St. Mary's University ቅድስት ማርያም ዩኒቨርስቲ

Committed to Excellence

ST. MARY'S UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE

DESIGNING A STEMMING ALGORITHM FOR KAMBAATA TEXT: A RULE BASED APPROACH

BY

JONATHAN SAMUEL SUMAMO

MARCH, 2018

ADDIS ABABA, ETHIOPIA

DESIGNING A STEMMING ALGORITHM FOR KAMBAATA TEXT: A RULE BASED APPROACH

BY

JONATHAN SAMUEL SUMAMO

A Thesis Submitted to School of Graduate Studies of
St. Mary's University in partial fulfillment of the Requirements
for the Degree of Master of Science in Computer Science

MARCH, 2018
ADDIS ABABA, ETHIOPIA

ST. MARY'S UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF INFORMATICS
DEPARTMENT OF COMPUTER SCIENCE

DESIGNING A STEMMING ALGORITHM FOR KAMBAATA TEXT: A RULE BASED APPROACH

BY

JONATHAN SAMUEL SUMAMO

APPROVED BY BOARD OF EXAMINERS

_____	_____	_____
DEAN, GRADUATE STUDIES	SIGNATURE	DATE
SOLOMON TEFERRA (PhD)	_____	_____
ADVISOR	SIGNATURE	DATE
MILLION MESHESHA (PhD)__	_____	_____
EXTERNAL EXAMINER	SIGNATURE	DATE
MICHAEL MELESE (PhD CANDIDATE)	_____	_____
INTERNAL EXAMINER	SIGNATURE	DATE

DEDICATION

This work is dedicated to my family.

ACKNOWLEDGMENT

All praise, honor and glory to my Lord Jesus Christ for His richest grace, mercy and giving me the strength for the accomplishment of this thesis.

First and foremost, I wish to place sincere thanks to my advisor Dr. Solomon Teferra, School of Information Sciences, Addis Ababa University, for his interest and enthusiasm about this thesis work and for his priceless guidance and support throughout my research work. Thank you Dr. Solomon for your sincerity and motivation which deeply inspired me to complete this thesis!

I am also very grateful and would like to extend my sincere thanks to Asrat Mulatu, PhD Candidate at IT PhD program, Addis Ababa University, for his consistent support and useful comments during this study.

I would also like to express my heartfelt thanks to Dr. Treis Yvonne, a Deputy Director of Languages and Cultures of Sub-Saharan Africa (LLACAN) and researcher at CNRS located at Paris, France for sharing the linguistic materials and her journal articles on Kambaata language and her comments during email conversation.

I would also like to express my sincere gratitude to Kambaata Tembaaro Zone Education Department, especially Ato Alemu Banta, the author of Kambaatissa-Amharic-English dictionary, for his cooperation in providing all the necessary resources for text corpus including school text books and his own dictionary.

I am also thankful to all my classmates and friends for the moral support, especially Eden Getachew and Yonas Fiseha for their constructive comments and many other individuals that I couldn't name here who have directly or indirectly helped me in completion of my thesis.

Last but never be the least, I would like to thank my beloved families, for their love, encouragement, prayer and support towards my study.

Jonathan Samuel Sumamo

March, 2018

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDIXES

LIST OF ACRONYMS AND ABBREVIATIONS

Abbreviations/Acronyms	Definitions
1PL	1st Person Plural
1SG	1st Person Singular
2PL	2nd Person Plural
2PL/HON	2nd Person Plural/Honorary
2SG	2nd Person Singular
3F/PL	3rd Person Feminine/3rd Person Plural
3HON	3rd Person Honorary
3M	3rd Person Masculine
3mO	3rd Person Masculine Object
ABL	Ablative
ACC	Accusative
AGR	Agreement
ASSOC	Associative
CV	Consonant - Vowel sequence
CC	Consonant - Consonant sequence
NCSR	National Centre for Scientific Research
CVB	Converb
DAT	Dative
DS	Different Subject
EX	Existential
GEN	Genitive
HEC	Highland East Cushitic
ICO	Imperfective Converb
ICP	instrumental-comitative-perlative
IMP	Imperative
INACT	Past tense, counterfactual
INALCO	National Institute of Oriental Languages and Civilizations
IPA	International Phonetic Alphabet
IPV	Imperfective
JUS	Jussive
KT	Kambaata and Tambaaro
LOC	Locative
NEG	Negation
NIPV	Non-imperfective
NOM	Nominative
NP	Noun Phrase
OBJ	Object marker
OBL	Oblique case
OI	Over stemming index

P/G	Palatalization and Gemination
PL	Plural
PRED	Predicate
PROG	Progressive
PVE	e-perfective
PVO	o-perfective
REL	Relative
RVs	Relative Verbs
SNNPR	Southern Nations, Nationalities, and Peoples Region
SOV	Subject Object Verb
SS	Same Subject
ST	Standard
SW	Stemming weight
UI	Under stemming index

ABSTRACT

Stemming is the process of reducing inflectional and derivational variants of a word to its stem. It has substantial importance in several natural language processing applications. In this research, a rule based stemming algorithm that conflates Kambaata word variants has been designed for the first time. The algorithm is a single pass, context-sensitive, and longest-matching designed by adapting rule-based stemming approach. Several studies agree that Kambaata is a strictly suffixing language with a rich morphology and word formations mostly relying on suffixation; even though its word formation involves infixation, compounding and reduplication as well.

The output artefact of this study is a context-sensitive, longest-match stemming algorithm for Kambaata words. To evaluate the stemmer's effectiveness, error counting method was applied. Two different test sets of 1385 and 1040 distinct words were used to evaluate the stemmer. The combined output from the first stemmer indicates that out of 2425 words, 2271 words (93.65%) stemmed correctly, 138 words (5.69%) over stemmed and 16 words (0.66%) under stemmed.

To minimize the problems identified in the first version of Kambaata stemmer, certain improvement was undertaken by identifying additional affixes and rules. Accordingly, the errors of over stemming and under stemming were reduced to 2.60% (63 words) and 0.54% (13 words), respectively. Consequently, the overall performance of the stemmer has been enhanced to 96.87%. What is more, a dictionary reduction of 67.52% has also been achieved for correctly stemmed words on the evaluation.

The main factor for errors in stemming Kambaata words is the language's rich and complex morphology. Hence a number of errors can be corrected by exploring more rules. However, it is difficult to avoid the errors completely due to complex morphology that makes use of concatenated suffixes, irregularities through infixation, compounding, blending, and reduplication of affixes.

Keywords: stemming algorithm; Kambaata stemmer; rule-based stemmer; longest-match stemmer; Kambaata language

CHAPTER ONE
INTRODUCTION

1.1. BACKGROUND OF THE STUDY

Variants of a basic word commonly exist in natural language texts [6]. Morphological variations are usually the most typical, along with other sources such as alternate spellings, miss-spellings, and variations coming from transliteration and abbreviation [6]. Stemming solves the challenges that arise via varying morphological forms by effectively reducing semantically related words to a common stem [1], [6].

As stated in [1], stemming algorithms are automated rules to reduce all terms with the identical root to a common form, normally by eliminating the words' morphological affixes. The researcher also discusses that stemming researchers are most desirable today in many fields of computational linguistics and IR, but for numerous motives. In morphological analysis, the stem of a term could possibly be of much less quick desire than its affixes, which is often used as hints to grammatical structure [1], [3].

The reason behind research works on stemming algorithms is the need to enhance information retrieval accuracy [1], and nowadays, stemmers are widely applied in different fields of NLP such as IR, text classification, text summarization and automatic machine translation [1].

According to Sharma [2], in the manual approach, a word in a document is queried by searching one of its variant at a time. The same researcher discusses that this technique is very tiresome and misses the related information of same importance [2]. Hence, that is why stemming is broadly applied in several information retrieval systems to avoid such kinds of difficulties and to enhance retrieval performance [3].

Stemming is applied as preprocessing stage in the development of automated text summarization systems. Stemming algorithm is also used in machine translation to get stemmed words or sentences [14].

Designing stemming algorithm for Kambaata language has a benefit of developing other natural language processing applications such as, text classification, text categorization and morphological analyzer [16].

An example of a stem can be the word "*mar*" (go - 2M) which is the stem for the variants "*marro*" (goes), "*marree(u)*" (went), "*marimba'a*" (didn't go), "*marano*" (will go), "*marayyoo(u)*" (is going), and "*marota*" (to go).

Morphology is the study of structure of words and defines word formations in a language. The most common ways of word variant formation in natural language text are suffixing and prefixing [5]. Inflectional and derivational morphologies are the two types of morphology [6]. Inflectional morphology is a creation of different forms of the same word without changing its part of speech. Usually, the variations are results of changes in person, number, tense and gender. As stated in [7], such variations have not effect on a word's class; that means, a verb still remains verb after its tense form is changed. For example, "*agud*" (look), "*agujjo*" (looks), "*agudayyoo(u)*" (looking), "*agujjee(u)*" (looked).

In another way, derivational morphology results in change of the word's class [7]. For instance, affix changes a word from adjective to nouns, from verb to nouns, from noun to verbs, and so on; like "*jaalu*" (friend), "*jaalloomaan*" (friendly), "*jaalloomat*" (friendliness) and "*jaalloomata*" (friendship).

Based upon the rich morphological property of individual languages, several variations of terms could possibly be resulted out of single stem [2]. This huge variant existence has powerful impact on information retrieval programs. As a result, right now there is a demand for automated procedure that can minimize the size of various terms to controllable level, and also record the strong connection that present amongst diverse word types [8]. Even if the several languages have various degree of morphological complexity, stemming is generally employed in information retrieval, with the fundamental reason that morphological variations represent equivalent meaning [16].

Morphological processing is a commonly used application for powerful and successful information retrieval, machine translation and word summarization [8], [9]. Consequently, it becomes extremely crucial for IR as it needs figuring out the proper word variations as index [10]. According to Salton [11], automatic IR system is a computer software element that helps request and access of information from databases by diverse end users.

According to Baeza-Yates [5], based upon on their particular stemming strategy, stemmers are grouped in to four. These are: affix removal, table lookup, successor variety, and n-gram stemmers.

Rule-based also known as Affix removal technique is a strategy that is implemented easily and efficiently [63]. In this strategy, affixes are eliminated from the terms resulting in stems. This technique was applied by [1], [12], [15], [16] [17], and [18]. Table lookup also called Dictionary-based strategy look ups the stem of a word in a table of dictionary. Table lookup method was employed by [14] to stem Amharic words. This approach is straightforward and relies on the dimension of stem dictionary. The strategy also requires significant storage space. Successor variety technique is centered on the identification of morpheme boundaries of terms and makes use of expertise from structural linguistics. This approach is much more complicated compared to that of affix removal technique. N-gram method is primarily based on the recognition of n-grams for instance bi-grams and tri-grams. This method was utilized by [13] to stem language independent words making use of uni-gram.

As opposed to morphologically simple languages for instance English, Cushitic languages for example Sidaama and Kambaata have very complex morphology [19]. According Treis [19] and [20], Kambaata does not make use of prefixes for word formation. Nevertheless, complicated terms can be created by suffixation, infixation, compounding and reduplication, specifically by full reduplication or by reduplication of portion of the word. The reduplicated section of the syllable is prefixed in Kambaata [19].

Kambaata is known as "*Kambaati afoo*" literally means 'the mouth of Kambaata' in Kambaata language. It belongs to the Highland East Cushitic branch that encompass languages spoken in south-central Ethiopia, such as Hadiyya, Libido, Kambaata, Alaaba, Qabeena, Sidaama, Gedeo, and Burji [19]. The language is spoken and institutionalized in Kambaata and Tambaaro (KT) Zone, which is located at northeastern part of Southern Nations, Nationalities, and Peoples Region (SNNPR) of Ethiopia and situated (the Zone) 250 km south west of Addis Ababa, Ethiopia's capital. The language is also spoken by Kambaata migrants in other parts of the country and abroad. For instance, there is significant population of Kambaata speaking migrants in South Africa.

The Kambaata people's name and the language that they communicate is available in numerous spellings in the literary works in addition to Kambaata; the most frequent ones include *Kambata, Kambatta, Kembata, Kembatta, Cambata, Cambatta, Kambara, Kemata and Donga*. The people of Kambaata call their language by the name *Kambaatissata* or *Kambaatissa*. It is also called *Kambaatigna/ Kambaatinya* (in Amharic) or ከምባትኛ (Ge'ez

script - Amharic), and sometimes Kambatic (in English, just like the 'ic' ending of "Amharic or Arabic") [19], [22].

Kambaata is a Highland East Cushitic language, part of the Cushitic and the much bigger Afro -Asiatic group and spoken by the people of Kambaata. Kambaata dialects (with lexical similarity between dialects) are: Tambaaro (95%), Alaaba (81%), Kabeena/ Qabeena (81%). Kambaata also has higher lexical similarity with other HEC groups, i.e. Sidaamo (62%), Libido (57%), Hadiyya (56%), and Gedeo (54%) [22].

Kambaata is as well the name of a smaller Highland East Cushitic division, the Kambaata group, which comprises of Kambaata (itself-being the main) and Tambaaro and also Alaaba and Qabeena which usually known as its dialects [19], [23].

1.2. STATEMENT OF THE PROBLEM

Kambaata is one of the Zonal languages in the SNNPR of Ethiopia. At present, it is estimated to be spoken by more than one million people [22], [23]. Currently, the language serves as a medium of instruction in the primary schools, and is also provided as a subject in the junior and secondary high schools and preparatory schools of KT Zone. The first Kambaata-Amharic dictionary was published by today's KT Zone Culture and Tourism Department (1995 E.C./2003) and the second dictionary, 'Kambaatissa-Amharic-English' dictionary was published by Alamu Banta (2009 E.C/2016). Kambaata Old Testament Bible translation in the official Latin orthography is 71% completed as of today according the data from Bible Society of Ethiopia [68]. Booklets having bible stories like "*Haaroo Woqqaa*" 'New Way', written in both Ge'ez and Latin script, has as well been published by the Bible Society of Ethiopia [19]. Kambaata language Proverbs, Tales and Legends are few of the works which has been accomplished partially until now [33].

Plenty of translation works have already started in translating materials from other languages to Kambaata. The language is being studied at numerous levels both locally and by researchers from abroad. As an example, now there are different research works performed for this language by research unit of Languages and Cultures of Sub-Saharan Africa (LLACAN) which is affiliated to the National Centre for Scientific Research (NCSR) and the National Institute of Oriental Languages and Civilizations (INALCO) [24]. Such possibilities open up opportunities to produce a lot more written materials in the language.

The advancement of technology and digital media in Ethiopia is expanding progressively and more quickly. Accessibility and usage of the Internet and search engines are getting part of everyday activity not only in Ethiopia but also in the rest of the Africa and the World. Textbooks, reference books, publications, articles and various other documents can be accessed digitally on all pervasive devices and support flexible access. Networking of educational institutions (i. e. universities, colleges, high schools) and corporations as well as businesses is in progress and a number of research projects with national and international institutions on study of Kambaata language have been started recently.

Together with having access to the Internet, there is proof of a swiftly growing number of Kambaata educational, cultural, religious, journal articles and other kinds of documents in electronic media. Nowadays, one of the hot issues in the field is the mechanism for storing and accessing this pervasive information in an effective and efficient way. Therefore, document summarization, classification and information retrieval are fields that attempt to deal with these kinds of problems [2].

Kambaata language has rich morphology [19], [54]. It makes use of the two types of morphologies, i.e. inflectional and derivational for word formation. For instance, more than two hundred variants can be formed from a single stem by inflection and derivation (see Appendix VII) [55]. Example is given for a progressive form of a verb "*kul*" (tell) inflection. For full list, see - Appendix VI and Appendix VII.

TABLE 1-1: MORPHOLOGICAL EXAMPLE - PROGRESSIVE FORM

Person	**e.g. "*kul*" 'tell'**
1SG	*kul-ayyoom(m)*
2SG	*kul-tayyoont*
3M	*kul-ayyoo(u)*
3F/ 3PL	*kul-tayyoo(u)*
3HON	*kul- eenayyoomma*
1PL	*kun- nayyoom(m)*
2PL/2HON	*kul- teenayyoonta*

However, according to the researcher's knowledge, Kambaata language has very few linguistic resources and absolutely no computational works have been carried out to computerize / automate the language in relation to NLP applications.

As reported by [7], the morphological complexity of a language could end up in extremely significant amounts of variations for a word. Subsequently, word variations may induce a substantial influence on the efficiency of IR systems as well as on morphological analysis tools. Since Kambaata is morphologically complex language [19], there is a need for automated programs that can easily stem words and decrease the size of a words for required storage space minimization, text summarizers and retrieval applications, and also determine the strong associations existing among various word varieties in the language [17].

Stemming also has a crucial role indetermination of stem from a word by the removal of both inflectional and derivational affixes, and therefore, there have been much desire for stemmers with this goal [5], [7]. This demand is growing further and most likely to boost in the future as a lot more text-processing applications turn out to be of crucial importance [10].

According to the researcher's observation during visits to the zone, at this time, there is already a need for stemming algorithm and other applications for Kambaata language text. This requires designing an automated procedure that removes inflectional and derivational affixes of Kambaata words; and requires exploring how semantically related terms in the language can be conflated together with one another automatically using rule based approach which is dependent exclusively on the morphology of the particular language. Exploring word conflation technique and designing the stemming algorithm is totally dependent on the morphological property of that specific language due to the fact that every language has distinct morphological structure [1], [2], [6]. Thus, it requires finding patterns for stemming and defining and developing a program/stemmer based on the language's morphology and word formations [1], [15], [16], [17], [18].

The formerly pointed out factors helped to determine the need to design an algorithm that conflates Kambaata texts effectively for the users of the language.

Stemming algorithm researches have been conducted to several languages both internationally and locally. Locally, stemmers have been attempted for Amharic [7], [14], Afaan Oromo [15], [29], Tigrigna [12], [16], Wolaytta [17], Silt'e [18] and few others. To the best of the researchers' knowledge, there is no any research carried out to explore stemming methods for Kambaata words and there has never been any attempt done to design a rule based stemmer for the Kambaata language text. Thus, the researcher used this

opportunity to do a research on exploring stemming rules and designing an appropriate algorithm for stemming Kambaata words.

Therefore, the purpose of this particular study is to explore methods and rules for stemming Kambaata words and design a rule based stemmer in order to provide automatic word conflation method for documents.

Hence, the research attempts to respond to the following research questions in the study course of action: -

- What are the morphological properties and how are words formed in Kambaata?
- What are the challenges in designing rule based stemmer for Kambaata?
- What optimal stemming performance may possibly be achieved on a given test corpus?

1.3. OBJECTIVE OF THE STUDY

The research has the following general and specific objectives.

1.3.1. GENERAL OBJECTIVE

The main objective of this research is to design a stemming algorithm for Kambaata text using Rule based approach.

1.3.2. SPECIFIC OBJECTIVES

To achieve the above general objective, the following specific tasks and/or objectives were performed in the research work.

- To carry out a literature analysis on stemming algorithm researches.
- To review morphological behavior of the language to become acquainted with word formations.
- To prepare corpus that is needed to identify affixes and stems in the language.
- To construct affixes list used in Kambaata from the corpus and different literatures.
- To explore/ adapt techniques and define rules for stemming Kambata words.
- To design a rule based stemmer for Kambaata words that conflates inflectional and derivational affixes of the language.
- To experiment the stemmer on selected test set and measure the performance of the stemmer designed in the study.

1.4. METHODOLOGY

1.4.1. GENERAL APPROACH

The general research approach adopted for this study is a Design Science Research methodology which is employed for the design of the algorithm. As stated in [70], the design science research requires the creation of an innovative, purposeful artifact for a specified problem domain. This research process involves problem identification, solution suggestion, development, evaluation and conclusion [70].

Problem identification: The research problem in this study has been identified by reading NLP research problems in Ethiopia, more specifically Stemming research gaps. Consequently, reading the research gap in the field provided the researcher an opportunity to get aware of the limitation of stemming research and helped to easily identify which languages have not been studied in this regard.

Suggestion: After problem identification, a research proposal has been prepared with a need to apply an existing knowledge of stemming to new area of Kambaata language as a new research effort.

Development: As part of this process, a rule based stemming technique was selected and the appropriate algorithm was designed for the Kambaata language based on the detail study of its morphology. Finally, an artifact of the study (the stemming algorithm) is developed and implemented using python programming language through context sensitive and longest match approaches.

Evaluation: After the design of the algorithm, the stemmer was evaluated using error counting and dictionary reduction methods. The evaluation results were measured in terms of correctly stemmed, over stemmed and under stemmed words.

Conclusion: At the end of the research process, conclusions have been derived from the main research findings. The challenges during designing stemmer for the Kambaata language are also discussed and the summarized behavior of the artifact is also discussed in this phase.

1.4.2. REVIEW OF LITERATURE

To understand stemming algorithms and development or adapting strategy, several research works for stemming algorithms on various languages such as English, French, Greek, Amharic, Afaan-Oromo, Tigrigna, Wolaytta, Silt'e have been reviewed. To understand the morphological properties of Kambaata, review of researches on the language is performed by advising diverse sources such as books, journal articles, dictionary, textbooks. For literature review of the morphology of the Kambaata language, books and journal articles are downloaded from the online journal libraries and further information is also compiled via email with appropriate individuals (linguists) of the language.

1.4.3. DATA COLLECTION AND CORPUS PREPARATION

A corpus is the fundamental data source needed in the development of stemming algorithm [16]. A text data is collected and literature survey is applied for compiling affixes. As stated in [16], a large sized text can show a reasonable language morphological behavior. Selection of much larger sized text is therefore, an essential element in designing a stemmer [18]. Hence for the purpose of this research, the researcher utilized a corpus of 129,929 word tokens that is believed to be a representative of the language because of its size which

is collected from school textbooks from Kambaata educational offices and high school. As stated in limitation section, the researcher could not be able to get a corpus of different domains.

1.4.4. DEVELOPING TOOLS AND TECHNIQUES

To implement the stemmer, Python programming language has been utilized for the reason that it provides extensive support libraries for string operation [18] and in addition the researcher is much more familiar with Python than any other language. Kambaata language has rich morphology [54]; hence the process of stemming Kambaata words involves dealing with mainly suffix stripping and infixation and other irregular words at lesser degree. The algorithm is designed by examining morphological rules of the language.

For the development of the Kambaata stemmer, Affix removal (often called Rule-based) with longest match technique is employed since it's a broadly used stemming approach. Rule based stemming is easier and can be implemented efficiently than other techniques if we know the rules for affix removal in the language [63]. Most of the stemmers that have been developed until now are based on this approach [2]. Sharma [2] has mentioned that Rule based technique has the following advantage over Statistical approaches:

1. Stemming programs constructed using Rule based technique are faster as compared to Statistical stemmers. Thus, stems can be obtained within short computing time using rule based approach.

