapping morphological analysis of Gĩkũyũ using unsupervised maximum entropy learning*. In H. Van hamme & R. van Son (Eds.), Proceedings of the Eighth Annual Conference of the International Speech Communication Association. Antwerp, Belgium.
9. Fridah , E. K., (2011). *Meru Dialects: The Linguistic Evidence.* Nordic Journal of African Studies 20(4): 300–327.
10. Muriithi, B. K., (2008). *Word processor for Gĩkũyũ language with spell check.* MSc. University of Nairobi.
11. N´emeth, L., (2010). Hunspell. [Online]. Available: http://hunspell.sourceforge.net (accessed: July 2012).

12. Ronoh, W., *(*2011). *Open source Kipsigis spell checker and language tool*. MSc. University of Nairobi.
13. Wagacha, P. W., Guy, D. P., and Gilles-Maurice, D. S., (2007). *Automatic diacritic restoration for resource scarce languages*. [e-book]. Springer. [Accessed 9 June 2013].
14. Wagacha, P.W., De Pauw, G. & Getao, K., (2006)a. *Development of a corpus for Gĩkũyũ using machine learning techniques*. In J.C. Roux (Ed.), Proceedings of LREC workshop - Networking the development of language resourcesfor African languages. Genoa, Italy: European Language Resources Association, pp. 27–30.
15. Wagacha, P.W., De Pauw, G. & Githinji, P.W., (2006)b. *A grapheme-based approach for accent restoration in Gĩkũyũ*. In Proceedings of the Fifth International Conference on Language Resources and Evaluation. Genoa, Italy: European Language Resources Association, pp. 1937–1940.
16. Wamberia, K., (1993). *Kitharaka Segmental Morphophonology with Special Reference to the Noun and the Verb*. Unpublished PhD Dissertation, University of Nairobi.