2. The performance of stemmed words by Rule based stemmers are quite higher.

1.4.5. TESTING THE ALGORITHM

Error counting technique is commonly applied method for evaluating stemmer performance. This technique is used to examine the effectiveness of the stemmer. Identification of correctly stemmed, over-stemmed, under-stemmed words and dictionary size reduction is conducted to observe the result of the stemmer. The result is represented in numbers and percentage. The percentage is used to demonstrate the accuracy of the stemmer.

1.5. SCOPE, DELIMITATION AND LIMITATION OF THE STUDY

In this study, the very first stemmer for Kambaata words that removes the affixes for different NLP applications have been designed. Kambaata is a strictly suffixing language; consequently, it has no known prefixes till now [19], [20]. This study is the first attempt to explore stemming technique and design the algorithm for Kambaata language words and the stemmer is the first of its kind. In this research, rules to stem words have been explored and an algorithm to handle suffixes, infixes and some irregular words in Kambaata has been designed. The stemmer is not only suffix removal but it also has context sensitive and recoding rules to transform words that are not handled by suffix removal rules. Reduplicated and compound words in the language are very few and are not in the scope of this study because of the complexity of the morphological behavior of the language and other resource limitations. The limitation of the research is that the corpus used for this study is not of different domain. The corpus is mainly of educational domain and the researcher is not able to find other domain corpus due to unavailability of the resources to the researcher.

1.6. SIGNIFICANCE OF THE STUDY

Designing stemming algorithm for Kambaata words helps the language's speakers to discover information they desire quickly without having any kind of problems while querying words. The artifact of this study could also be a foundation to explore and develop various other NLP applications such as, IR systems, text summarizers, machine translation, text categorization applications and morphological analyzers for Kambaata. Kambaata word processing tools could also need stemming algorithm that functions together with spell checker software to enhance the efficiency of spelling checking [69].

Kambaata word stemmer could also give an advantage of reducing the size of documents [14]. Because an individual stem usually corresponds to several complete terms, by storing stems rather than words, a data compression rate of more than 50 percent could possibly be attained [2].

In Kambaata, a word has got quite large variants and conflating all these variants increases performance of the retrieval [64]. It also decreases storage space needed for index documents [2]. Moreover, exploring stemming techniques for the language's words could also provide the following advantages: -

- The study helps to design tools such as term frequency counter.
- It can be used to decrease word variations and to reduce total number of documents.

1.7. ORGANIZATION OF THE THESIS

The thesis document is structured in five chapters. The first chapter is the introduction that features background, statement of the problem and its justification, the objective, methodology, scope and limitation and the importance/application of the study.

The second chapter explains principles and strategies of stemming algorithms. Comprehensive discussions are made on methods to stemming. Review is also made on stemmers developed for local and foreign languages as part of related works.

Kambaata language morphology is introduced and reviewed in the third chapter. In depth outline of Kambaata morphology is provided in this chapter.

Chapter four is the key portion of the thesis work. It presents the exploration and design of the Kambaata stemming algorithm with short introduction followed by corpus preparation for the algorithm. The discussion proceeds with the collection of Kambaata affixes succeeded by the implementation of the stemmer. Finally, the evaluation of the stemming algorithm is discussed in depth.

The last chapter presents conclusions comprehended from the findings and recommendations for future study.

CHAPTER TWO
LITERATURE REVIEW AND RELATED WORKS

2.1. OVERVIEW

This chapter discusses the review of concepts of stemming algorithms and conflation techniques. The classification of stemming algorithms based on the stemming approach has also been discussed in detail. Review of related works for several international and local languages is also discussed in this chapter in detail.

Stemming is an automated procedure helpful to cut down morphological variations of terms to their citation or dictionary form [14]. As stated by Lovins [1], stemming algorithm "is a computational procedure, which reduces all words with the same root to a common form by striping each word of its derivational and inflectional affixes".

2.2. CONFLATION TECHNIQUES

Word conflation is a method of matching morphological variations of terms that are associated in meaning [3]. Conflation or removal of affix from words to convert to stems is required in document indexing processes to specify appropriate content identifiers and also in IR to determine appropriate query words that complement indexing terms in a document [71]. Hence, word conflation might be conducted at indexing as well as at search time [31].

Conflation is carried out in two tactics; either manually or automatically by means of software programs referred to as stemmers [15], [16], [17], [18], [43]. According to Srinivasan [71], Right hand truncation is among the frequently utilized methods for manual conflation. Many traditional internet-based systems, for example ERIC (Education Resources Information Center) which is an online digital library system, permit the user to truncate query words by using wildcard characters i. e. asterisk (*) [43]. For example, more records on the topic COMPUTATION will be retrieved if the initial lookup term is truncated to COMPUT*. However, users usually are not familiar with the truncation technique. Willett [31] mentioned that two main difficulties are linked with the manual right hand truncation:

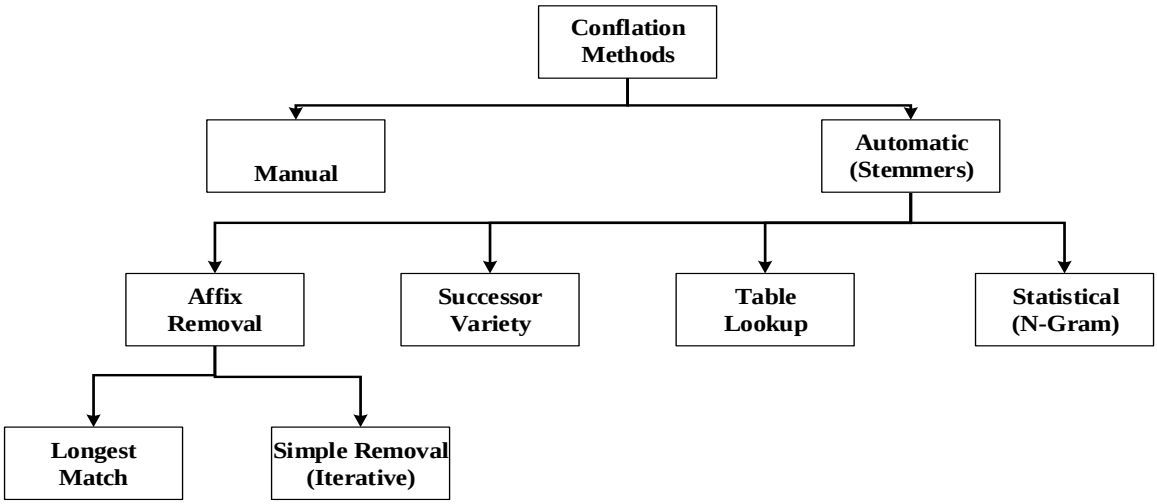
- Over truncation meaning the leftover stem of a word is very short right after truncation;

- Under truncation which usually leads to retrieval of very few similar appropriate terms.

For instance, during over truncating the word EDUCATION to EDUC, all words related to EDUCATE and EDUCATION and also totally unrelated words are retrieved. In the case of a word being under truncated, very few of any relevant word is retrieved e. g. if the word COMPUTERS is truncated to COMPUTER, then all relevant documents related to COMPUTING and COMPUTATIONAL are not be retrieved. Walker and Jones [34] also discovered that manual truncation is seldom used by users as it requires certain experience and skills. Therefore, the use of manual word conflation in information retrieval systems and OPACs demands qualified intermediaries who can assist users to conquer the aforementioned challenges [43].

Automated word conflation is performed through computer programs known as stemmers, that remove morphological variants of a word to form stems [1]. The next diagram demonstrates the variety of types of conflation strategies and stemmers.

FIGURE 1-1: WORD CONFLATION METHODS [2]



2.3. STEMMING ALGORITHMS

User search words needs to be matched up to the terms in databases or documents for effective retrieval of the required information [11]. Query words and those in the files must fulfill the criteria for retrieval; or ranked, where the documents with the better relevance to the request are retrieved as outputs. There needs to be a way for deciding whether or not a provided query term matches to a given word in a document. The most straightforward is to permit exact matching only; for example, “business” would match itself only and a

document that contained “business” but “Business” would not be recognized as a match. Another simple correspondence method is case-insensitive exact match [18].

Stemming is also a correlation process, whereby search terms and words stored in a document are reduced to a common root word by removing affixes. Words relate if they have a common root. For example, “READS” might be stemmed to “READ”, which would match “READ”, “READING”, and of course “READS”. Each word thus has a set of conflations, that is, words that have the same stem [18].

The features of stemming algorithms differ considerably depending on whether stem dictionary and suffix lists are being utilized, and also on the aim for which the stemmer is developed [1], [17]. However, most of the stemmers are based on specific rules and techniques [35]. These techniques include a removal of the single longest matching suffix or the iterative elimination of numerous suffixes. The reason behind iterative strategy is the fact that suffixes are affixed to stems one after the other. In this approach, suffixes are removed from the stem in the order of their derivational rules. The suffix removal begins from the end of the word and performing in the direction of the starting. For example, if we look at a word COLOURFULNESS, the suffix -NESS is taken off in the very first iteration and re-considering COLOURFUL, the suffix -FUL is removed leaving behind COLOUR as a last stem. The affix removal is accomplished according to the class order that is defined by the developer. Different techniques for instance the one recommended by [36] could also present, which are usually iterative although have no their endings classified. The application of an iterative method has also its own challenge [29]; the need of analyzing huge quantity of suffixes along with their own sequential relationship, which is a tough activity and construction of list of sequential class.

Porter used iterative stripping approach, which operated in five stages, using five different classes of 60 suffixes to recreate the inflectional and derivational practice of terms. Normally, a number of the rules tend not to remove the suffix however they recode endings of the stem to new endings. For example, if we eliminate the suffix “-ING” from “DEPORTING” and “-ION” from “DEPORTATION”, it provides us with “DEPORT” and “DEPORTAT” that happen to be distinct stems. However, making use of the recoding rule that converts endings from “-RTAT” to “-RT,” both words are conflated to exact same stem “DEPORT” [6].

The longest match strategy removes the lengthiest suffix feasible [7], [17]. For example, if we take the same word COLOURFULNESS, the suffixes attached to the word are: -NESS, -FUL, and FULNESS. Consequently, the algorithm deletes -FULNESS in the term. The challenge of employing longest match method in comparison to iterative method consist of the need to have most feasible combinations of affixes, processing space and memory and storage space required, and the modification of affix while concatenating [18].

There is also other kind of stemming algorithms classification, context-free or context-sensitive stemmers [7], [16], [17], [18]. Context is related to any sort of conduct which can be linked to the leftover stem and usage of the suffix [37]. Not any restriction is applied on the stem and therefore no extra procedures are needed to examine restrictions in context-free algorithms. However, the majority of stemming researches [1], [6], [30] suggested that much better output could be attained by incorporating restrictions to stripping procedures, which can be, applying context-sensitive conditions, that are generally language dependent [17]. Context sensitive rules identify certain circumstances by which each suffix could be removed from an input word [7]. Savoy [30] discusses three common kinds of restrictions: quantitative, qualitative and recoding rule.

"In quantitative constraints, minimum length for the remaining stem is set when a suffix is removed. This helps not to remove ending from a stem in which the ending is in the suffix list but actually not a suffix for that stem. For example, for the word ABILITY and suffix ABILITY, as the remaining stem must not be zero, the suffix -ABILITY will not be stripped from the word." Qualitative constraints determine situations to be fulfilled by ending of the remaining stem. For instance, remove the suffix "-ize" if the remaining stem does not end with "e". In this instance, removing "-ize" from "seize" is prohibited [30].

Correcting and modifying spelling rules need to be applied to conflate the terms to precise stem [3]. To this end, a recoding rule that transforms MxN →MyN, where M and N define the context transformation, while X being input string, and Y being the transformation string must be used [3], [18]. Stemming algorithms usually range from poor stemmers that remove just plural markers to sophisticated ones that deletes suffixes, infixes and prefixes. Stemming algorithms nowadays are inspired by the techniques used by popular English stemmers developed by Lovins and Porter. They are considered as pioneers and foundations for the majority of suffix removal stemming algorithms. Lovins algorithm was designed in longest match approach, and it works by using a list of 260 endings arranged in decreasing

order of suffix length [1]. As an illustration, a term PRIVATIZATION can be stemmed to PRIVAT if the suffix -IZATION was incorporated in the suffix dictionary. Once a suffix is taken off, the remaining stem is compared to one of the 34 recoding rules to take into account spelling exceptions. The recoding rules determine, for example, the treatment of a suffix preceded by double consonants such as TRAVELLING to TRAVEL, or minimal stem length has to be maintained, for example, the removal of ING from DOING but not from BRING etc. The challenge in longest match strategy is having a very significant affix list: both simple suffixes and the concatenated ones. This demands storage spaces and collection and processing the affix list [3], [15], [18].

Several alternative solutions are already suggested to deal with issues of stemming. The problem of stop words is likewise one more problem that must be discussed with regards to retrieval effectiveness [17]. Stop words removal in retrieval process, offers increase in performance by minimizing storage need and increasing the matching speed of a query with index terms of a document [30]. Thus, stop word compilation is crucial in developing algorithms for stemming words in a language [16], [17]. These words might be collected either by sorting a words of a corpus and by selecting based upon word occurrence [6] or collecting by applying the linguistic knowledge of the language which is much more successful one [1], [2], [3], [15], [16], [17]. Salton [4] encourages choice according to term frequency by saying that the frequency characteristics of terms in the documents of a selection have been used as signals of term importance for content analysis and indexing purpose.

Stemming algorithms are categorized as Affix Removal (Rule-based), Dictionary-based (Table Lookup), Successor Variety, and Statistical (N-gram) based on the development strategy applied [3].

2.3.1. AFFIX REMOVAL ALGORITHMS

Rule based strategy is solely reliant on the morphological behavior of the particular language [2], [7]. In this method, the algorithm deletes affixes from terms resulting in the stem. Occasionally, the output stem is recoded. Most popular stemmers of this kind presently being used are iterative and longest match stemmers. The longest match stemmer was initially developed by [1] and followed by [26]. An iterative stemmers reported by [6] and [27], remove the basic suffixes iteratively and continue repeatedly until no more affixes

can be removed. Despite the removal of all affixes, stems could probably be not effectively conflated.

Rule based Iterative method is a broadly employed stemming approach and this approach was first presented by [6], [16], [18]. The Porter English stemming algorithm removes suffixes from a given word iteratively, resulting in its stem. Despite the fact that the iterative technique possesses its own disadvantages, it is the widely accepted for its higher accuracy and storage consumption [18].

2.3.2. DICTIONARY-BASED TECHNIQUE

As stated in [38], developing an extremely large dictionary for storing all words in documents along with their respective inflectional and derivational variations is a foundation for a Table lookup strategy. The dictionary in this technique is expected to include stems, roots, and affixation. There shall be only one unique entry per each word in a look up table. As an example, PUMPING, PUMPED and PUMPER are words to be stemmed to PUMP. The stemming process is conducted manually, where the stems are identified for each term and kept in organized form for instance table [16], [18].

Table lookup approach arranges all words in a sorted alphabetical list with their respective variant forms [2], [18]. During stemming a word, the table is queried to discover an equivalent variation of a term. If a related variant of the query word is found, the connection root is provided as output. Query words and indexes might then be stemmed by way of table look up [16]. A hash table or binary lookup list can be utilized to improve the search. Despite the fact that this technique results in precise outputs, it has problems such as dictionary maintenance; to maintain with a constantly changing language, and this is really quite a challenge [17], [18]. In addition to the construction of the dictionary, this also needs frequent upgrading and also storage space is yet another issue and as well required processing duration for retrieval of data is usually a concern in this technique [18].

The primary advantage of this approach is the generation of correct stems [16]. The limitation of this method is, it retrieves only words defined and stored in the lookup table. Storage space for the dictionary is likely to increase as the corpus increases, which make the retrieval process inefficient and slower [18].

2.3.3. SUCCESSOR VARIETY

According to Hafer [25], the successor variety of a string is the number of various characters that follow it in words in some body of text. The successor variety of substrings of a word reduces as more characters are included till a segment boundary is attained. Stemmers of this kind are dependent on work in structural linguistics that attempts to determine word and morpheme boundaries based upon the distribution of phonemes in a large text corpus [17]. Stemming based on this strategy makes use of letters instead of phonemes, and a body of text in place of phonemically transcribed words. It reads the word to be stemmed and finds out the cut point where the successor variety raises sharply [15], [16]. Many variations are feasible, which includes: cut-off, peak and plateau, complete word and entropy. The successor variety algorithm has the benefit of avoiding the need of affix removal rules that are based on the morphological property of the language [2].

2.3.4. STATISTICAL APPROACH

This technique is a string similarity method to word conflation. It uses similarity measures depending on the number of n-grams in common rather than words, then applies clustering techniques. N-gram technique involves statistical methods whereby, through a process of inference and based on a corpus, rules are formulated regarding word formation. Some of the methodologies adopted are N-gram [39] and Hidden Markov Models (HMM) [40].

Developing stemmer based on HMM is also realized using N-gram technique [41]. This strategy doesn't require a prior linguistic knowledge or a manually developed training data. Rather, it utilizes unsupervised training that can be executed at indexing time. Jinxi and W. Bruce [42] recommended a method which allows correcting "rude" stemming results based on the statistical properties of a corpus utilized. The fundamental concept is to produce equivalence classes for words with a classical stemmer and then "separate back" some conflated words based on their co-occurrence in the corpora [16]. N-gram technique does not need any kind of linguistic knowledge at all, being completely independent of the morphological behavior of the language [2], [16].

2.4. EVALUATION METHODS FOR STEMMERS

As discussed in [7], [14], [15], [16], [17] and [18], there are different methods to measure the accuracy of stemmers. The manual method, vocabulary reduction and Paice's method are known stemmer performance analysis approaches. In the manual evaluation technique, an individual that makes a decision on the accurate stem for each word executes the

assessment. We have three evaluation parameters: the number of correctly stemmed words; the number of over stemmed words; and the number of under stemmed words. Among the benefits of stemmers is to reduce the size of the dictionary for indexing needs. The dictionary compression is attained by dividing the number of terms in the text by the number of stems produced, eliminating duplications.

We can also evaluate stemmers using Paice's method which is based on error counting [46]. In this technique, measures of under stemming and over stemming decide precisely how great a stemmer is beyond retrieval context. In Paice's approach, three measurements are applied so as to make a qualitative contrast among various stemmers: the over stemming index (OI); the under stemming index (UI); and the stemming weight (SW). This technique needs a word sampling, without any repetitions, divided into conceptual groups where terms are semantically and morphologically associated. The SW is provided by the ratio OI/UI. Paice has compared several English stemming algorithms in isolation from the context of an IR system and he did not use the conventional precision/recall variables, an ideal stemmer should stem all words in a group to the identical stem. If a stemmed group consists of in excess of one distinct stem, the stemmer has produced under stemming errors. In an IR system, this refers to a negative effect on recall. If a stem of a certain group also happens in various other stemmed groups, the stemmer has produced over stemming errors, which decrease precision. A good stemmer should thus generate as few under and over stemming errors as possible.

2.5. RELATED WORKS

2.5.1. STEMMING ALGORITHMS FOR FOREIGN LANGUAGES

Stemming algorithms have also been designed for several languages up to now which includes English by Lovins [1], Frakes [3], Porter [6], Dawson [26], Paice [27] and Lennon [35]; French by Savoy [30], Greek by Kalamboukis [44], Slovene by Popovic [43], Arabic by Khoja [32] and a number of other languages.

2.5.1.1. ENGLISH STEMMING ALGORITHMS

LOVINS STEMMING ALGORITHM

The first popular and effective stemmer was proposed in 1968 by Lovins [1]. This stemmer performs a lookup on a table of 294 endings, 29 conditions and 35 transformation rules. The stemmer is a context-sensitive and works on a longest match first principle. A word is

stemmed if an ending with a satisfying condition is found. A suitable transformation rule is applied next, its aim being to deal with doubled consonants and irregular plurals.

Even if the stemmer was innovative at the time, it has the problematic task of trying to address two areas (IR and Linguistics) and cannot do well at either. The approach does not give good results with linguistics, as it is not complex enough to stem many suffixes due to absence of complete list of suffixes in the rule. There were also problems regarding the reformation of words. The stemming process also uses recoding rules to reform the stems into words to ensure they match stems of other similar meaning words. The major issue with this process is that it is highly unreliable and recurrently fails to form words from the stems, or matches the stems of like meaning words. The stemmer does not satisfy from the IR or Linguistic viewpoint either, as its large rule set and its recoding stage affect its speed of execution. The Lovins stemmer removes a maximum of one suffix from a word, due to its nature as single pass algorithm. It used a list of about 250 different suffixes, and removes the longest suffix attached to the word, ensuring that the stem after the suffix has been removed is always at least 3 characters long.

DAWSON STEMMING ALGORITHM

Dawson stemmer is an extended version of the Lovins stemmer except that it covers a much more comprehensive list of about 1200 suffixes [26]. Similar to Lovins' stemmer, this stemmer is also a single pass stemmer and hence is very fast. The suffixes are stored and arranged in the reverse order indexed by their length and last letter. In fact, they are organized as a set of branched character trees for rapid access. Dawson did not use recoding technique in his algorithm to handle stems and instead used an extension of the partial matching procedure also defined within in Lovins stemmer. The basic principle of Dawson's algorithm is that if two stems match up to a certain number of characters and the remaining characters of each stem belong to the same stem ending class, then two stems are of the same form. The advantage is that it covers more suffixes than Lovins and is fast in execution. The disadvantage is that it's very complex and lacks a standard reusable implementation.

PORTER STEMMING ALGORITHM

The Porter stemmer is one of the most popular stemmers today, which is proposed in 1980 [6]. The stemmer is based on the idea that the suffixes in the English language (about 1200) are mostly made up of a combination of smaller and simpler suffixes. The stemmer has five

steps; and within each step, rules are applied until one of them passes the conditions. If a rule is accepted and meets the condition, the suffix is removed accordingly and the next step is performed. This process continues for all five classes sequentially, the resultant stem being returned by the stemmer after control has been passed from final class, step five. For example, such a condition may be the number of vowel characters, which are followed by a consonant character in the stem (measure), must be greater than one for the rule to be applied.

As stated in the article, the aim for the development of the stemmer was to improve the performance of IR. The Porter algorithm consists of a sets of condition rules. The conditions are divided into 3 classes; these are conditions on the stem, condition of the suffix and conditions on the rules. Porter's algorithm uses a dictionary of about 60 suffixes and has only few context-sensitive and recoding rules, and therefore is economical in storage and computing time and is very easy to comprehend.

PAICE/HUSK STEMMING ALGORITHM

The Paice/Husk stemmer is an iterative algorithm with one table containing about 120 rules indexed by the last letter of a suffix [27]. The stemmer uses a separate rule file, which is first read into an array or list. This file is divided into a series of sections, each section corresponding to a letter of the alphabet. The section for a given letter, say "e", contains the rules for all endings with "e", the sections being ordered alphabetically. An index can thus be built, leading from the last letter of the word to be stemmed to the first rule for that letter. During word processing, the stemmer takes its last letter and uses the index to find the first rule for that letter. If the rule matches, then it is applied to the word; and if not accepted, the rule index is incremented by one and the next rule is applied. However, if the first letter of the next rule does not match with the last letter of the word, this indicates that no ending can be stripped, so the process ends. Once a rule has been found to match, it is not applied at once, but must first be checked to confirm that it would leave an acceptable stem. When a rule is applied to a word, this usually means that the ending of the word is removed or replaced. The advantage is that it's a simple form and each iteration taking care of both deletion and replacement as per the rule is applied. The drawback is that it's a very heavy stemmer and over stemming may occur.

KROVETZ STEMMING ALGORITHM

The Krovetz stemmer was developed by Bob Krovetz, at the University of Massachusetts, in 1993 [45]. It is quite a light stemmer as it makes use of inflectional morphology. The stemmer effectively and accurately removes inflectional suffixes in three steps, the conversion of a plural to its single form (e.g. '-ies', '-es', '-s'), the conversion of past to present tense (e.g. '-ed'), and the removal of '-ing'. The transformation process firstly removes the suffix, and then checks in a dictionary for any recoding, and finally returns the stem to the input word.

2.5.1.2. FRENCH STEMMING ALGORITHM

According to Savoy [30], French is an inflective language and has a number of irregularities in morphology and orthography. Even the application of the weakest English stemmer for the French language will require a comprehensive suffix dictionary of about 3,000 inflectional suffixes. French terms also have differences between linguistic and semantic meanings. According to the paper, the stemming procedure for French texts consists of two stages:

- Morphological analysis of terms;
- Removal of derivational suffixes according to the grammatical categories.

The morphological analysis requires a dictionary file and a declension file. In dictionary file, each term is associated with a certain declension number, gender and grammatical category. For example, the term ROBUSTE (robust) is characterized as adjective, which uses declension number five and the term is masculine in singular form. Declension number five can be found in the declension file which states that ending -s will be removed if the term is in masculine or feminine and in plural form. All declension forms are organized in a truncated digital search tree which determines that the morphological analysis starts from the end of a word. Apart from removing inflectional suffixes, the morphological analysis evaluates the past participle and returns the infinitive form of the verb. For example, the term NEUVES (new) will be processed in following way: first of all, three last characters in reverse order i. e. -SEV will be removed from the term (one character at a time) and after that character, F will be added to the remaining stem NEU forming the stem NEUF. The derivational process is similar to Porter's iterative affix removal approach.

As stated in the paper, derivational suffixes can be determined by using a suffix list based on four tables which correspond to four grammatical categories: nouns, adjectives, verbs

and adverbs. Each grammatical category covers special rules and several restrictions regarding gender and/or the remaining stem length. When the grammatical category and suffix of the term are determined, it is possible to find a term's stem and the grammatical category of the corresponding stem. For example, for the adjective VOLCANIQUE (volcanic) suffix -IQUE will be removed leaving the stem VOLCAN (volcano) which is a noun.

According to the researcher, the French stemmer has been evaluated using three basic tests. The first experiment covered weak stemming which removed inflectional suffixes (plurals, past participle) from 50 test terms. According to the result, all 50 terms were stemmed correctly. The second test was dealing with prefix removal and the success rate was also high. Finally, the third test which evaluated suffix removal procedure from terms containing only derivational suffixes also revealed high ratio of correct results. It was also observed that the use of grammatical categories can decrease the number of over stemmed terms.

2.5.1.3. GREEK STEMMING ALGORITHM

Suffix removal stemmer for Greek was one of the first attempts to construct a stemming algorithm for the language which is based on non-Latin character set [44]. The grammatical structure of Greek language covers a rich inflectional system which includes 41 forms of suffixes. Nouns in the Greek language have four different cases and the declension is carried out according to 41 categories of nouns, e.g., 14 for the masculine, 14 for the feminine and 13 for neuter. This iterative algorithm was based on two-stage suffix removal procedure: analysis and removal of inflectional suffixes; removal of derivational suffixes which correspond to their grammatical categories.

The evaluation based on two small test collections covering documents in medicine and computing as well as analysis of user enquiries revealed that the majority of errors were caused by under stemming. Although, the algorithm has not been implemented into any of Greek databases and/or tested using large document collections, the initial evaluation showed that the stemmer produced 90% correct stems.

2.5.2. STEMMING ALGORITHMS FOR LOCAL LANGUAGES

Researches in the areas of stemming for Ethiopian languages are presented as follows.

2.5.2.1. STEMMER FOR AMHARIC

The first Amharic stemming algorithm that conflates words for information retrieval was developed by Nega Alemayehu and Peter Willett [7]. Their work was one of the earliest main works in Amharic NLP researches. They used a previous work for Slovene [43] and developed stemming algorithm for Amharic in similar way.

As stated in [7], the stemmer was iterative that removes prefixes and suffixes and also considered letter inconsistency and reiterative verb forms. The same researcher explains that algorithm first identifies a set of stop-words and then a set of affixes associated with the remaining content-bearing words. The researchers have used the characteristics of the resulting affixes which used to guide the development of the stemmer. The stemmer removes affixes by iterative procedures that employ a minimum stem length, recoding and context sensitive rules, with prefixes being removed before suffixes. Once the stem of the word is obtained, the root is obtained by stripping all the remaining vowels from it.

As mentioned by this researcher, the stemmer uses a context-sensitive iterative procedure that removes both prefixes and suffixes. The performance of the stemmer was measured on a sample data of 1221 words. The result of the experiment shows that the stemmer performed at an accuracy of 95.9%.

Following the first Amharic stemmer work, Atelach and Lars developed another Amharic stemmer which is based on table lookup strategy [14]. They have presented the design and development of an Amharic stemmer which reduces words to their citation forms for the purpose of dictionary lookup in Cross Lingual Information Retrieval systems. The stemmer finds all possible segmentations of a given word according to the morphological rules of the language and then selects the most likely prefix and suffix for the word based on corpus statistics. It removes the prefix and suffix and then attempts to look up the remaining stem (or alternatively, some morphologically driven variants of it) in the stem dictionary to check that it is a potential stem of the word. The frequency and distribution of prefixes and suffixes over Amharic words are based on a statistical analysis of a 3.5 million word Amharic news corpus and some old-fashioned words from Amharic fiction.

As discussed in [14], the stemmer first creates a list consisting of all possible segmentations of the word that is to be stemmed. In a second step, each such segmentation is then verified by matching each candidate stem against the machine-readable dictionary. If no stem matches the dictionary, the stemmer will modify the stem and re-do the matching. If more

than one stem found, the most related stem is chosen after disambiguating between the potential stems based on statistical and other properties of the stems. In the case of one exact match in the dictionary, that stem will be presented as the output of the stemmer. This stemmer had an accuracy of 76% on news corpus and 60% on old-fashioned words when evaluated on a limited text consisting of 1503 and 470 words respectively.

2.5.2.2. STEMMER FOR AFAAN OROMO

The first rule based Afaan Oromo stemmer was developed by M. Wakshum [29]. This stemmer used suffix table in combination with rules that strips off suffix from a given word by looking up the longest match suffix in the suffix list. 342 suffixes were compiled automatically by counting and sorting the most frequent endings. Other linguistically valid suffixes were also included manually. The stemmer finds the longest suffixes that match the end of a given word and remove. This stemmer uses the longest-match, context-sensitive approach and rules that remove prefix and suffix. The stemmer was evaluated by counting stemming errors and also reduction of dictionary size. It performed an accuracy of 92.52% based on the test data of 1061 words.

Another Afaan Oromo stemmer was developed by D. Tesfaye and E. Abebe [15] to improve weakness of stemmer developed by M. Wakshum which had no rules to stem irregular and duplicated words.

As stated in [15], the stemmer was developed using hybrid approach (combining both Affix removal and n-gram techniques) and is based on a series of steps that removes a certain type of affix by way of substitution rules and suffix removal. These rules apply for specific conditions, for example, the resulting stem must have a certain minimal length. Most of the rules have a certain condition based on the measure (number of vowel-consonant sequences that present in the output stem). This condition must prevent that letters which look like a suffix but are just part of the stem will be removed.

The researcher discussed that the stemmer was tested on set of 5000 words. According to the paper, the corpus from which the stemmer developed was completely different from the test data so as to predict the performance of the stemmer in real world data. The output from the stemmer indicates, out of 2458 words 90 words (3.66 %) were under stemmed and 15 words (0.61 %) were over stemmed. Overall, this stemmer generated 105 words (4.27 %) stemming error. As a result, the accuracy of the stemmer was 95.73%.

2.5.2.3. STEMMER FOR TIGRIGNA

The first iterative Tigrigna stemmer was designed by G. Berhe in 2001 [12]. This stemmer was developed based on iterative procedure and uses context-sensitive rules that remove prefix, suffix, prefix-suffix pair and reduplication of single and double letters. The algorithm works in iterative approach but when it finds two affixes that match with the word, it removes the longest one.

As discussed in [12], the stemmer used five step rules for the purpose of removing affixes. The first step takes the word to be stemmed as an input and removes double letter reduplication. The second step removes prefix-suffix pair. This step takes the output of the first step as an input and checks if the words contain a match with any prefix-suffix pair. If the word contains a match and the remaining string has a length greater than the minimum length, then the prefix and suffix are removed from a word. The third step removes prefixes and takes the output of prefix-suffix stripping. In removing a prefix, checking for match in the prefix list and counting length of the remaining string is done. The fourth step removes suffixes by accepting the output from the previous stem and checks if the word contains any match from the list of suffixes. If the word has a match and the remaining string is greater than the minimum length, the suffix is removed from the word. In the last step, the algorithm stems reduplication of a single letter. This algorithm has recoding rule that is applied after each step is applied for checking some spelling exceptions and making readjustment. The accuracy of the stemmer was tested based on error counting technique and the result showed an accuracy of 84% and brought a dictionary reduction of 32.40%.

Another Tigrigna stemmer was developed by Y. Fiseha in 2011 [16]. The researcher has introduced the algorithm in order to minimize the gap of the existing algorithm by G. Berhe. and to create a more effective Tigrigna stemmer. Based on the previous research finding of G. Berhe, the researcher created a system that removes the affixes of the Tigrigna words in a relatively larger corpus by using specific grammatical rules to improve the effectiveness of the stemmer. However, this algorithm is limited to handle only prefixes and suffixes. Reduplication, compounding and irregular words were not handled in this research according to the paper.

The researcher further discussed that based on the experiments carried out for this study, the algorithm stems the words with an accuracy rate of 86.1%. As discussed by the researcher, context-sensitive rules were not included in the stemmer due to lack of common

context-sensitive rules that work commonly for a group of words. The stemmer also does not give similar accuracy on different datasets.

2.5.2.4. STEMMER FOR WOLAYTTA

The first attempt to develop stemmer for Wolaytta language was conducted by L. Lessa [17]. This Wolaytta stemmer is a rule based iterative stemmer which removes each base suffix one by one iteratively. This stemmer is also a context sensitive.

The researcher employed a semi-automatic and semi-manual means to prepare the potential suffixes. The stemmer first gets a word to be stemmed then checks if a suffix from the list is attached to the word. Next, these suffixes are iteratively stripped from the word and after application of necessary condition, the final word is considered as a stem. The stemmer was evaluated in a sample of 884 words and the performance registered was 86.9%.

2.5.2.5. STEMMER FOR SILT'E

The first stemmer for Silt'e language was developed by M. Kedir in 2012 [18]. According to the study, the stemmer contains prefix striping, suffix stripping and letter reduplication stripping modules.

The researcher indicated that the stemmer implemented is the iterative but longest match first and the lists of affixes are checked against the word. It removes affixes iteratively until the entire affixes are removed. In the stemmer, three steps were used for the purpose of removing affixes. The first step removes prefixes. In the second step, the removal of suffix is done and in the third step reduplication of letters is removed from the word. The stemmer was tested on a data set of 1486 words. The stemmer performed at accuracy of 85.71% and reduced dictionary size by 34.99%.

2.5.2.6. RESEARCH GAP

From the literature review, the researcher observed that stemming algorithms have been designed to different local languages. The languages in which stemming researches conducted include Amharic [7], [14], Afaan Oromo [15], [29], Tigrigna [12], [16], Wolaytta [17], Silt'e [18] and few others. However, there is no any research carried out to design stemming algorithm for Kambaata language text. Thus, the researcher used this opportunity to do a research on designing a rule based stemming algorithm for Kambaata words.

CHAPTER THREE

MORPHOLOGY OF KAMBAATA LANGUAGE

3.1. OVERVIEW OF THE KAMBAATA LANGUAGE

As mentioned below, Kambaata has its own writing system and uses both Ethiopic and Latin script; the latter being used primarily. Since Emperor Haile Selassie (head of Ethiopian government) started a literacy campaign in 1955 known as 'Yefidel Serawit" in Amharic, which literally means 'an alphabet army', Ge'ez-based Ethiopic writing system was introduced into the languages spoken all over the country including Kambaata [67]. The literacy campaign lasted until 1991 when the current central government came to power; instead of continuing the same campaign, the new government started to allow the speakers of non-Semitic languages to develop their own writing systems. Accordingly, Latin-based alphabet system was designed in 1992 (1984/1985 E.C.) to be used by Kambaata language, but the literacy rate at present still seems to be low and the language has been poorly documented [20], [21].

3.2. THE WITING SYSTEM OF KAMBAATA LANGUAGE

As stated in [19], both the Latin script and Ethiopic also known as Ge'ez script are used for writing Kambaata texts. For instance, the Bible New Testament and part of the Old Testament is written in Ge'ez script. However, according to linguists and the researcher's practical observation and experience, the official writing system (orthography) taught currently in primary and secondary schools in the Zone is Latin-based and there is some deviation from the International Phonetic Alphabet (IPA) convention. For example, the word *Kambata* is written as *Kambaata* where double letters "*aa*" indicate the length.

Most Kambaata languages publications as of today belong to religious and educational domains. After a Latin-based orthography had been designed in 1992, school books for all subjects were developed in Kambaata language for primary school (grades 1-6) and the books for Kambaata lessons (Kambaatissata: Rosaanchi Maxaafa) for grades 7 and 8 at the start. Recently, the language is medium of instruction in primary schools (grades 1-4), taught as a subject in junior and secondary schools and preparatory level (or grades 5 – 11) and curriculum is under development for grade 12. As stated in [19], Latin based orthographic rules of Kambaata were written with some explanations and examples for the first time in 1992.

The researcher also discussed that for many of the Kambaata speakers, reading and writing in their mother tongue is restricted to the context of primary and junior high school as Amharic is dominant official writing language in work places both at country and regional level. Although, formally educated young generation has attended the Kambaata classes in school, the Kambaata language has not yet become part of the "linguistic landscape". The national language, Amharic is still the preferred language in governmental institutions, religious organizations, announcements, traffic signs, and notice boards [19].

3.2.1. ALPHABETS

Kambaata orthography is Latin-based. As stated in [19], some consonant phonemes are written by more than one series of characters: "**'**", "**s**", "**y**" and "**h**". Kambaata has 28 consonants and five short vowels: "**a**", "**i**", "**e**", "**u**", "**o**" and their long counterparts "**aa**", "**ii**", "**ee**", "**uu**" and "**oo**".

As discussed by [19], the official orthography of Kambaata deviates from the IPA conventions as follows: "**ph**" = pʼ, "**x**" = tʼ, "**q**" = kʼ, "**j**" = dʒ, "**c**" = tʃʼ, "**ch**" = tʃ, "**sh**" = ʃ, "**ny**" = ɲ, "**zh**" = ʒ, "**'**" = ʔ, "**'l**" = lʼ and "**'r**" = rʼ. Length is specified by double vowel letters and double consonant represents stress in speech, for example: "**chch**" = tʃː, "**aa**" = aː, "**bb**" = bː, and "**shsh**" = ʃː. By convention, the second consonant of a glottal stop-sonorant cluster is written as double, although the cluster only consists of two phonemes, e.g. "**'rr**" = ʔr, "**'ll**" = ʔl. Consequently, the digraphs "**'l**" and "**'r**" are free to be used to mark glottalised sonorants. Treis [19] also further discussed that word-final unstressed /i/ does not occur orthographically, irrespective of its phonological status [19].

TABLE 3-1: KAMBAATA ALPHABETS AND THEIR ETHIOPIC COUNTERPARTS [49]

Alphabet		Amharic Equivalent	Alphabet		Amharic Equivalent
Capital	Small		Capital	Small	
A	a	አ/ዐ	O	o	ኦ/ዖ
B	b	ቡ	P	p	ፐ
C	c	ጭ	PH	ph	ጵ
CH	ch	ች	Q	q	ቀ
D	d	ድ	R	r	ር
E	e	ኤ/ዔ	S	s	ስ/ሥ
F	f	ፍ	SH	sh	ሽ

G	g	ግ	T	t	ት
H	h	ሀ/ሐ/ኅ/ኸ	TS	ts	ፅ/ጽ
I	i	እ/ዕ	U	u	ኡ/ዑ
J	j	ጅ	V	v	ቭ
K	k	ክ	W	w	ው
L	l	ል	X	x	ጥ
M	m	ም	Y	y	ይ
N	n	ን	Z	z	ዝ
NY	ny	ኝ	ZH	zh	ዥ

3.2.1.1. VOWELS

Kambaata orthography has a five-vowels. Vowel length is indicated by double vowels. Long vowels are bi-phonemic, i.e. combinations of two short vowels.

TABLE 3-2: KAMBAATA VOWELS (SHORT AND LONG) [19]

	Front	Central	Back
High	i, ii		u, uu
Mid	e, ee		o, oo
Low		a, aa	

Vowel phonemes appear in the middle and end of a word after any consonant. Phonemic opposition between short "***o***" and "***a***" is neutralized only after the approximant "***w***".

3.2.1.2. CONSONANTS

Kambaata writing system has 28 consonant graphemes. The consonant graphemes are presented in Table 3-3. Graphemes that are not indicated inside pointed brackets <…> are identical to the IPA standard in phonemic and orthographic representation of a phoneme. Starred graphemes (⊗) are exclusively used for writing loanwords [19]. The remaining graphemes indicated inside pointed brackets are customized for Kambaata orthography.

TABLE 3-3: CONSONANT PHONEMES AND THEIR ORTHOGRAPHIC REPRESENTATION [19]

		Labial	Alveolar	Palato - alveolar	Velar	Glottal
Stops	voiceless	<p>⊗	t	<ch>	k	(<'>)
	voiced	b	d	<j>	g	
	glottalic	<ph>	<x>	<c>	<q>	

Fricatives	voiceless	f	s	<sh>		h
	voiced	<v>⊗	z	<zh>		
	glottalic		<ts>⊗			
Nasals		m	n	<ny>		
Vibrants	plain		r			
	glottalized		<'r>			
Laterals	plain		l			
	glottalized		<'l>			
Glides		w		y		

3.3. KAMBAATA MORPHOLOGICAL SYSTEM AND WORD FORMATION

Morphology a discipline of natural language study that deals with the inner composition of terms and their construction, such as affixation, roots, and pattern properties [47]. According to Spencer [47], morphology is the origin of word variation in natural language text, with suffixing and prefixing being the most prominent approaches of word variant formation. Morphology can be divided in two; inflectional or derivational. Inflection is variation or change of form that words experience to indicate differences of case, gender, number, tense, person, mood, voice. Inflectional morphology is employed to a given stem with expected formation. It does not impact the word's part of speech, such as noun, verb, etc. Case, gender, number, tense, person, mood, and voice are some good examples of properties that might be impacted by inflection. Derivational morphology is derivation of a given term to create diverse term variations from a one stem. In cases like this, the word's grammatical as well as syntactic class could possibly be impacted and altered [16].

Usually, terms can have numerous word forms, e.g., the word "***waal***" (come) can take the forms "***waalleeu***" (came) and "***waalayyoou***" (is coming), usually called inflected forms. The root is the original form of the word before any transformation process, "***waal***" in our case, and it plays an important role in language studies. The root is the form of a word from which the other forms can be derived using the morphological rules of a language. A morpheme is the smallest unit of a language that has a meaning and cannot be broken down further into meaningful or recognizable parts and should impart a function or a meaning to the word which they are part of. An affix is a morpheme that can be added before (prefix)

or after (suffix), or inserted inside (infix) a root or a stem to form new words or meanings [48]. Morphological behavior of one language is beneficial for several applications in the field of NLP for instance, stemming, morphological analysis, text summarization, machine translation, information retrieval and so on.

Kambaata is a rigidly head-final (i.e. word order = left-branching (SOV)) and strictly suffixing language with a rich nominal and verbal morphology [50]. It is an agglutinative language, where almost all derivational morphology and all inflectional morphology involve affixation. It has been emphasized in [50] that Kambaata is exclusively suffixing language and that there are no prefixes in the language. However, the research is able to find very few words with prefix affixation as indicated in Chapter four.

Inflectional affixes describe word as stems combined with grammatical markers for things like person, gender, number, tense, and case. Regarding parts of speech in Kambaata, there are five open word classes (nouns, verbs, adjectives and ideophones and interjections); and several closed word classes (pronouns, numerals and quantifiers, demonstratives; hardly any conjunctions and adverbs) [50]. Ideophones and interjections are morphologically invariant [19]; prepositions and conjunctions are totally unproductive for IR purpose or other natural language processing. The same happens for adverbs which are few in number. Therefore, the discussion of derivational and inflectional morphology concentrates on the main three parts of speech, namely verbs, nouns and adjectives.

3.3.1. VERB MORPHOLOGY

Kambaata is an exclusively suffixing language and all inflectional morphemes are located after the verbal stem. A draft of the morpheme structure of a (declarative affirmative) main verb is given in Table 3-4 and example is provided in section 3.3.1.1 [58].

As stated in [51] and [58], the verbal stem consists of the root and derivational morphemes. Each affirmative declarative main verb has two subject agreement markers. The first agreement slot is occupied by the inherited Afro-asiatic subject morphemes; the second slot contains agreement morphemes. Aspect morphemes are placed in the slot between the subject agreement positions. As discussed in [58], four aspectual values are distinguished in main verb paradigms: (i) Imperfective (IPV), (ii) Perfective with a characteristic e-vowel (PVE), (iii) Perfective with a characteristic o-vowel (PVO), and (iv) Progressive (PROG).

Kambaata is primarily an aspect-marking language. Verbs are inflected for aspect (the primary differentiation of verb forms is between perfective (P) and imperfective (I) paradigms (some of which are given in (1)), mood (indicative, imperative/jussive, and preventive), subordination (main vs. subordinate verb) and subject agreement (person, gender, number and social status of the subject). Verbs are situated at the rightmost end of the clause [51].

Important distinction between *main (final)* verbs and *subordinate (non-final* verbs) in Kambaata is that main verbs end a sentence, and have the most elaborate inflectional potential (more distinctions for subject agreement, aspect, mood) while subordinate verbs are found sentence medially (relative verbs, converbs, purposive verbs, verbal nouns), and have a reduced inflectional potential (less or no distinctions for subject agreement and aspect, no mood marking), some are marked for switch-reference [53].

(1)	***it-***	**eat**	
I	*it-aamm*	I will eat. / I eat habitually.	imperfective main verb
P	*ichch-eemm*	I ate.	e-perfective main verb
P	*ichch-oomm*	I ate.	o-perfective main verb
I	*it-an(i)*	… (I) eating …	imperfective SS converb
P	*ichch(i)*	… (I) having eaten …	perfective SS converb
I	*it-ani-yan*	… (I) eating …	imperfective DS converb
P	*ichchi-yan*	… (I) having eaten …	perfective DS converb

3.3.1.1. MORPHOLOGICAL STRUCTURE OF VERBS

TABLE 3-4: STRUCTURE OF A DECLARATIVE AFFIRMATIVE MAIN VERB [54]

Stem: Root (+ Derivation)	Inflection			(Object Suffix)	(*ikke*)
	Subject Agreement *	Aspect	Subject Agreement		
	1SG: *-Ø* 2SG: *-t* 3M: *-Ø* 3F/PL: *-t* 3HON: *-een* 1PL: *-n*	IPV: *-a(a)* PVE: *-e(e)* PVO: *-o(o)* PROG: *-ayyoo*	1SG: *-m(m)* 3M: *var.* 2SG: *-nt* 3F/PL: *(-'V)* 3HON: *var.* 1PL: *-m(m)* 2PL: *-nta(a'u)*	1SG: *-'e* 3M: *-s* 2SG: *-kke* 3F: *-se* 3HON/PL: *-ssa* 1PL: *-nne*	

	2PL/HON: -*teen*			2PL/HON: -*(kki)'nne*

* In the singular, the persons are ordered in an unorthodox way to point out the common neutralization pattern 1SG=3M and 2SG=3F/PL.

At the right edge: pronominal object suffixes and inactual ("past") morpheme ikke is added. "***ikke***" is used to show something that was habitual in the past. It can be represented by 'used to'.

e.g.	*xuudanos ikke*	*xuud-Ø-a-no-s*	*ikke*
		see-3m-IPV-3m-3mO	INACT

'He used to see him.'

3.3.1.2. ASPECT MARKING ON MAIN VERBS

IMPERFECTIVE (IPV)

Characteristic aspect vowel (between agreement morphemes): a(a)

TABLE 3-5: IMPERFECTIVE MAIN VERB [53]

	-AGR	*-a(a)*	-AGR	e.g. *kul* - 'tell'
1SG	*-Ø*	*-aa*	*-m(m)*	*kul-**aam(m)***
2SG	*-t*	*-aa*	*-nt*	*kul-**taant***
3M	*-Ø*	*-a*	*-no*	*kul-**ano***
3F/3PL	*-t*	*-aa*	*(-'V)*	*kul-**taa('u/'a/')***
3HON	*-een*	*-Ø*	*-no*	*kul-**eenno***
1PL	*-n*	*-aa*	*-m(m)*	*kun-**naam(m)***
2PL/2HON	*-t-een*	*-a*	*-nta*	*kul-**teenanta***

The imperfective marks an event as non-completed, either because the event is 1) habitual, a general truth, 2) carried out or happening at the speech time, in the future from the perspective of the speech time, or 3) in the future relative to the time of a past reference event. The IPV is not only used to encode events that are non-completed at the speech time

but also at a reference time prior to the speech time, i.e. the imperfective can mark habitual events in the past [53], [55].

PROGRESSIVE (PROG)

The complete progressive paradigm of the verb *kul-* 'tell' is given in Table 3-6 [55].

TABLE 3-6: PROGRESSIVE MAIN VERB [53]

	-AGR	*-ayyoo*	-AGR	e.g. *kul* - *'*tell*'*
1SG	*-Ø*	*-ayyoo*	*-m(m)*	*kul-**ayyoom(m)***
2SG	*-t*	*-ayyoo*	*-nt*	*kul-**tayyoont***
3M	*-Ø*	*-ayyoo*	*(-'V)*	*kul-**ayyoo('u/'i)***
3F/3PL	*-t*	*-ayyoo*	*(-'V)*	*kul-**tayyoo('u/'i)***
3HON	*-een*	*-ayyoo*	*- mma*	*kul-**eenayyoomma***
1PL	*-n*	*-ayyoo*	*-m(m)*	*kun-**nayyoom(m)***
2PL/2HON	*-t-een*	*-ayyoo*	*-nta*	*kul-**teenayyoonta***

The progressive marks a durative event as being in the process of happening at the reference time (which does not have to be the speech time). Progressive marking encodes iteration (*ub-ayyoo'u* 'he keeps on falling') with punctual verbs (e.g. *ub-* 'fall') and an incipient change of state (*qeree'rr-ayyoo'u* 'he is growing tall') with inchoative-stative verbs (*qeraa'rr-* 'be(come) tall') [53], [55].

PERFECTIVE (PVO)

There are two perfective paradigms: e-form "Simple Perfect" vs. o-form "Present Perfect" in Kambaata [53], [55].

- Characteristic aspect vowel between agreement morphemes: o(o) (except in 3HON and 2HON/PL).
- Characteristic morphophonological process: palatalization (P) and gemination (G) in 1SG and 3M.
- Overlap perfective/perfect: 3HON and 2PL/HON forms shared with the perfect paradigm.
- Defectiveness: 1SG and 3M perfective form can only be formed from verb stems ending in -C (but not -CC) > verb stems in -CC do not distinguish between perfective and perfect (e.g. *barg-* 'add' > *barg-eemm* 1SG perfective/perfect).

- Neutralization: Distinction between perfective and perfect is neutralized in the negation.

TABLE 3-7: PERFECTIVE MAIN VERB [53]

	-AGR	*-o(o)*	-AGR	e.g. *kul* - 'tell'
1SG	*(P/G)-Ø*	*-oo*	*-m(m)*	*kull-**oom(m)***
2SG	*-t*	*- oo*	*-nt*	*kul-**toont***
3M	*(P/G)-Ø*	*- o*		*kull-**o***
3F/ 3PL	*-t*	*- oo*	*-('V)*	*kul-**too('u)***
3HON	*-een*	*-Ø*	*- ma(a'V)*	*kul- **eemma(a/u/)***
1PL	*-n*	*- oo-*	*-m(m)*	*kun- **noom(m)***
2PL/2HON	*-t-een*	*- Ø*	*-nta(a'V)*	*kul- **teenta(a/u/)***

The perfective form is used in texts for sentences which advance the story. The sentences advancing the story end in true perfective verbs or potential perfective verbs.

PERFECT (PVE)

- Characteristic aspect vowel between agreement morphemes: ***ee*** (except in 3HON and 2PL/HON).
- Characteristic morphophonological process: palatalization (P) and gemination (G) in 1SG and 3M.
- Accentuation of the verb form dependent on structure of the stem (except in 3HON and 2PL/HON).
- Overlap perfect/perfective: 3HON and 2PL/HON forms shared with the perfective paradigm.

TABLE 3-8: PERFECT MAIN VERB [53]

	-AGR	*-ee*	-AGR	e.g. *kul* - 'tell'
1SG	*(P/G)-Ø*	*-ee*	*-m(m)*	*kull-**eem(m)***
2SG	*-t*	*- ee*	*-nt*	*kul-**teent***
3M	*(P/G)-Ø*	*- ee*	*-('V)*	*kull-**ee('u)***
3F/ 3PL	*-t*	*- ee*	*-('V)*	*kul-**tee('u)***
3HON	*-een*	*-Ø*	*- ma(a'V)*	*kul-**eemma(a/u/)***
1PL	*-n*	*- ee-*	*-m(m)*	*kun- **neem(m)***
2PL/2HON	*-t-een*	*- Ø*	*-nta(a'V)*	*kul- **teenta(a/u/))***

The perfect is used for processes and actions that are completed but whose ensuing/ resulting state continues to the reference time [53].

3.3.1.3. NEGATION OF VERBS

NEGATION OF MAIN VERBS

Negative imperfective main verbs (3) are merely marked by the addition of a morpheme *-ba'a* to the affirmative form (2). The accentual structure of the imperfective verb is not altered by the additional negative morpheme [20].

(2) Imperfective affirmative

Stem	Subject agreement	Aspect	Subject agreement	(Object suffix)

e.g. *xuud-deenanta-s* < *xuud-teen-a-nta-s*
see-2PL.IPV-3M.OBJ
'you (PL) see him'

(3) Imperfective negative

Stem	Subject agreement	Aspect	Subject agreement	(Object suffix)	*-ba'a*

e.g. *xuud-deenanta-si-ba'a* < *xuud-teen-a-nta-s-ba'a*
see-2PL.IPV-3M.OBJ-NEG
'you (PL) do not see him'

Perfective (4) and progressive (5) main verbs share one negative paradigm (6), which is characterized by a morpheme *-im* (a marker for non-imperfective aspect) after the first subject agreement morpheme and by a subsequent negative morpheme *-ba*(*'a*). The negative perfective lacks the second subject agreement marker. Object suffixes occur after the negative morpheme and trigger the loss of the "glottal appendix", i.e., before an object suffix, the negative morpheme is realized as *-ba* (6) [53], [55].

(4) Perfective affirmative

Stem	Subject Agreement	Aspect	Subject Agreement	(Object suffix)

e.g. *xuud-deenta-s* < *xuud-teen-nta-s*
see-2PL.PVE-3M.OBJ

'you (PL) saw him'

(5) Progressive affirmative

Stem	Subject Agreement	Aspect	Subject Agreement	(Object suffix)

e.g. *xuud-deenayyoonta-s* < *xuud- teen-ayyoo-nta-s*

see-2PL.PROG-3M.OBJ

'you (PL) are seeing him'

(6) Perfective ["non-imperfective"] negative

Stem	Subject Agreement	*-im*	*-ba('a)*	(Object suffix)

e.g. *xuud-deenim-ba-s* < *xuud- teen-im-ba-s*

see-2PL.NIPV-NEG-3M.OBJ

'you (PL) did not see him; you (PL) are not seeing him'

Kambaata has several unrelated negation morphemes. Apart from the morpheme *-ba'a*, whose use for the negation of indicative main verbs was exemplified in (3) and (6), the morpheme *-u'nna* serves to negate converbs, the morpheme *-ka* is applied to negative jussive verbs, and the morpheme *-oot* signals negative imperative verbs [53], [55].

NEGATION OF RELATIVE VERBS

Relative verbs are marked as negative by the morpheme *-umb*, which does not seem to be related to the aforementioned negative morphemes. The negative RVs, whose paradigm is presented in Table 3-9, are not derived from negative main verb forms [20].

TABLE 3-9: PARADIGM OF NEGATIVE RELATIVE VERB FORMS

Person	*Morpheme*	*e.g. kul - 'tell'*
1SG and 3M	*-Ø-umb-u*	*kul* - ***umb-u***
2SG and 3F/PL	*-t-umb-u*	*kul* - ***t-umb-u***
3HON	*-een-umb-u*	*kul* - ***een-umb-u***
1PL	*-n-umb-u*	*kul* ***-n-umb-u***
2PL/HON	*-teen-umb-u*	*kul* - ***teen-umb-u***

OTHER NEGATION FORMS

TABLE 3-10: OTHER NEGATION SUFFIXES OF VERBS IN KAMBAATA

Person	*Morpheme*
3M	*-u'nna*
1SG	*-ba('a)*
3M	*-ka*
2SG	*-toot*
2PL	*-een-oochch-e*

e.g. hujat - 'work'	*Translation*
hujat- ***u'nna***	*'…(He) before/without working'*
*hujat -****aam-ba'a***	*'I don't work'*
hujat - ***un-ka***	*'Don't let him work'*
*hujat -****toot***	*'Don't do(work)'*
*hujat-****een-oochch-e***	*'Don't do(work)'*

In Kambaata languages, a considerable number of different negative morphemes are found, because different clause types require different negation strategies. An overview of the forms and functions of all negative morphemes discussed in the preceding sections is given in Table 3-11 [59].

TABLE 3-11: CLAUSAL NEGATION IN KAMBAATA SUMMARIZED

Standard (ST)	Existential (EX)	Imperative (IMP)	Jussive (JUS)	Converb (CVB)	Relative (REL)
-ba('a)	*-ba('a)*	*-oot; -oochch*	*-(n)ka*	*-u'nna;* *-u'nnaan;* *-u'nnaachch*	*-um(-)b-*

3.3.1.4. DERIVATION

Unlike other word categories, derivation of verbs from other part of speech is not common [62]. By removing the adjectival suffixes, verbs can be derived from adjectives as shown in Table 3-12 [19].

TABLE 3-12: DERIVED VERBS AND THE CORRESPONDING ADJECTIVES

MORPHEME	ADJECTIVE	TRANSLATION	DERIVED VERB	TRANSLATION
-a(-ta)	*abb-a(-ta)*	big; much, many	*abb-*	be(come) big
-a(-ta)	*annann-a(-ta)*	different	*annann-*	be(come) different
-aan-ch-u(-ta)	*ros-aan-ch-u(-ta)*	student	*ros-*	learn
-aan-ch-u(-ta)	*ros-is-aan-ch-u(-ta)*	teacher	*ros-is-*	teach
-im-a(-ta)	*hoog-im-a(-ta)*	weak, tired	*hoog-*	be(come) tired
-all-a(-ta)	*baab-allu(-ta)*	frightened	*baab*	be(come) frightened

-eem-a(-ta)	*biddiqq-eem-a(-ta)*	flat, even, spread out	*biddiqq y-*	be(come) flat, even, be spread out

3.3.1.5. INFLECTIONAL SUFFIXES FOR A VERB IN KAMBAATA

For general verb inflection, the researcher demonstrated some possible suffixes for a single verb "*kul* (tell)" but the list might not be exhaustive as the language has very rich morphology [20], [54], [55], [56], [57].

Kambaata verb inflection for a verb "*Kul*" 'tell' within different tense aspect mood is provided in Appendix VI (Table VI-1) and exhaustive list (as much as possible) is provided in Appendix VII.

Purposive: is subordinate verb form used in purpose clauses. SS purposive are used in questions about one's intentions or plans [55]. For other forms of negations, examples are given using the stem "kul" 'tell' in Table 3-13.

TABLE 3-13: OTHER FORMS OF NEGATION

Person	*Morpheme*	*e.g. kul - 'tell'*
3M	*-u'nna*	*kul-* ***u'nna***
3M	*-ka*	*kul -* ***un-ka***
2SG	*-toot*	*kul* ***-toot***

Other inflectional forms of the verb can be represented with equivalent compound word with other suffixes.

TABLE 3-14: OTHER INFLECTIONAL FORMS

Word	suffix	Equivalent compound word	New suffix
kullebe	*-l-ebe*	kamme-kulle	*-le*
kullela	*-lela*	hikkada-kulle	*-le*
kultaada	*-taada*	kulii-hassooda	*-ii*

3.3.2. NOUN MORPHOLOGY

Kambaata has fairly rich nominal morphology. It has a marked nominative case system. Segmentally, case is marked by suffixes, by a specific stress position. Eight case forms are distinguished: nominative (NOM), accusative (ACC), genitive (GEN), dative (DAT), ablative (ABL), instrumental-comitative-perlative (ICP), locative (LOC) and oblique (OBL). Kambaata distinguishes two genders: masculine and feminine. The case endings of

a feminine noun and a masculine noun of two selected declensions are given in Table 3-15 [55], [60].

TABLE 3-15: TWO EXEMPLARY NOMINAL DECLENSIONS

Declension	ACC	NOM	GEN	DAT	ABL	ICP	LOC	OBL
F1a	*-a-ta*	*-a-t*	*-a*	*-aa(-ha)*	*-aachch*	*-aan*	*-aan*	*-a*
M1	*-a*	*-u*	*-í*	*-ii(-ha)*	*-iichch*	*-iin*	*-aan*	*-a*

3.3.2.1. MORPHOLOGICAL STRUCTURE OF NOUNS

The morphological structure of nouns in Kambaata is provided in Figure 3-1. The stem of a noun consists of a root plus derivational morphemes. The derivational morphemes closest to the root are predominantly word class-changing formatives. The second type of derivational morphemes derives singulative and plurative nouns. In the order of morphemes, number markers are situated between (word class-changing) derivation and inflection. Number cannot be regarded as an inflectional category such as case. However, it may also not be unhesitatingly considered a derivational category, though it certainly has more traits of derivation [19].

In Kambaata, all nominal roots and stems end in a consonant or consonant cluster. An obligatory (primary) case suffix follows this / these consonant(s), i.e. the minimal noun consists of a root plus case vowel. Bare roots or bare stems are never used in isolation, but are merely units of the linguistic analysis. After the (primary) case morpheme, a (secondary) case / gender suffix and a possessive suffix can be attached. Coordination is generally marked at the right-most end of a noun [19].

FIGURE 3-1: MORPHOLOGICAL STRUCTURE OF NOUNS [19]

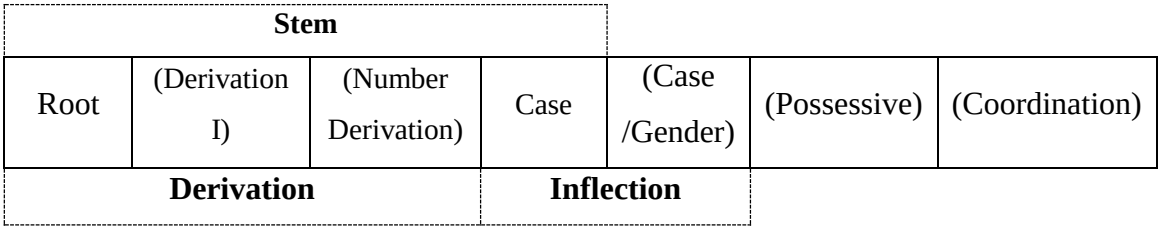


Figure 3-1 provides only an incomplete picture of the noun structure. Interdependencies between morphemes, such as the incompatibility of the gender marker with certain case suffixes, are not integrated into the figure above. Besides, purely pragmatically determined morphemes, *-n*, *-be*, *-'nnu*, and *-ma*, and enclitics are omitted.

3.3.2.2. CASE

Case has a significant importance in Kambaata, because it is the most important means to encode the syntactic relations in phrases and sentences. Kambaata differentiates between eight cases: NOM, ACC, GEN, DAT, ABL, ICP, LOC and OBL [19], [61].

TABLE 3-16: NOMINAL DECLENSIONS [61]

Declension	ACC	NOM	GEN	DAT	ABL	ICP	LOC	OBL
F1a	*-a-ta*	*-a-t*	*-a*	*-aa(-ha)*	*-aachch*	*-aan*	*-aan*	*-a*
F1b	*-a*	*-a*	*–a*	*-aa(-ha)*	*-aachch*	*-aan*	*-aan*	*-a*
F2a	*-i-ta*	*-i-t*	*-e*	*-ee(-ha)*	*-eechch*	*-een*	*-een*	*-e*
F2b	*-e*	*-/i/*	*-e*	*-ee(-ha)*	*-eechch*	*-een*	*-een*	*-e*
F3a	*-u-ta*	*-u-t*	*-o*	*-oo(-ha)*	*-oochch*	*-oon*	*-oon*	*-o*
F3b	*-o*	*-u*	*-o*	*-oo(-ha)*	*-oochch*	*-oon*	*-oon*	*-o*
F4	*-aa-ta*	*-aa-t*	*-aa*	*-aa(-ha)*	*-aachch*	*-aan*	*-aan*	*-aa*
F5	*-ee-ta*	*-ee-t*	*-ee*	*-ee(-ha)*	*-eechch*	*-een*	*-een*	*-ee*
F6	*-oo-ta*	*-oo-t*	*-oo*	*-oo(-ha)*	*-oochch*	*-oon*	*-oon*	*-oo*
M1	*-a*	*-u*	*-i*	*-ii(-ha)*	*-iichch*	*-iin*	*-aan*	*-a*
M2	*-i*	*-u*	*-i*	*-ii(-ha)*	*-iichch*	*-iin*	*-een*	*-e/()*
M3	*-u*	*-u*	*-i*	*-ii(-ha)*	*-iichch*	*-iin*	*-oon*	*´-o*
M4a	*-a*	*-/i/*	*-i*	*-ii(-ha))*	*-iichch*	*-iin*	*-aan*	*´-a*
M4b	*-o*	*-/i/*	*-i*	*-ii(-ha)*	*-iichch*	*-iin*	*-oon*	*´-o*
M5a = F1b	*-a*	*-a*	*-a*	*-aa(-ha)*	*-aachch*	*-aan*	*-aan*	*´-a*
M5b = F2b	*-e*	*-/i/*	*-e*	*-ee(-ha)*	*-eechch*	*-een*	*-een*	*´-e*
M5c = F3b	*-o*	*-u*	*-o*	*-oo(-ha)*	*-oochch*	*-oon*	*-oon*	*´-o*
M6	*-aa(-ha)*	*-oo(-hu)*	*-ee*	*-ee(-ha)*	*-eechch*	*-een*	*-aan*	*-aa*
M7	*-ee(-ha)*	*-oo(-hu)*	*-ee*	*-ee(-ha)*	*-eechch*	*-een*	*-een*	*-ee/()*
M8	*-oo(-ha)*	*-oo(-hu)*	*-ee*	*-ee(-ha)*	*-eechch*	*-een*	*-oon*	*-oo*
M9	*-uu(-ha)*	*-uu(-hu)*	*-ii*	*-ii(-ha)*	*-iichch*	*-iin*	*-uun*	*-uu/()*

Explanations and Examples:

F1a: largest feminine declension; e.g.: *macc-a-ta* ‘ear’, most plurative nouns, e.g. *boorr-a-ta* ‘oxen’

F1b: minor declension of feminine proper nouns, e.g. *Bes-a* (women’s name)

F2a: e.g. *gat-i-ta* ‘backyard’

F2b: most feminine proper nouns, e.g. *Aacaam-e* (women’s name); some common nouns, e.g. *char-e* ‘type of bird’

F3a: e.g. *xinkuta* ‘riddle’, feminine singulative nouns in *-(ich)chuta* and feminine agent nouns in *-aanchuta* e.g. *Kambaat-ichchu-ta* (SG) ‘Kambaata woman’

F3b: minor declension of feminine proper nouns, e.g. *Ayyaant-o*, some common nouns, e.g. *xorb-o* 'ball'

F4: e.g. *mashsh-aa-ta* 'type of knife (for enset food)', associative nouns in *-'aata*

F5: e.g. *quncul-eeta* 'scraper', associative nouns in *-'eeta*

F6: e.g. *hiz-oo-ta* 'sister', associative nouns in *-'oota*, e.g. *Xummiso'oota* '*Xummiso* and his associates'

M1: largest masculine declension, *ishima* 'brother of mother', most loanwords, e.g. *muuz-a* < Amh. *muz* 'banana'

M2: e.g. *fool-i* 'soul'

M3: e.g. *utubu* 'center pole (house)', masculine singulative nouns in *-(ich)chu* and masculine agent nouns in *-aanchu*

M4a: minor declension of masculine proper nouns, e.g. *Duuball-a* (men's name);

M4b: masculine proper nouns ending in *-aam-o*, -am-o, *-eeb-o*, and *-aab-o*, e.g. *Wo'llaam-o, Latam-o, Makkeeb-o, Ansheeb-o, Alaab-o*

M5a: terms of address, *ann-a* and *abb-a* for father; terms belonging to affectionate language, e.g. *ha'-a* 'disgusting, yucky thing'; morphologically identical to F1b

M5b: minor declension of masculine proper nouns, morphologically identical to F2b; e.g. *Moc-e, Doboc-e, Qaallor-e* (men's names); *Gurr-e, Booq-e* (bull names);

M5c: most masculine proper nouns, morphologically identical to F3b; e.g. *Hawaando, La'llaag-o, Habsiis-o* (men's names); *Ebal-o* 'so-and-so';

M6: e.g. *zaanzaa* 'centre of enset corm', *adab-aa* 'boy',

M7: (so far only) *qoqee* 'throat'

M8: e.g. *eloo* 'pit', *af-oo* 'mouth', *max-oo* 'rainy season',

M9: (so far only) *haguu* 'dry season'

Case forms of a woman's name (*Muccure*) and a letter name (*i*) compared as an example for the eight case forms is given in Table 3-17 below [19].

TABLE 3-17: CASE FORMS OF A WOMAN'S NAME (*MUCCURE*) AND A LETTER NAME (I) COMPARED [19]

ACC	NOM	GEN	DAT	ABL	ICP	LOC	OBL
Muccur-*e*	Muccur-*/i/*	Muccur-*e*	Muccur-*ee(-ha)*	Muccur-*eechch*	Muccur-*een*	Muccur-*een*	Muccur-*e*
i-*h*-*e*	*i*	i-*h*-*e*	i-*h*-*ee(-ha)*	i-*h*-*eechch*	i-*h*-*een*	i-*h*-*een*	*i*

3.3.2.3. GENDER

Kambaata distinguishes two genders. There is semantic justification for giving these two genders the labels ‘masculine’ and ‘feminine’, since nouns denoting males are assigned to the first gender and those denoting females to the second. The nouns *sa’-a*(M) ‘cow’ and *meent-u*(M) ‘women’ are two exceptions to this rule; here the female sex of the referent(s) contradicts the masculine gender of the nouns. Nouns which do not denote living beings are not assigned to a single gender but equally distributed across masculine and feminine gender [19].

Feminine gender is marked by an accent-neutral morpheme *-ta* in the accusative and *-t/i/* in the nominative. See also most of the feminine declensions in Table 3-18 in Section 3.5.2.2.

e.g. *am-a-ta* ‘mother’
hix-i-ta ‘grass’

Plurative nouns have generally the form of feminine nouns. They belong to the feminine declension **F1a** and trigger feminine agreement on the verb.

e.g. *am-aakk-a-ta* ‘mothers’ (plurative of *am-a-ta* ‘mother’)
hiz-aakk-a-ta ‘brothers’ (plurative of *hiz-oo* ‘brother’ / *hiz-oo-ta* ‘sister’)
minn-a-ta ‘houses’ (plurative of *min-í* ‘house’)

Masculine nouns are usually unmarked for gender.

e.g. *ann-a* ‘father’
buul-a ‘mule (m)’
bonx-a ‘leaf / leaves’

Only on masculine nouns with a long final accusative vowel a morpheme *-ha* ACC / *-hu* NOM can occur optionally.

e.g. *ar-oo*(*-ha*) ‘husband’
hag-uu(*-ha*) ‘dry season’

3.3.2.4. NUMBER

A minimal Kambaata noun consists of a stem and a (primary) case morpheme. The stem can be a simple root or a root plus (a) derivational morpheme(s). Nouns inflect for the categories of case and gender. In Kambaata, number is marked, to different degrees, on common nouns, proper nouns and pronouns [60].

e.g. <u>Basic</u> <u>Singulative</u>

	kin-u	*kin-ch-u*	‘stone’
e.g.	Basic	Plurative	
	min-i	*min-n-a-ta*	‘houses’

Where, the terms “basic form”, “singulative”, and “plurative” are used with respect to the form of a noun. The “basic form” is the form of the noun which is formally unmarked for number. The singulative form is the basic form plus a singulative morpheme; the plurative form contains a plurative morpheme [19].

3.3.2.5. WORD FORMATION

Complex nouns are created by derivation, compounding and blending. Derivational patterns often change the word class of the base. Nouns can be derived from adjectives and verbs [19].

3.3.2.6. DERIVATION

Nouns can be derived from adjectives and verbs using suffixes (morphemes) [19].

TABLE 3-18: DE-ADJECTIVAL QUALITY NOUNS [19]

MORPHEME	ADJECTIVE /IDEOPHONES	TRANSLATION	DERIVED NOUN	TRANSLATION
-im-a-ta	*abb-a(-ta)*	big	*abb-im-a-ta*	being big, size
-inn-i(-ta)	*hodeem-a(-ta)*	pregnant	*hodeem-inni-ta*	pregnancy
-itt-a	*baqq y-*	wake up	*baqq-itt-a*	waking up

TABLE 3-19: DE-VERBAL NOUNS GENERATED THROUGH GEMINATION AND PALATALIZATION [19]

MORPHEME	VERB BASE	TRANSLATION	DERIVED NOUN	TRANSLATION
-ch-a	*hiir-am-*	be translated	*hiir-an-ch-a*	translation, meaning
-ishsh-a(-ta)	*odd-i(i)s-*	dress	*odd-i(i)shsh-a-ta*	clothes
-ishsh-a(-ta)	*ros-*	learn	*roshsh-a-ta*	lesson; learning
-a	*buuh-*	go moldy	*buuh-a*	mold
-a-ta	*wix-*	sow	*wix-a-ta*	grain
-í	*min-*	build	*min-i*	house
-í-ta	*ibiib-*	get lice	*ibiib-i-ta*	lice

3.3.2.7. REDUPLICATION

Complex nouns can be formed by reduplication, more specifically by full reduplication or by reduplication of (a part of) the first syllable. The reduplicated (part of the) syllable is prefixed.

e.g. *aleen →**al**-al-een*

*shiinaan →**shi**-shiin-aan*

3.3.2.8. COMPOUNDING AND BLENDING

COMPOUNDING

Kambaata makes little use of compounding to form complex words. Only very few compounds are found in Kambaata. However, in these few compounds, stems of various word classes may be combined to a single constituent. The compound is a single unit phonologically, because it has only a single accent [19].

e.g. ***min-i*** 'house'+ ***ann-a*** 'father' → ***min-ann-a*** 'head of the family, householder'

abb-a 'big' + ***am-a-ta*** 'mother' → ***abb-am-a-ta*** 'grandmother'

abb-a 'big' + ***ann-a*** 'father' → ***abb-ann-a*** 'grandfather'

BLENDING

As discussed by Yvonne [19], blending is another morphological tool to make replacements for new word for newly brought concepts. Blends contain either initial morpheme strings of two words or one word-initial and one word-final string.

For instance, when Biology textbooks were prepared initially, there were no appropriate translations for 'plant' and 'animal' in Kambaata. Thus, the terms given in example below were created. In both examples, the words of a smallest and a biggest species were combined into one word [19].

e.g. ***alg-od-a*** 'plant'← ***alg-ee*** 'alga' + ***od-ee-ta*** 'type of tree (Ficus sycomorus)'

amee-zan-a 'animal'← ***ameeb-a*** 'ameba' + ***zan-aa*** 'elephant'

hans-qaala'-a 'area of medium height' ← ***hansaww-a*** 'highlands' + ***qaala'-a*** 'lowlands'

3.3.3. ADJECTIVE MORPHOLOGY

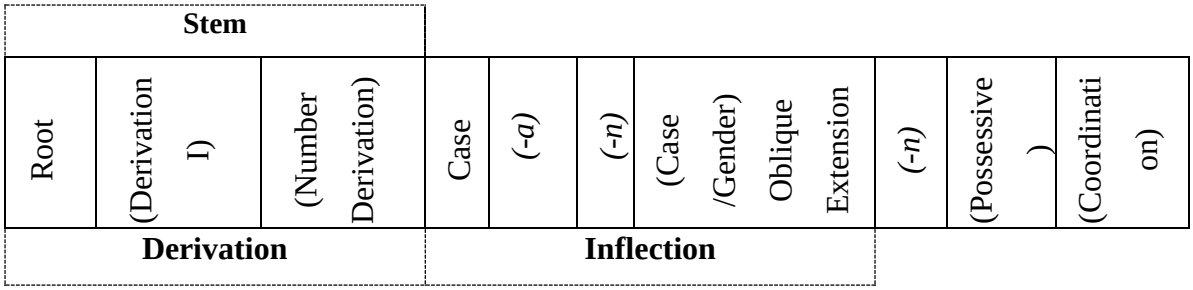
Kambaata is a language with a large open adjective class. Through derivation, an infinite number of adjectives can be created. Adjectives are defined in Kambaata as a class of

lexemes that display case and gender-agreement with a head if used as modifiers in a noun phrase [19]. The class of adjectives subdivides into true adjectives, demonstrative determiners and cardinal numerals, all of which distinguish two genders (masculine, feminine) and three cases (accusative, nominative, oblique) if used as modifiers in a noun phrase [52].

3.3.3.1. MORPHOLOGICAL STRUCTURE OF ADJECTIVES

The morphological structure of a maximally complex attributive adjective is given in Figure 3-2. Various compulsory and optional morphemes can be attached to the root. The root always ends in a consonant [19].

FIGURE 3-2: MORPHOLOGICAL STRUCTURE OF ATTRIBUTIVE ADJECTIVES [19]



The morphological structure of an independent adjective corresponds to that of a noun.

e.g. **Qeraa'rr-ut** maranch-at ees sagab-unta ass-itee'u.
long-F.NOM walk-F.NOM 1SG.ACC become: thirsty-1SG.PURP.DS do-3F.PVE
The long walk made me thirsty.

danaam-e ~ **danaam-i-ta** meent-ichch-oo
beautiful-F.OBL ~ beautiful-F.OBL-EXTENSION women-SG-F.DAT
to the beautiful woman
muccur-o ~ **muccur-u-a** wo'-aan
clean-M.OBL ~ clean-M.OBL-EXTENSION water-M.LOC
in the clean water

3.3.3.2. CASE AND GENDER INFLECTION

Attributive adjectives have a unique morphology that sets them apart from nouns; they distinguish between two genders (masculine and feminine) and three case forms (accusative, nominative, and oblique).

TABLE 3-20: ADJECTIVAL DECLENSIONS [19]

Declension		ACC	NOM	OBL
A1	M	*-a*	*-u*	*-a ~ -a-a*
	F	*-a-ta*	*-a-t*	*-a ~ -a-ta*
A2	M	*-u*	*-u*	*-o ~ -u-a*
	F	*-u-ta*	*-u-t*	*-o ~ -u-ta*
A3	M	*-u*	*-u*	*-o ~ -u-a*
	F	*-i-ta*	*-i-t*	*-e ~ -i-ta*
A4	M	*-a*	*-u*	*-a ~ -a-a*
	F	*-i-ta*	*-i-t*	*-e ~ -i-ta*
A5	M	*-oo*	*-oo*	*-oo ~ -oo-ha*
	F	*-oo-ta*	*-oo-t*	*-oo ~ -oo-taa*

A1: A1 is the largest declension of underived adjectives. In addition, it contains derived adjectives with the formatives *-aashsh-a*(*-ta*), *-eem-a*(*-ta*), and *-im-a*(*-ta*), compound adjectives, as well as plurative adjectives formed by the reduplication of the stem-final C.

The distinction between masculine and feminine gender is neutralized in the oblique form. Examples: *abb-a*(*-ta*) 'big', *biillaashsh-a*(*-ta*) 'light, easy'. When used as the head of an NP, A1 adjectives belong to the nominal declensions M1 or F1a, respectively (Table 3-16).

A2: A2 contains underived adjectives, e.g. *qeraa'rr-u*(*-ta*) 'long, high, tall', *qixx-u*(*- ta*) 'equal; fitting', as well as many derived adjectives, which are either formed with the highly productive agentive morpheme (*-aan*), e.g. *sal-aan-ch-u-ta* 'pregnant', or the fossilized -all morpheme, e.g. *duub-all-u*(*-ta*) 'rich'. The distinction between masculine and feminine gender is neutralized in the oblique form. When used as NP heads, A2 adjectives inflect like nouns of the declensions M3 or F3a, respectively (Table 3-16).

A3: A3 is characterized by a vowel change between the masculine and the feminine form. It contains an unlimited number of adjectives derived through the proprietive morpheme *-aam*, e.g. *dan-aam-u / dan-aam-i-ta* 'good, beautiful', *wo'- aam-u / wo'-aam-i-ta* 'watery, liquid, juicy'. As NP heads, feminine adjectives of A3 inflect like nouns of declension F2a, masculine adjectives like nouns of M3. Most numerals inflect like A3 adjectives.

A4: A4 contains only a single adjectival quantifier: *hoolam-a / hoolam-i-ta* 'many'. The adjective is characterized by a vowel change '*a'* vs. '*i'* between masculine and feminine

forms in the accusative case. As an NP head it receives the nominal morphology of the declension M1 or F2a. Some numerals inflect like the A4 adjective.

A5: A5 has (as of yet) only one example: *haar-oo*(*-ta*) 'new'. As an NP head it inflects like a noun of M8 or F6.

3.3.3.3. NUMBER MARKING

The description of number marking on adjectives meets with the same problems as number marking on nouns. Adjectives are pluralitized by geminating the stem-final consonant (if it is single) and by transfer into declension A1 (PL1 formation). If the adjectival stem ends in a consonant cluster, only the plurative 2 morpheme *-aakk-ata* is applicable.

e.g. *wiim-a* / *wiim-ata* → *wiim-**m**-ata*
full-M.ACC full-F.ACC full-PL1-F.ACC

ceemmallaashsh-a / ceemmallaashsh-ata → *ceemmallaashsh-**aakk**-ata*
lazy-M.ACC lazy-F.ACC lazy-PL2-F.ACC

3.3.3.4. WORD FORMATION

Complex adjectives can be formed through derivation, compounding, and reduplication. Simple adjectives share their stem with inchoative verbs. Complex adjectives are derived from simple adjectives, nouns, verbs, ideophones, and manner demonstratives.

3.3.3.5. DERIVATION

Adjectives can be derived from adjectives, nouns and verbs using suffixes [19].

TABLE 3-21: ADJECTIVES FORMED FROM NOUNS

MORPHEME	BASE NOUN	TRANSLATION	DERIVED ADJECTIVE	TRANSLATION
-aam-u / -i-ta	*godab-a*	stomach	*godab-aam-u / -i-ta*	greedy
-aam-u / -i-ta	*dan-a*	beauty, kindness	*dan-aam-u / -i-ta*	beautiful, good
-beel-u(-ta)	*wozan-a*	heart	*wozan-beel-u(-ta)*	heartless

TABLE 3-22: ADJECTIVES AND THEIR CORRESPONDING VERBS

MORPHEME	VERB	TRANSLATION	DERIVED ADJECTIVE	TRANSLATION
-a(-ta)	*abb-*	be(come) big	*abb-a(-ta)*	big; much, many

-a(-ta)	*annann-*	be(come) different	*annann-a(-ta)*	different
-aan-ch-u(-ta)	*ros-*	learn	*ros-aan-ch-u(-ta)*	student
-aan-ch-u(-ta)	*ros-is-*	teach	*ros-is-aan-ch-u(-ta)*	teacher
-im-a(-ta)	*hoog-*	be(come) tired	*hoog-im-a(-ta)*	weak, tired
-all-a(-ta)	*baab*	be(come) frightened	*baab-allu(-ta)*	frightened
-eem-a(-ta)	*biddiqq y-*	be(come) flat, even, be spread out	*biddiqq-eem-a(-ta)*	flat, even, spread out

3.3.3.6. REDUPLICATION

Adjectives are reduplicated in order to express 'each of the quality X'. As the examples in Table 3-23 illustrate, the reduplication is partial and also complete. In some cases, the adjective is fully reduplicated (e.g. *annann-u annann-u*). In other cases, the stem-initial C is additionally geminated (e.g. *qa-q-qahu*(*-ta*)) [19].

TABLE 3-23: REDUPLICATED ADJECTIVES [19]

NOTE	ADJECTIVE	TRANSLATION	REDUPLICATED FORM	TRANSLATION
CV-C-prefix	*qahu(-ta)*	small	*qa-q-qahu(-ta)*	small-each
Full	*annann-a(-ta)*	different	*annann-a(-ta) annann-aa(-ta)*	different each

3.3.3.7. COMPOUNDING

Complex adjectives could be made by the composition of an adjectival or numeral stem plus a nominal stem [19]. See example below.

e.g. ***mat-u*** 'one' + ***hagar-a*** 'type' → ***mathagar-a(-ta)*** 'of one type';
mat-i-ta 'one' + ***ill-i-ta*** 'eye' → ***matill-a(-ta)*** 'one-eyed'
mexx-u-ta 'single' + ***hagall-u-ta*** 'branch' → ***mexxagall-a(-ta)*** 'of the same type'
lam-u 'two' + ***fool-i*** 'soul' → ***lamfool-a(-ta)*** 'highly pregnant' (literally "two-souled")

3.4. CHALLENGE OF THE LANGAUGE FOR STEMMING

The main challenges in Kambaata for stemming words is its rich and complex morphology, i.e. words are formed making use of

- Multiple (concatenated) suffixes
- Irregularities through infixation, compounding, blending, and reduplication of affixes.

CHAPTER FOUR

DESIGN OF THE STEMMING ALGORITHM

4.1. INTRODUCTION

As discussed in the third chapter, Kambaata word formation process involves mainly suffixing. However, words can also be formed by infixing, reduplication and compounding in some cases. In developing IR systems and some other NLP applications for the language, reducing these kind of morphological variants into corresponding stem increases the retrieval performance [1]. This could be accomplished by a conflation technique, which is also known as stemming. The purpose of stemming algorithm is reducing various variants of words in to their common representation also referred to as stem. Therefore, the goal of the chapter is exploring stemming techniques and designing rules and developing a prototype stemmer for Kambaata language. Thus, the next sections present all these processes. The compilation of affixes, the algorithm design and evaluation of the stemmer has also been introduced as well.

4.2. THE CORPUS

Various text documents that contain the Kambaata words for the definition of rules and experimentation of the prototype stemmer has been compiled from Durame Senior Secondary and Preparatory School and KT Zone Education Department. The corpus that utilized for the identification and analysis of affixes and word formations contained 117,198 total word tokens with 26,731 distinct words. An additional corpus with 12,731 tokens containing 4,914 distinct words has been used to prepare a test data. The test data was collected from separate corpus to examine the algorithm from another corpus. Hence, the total corpus used for the algorithm design and testing of the stemmer is of 129,929 word tokens with 28,853 distinct words.

TABLE 4-1: WORD DISTRIBUTION RATIO OF SAMPLE KAMBAATA TEXT

Text	Total words	Distinct words	Word ratio
Corpus for affix identification	117,198	26,731	22.81%
Test set	12,731	4,914	38.60%
Total Corpus	129,929	28,853	22.21%

4.3. NORMALIZATION AND TOKENIZATION

Normalization and tokenization are essential data preprocessing steps in NLP applications like stemming. In the preprocessing stage, file formats, character sets, and variant forms are often transformed, so that all text, irrespective of its source, is in the identical format. Preprocessing must ensure that the source text be offered to stemming program in a form usable for it. As an example, stemmers normally require their input to be tokenized, i.e. text components, generally word forms or sentences are determined and positioned on distinct lines of the input [65]. In this step, this study addresses normalization and tokenization.

All punctuation marks with the exception of 'apostrophe' (**'**), which is used as glottal sound 'i' (e.g. "***asi'm***" 'look at") and also separate consecutively happening similar vowels in words (e.g. "***ga'aa***" 'tomorrow'), control characters, numbers and special characters are removed from the text before the data is processed. After all punctuation marks and special characters with the exception of (**'**) have been changed to spaces; space is marked as a word splitting up border. As a result, if a series of legitimate characters is followed by space, that sequence is determined as a word in tokenization process.

FIGURE 4-1: ALGORITHM FOR NORMALIZATION AND TOKENIZATION

```
Open the text file for processing
Create string container
Do
        Read the content of the file line by line and split to string by space
        Put to container for each strings
        For word in container and Special _characters= ".?/|\ @*=^&
        ()÷…¤¤¤```+_;:;" '!,#$% []{}<>-1234567890"
                If word contains Special _characters
                Replace them with a space
        End for
While end file
```

The above algorithm normalizes and tokenizes the document as follows. First, the text document is opened and the content of the file is read line by line. Second, split them by using space as a boundary in to a list of words. Third, if word within the list contains punctuation marks, control characters, numbers and special characters of Kambaata which

are defined as ***Special characters*** are found, replace them by space. This continues until end of line is reached.

4.4. COMPILATION OF AFFIXES

The known affixes of Kambaata language are suffixes, infixes, reduplication and compounding. In contrary to English stemmers that perform very effectively by removing suffixes along with prefixes to get the stems, an effective and powerful Kambaata stemmer not only able to remove suffixes, but also remove infixes and define context sensitive rules to transform irregular words to their stems as well. Without removing all these affixes, the stemmer cannot be effectively used to stem Kambaata documents.

Kambaata is a language that strongly depends on suffixation based on the reviewed literature [19], [54], [58] and that is backed by the researchers' linguistic knowledge as a user of the language as well. Morphemes that are used to represent a prefix and prefix-suffix pairs in other languages are all represented by a suffix in the case of Kambaata. If we consider an example from Amharic word "**ከ-ቤት**" 'from home', the prefix "**ከ-**" 'from' is prefixed to the word "**ቤት**" 'home'. Its equivalent in Kambaata is "***min-iichch***" where "***-iichch***" is the suffix which represents the Amharic prefix "**ከ-**". An additional illustration could be observed by looking at the Amharic word "**ከ-ቤት-ም**" 'also from home'. In this example, the prefix "**ከ-**" and the suffix "**-ም**" occurred at the same time creating prefix-suffix pair "**ከ-ም**". Its equivalent in Kambaata is "***min-iichch-ii***" where the suffix "***-iichch*** is equivalent to the prefix "**ከ-**" and the last suffix "***-ii***" represents the suffix "**-ም**".

We can also see the case of negation or opposite word formation by taking one example from Amharic word "**አት-ሥራ**" 'do not work', the prefix "**አት-**" 'do not' is prefixed to the word "**ሥራ**" 'work'. Its equivalent in Kambaata is "***hujat-t-oot***" where "***-t-oot***" is the suffix which represents the Amharic prefix "**አት-**".

Affix collection relies on the strategy or type the stemming technique chosen, longest match or iterative [7]. The longest match technique requires all forms of the affixes, simple suffix (a minimum form of suffix that cannot be decomposed into a lot more suffixes) and concatenated (a suffix formed by combination of two or more simple suffixes), for successful stemming. Whereas, the iterative method simply requires a list of basic affixes and removes them iteratively [17].

Suffixes concatenation is frequent in Kambaata words. Consequently, much more base suffixes can be combined with each other and attached to a word. Such combination is often extremely huge complicating the identification process of the full list of concatenations. Concatenation additionally makes suffixes lengthier by connecting one suffix to the other. Thus, collecting large data for affix analysis is regarded as the ideal choice to compile the biggest possible suffixes from Kambaata language texts to be able to utilize for the development of the stemmer. Therefore, 6299 unique suffixes (ranging from length of one character to twenty characters) and more than 300 exceptional word formations that require context sensitive and recoding rules have been identified in this study.

4.4.1. COMPILATION OF SUFFIXES

The suffix collection ranges from simple suffixes for instance "***ii***", "***ikke***", "***aan***", "***indo***" to concatenated suffixes for example "***anniichchisin***", "***eemmahanniichch***", "***iishshoomaantassa***". Table 4-2 shows some of the suffixes collected for the design of the algorithm. The complete list of suffixes identified is too much. Hence, it is difficult to present all the suffixes in this document. Therefore, randomly selected 14.23% of the whole suffixes (or 899 suffixes) are presented in Appendix I.

TABLE 4-2: SAMPLE SUFFIXES OF KAMBAATA

Basic suffixes	**Concatenated suffixes**
ii	aqqansiisseenata
een	isaanniichchissa
aan	oomaniichchissan
ba'a	iishshoomaantassa
ihaa	iteentahanniichch
ikke	nteentahanniichch
indo	aqqansiisaanchiichch
iqqi	ansiissaahaarranii
iyye	atteentahanniichch
ta	attoohanniichch
it	eenayyoommabe
oo	eemmahanniichch
ua	isaanchooha'nne
un	anniichchisin

4.4.2. COMPILATION OF INFIXES

Occasionally, "***-n-***" and "***-m-***" are infixed in Kambaata words [54]. Table 4-3 shows "***-n-***" infixation and the only known "***-uu-***" infix in the derivation of a verb "***xaaf***" 'write' to a noun "***xuuf***" 'writing'. There is also pragmatically known prefix "***ma-***" in the formation of a noun "***ma-xaaf-a***" 'book' from the same verb "***xaaf***" 'write' even though it's concluded in literatures that Kambaata is exclusively suffixing language [19], [58].

TABLE 4-3: INFIXATION IN KAMBAATA

Word	Infix	stem
a-n-**f**-oommi	n	af
aa-n-**g**-oommi	n	aag
da-n-**g**-umbudda	n	dag
ha-n-**s**-een	n	has
hoo-n-**g**it	n	hoog
huja-n-**t**-eemm	n	hujat
x-uu-**f**-iichcch	uu	xaaf
ma-**xaaf**-f-aachch		xaaf

4.5. THE RULES

4.5.1. CONTEXT SENSITIVE RULES

A context-free stemmer is one which eliminates strings without having any account of the leftover stem, and may therefore remove strings that are similar to, but which in fact are not legitimate affixes [7]. For instance, removing "un-" from "uniform" or "-al" from "meal" has this kind of behavior and so bad outcomes are acquired if context free approach is followed for Kambaata. For example, ***xaw-aaqqitoou*** 'they talked' ***aaqqitoou*** 'they took', have same word ending ***aaqqitoou*** which is in the list of suffix. Removing this ending string result in ***xaw,*** that is the correct stem for the first word. However, if we remove the same string from the second word ***aaqqitoou***, the stemmer will remove the word completely; hence we couldn't get the intended output where the correct stem for this case is ***aaqq*** 'take' even if the suffix ***itoou*** exists in the suffix list. According to the researcher's linguistic knowledge, there are many such scenarios in Kambaata. Therefore, the researcher decided that context-sensitive approach should be adopted, with the longest match suffix removal that is being controlled by two action codes and four conditions. The three types of context sensitive actions applied are:

Action 1 (A1): Don't perform affix removal

Action 2 (A2): Transform or substitute the word with others partly or completely as specified in the rules.

Action 3 (A3): Remove affixes.

The tree types of conditions applied are the following:

Condition 1 (C1): Check characters at the beginning of the stem and characters at the end of the suffix with the rules defined if they match. This is to avoid the removal of non-genuine affixes.

For example, if word starts with "g" and ends with "ntaa", replace "ntaa" with "m" (*giphpha-ntaa* 'oppose each other'→*giphpham* 'oppose') with the exception to words that start with "*gaan*" and "*gix*" as their respective stems are "*gaan*" and "*gix*".

FIGURE 4-2: ALGORITHM FOR CONDITION 1

```
if (word starts with "g" and word ends with "ntaa" and not
word starts with "gaan" and not word starts with "gix") {
    replace "ntaa" with "m";
    }
```

This algorithm is one typical example that replaces "ntaa" with "m" for a word "giphpha-ntaa" but not for words like "gaan-taa" and "gix-antaa". The action taken for this condition is Action 2 when the condition is satisfied.

Condition 2 (C2): For words with characters less than four but greater than one (or words with length of 2 or 3). These are words directly taken as output stems from test corpus if exist.

For example, a word "kei" and "tam" are both words and stems that exist in the test set. Since minimum stem length in Kambaata is 2 and most stems in Kambaata have length greater or equal 3, the stemmer is designed in a way it outputs such words without going through stemming process.

FIGURE 4-3: ALGORITHM FOR CONDITION 2

```
if (length of Word is 2 or 3) {
```

```
return Word as stem;
}
```

The action taken for this condition is Action 1 when the condition is satisfied.

Condition 3 (C3): A minimum stem length should be greater or equal to two characters. This is to maintain the minimum stem length of the word in the language. In Kambaata meaningful word stem or word has a minimum length of 2.

FIGURE 4-4: ALGORITHM FOR CONDITION 3

```
if ((length(WORD)-length(SUFFIX)>1) {
    Remove Suffix;
}
```

TABLE 4-4: CONTEXT SENSITIVE WORDS EXAMPLE 1

Word starts with	Followed by	Word end with	Stem	Condition
mann	∅	ann	mar ‘go’	C1,A2
mannee	-	-	mar ‘go’	C1,A2
mann	aam/ayy/un	-	mar ‘go’	C1,A2
mann	∅	nna	manch ‘human’	C1,A2
mann	aakkata/iichch	-	manch ‘human’	C1,A2
manni	-	-	manch ‘human’	C1,A2
manno	∅	omata	manch ‘human’	C1,A2

From Table 4-4 above and Table 4-5 below, we can see that word formation in Kambaata is complex and there is no easy way to remove morphological variants of words to form their respective stems. It clearly shows that the affixation in Kambaata words involves irregularities which makes a very complex situation for stemming. Hence it requires a detail analysis of Kambaata word formation for designing the algorithm.

TABLE 4-5: CONTEXT SENSITIVE WORDS EXAMPLE 2

Word ends with	Actual suffix	Substitution	Scenario	Stem
ntaa	ntaa	ntaa→Ø	ga'aa-ntaa	ga'aa
ntaa	ntaa	ntaa→m	ka-ntaa	kam
ntaa	ntaa	ntaa→m	wii-ntaa	wiim
antaa	antaa	antaa→Ø	gix-antaa	gix
assi	assi	assi→Ø	xoqq-assi	xoqq
assi	assi	assi→Ø	binn-assi	binn
assi	i	i→Ø	caqass-i	caqas
assi	i	i→Ø	ass-i	ass
aassi	i	i→Ø	k-aass-i	kaas
aassi	i	i→Ø	aass-i	aass
aassit	it	it→Ø	z-aass-it	zass
aassit	it	it→Ø	aass-it	aass
aaqqitoou	aaqqitoou	aaqqitoou→Ø	xaw-aaqqitoou	xaw
aaqqitoou	itoou	itoou→Ø	aaqq-itoou	aaqq
ittaa	ittaa	ittaa→Ø	saww-ittaa	saww
ittaa	aa	aa→Ø	itt-aa	it
aaqqiyye	aaqqiyye	aaqqiyye→Ø	xaw-aaqqiyye	xaw
aaqqiyye	iyye	iyye→Ø	aaqq-iyye	aaqq

4.5.2. RECODING RULES

Substitution rules are defined to handle some of the suffixes individually. As mentioned in challenges section at the end of this chapter, most of the rules in context sensitive and substitution part are defined for either individual words or work for few group of words.

FIGURE 4-5: ALGORITHM FOR SUBSTITUTION

if ((*word ends on* ***ccano|jjo) && (****if word starts with* ***xaa)) {***

replace ***ccan or jjo*** *by* ***z;***

}

For example, the pseudocode for the algorithm shows that for a word that starts with '***xaa***' and having '***ccano***' or '***jjo***' at its ending, will be recoded as follows:

Ending	**Replaced by**	**Word**	**Stem**	**Condition**
ccano →	z	xaa-ccano	xaaz	if word stars with '*xaa*'
jjo →	z	xaa-jjo	xaaz	

The following rules are examples and the full list is provided in Appendix II.

TABLE 4-6: EXAMPLE SUBSTITUTION RULES

Word	**Suffix**	**Substitution**	**Stem**	**Condition**
qora-mbun	-mbun	b	qorab	
xuu-jjoomm	-jjoomm	d	xuud	if word stars with 'xuu'
agu-xxantaau	-xxantaau	d	agud	
a-nfaamm	-nfaamm	f	af	
a-ngaamm	-ngaamm	g	ag	
bee-qqameemma	-qqameemma	h	beeh	
waa-nneemm	-nneemm	l	waal	
akee-nkeemm	-nkeemm	k	akeek	
giphpha-ntaassa	-ntaassa	m	giphpham	
ma-nnoommida	-nnoommida	r	mar	
ro-nsoomm	-nsoomm	s	ros	
i-chchaan	-chchaan	t	it	
i-nto	-nto	t	it	
huja-xxayyoo	-xxayyoo	t	hujat	
bii-cco	-cco	x	biix	
xaa-ccano	-ccano	z	xaaz	if word stars with 'xaa'
xaa-jjo	-jjo	z	xaaz	if word stars with 'xaa'
xaa-nzan	-nzan	z	xaaz	

4.5.3. SUFFIXES REMOVAL RULES

To cope with each and every suffix separately, 20 groups of suffix removal rules are designed. The rules begin with stemming the longest suffix first and the smallest suffix last together with other conditions. Because of the large number of the suffixes within each

rule, the researcher introduced just couple of suffixes for rules 5 to 6 and 8 to 18. Rules 1 to 4, 7, 19 and 20 are provided with full list of suffixes. The rules are introduced in Appendix III in pseudo-code.

4.6. THE PROPOSED STEMMING ALGORITHM

Figures 4-6 and 4-7 below present a rule based longest match and context sensitive algorithm for stemming Kambaata word variants. The minimal length for a meaningful Kambaata word/stem is two according to the researcher's own learning experience and observation from the corpus as well as textbooks and dictionary. In a given corpus of 4100 Kambaata word (which is taken from test set corpus for the analysis of this purpose), only 0.39% (16 words) have character length of 2 and 3. Therefore, the researcher has decided not to conflate words with length less than 4 in order to save the computational time of the algorithm. The stemmer will stem a word if the remaining stem after the suffix removal is a word having length greater or equal to 2, because most of the stems in the language are 3 or more characters long or have two or more radicals [19]. In Kambaata, verbal root necessarily ends in a single consonant or maximally two Cs (a consonant cluster) [19], [53]. Therefore, the researcher also considered such scenarios and defined rules accordingly. For the suffix removal component, rule has been defined to remove one consonant if the remaining stem length is greater than 4 after suffix removal and if it ends with double letter. However, double consonant at the end of the stem with length less than or equal to 4 character is not removed because the literature shows that there is a possibility of double consonant occurrence [19]; and such conditions usually happen in the shorter stems according to the linguistic knowledge of the researcher.

FIGURE 4-6: FLOW CHART FOR STEMMING PROCESS OF KAMBAATA STEMMER

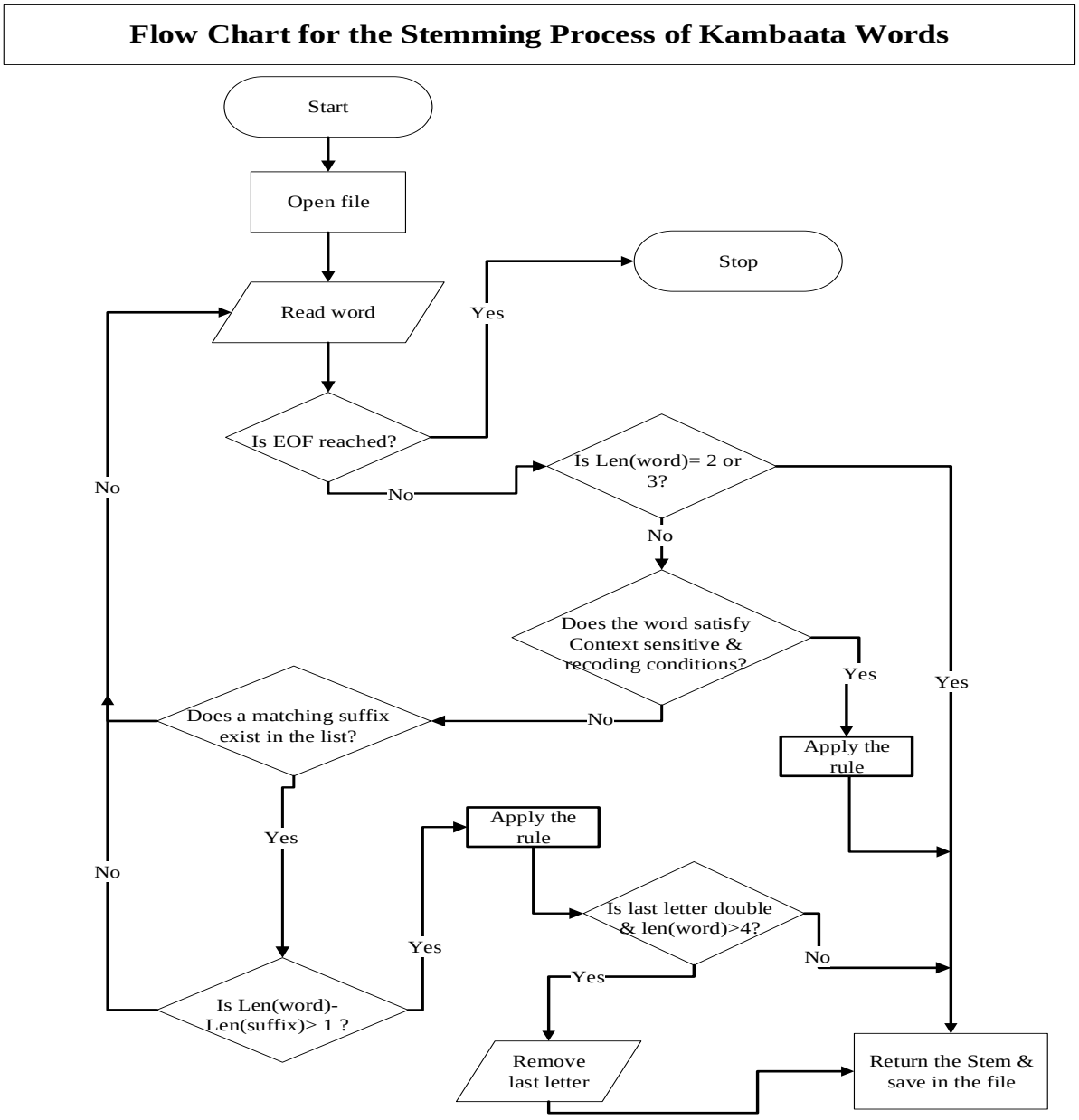


FIGURE 4-7: THE PROPOSED STEMMING ALGORITHM

While Not End of File (EOF)

Do

1. Get the WORD and measure the length(WORD) to be stemmed

2. IF length (WORD) = 2 or 3

Return WORD

ELSE

CONTINUE

3. IF the length(WORD)>=4

3.1. Determine the WORD beginning and ending list in the rules and Search a list of transformation for a match to the WORD being stemmed

IF a match found

Recode the WORD according to the rule and

Return STEM

ELSE

CONTINUE

3.2. Determine the SUFFIX and Search for the suffix in the ending list

IF a match found

IF (length(WORD)-length(SUFFIX)>1

Remove suffix

Return STEM

IF last letter of the remaining stem is double & length (WORD)> 4

Remove last letter

Return STEM

End for

End While

The algorithm works in the following way: first, a word file is opened and word read sequentially. If there is no next word or EOF is reached, the stemmer stops processing otherwise it continues. Next, the length of the word is measured and if the word has length of 2-3 characters, the word will be returned as a stem without going through the stemming

process. If not, it adheres the context sensitive and recoding procedures according to the specific rules provided, i.e. if the length of the word is greater or equal to 4. If the word satisfies the conditions, it will be transformed and recoded. Otherwise, word to be stemmed is provided to suffix removal rules lastly when they tend not to fulfill the conditions presented in the context sensitive and recoding rules. The suffix stemming rules range from one-letter suffixes to twenty-letter suffixes. The algorithm begins stemming the words from the longest twenty-letter suffixes to the shortest one-letter suffixes sequentially. The minimum stem length rule is defined under the suffix removal component and is checked before the suffix is completely removed. The stemmer also removes last double letter for stems with character length of greater than 4 after suffix removal and if the last letter is double. The procedure continues reading next word until EOF is reached. If EOF not reached, the process continues until all words are stemmed. The detail working procedure is explained with example in the subsequent section.

4.7. IMPLEMENTATION OF THE STEMMER

The algorithm's rules are implemented as a sequential program by using Python programming language. It is implemented in longest match technique and affixes are removed through the process of matching the input word ending to the list of affixes in the rules. The algorithm removes single and concatenated suffixes without iteration. The researcher has collected the affixes used in Kambaata to form different word variants. Using these affixes all possible combinations of the affixes are created and the correct ones are selected to form the rules.

The way in which the algorithm functions is described using illustration as follows. First, a word is given to the stemmer. Let the stemmer obtain the word "***rosisaanchiihanki'nne***" 'for your teacher'. In this word, we can get eleven meaningful suffixes "***-is***", "***-isa***", "***-isaa***", "***-isaanchii***", "***-isaanchiiha***", "***-isaanchiihanki'nne***", "***-iihanki'nne***", "***-anki'nne***", "***-'nne***", "***-nne***", and "***-e***". The stem is "***ros'***' learn'. After getting the word "***rosisaanchiihanki'nne***", the number of letters appearing in the word is counted. The word "***rosisaanchiihanki'nne***" has 21 letters which is not less than 4, the minimum required length for a word to be stemmed, and then the word passes to the next step. Next, the word beginning and ending is checked and goes through context sensitive and recoding rules, if a match exist, specific transformation rule is applied to the word. If the matching condition doesn't exist in the rules, the word passes to the next suffix removal component. Since we

have all the eleven suffixes mentioned above are kept in the suffix dictionary in the longest match suffix list; When reading suffix from the suffix list provided, the stemmer gets the suffix "***-isaanchiihanki'nne***". While scanning list of suffixes, it may also get suffixes like "***-iihanki'nne***", "***-anki'nne***". As the stemmer is longest match and context sensitive it checks the rules if any and then it immediately removes only "***-isaanchiihanki'nne***" resulting a word "***ros***" if the remaining stem has a length of more or equal to 2 characters. Since the remaining word has 3 letters it will pass to the next step. Finally, minor transformation such as last letter reduplication is removed if the length of the remaining word (stem) is greater than 4. In our case, "***ros***" has no reduplicated last word and also the remaining word length is not greater than 4. Therefore, "***ros***" is provided as the output of the stemmer.

4.8. EXPERMENTATION AND EVALUATION

The stemmer in this research is evaluated against correctness and dictionary reduction methods. Error counting strategy is employed to evaluate the algorithm with regards to the number of accurately stemmed results. The quantity of properly conflated words and erroneously conflated ones are counted for analysis. The output of the stemmer was examined in contrary to the respective anticipated linguistically valid stems. The errors were then described in terms of under stemming and over stemming. When a term is over-stemmed, an excessive amount of it is eliminated. Over-stemming can lead to unrelated words to be conflated. Under-stemming is a situation wherever very little of a word is eliminated. Under-stemming will prevent related terms from being conflated.

4.8.1. EXPERMENTATION OF THE STEMMER

To examine the overall performance of the stemmer, experiments have been carried out. The stemmer had been run on small sample data sets to check its effectiveness throughout implementation. Two separate experimentation data sets were taken out randomly from the corpus which was utilized for the affix analysis. The experimentation was conducted in two test sets having different data, where the first test set was comprised of 108 words and the second test set contained 102 distinct words. Table 4-7 sums up the result prior to the evaluation of the stemmer which is discussed on the subsequent sections. The purpose of this experimentation exercise is not to evaluate the stemmer rather it is to define more rules during design of the algorithm and this is the reason behind using small data sets.

TABLE 4-7: EXPERIMENTATION OF SAMPLE DATA

Experiments	Total distinct words	Correctly stemmed	Over stemmed	Under stemmed	Accuracy
Experiment 1	108	96	11	1	88.89%
Experiment 2	102	96	6	0	94.12%

The improved result in experiment two (94.12%) was the result of additional rules defined after the first experiment conducted which registered 88.89% accuracy. In the first experiment, out of 108 words, 10.19% (11 words) were over stemmed and 0.93% (1 word) was under stemmed. In the second experiment, all the errors, i.e. 5.88% (6 words out of 102 words) were generated by over stemming. The evaluation of the final stemmer on larger test sets is discussed in sections 4.8.2 and 4.8.3.

4.8.2. EVALUATION AND DISCUSSION OF THE FIRST STEMMER

After experimentation of the stemmer on sample data as discussed in section 4.8.1 above, 29 additional rules were defined to increase the performance of the stemmer on the test data during evaluation. The next section explains the results of evaluation of the stemmer.

4.8.2.1. THE RESULTS

The first stemmer was run in two various test sets of TS1 and TS2 having 1385 and 1040 distinct words which contain different words. The corpus from which the rules of the stemmer were developed is totally different from the test set. This was performed deliberately in order to see the efficiency of the stemmer in the real data. The summarized results of the first stemmer (first evaluation) are provided in Table 4-8.

TABLE 4-8: ACCURACY OF THE FIRST STEMMER

Test Set	Total Word Count	Correctly Stemmed Words	Over Stemmed Words	Under Stemmed Words	Stemmer Accuracy	Error Rate
TS1	1385	1302	76	7	94.01%	5.99%
TS2	1040	969	62	9	93.17%	6.83%
Combined	**2425**	**2271**	**138**	**16**	**93.65%**	**6.35%**

The combined output (TS1 + TS2) from the stemmer indicates that, out of 2425 words, 2271 (93.65%) words were stemmed correctly. 138 (5.69%) words were over stemmed and 16 (0.66%) words were under stemmed. Totally, this stemmer generated 154 (6.35%) words inaccurately stemmed words. As a result, the accuracy of the stemmer is 93.65% on the combined test set.

TABLE 4-9: SAMPLE OF STEMMED WORDS BY THE FIRST KAMBAATA STEMMER

Unstemmed Term	Expected Stem	Output of the Stemmer	Error Type
aagaqqancha	aag	aag	
aassantoossa	aass	aass	
afeesi	afoo	afoo	
affaau	af	af	
afuuIlleeii	afuuIll	afuuIll	
ageen	ag	ag	
agudaa	agud	agud	
akeekaakkata	akeek	akee	Over Stemmed
angassa	ang	ang	
assussa	ass	ass	
gantoosi	gan	gantoosi	Under Stemmed
gardabbaakka	gardab	gardab	
garegiin	gar	gar	
gooffeeu	goof	goof	
hanqafantaa	hanqaf	hanqaf	
hanxishshosiga	hanxishsh	hanxishsh	
haraaraga	haraar	haraa	Over Stemmed
hasammoru	has	has	
hawwa	haww	haww	
hujachchessa	hujat	hujat	
ilteeu	il	ilt	Under Stemmed

TABLE 4-10: EXAMPLES OF WRONGLY CONFLATED TERMS BY THE FIRST VERSION OF KAMBAATA STEMMER (OBTAINED FROM THE TS1)

Unstemmed Term	Expected Stem	Output of the First Stemmer	First Stemmer Error Type
hegeegissataa	hegeeg	hegee	Over stemmed
higano	hig	hi	Over stemmed
hogobo	hogob	hog	Over stemmed

honso	hons	hon	Over stemmed
hoolchutaans	hoolch	hoolc	Over stemmed
inkiilantee	inkiil	inkii	Over stemmed
jaallaaakkaahaansa	jaal	jaall	Under stemmed
kajjelano	kajjel	kajje	Over stemmed
keeImmii	keeIm	keeI	Over stemmed
kullessa	kul	kull	Under stemmed
kuushebiihaa	kuush	kuus	Over stemmed
lallabaanchoomaantas	lallab	lalla	Over stemmed
lallat	lal	lall	Under stemmed
leinu	lei	le	Over stemmed
lokkaan	lokk	lo	Over stemmed
maanata	maan	ma	Over stemmed
maarka	maark	maar	Over stemmed
maccooccisiishshessa	maccoocc	maccoocc	Under stemmed
maleeshshata	malees	mal	Over stemmed
maru	mar	ma	Over stemmed

4.8.2.2. WORD COMPRESSION

The stemmer is as well evaluated in terms of word compression rate. For determining the word compression rate (C), or reduction of dictionary is calculated using the formula [66]:

$$C = 100 * (W - S)/W$$

Where,

C - is the compression value (in percentage)

W - is the number of the total words

S - is a distinct stem after conflation

TABLE 4-11: WORD COMPRESSION RATIO OF TOTAL WORDS

Test set	Stem (S)	Word (W)	Compression Ratio (C)
TS1	448	1385	67.65%
TS2	394	1040	62.12%
Overall	842	**2425**	**65.28%**

The percentage of compression for Kambaata text based on the test set text for this first stemmer becomes 100 * (2425- 842) / 2425 = **65.28%.**

TABLE 4-12: WORD COMPRESSION RATIO OF CORRECTLY STEMMED WORDS

Test set	Correct Stem (S)	Correctly Stemmed Word (W)	Compression Ratio (C)
TS1	388	1302	70.20%
TS2	336	969	65.33%
Overall	**724**	**2271**	**68.12%**

The percentage of compression for correctly stemmed words for this first stemmer becomes 100 * (2271- 724) / 2271 = **68.12%**.

From this result, we can observe that the stemmer can reduce the Kambaata morphological variants of words and reduce the size of the file by 68.12% which is very significant reduction.

4.8.2.3. PROBLEMS OBSERVED

Reasons for the observed problems in the first stemmer are:

i. Due to the complexity of the morphological variations in Kambaata words, it is difficult to define a rule that works for all or for some group of words. For example, suffix "***ntaa***" which is recoded as "***m***" for some words that start with letter "***g***" in Kambaata, does not work for words that start with "***gaan***" and "***gix***" even though both words start with the same letter "***g***".
ii. It was difficult to come up with the complete affixes and/or rules because of the complexity of the language. More conditions/rules are required based on detailed study of the morphology of the language. See examples in Section 4.5.1.1. Recoding and Context Sensitive Rules.
iii. Some suffixes which perfectly fit for some words cause over stemming and under stemming errors for the other words. For example, a suffix "***-eeda***" is a genuine suffix for a word "***waalleeda***" 'if he came' where the stem is "***waal***" 'come'. However, if we have a word "***bareeda***" 'is good' the suffix "***-eeda***" cause over stemming error for this word because the stem for "***bareeda***" is "***bareed***" 'good' but not "***bar***" which a root word for 'day'.
iv. Irregular words require special rules to handle their affixation.
v. Disagreement of minimum stem length controlling rules with the suffixes will also generate errors, i.e., it causes the stemmer to generate unrelated output to the input

word. For example, "***aaqqitoou***" 'they took' has stem "***aaqq***" 'take'. But the suffix "***aaqqitoou***" also exists in the suffix list for words such as "**xaw-*aaqqitoou***" "they talked" where the stem is "***xaw***" 'talk'. Unless we define a special rule that removes only "***itoou***" to handle word "***aaqqitoou***", it will create confusion for the longest match suffix removal stemmer to remove "***aaqqitoou***" from words "***aaqqitoou***". Therefore, minimum stem length controlling rules have been defined in the suffix removal part that checks the length of the remaining stem during suffix is removal. Hence unless special condition is defined for the word "***aaqqitoou***", the controlling function pushes the stemmer generate wrong output.

4.8.3. EVALUATION AND DISCUSSION OF THE IMPROVED STEMMER

To resolve the issues determined on the first version of the stemmer, the following minor enhancements were introduced to the algorithm.

1) To solve the problem described under section 4.8.2.3(i), twenty-three more conditions and rules were added. Due to the complex word formation in the language, it is difficult to find generic rules.
2) To solve the problem described under section 4.8.2.3(ii), more suffixes which were not previously available in the suffix list are included manually. For example, "***achchiichchin***", "***achchuna***", "***eennossagiihaa***", "***igiihaa***" and "***aawwisu***" were added to the suffix list.
3) To solve the problem described under section 4.8.2.3(iii), some suffixes which are previously available in the suffix list are removed and instead other rules were defined. For example, "***ccii***", "***eeda***", "***innitii***" and "***innita***" were removed from the suffix list.
4) To solve the problem described under section 4.8.2.3(iv), some context sensitive and substitution rules were defined.

4.8.3.1. THE RESULTS

The stemming procedure of the improved stemmer is identical to the first stemmer; the difference is the number of rules added, affixes added and removed, and conditions defined to the latest stemmer. The improved stemmer output in comparison with the first version is shown in Table 4-13.

TABLE 4-13: COMPARISON OF PERFORMANCE BETWEEN THE FIRST AND THE IMPROVED STEMMER

Un-stemmed Term	Expected Stem	Output of the First Stemmer	First Stemmer Error Type	Output of the Improved Stemmer	Error after Improvement
hegeegissataa	hegeeg	hegee	Over Stemmed	hegeeg	No error
higano	hig	hi	Over Stemmed	hi	Over Stemmed
hogobo	hogob	hog	Over Stemmed	hogob	No error
honso	hons	hon	Over Stemmed	hon	Over Stemmed
hoolchutaans	hoolch	hoolc	Over Stemmed	hoolch	No error
inkiilantee	inkiil	inkii	Over Stemmed	inkiil	No error
jaallaaakkaahaansa	jaal	jaall	Under Stemmed	jaall	Under Stemmed
kajjelano	kajjel	kajje	Over Stemmed	kajje	Over Stemmed
keeImmii	keeIm	keeI	Over Stemmed	keeI	Over Stemmed
kullessa	kul	kull	Under Stemmed	kull	Under Stemmed
kuushebiihaa	kuush	kuus	Over Stemmed	kuush	No error
lallabaanchoomaantas	lallab	lalla	Over Stemmed	lallab	No error
lallat	lal	lall	Under Stemmed	lall	Under Stemmed
leinu	lei	le	Over Stemmed	lei	No error
lokkaan	lokk	lo	Over Stemmed	lo	Over Stemmed
maanata	maan	ma	Over Stemmed	ma	Over Stemmed
maarka	maark	maar	Over Stemmed	maar	Over Stemmed
maccooccisiishshessa	maccoocc	maccoocc	Under Stemmed	maccoocc	Under Stemmed
maleeshshata	malees	mal	Over Stemmed	malees	No error
maru	mar	ma	Over Stemmed	ma	Over Stemmed

After improvement have been done, the new stemmer was evaluated on the same set of test data (TS1 with 1385 words and TS2 with 1040 words) that was used for the first version of Kambaata stemmer discussed in section 4.8.2. This was done in order to see the effect of the improvements on the performance of the stemmer. Accordingly, the percentage and number of over stemmed and under stemmed words were reduced to 2.60% (63 words) and

0.54% (13 words) respectively. The total errors account for 3.13% (76 words) and the performance of the stemmer is enhanced to 96.87%.

TABLE 4-14: ACCURACY OF THE IMPROVED STEMMER

Test Set	Total Word Count	Correctly Stemmed	Stemmer Accuracy
TS1	1385	1344	97.04%
TS2	1040	1005	96.63%
Combined	2425	2349	**96.87%**

Table 4-15 summarizes the overall enhancement in comparison with the first version. The partial input test data from TS2 for this stemmer is given in Appendix IV and the corresponding output of the stemmer is provided in Appendix V.

TABLE 4-15: PERFORMANCE COMPARISON OF THE FIRST VS. IMPROVED STEMMER

Stemmer Version	Over Stemmed		Under Stemmed		Total Errors		Correctly Stemmed Words	Total Word	Performance (Accuracy)
	Words	pct.	Words	pct.	Words	pct.			pct.
First Stemmer	138	5.69%	16	0.66%	154	6.35%	2271	2425	**93.65%**
Improved Stemmer	63	2.60%	13	0.54%	76	3.13%	2349	2425	**96.87%**

4.8.3.2. WORD COMPRESSION

As provided in Table 4-16, in terms of dictionary size, the compression for the whole test data becomes 100 * (2425- 828) / 2425 = **65.86%**. This result also shows the compression of the dictionary size is increased by 0.4%.

The compression for correctly stemmed words becomes 100 * (2349-763) / 2349 = **67.52%**. However, we could not get better compression ratio for this category which is decreased by 0.6%.

TABLE 4-16: WORD COMPRESSION RATIO

Test set	Stem (S)	Word (W)	Compression Ratio (C)	Correct Stem (S)	Correctly Stemmed Word (W)	Compression Ratio of Correct (C)
TS1	440	1385	68.23%	406	1344	69.79%
TS2	388	1040	62.69%	357	1005	64.48%
Overall	828	**2425**	**65.86%**	763	2349	**67.52%**

4.8.3.3. PROGRAM EXCUTION TIME

To determine the execution time of the algorithm, the Python built in ***timeit.default_timer()*** function from ***timeit*** package is used. The function starts counting when the stemmer starts, and stops when the stemmer finishes execution. The program calculates the difference between the initial and completion time and outputs the duration in seconds. The test is run on Core i5 laptop computer that has 2.6 GHz processing unit, 4 GB memory and 500 GB hard disk. Accordingly, it took 4.13 seconds to conflate 1385 words in test set one and it took 3.208 seconds to conflate 1040 words in test set 2. Hence, the program is efficient and faster. Multiple programs/processes running on the machine during the execution might interfere and may elapse more time. During this program execution, only MS Office applications (Excel, Word), Python IDLE Shell editor were running.

In conclusion, reasons for under stemming and over stemming are:

1) It absolutely was hard to bring the full list of suffixes mainly because of the rich morphological behavior of the Kambaata language.
2) It has been challenging to define comprehensive list of context sensitive and recoding rules. More conditions/rules are required based on a further study of the morphology of the language.
3) Number of loan words such as ***forograammata*** (programme), ***tiraatiri*** (theater) and ***aksuumaakka*** (Aksumite) are not conflated correctly.
4) Words with spelling errors usually result in incorrect results.

4.9. FINDING OF THE STUDY

In this study, a context-sensitive longest match stemmer is designed using rule based approach for stemming Kambaata text and it conflates word variants to their respective root. Sample test set of 2425 words were used as a test set to evaluate the performance of the stemmer and the result showed 96.87% accuracy for the improved final stemmer. A dictionary reduction of 67.52% is also achieved for correctly stemmed words. The next chapter discusses conclusions of findings and recommendations for future research.

CHAPTER FIVE
CONCLUSION AND RECOMMENDATION

5.1. CONCLUSION

Analysis of the morphology of Kambaata words presented in Appendix VI and VII and literature review reveals that the language is rich morphologically. The types of affixations such as suffixes, infixes, reduplication and compounding and concatenation of suffixes in the language contribute a lot in generating rich morphological variants and make the word formation process complicated in Kambaata.

This is why attempting to conflate Kambaata words manually is very tedious and extremely difficult. For this reason, applying automated conflation procedure such as stemmer is extremely important in NLP applications.

In this study, the researcher employed rule based stemming approach with longest much technique. The stemmer has 20 groups of suffix removal rules that remove more than 6299 suffixes with longest match first remove fashion. The stemmer is also context sensitive with 255 context sensitive and recoding rules defined in it. So as to apply the longest match technique, all possible long suffixes (whole list of concatenated suffixes with in the corpus) in addition to the basic suffixes were collected from large corpus of 117,198 words with 26,731 distinct words.

This stemming algorithm does not just remove suffixes, but also takes exceptional scenarios into consideration and stems them by applying substitution and context sensitive rules. The suffix removal and substitution and context sensitive rules are utilized only if specific circumstances are satisfied, e.g. the resulting stem must have a certain minimal length. The majority of rules have got a condition depending on the length of the remaining stem. The length is the number of characters or letters (either vowel or consonant) that exist in the resulting stem, where each letter is counted as one. These conditions have to stop removing of characters which appear to be a legitimate suffix but are portion of the stem. The existence of these rules makes the stemmer a context sensitive. The contexts are discovered based upon the comprehensive study of the morphological behavior of Kambaata language.

From the evaluation carried out on the selected test sets of the study, it can be demonstrated that the algorithm stems words with an accuracy of 96.87% with an error rate of 3.13%.

- It has been noticed that good performance cannot be attained with only suffix removal in Kambaata language.
- The explored rule-based stemmer is both precise by stemming words effectively with accuracy of 96.87% and very fast by stemming 330 words per second in average.
- The improved stemmer stems words of separate data sets with nearly comparable performance. i.e. 97.04% accuracy registered on test set one (TS1) and 96.63% accuracy registered on test set two (TS2), the overall accuracy being 96.87%. Nevertheless, this result may possibly change on various data sets of other domain.
- The main challenges in Kambaata for stemming words is its rich and complex morphology, i.e. words are formed making use of multiple (concatenated) suffixes, irregularities through infixation, compounding, blending, and reduplication of affixes.
- The most difficult thing in exploring stemming rules for Kambaata words is discovering a rule which works efficiently for most or several set of words. Generally, there is usually exception whenever one rule is defined for a number of group of words.
- The other challenge is that the language is a little explored with regards to its linguistic feature, most importantly, its morphology. The morphology of Kambaata is not well studied in the linguistic discipline. Thus, the researcher could not find enough linguistic resources except Dr. Treis Yvonne's research works.

5.2. RECOMMENDATION

The researcher believes this study as a big step towards finding algorithm for stemming Kambaata words for different NLP applications. This research is believed to be a baseline for future scientific studies in the field of NLP in particular and Computer Science and Information Sciences in broad sense.

This research work could possibly have downsides and even more enhancements might be needed to improve the algorithm's accuracy and the efficiency of the stemming process. As observed in the study process, errors produced can mostly be improved possibly by identifying a lot more context sensitive and substitution rules and/or incorporating more suffixes to the suffix dictionary. In some cases, removing suffixes that adversely affect some words may also improve the error rate for the other group of words.

A stemming algorithm design is attempted for stemming Kambaata words with the exception of the reduplicated and compound words. A considerable move for future enhancement of Kambaata words stemming algorithm could be a study on stemming techniques for reduplicated and compound words.

The researcher recommended potential open research areas in one or combination of the following areas.

- The stemmer can be one components for designing other NLP tools like morphological analyzer, machine translation, word frequency counter, and automatic text summarizers for Kambaata language.
- Researches or research projects could be performed on building Kambaata information retrieval system making use of this stemming algorithm to access large amount of Kambaata words and observe its impact over recall and precision.
- It would also be really fascinating to explore stemming algorithm applying table lookup and statistical conflation techniques and notice how it performs for Kambaata language as an alternative stemming technique.
- NLP applications need standard corpus preparation. Hence, preparing the standard corpus for Kambaata NLP researches could also be another research opportunity in the field.

REFERENCES

[1] J. B. Lovins, "Development of a stemming algorithm," *Mechanical Translation and Computational Linguistics*, vol. 11, no. 1 and 2, 1968.

[2] D. Sharma, "Stemming Algorithms: A Comparative Study and their Analysis," *International Journal of Applied Information Systems*, vol. 4, no. 3, pp. 1-6, 2012.

[3] W. B. Frakes, "Stemming algorithms. In Frakes," in *Information retrieval: data structures and algorithms*: Prentice-Hall, 1992, pp. 131-160.

[4] G. Salton, "Introduction to Modern Information Retrieval," McGraw-Hill, 1983.

[5] R. Baeza-Yates and B. Ribeiro-Neto, *Modern information retrieval*, New York: Addison Wesley, 2011.

[6] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130-137, 1980.

[7] N. Alemayehu and P. Willet, "Stemming of Amharic Words for Information Retrieval," *Literary and Linguistic Computing*, vol. 17, no. 1, pp. 1-17, 2002.

[8] B. O. Alijla, "Stemming in Natural Language," Faculty seminar, Department of Information Technology System, Faculty of Information Technology, The Islamic University of Gaza, 2009.

[9] J. A. Hugh, E. Williams and S.M.M Tahaghoghi, "Stemming Indonesian language," *28th Australian Computer Science Conference (ACSC), Conferences in Research and Practice in Information Technology*, vol.38. pp.1-8, 2005.

[10] H. Harmanani, W. Keirous and S. Raheel, "A Rule-Based Extensible Stemmer for Information Retrieval Application to Arabic," *The International of Arab Journal of information Technology*, vol.3, no.3, pp.265-271, 2006.

[11] G. Salton, *Automatic text processing: The Transformation, Analysis, and Retrieval of Information by Computer*, 1st ed. Reading, Mass. [etc.]: Addison-Wesley, 1989.

[12] G. Berhe, "A Stemming algorithm development for Tigrigna language text documents", Master's Thesis, Addis Ababa University, Addis Ababa, 2001, unpublished.

[13] P. Birungi, "Improved Strategies for Employment and Human Resources Utilization in the Information and Documentation Sector," *Strategies for Human Resources Development for information management in Africa*, Ed. 49-57, Addis Ababa: UNECA, PADIS, 1995.

[14] A. Alemu and L. Asker, "An Amharic Stemmer: Reducing Words to their Citation Forms," *The Association for Computational Linguistics*, Prague, Czech Republic, June 2007.

[15] D. Tesfaye, and E. Abebe, "Designing a Rule Based Stemmer for Afaan Oromo Text," *International journal of computational linguistics (IJCL)*, vol. ED-1 (2), October 2010.

[16] Y. Fisseha, "Development of Stemming Algorism for Tigrigna Text," Master's Thesis, Addis Ababa University, Addis Ababa, June 2011, unpublished.

[17] L. Lessa, "Development of stemming algorithm for Wolaytta text," Master's Thesis, Addis Ababa University, Addis Ababa, July 2003, unpublished.

[18] M. Kedir, "Designing a Stemming Algorithm for Silt'e Language," Master's Thesis, Addis Ababa University, Addis Ababa, June 2012, unpublished.

[19] Y. Treis, *A grammar of Kambaata (Ethiopia), Part I: Phonology, Nominal Morphology and Non-verbal Predication*, 1st ed. Köln: Rüdiger Köppe, 2008.

[20] Y. Treis, "Relativization in Kambaata from a typological point of view," *In: Zygmunt Frajzyngier and Erin Shay (eds.), Interaction of morphology and syntax: Case studies in Afroasiatic*, pp. 161-206, Amsterdam/Philadelphia: Benjamins. 2008b.

[21] K. Kawachi, *A grammar of Sidaama (Sidamo), a Cushitic language of Ethiopia*, 1st ed. Buffalo, NY: State University of New York, 2007.

[22] "Ethnologue: Languages of the World," *Ethnologue*, 2017. [Online]. Available: https://www.ethnologue.com.country/ET [Accessed: 12- May- 2017].

[23] J. Project, "Language – Kambaata, Joshua Project," *Joshuaproject.net*, 2017. [Online]. Available: https://joshuaproject.net/languages/ktb. [Accessed: 11- Dec- 2017].

[24] "Languages and Cultures of Sub-Saharan Africa," 2017. [Online]. Available: http://llacan.vjf.cnrs.fr/index_en.php. [Accessed: 12- May- 2017].

[25] M. Hafer and S. Weiss, "Word Segmentation by Letter Successor Varieties," *Information Storage and Retrieval*, vol. 10, no. 371-385, 1974.

[26] J. Dawson, "Suffix removal for word conflation," *In Bulletin of the Association for Literary and Linguistics computing*, vol. 2, No. 3, pp. 33-46, 1974.

[27] C. D. Paice, "Another stemmer," *ACM SIGIR Forum*, vol. 24, no. 3, pp. 56-61, 1990.

[28] G. Adamson and J. Boreham, "The Use of an Association Measure Based on Character Structure to Identify Semantically Related Pairs of Words and Document Titles," *Information Storage and Retrieval*, vol. 10, pp. 253-260, 1974.

[29] M. Wakshum, "Development of Stemming Algorithm for Afaan Oromo Text," M. Sc. Theses, Addis Ababa University, 2000, unpublished.

[30] J. Savoy, "Stemming of French Words Based on Grammatical Categories," *Journal of American Society for Information Science*, vol. 44, no. 1, pp. 1-9, 1993.

[31] P. Willett, "Automatic indexing of documents and queries," *Document retrieval systems*, vol. 21, no. 2, pp. 4-9, 1988.

[32] S. Khoja and R. Garside, "Stemming Arabic text," Computing department, Lancaster University, Lancaster, 1999.

[33] Y. Treis, "Kambaatissata: Yanna, maauta, heessa [The Kambaata language: proverbs, tales and legends]," *La Trobe University: Research Centre for Linguistic Typology*, 2008.

[34] S. Walker & R. M. Jones, "Improving subject retrieval in online catalogues," in *Stemming, automatic spelling correction and cross reference tables*, London: The Polytechnic of Central London, pp. 2, 1987.

[35] M. P. Lennon, D. Tarry, and P. Willett, "An evaluation of conflation algorithms for information retrieval," *Journal of Information Science*, vol. 3, pp. 177-183, 1981.

[36] C. D. Paice, "An evaluation method for stemming algorithms," *In Croft and van Rijsbergen*, pp. 42-50.

[37] T. Tewodros, "Word formation in Tigrinya" M.Sc. Theses, Addis Ababa University, 1993, unpublished.

[38] W. Frakes, R. Baeza-Yates, "Information Retrieval: Data Structures and Algorithms," *NJ: Prentice-hall*, 1992.

[39] J. Mayfield and P. McNamee, "Single N-gram stemming," *In proceedings of the 26th Annual International ACM SIGIR conference on research and development in information retrieval*, pp. 415-416, Toronto, Canada, ACM Press, 2003.

[40] M. Melucci, Introduction to Modern Information Retrieval, New York, 2003.

[41] G. Salton, H. Wu and C. Yu, "The measurement of term importance in automatic indexing," *Journal of the American Society for Information Science*, vol. 32, no. 3, pp. 175-186, 1981.

[42] J. Xu and W. Croft, "Corpus-based stemming using co-occurrence of word variants," *ACM Transactions on Information Systems*, vol. 16, no. 1, pp. 61-81, 1998.

[43] M. Popovic and P. Willett, "The Effectiveness of Stemming for Natural-Language Access to Slovene Textual Data," In *Journal of American Society for Information Science*, vol. 43, no. 5, pp. 384 -390, 1992.

[44] T. Z. Kalamboukis, "Suffix stripping with Modern Greek," *Program*, vol. 29, no. 3, pp. 313-321, 1995.

[45] R. Krovetz, "Viewing Morphology as an inference process," *In proceedings of the 16th Annual International ACM SIGIR conference on research and development in information retrieval*, pp. 191-202, ACM New York, 1993.

[46] C. Paice, "Method for evaluation of stemming algorithms based on error counting," *Journal of the American Society for Information Science*, vol. 47, no. 8, pp. 632-649, 1996.

[47] A. Spencer, "Morphological Theory: An introduction to word structure in generative grammar," *Oxford: Blackwell Publishers*, 1991.

[48] Gregory T. Stump, "Inflectional and Derivational morphology," Kentucky University, USA*: Cambridge University Press*, 2001.

[49] A. Banta, "Kambaatissa-Amharic-English Dictionary," KT Zone Education Bureau, Durame, Ethiopia, 2016.

[50] Y. Treis, "Kambaata Numerals and Denumerals Revisited," LLACAN.

[51] Y. Treis, "Motion events in Kambaata," *Annual Publications in African Linguistics*, vol. 5, pp.197-226, 2007.

[52] Y. Treis, "They are only two, like the teats of a donkey: Kambaata Denumerals Revisited," *Mechthildian Approaches to Afrikanistik: Advances in language based research on Africa - Festschrift fur Mechthild Reh, Rudiger Koppe*, pp.339-366, 2017.

[53] Y. Treis, "Past tense in Kambaata (Cushitic)," In *Workshop on the meaning of past tense morphology*, 2016.

[54] Y. Treis, "Categorial hybrids in Kambaata," *Journal of African Languages and Linguistics*, De Gruyter, pp. 215-254, 2012.

[55] Y. Treis, "The Perfective Paradigms of Kambaata: Double Agreement, Defectiveness and Overlaps," *19th International Conference of Ethiopian Studies*, Warsaw, 24-28 August 2015.

[56] Y. Treis, "Future in an aspect-marking language: The example of Kambaata," *Atelier sur le future (Operation "Le temps dans les langues africaines") CNRS-LLACAN* Villejuif, May 2011.

[57] Y. Treis, "Purpose-encoding strategies in Kambaata," *Afrika und Übersee*, vol. 91, pp.1-38, 2010.

[58] Y. Treis, “Expressing future time reference in Kambaata,” *Nordic Journal of African Studies*, vol. 20, no. 2, pp.132-149, 2012.

[59] Y. Treis, “Negation in Highland East Cushitic,” *Burning Issues in Afro-Asiatic Linguistics*, Newcastle upon Tyne: Cambridge Scholars Publishing, pp. 20-61, 2012.

[60] Y. Treis, “Number in Kambaata: A category between inflection and derivation,” LLACAN.

[61] Y. Treis, “Form and Function of Case Marking in Kambaata,” *Afrikanistik online*, 2006.

[62] B. Yimam, “Ye amargna sewasew (Amharic Grammar),” *EMPDA*, 1995.

[63] L. Galambos, “Lemmatizer for Document Information Retrieval Systems in JAVA,” *SOFSEM 2001: Theory and Practice of Informatics*, pp. 243-252, 2001.

[64] M. Lennon, D. Peirce, B. Tarry and P. Willett, “An evaluation of some conflation algorithms for information retrieval,” *Information Scientist*, vol. 3, no. 4, pp. 177-183, 1981.

[65] R. M. Kaplan, “A method for tokenizing text,” *Inquiries into words, constraints and contexts*, pp. 55, 2005.

[66] D. Harman, “How effective is suffixing?” *Journal of the American Society for Information Science*, vol. 42, no. 1, pp. 7-15, 1991.

[67] “The history of Ethiopian national literacy campaign,” *Ethiopian History*, 2017. [Online]. Available: http://www.ethiohistory.com. [Accessed: 26- Nov- 2017].

[68] “Translation-works”, *Bible Society of Ethiopia*, 2017. [Online]. Available: http://biblesociety-ethiopia.org. [Accessed: 26- Nov- 2017].

[69] Md. Islam, Md. Uddin and M. Khan, “A Light Weight Stemmer for Bengali and Its Use in Spelling Checker,” Center for Research on Bangla Language Processing, BRAC University, Dhaka, Bangladesh.

[70] V. Vaishnavi, W. Kuechler, and S. Petter, "Design Science Research in Information Systems," 20 12 2017. [Online]. Available: http://www.desrist.org/design-research-in-information-systems/. [Accessed 09 03 2018].

[71] S. Srinivasan, ZP. Thambidurai, “STANS Algorithm for Root Word Stemming,” *Information Technology Journal*, vol. 5, no. 4, pp. 685-688, 2006.

APPENDIXES

APPENDIX I: SUFFIXES COMPILED FOR THE STEMMER

Total suffixes compiled for the development of Stemmer for Kambaata are 6299. Here, randomly selected 899 suffixes (14.23%) are presented.

aabeechchii
aab’a
aachchinne
aada
aagan
aagu
aahaanki’nne
aahanki'nne
aahansa
aaii
aaindoo
aakas
aakka
aakkaachchina
aakkaahaansa
aakkaahas
aakkaaneet
aakkaantassan
aakkaatanki’nne
aakkabikkii
aakkakki’nne
aakkantissa
aakkassaa
aakkatan
aakkatansinii
aalla
aamata
aamiga
aamitii
aammatii
aammigan
aammindoo
aammiyarra
aamoonin
aamus
aanat
aanchichchu
aanchiihanse
aanchisiga
aanchoochch
aanchos
aanchuhaa
aanchunkus
aanchuta
aaneen
aanii
aani’nne
aankassaa
aankii’nne
aanki’nnetannee
aannatisu
aannihanniin
aanniihaa
aannin
aannua
aannunkus
aano
aansa
aanseemmada
aansiin
aansitaahaarra
aansiyye
aantaassa
aantakki’nne
aantannee
aantassa
aanteenan
aantoo
aaqq
aaqqaanchitannee
aaqqaannu
aaqqami
aaqqanoru
aaqqeei
aaqqi
aaqqinaamm
aaqqitunta
aaqqyye
aariichch
aarraan
aashaha
aashshaana
aashshassa
aashshimata
aasi
aassaarra
aataa
aatanne
aatayyoont
aatisano
aatuhuu
aautaa
aa’nnu
abii
abina
abut
achchenii
achchoo
ache
ae
aeemmaau
aeennogaa
aeet
aggiin
ahaarra
ai
aiinii
aisiisano
aiyyee
akk
akkaahaans
akkaantasii
akkachch
akkassaba
akkatii
akktaa
allu
amaanchchus
amaanchuhuu
amaaniin
amaannusii
amanina
amanobaan
amanonee
amanora
amanosiru
amanotanneehaat
amanuanii
amayyooi
amba’a
ameemmagiinii
ameennoba’a
amigiitannee
amiihansii
amiintassa
amisigiitannee
amm
ammahaa
ammataa
ammeei
ammeena
ammeet
amme’nne
ammiyye
ammoe
ammohanni
ammoo
ammooho
ammoondaa
ammoruu
ammossagiin
ammyye
amooha
amo’nne
amumbo
amumbu
amunbogga
amusee
amu’nnaachchin
anaka
anataa
anbikokii
anchaakkaachch
anchaakkase
anchaanii
anchassa
anchchi
anchigiinii

anchiihans
anchita
anchoontaabaibdo
anchunku
anchuu
aneentibe
anii
aniyan
anka
ankassa
ann
annase
annen
annibikkiinkaat
anniichch
anniinka
annu
anoba
anobiiha
anoee
anoga’a
anohaa
anoki’nne
anondoo
anonnega
anoohaa
anora
anos
anosi
anosiihaa
anositannee
anossaga
anota
ano’nee
anqaxaanin
ans
ansaan
ansaannu
ansat
anseemmada
anseennogana
ansiichchisin
ansiisaanchua
ansiisaannuhuu
ansiisiihaa
ansina
ansit
ansiteenan
ansitina
ansiyye
ansuanii
ansu’nnaachchii
antaada
antaai
antaaindoo
antaaraa
antaassaarra
antaba’a
antaniyan
antasen
anta’nne
anteeii
anteenanta
anteentaau
anteet
anti
antiyan
antoodaat
antooii
antoora
antoos
antoou
antumbogga
antumburra
antumbuu
antu’nnaan
anuarra
aqanchi
aqqa
aqqaankeru
aqqamayyoou
aqqami
aqqammanii
aqqammiyye
aqqancha
aqqanchi
aqqanchu
aqqansii
aqqansiishsha
aqqansiyye
aqqantaaiihu
aqqanteeii
aqqantoo
aqqataa
aqqeeiichchii
aqqeeu
aqqitaa
aqqitee’nne
aqqituntaa
aransitan
aro’ootaa
as
aseet
asibii
asiin
asiteenan
assaamm
assano
asseenan
assii
assinamm
assitaa
assitan
assiteenantaru
assiyye
asu
ataakka
ataba’a
atans
atenii
ato
atoon
atora
att
attaau
atteenta
attoohanniichch
atu
atussaa
atutauta
aumbogga
ayoou
ayyi
ayyooii
ayyoonne
ayyoou
a’aanchii
a’ammoochch
a’ayyoondoo
a’nnaamm
a’rro’oochchin
baassaa
baigaa
baina
ba’a
beenanta
biihuu
boommigiin
ccitaabuu
chchata
chcho
chchu
chi
choochch
chuhuu
chut
daakkaan
daneen
deentaga
doogiin
eano
ebii
echchahaa
eeananta
eebi
eeccanchiichch
eechchie
eechchi’i
eechchoohans
eechchora
eechchuhu
eechchuta
eedaat
eegu
eehaarra
eehansa
eehi
eehussa
eeiihu
eekkebaan
eem
eemaaneetindoo
eemataa
eemm
eemmaahaan
eemmaau
eemmaga
eemmaranka
eemmasi
eemmi
eemmiiha
eemu
eenaamm
eenaanki’nne
eenanatara
eenaniyansa
eenantaga
eenantasi
eenatansa
eenayyoommae

eeni
eenindoo
eenmgaa
eennaga
eenni
eennisi
eennoga
eennoki’nne
eennonii
eennootii
eennosi
eennossa
eennoua
eennuga
eenokkoonta
eenooma
eentaa
eentaatannee
eentahanniichch
eentandoo
eentase
eentassa
eentiraan
eenumbbussana
eenumboorraan
eenumburruu
eenumbuuta
eeqqa
eeraan
eesamimba’a
eesanondoo
eesanora
eesayyoou
eeseenno
eeshshaakka’nne
eeshshoebiihaat
eesimat
eessaasi
eesu
eetaa
eetaando
eetimba’a
eeu
ee’nneet
egan
eiihu
emata
enanta
ennaga
entagan
eraa
esi
esiru
eta
eyesuusenii
e’eenno
e’eetii
ga
gaantibay
gano
geenantaarra
giin
gitoou
gooiihu
gumbut
haantindoo
hans
hela
hsessa
hshaachchisin
hshaakkaachch
hshaakkas
hshaamita
hshaantassa
hshagiinii
hshanne
hshase
hshat
hsha’nne
hsheemmii
hshe’e
hshiyessa
hshodanii
hshoommiihu
hshossa
huu
ian
ibaaneet
ibiihaat
iccammeemm
iccantaa
icceeu
iccinaamm
iccu
ichchibiihuu
ichchina
ichchona
ichchube
ichchunkus
ichchutaa
ida
igaamint
igaamuan
igii
igoon
igoonnata
iguta
ihanki’nne
iibali
iichch
iichchiin
iichchis
iichchi’i
iichchoon
iichchuta
iigunta
iihaanse
iihank’nne
iihat
iin
iinin
iinkasen
iinneet
iintas
iintasi’nnu
iirana
iiseenno
iishhsata
iishshaachch
iishshaanii
iishshas
iishsheet
iisii
iissayoo
iita
iiteena
iittaakkaantas
iitu
ikk
ikkeei
ikkibii
ikkoo’nne
imaachchissan
imaaneetii
imaat
imanka
imat
imbahe
immaa
ina
inaamm
inaammiihu
inaan
inba’a
ineet
inka
innata
inneet
innitii
inommida
inoommida
inoommiriichch
ins
insan
insun
intis
inunta
inyoogiin
ira
iro’oos
isa
isaamm
isaanchiihaa’nne
isaanchiinta’nnee
isaanchoomata
isaanchuhuu
isaanchuse
isaanchu’nne
isaanki’nna
isaanniin
isaannunku
isaata
isameen
isammii
isan
isanno
isanoga
isanokke
isanonneerra
isanoohu
isanosba’a
isanosigiin
isanossa
isanossanii
isanoua
isansiyye
isantooda
isayyooii
iseemma
iseenaan

iseennoba’a
iseennose
iseenumburuu
isega
ish
ishsassa
ishshaakkasi
ishshagiin
ishshasigiin
ishshataa
ishsheei
ishshessa
ishshiyye
ishshohe
ishshoohu
ishshoommibiichch
ishshossa
isibiihu
isigiinii
isiichchissan
isiinii
isiisaancho
isiiseemma
isiishshe
isiisiin
isiissayyoossa
isiisseeu
isis
isitannee
isonneta
iso’oot
issaagiin
issaaiita
issaara
issaassara
issabii
issana
issantoo
issataa
isseenan
isseentaachch
isseeu
issiishshosseehu
issoobiichch
issoossa
issumbuarra
ista
isumbogga
isumbus
isunta’nne
it
itaaga
itaahaarra
itaaiit
itaakka
itaando
itaara
itaaru
itaassaba’a
itaau
itaneen
itanneehaa
itassa
itayyooi
itayyoossanii
iteega
iteena
iteenanii
iteenantandoo
iteenata
iteennan
iteent
iteentaahu
iteentada
iteentando
iteentasigii
iteenumboochch
iteese
iteeu
itentada
itimbua’a
itmannaakkat
itoobaanin
itooganka
itooiiha
itoont
itooraa
itoose
itoossada
itossata
ittaa
ittaakka
ittaakkaan
ittaakkachch
ittaakkasii
ittaakkatanneeha
ittaaklkataa
ittaantassa
ittakk
ittanne
itta’nne
ittichch
ittiichchii
ittiinin
ittisigiin
ittooii
itubu
itumboggaa
itumbuhee
itun
itunta
ituta
ixu
ixxaan
ixxiineet
iyaan
iyan
iyanse
iya’nne
iyeen
iyigaa
iyitooiihu
iyyaataakkat
iyyenii
i’nne
i’nnu
kkaaii
kkasii
kkoou
laakkata
lan
lataa
llaashshahaa
los
maata
mas
mbutanee
miin
mmee
mmoochch
moohu
n
nase
nchaakkaahaa
neen
nin
nkassa
nnaammii
nneya
noegiin
nsaanchu
ntaa
ntanne
nteendoo
ntis
nu
oaamu
obay
obiita
oda
oe
ogan
ogiinin
ohanni
okkega
ona
onta’nne
oobii
oochcheetindoo
oochchisee
oodiintase
ooga
oogiichch
oohaarra
oohanse
ooichch
ooiihuu
ooma
oomaanii
oomano
oomassa
oomida
oommanise
oommigan
oommita
ooneent
oonina
oonki’nneechch
oonnee
oontakk
oontasen
oonta’nnen
oontigiin
oontis
oorii
oos
oosida
oosirii
oota
ootanee

ootii
ootisseenan
ooxximaanii
ooxxitii
oo’oot
oseda
osida
osiraa
ossagiin
otaa
otanneehaat
o’oo
o’oochchisin
o’oonta
o’ootii
qo
raakka
reeindo
roda
saahaa
saakkagiin
saanchiihaa
saannu
san
sanseen
sanua
saqqu
sebairiinin
seenan
seennua
sega
shshiyyessa
shshose
sibaiiha
sibuu
siibiihu
siihaa
siisaanchuta
siisanoo
siishiyye
siishshata
siishshiyyessa
siisii
siissaau
siisu
sindo
sishshiyyeessa
sitaassa
siteenta
siyaa
sooba
sooiihu
sossagiin
ssaarii
ssabaiga
ssabikkii
ssariin
sseianan
ssoou
sumbut
sutaa
taabiin
taahaarra
taaindoo
taaraa
taassaga
taga
tanne
tans
tayyoou
teeii
teenaneen
teenoochche
teentaau
teente’e
teera
tessa
tindo
tiyaneet
toodaa
toogu
tooiihu
toontiichch
toot
tota
tumboga
tumburru
tun
tu’nnaachchin
uabe
uandoo
uansa
uariichchii
uarriinii
ua’ano
ue
uhu
ukk
umbanne
umboddan
umbogiihaatii
umbossaarra
umbuarra
umburru
umbussa
umbutii
ummaaha
unbu’nnee
unkaa
unki’nne
unkussa
unne
untakki’nne
untis
urru
usee
usi’nnu
ut
utans
uuha
u’nnaa
u’nnu
xxi
yeessa
yiteen
zaainii
’aau
’ammoomm
’eetaa
’nne
’oot

APPENDIX II: AFFIXES FOR RECODING RULES

Suffixes	Substitution	Condition:
amb, mbun, mbaamm, mbaammi, mbeemm, phph, phphee, phpheen, phpheennogii, phphi, phphii, phphiihaa, phphiin, phphinun, phphisiishsha, phphit, phphitaau, phphitan, phphitannee, phphitumboochch, phphitunta, phphu, phphua, phphuhaa, phphunta	b	if word does not start with "***a***"
jeeiya, jie, jj, jja, jje, jjee, jjeegiin, jjeehaa, jjeemm, jjeemmi, jjeen, jjeense, jjeet, jjeeu, jjela, jji, jjiyan, jjiyans, jjiyye, jjo, jjo'nneda, jjoda, jjodaa, jjoga, jjogiin, jjohendo, jjondoo, jjoo, jjoom, jjoomm, jjoommigiineet, jjoonsaahu, jjoosii, jjos, jjose, jjosee, jjosi, jjosibikkii, jjosiga, jjosindoo, jjossa, jjossagiin, ndaami, ndaamm, ndaammi, ndaammii, ndaammiihu, ndan, ndeemm, ndeemm, ndeemmii, ndeemmita, ndo, ndoombaan, ndoomida, ndoommi, ndoommida, ndoommidaa, ndoommiganka, ndoonsi, ndun, ujj, xamaantassa, xamata, xammata, xxaanta, xamaantassa, xxamanoba'a, xxamanoo, xxamat, xxamata, xxamataa, xxammaantassa, xxammas, xxammassa, xxammastana, xxammat, xxammata, xxammataa, xxammatana, xxammatansa, xxammee, xxanchu, xxans, xxansanossaru, xxansiyye, xxantaa, xxantaasira, xxantaassara, xxantaau, xxantun, xxmata	d	if word does not start with "***xa***"
nf, nfaamm, nfaammi, nfaammigu, nfaammii, nfan, nfoommi, phphaqqant, phphaqqantoou	f	if word starts with "***a***"
ngaamiru, ngaamm, ngaammiganka, ngaannu, nge, ngeemm, ngi, nginne, ngise, ngit, ngita, ngitaa, ngoommi, ngumbudda, ngumbutanneehaat, ngun, qqamaannu, qqaman, qqameenan, qqant, qqantooiihu, qqantooiihui, qqantoou	g	
kk, kkaaga, kkan, kkau, kkeemm, kkeenuntaa, kkichchu, kkichchua, kkoohu, kkunta, nk, qqamaamm, qqamaanchu, qqamaanniin, qqamaannu, qqamaannus, qqamanohanniga, qqameemma, qqami, qqamii, qqamiinii, qqamissa, qqamm, qqammaannii, qqamu, qqamuntaa, qqancha, qqanchaan, qqanchu, qqano, qqansinaammi, qqansu, qqant, qqantaa, qqantaau, qqanteen, qqanteenumburru, qqaqqansiin, qqee, qqiteeiita	h	if word starts with "***b***"
nn, nneemm, nneemmii, nnoommii	l	if word does not start with "***ma***"
nkeemm	k	
'mmami, 'mmamii, 'mmantaau, ansiiseemma, ncha, nchata, nchchuta, nchi, nchiin, nchu, nchuhuu, nsi,	m	

nsiisii, nsiisussa, nsishsho, nsitan, nsu, ntaa, ntaa'nneriichch, ntaaba'a, ntaahaarra, ntaahaarraa, ntaaii, ntaara, ntaasira, ntaassa, ntaassariin, ntaatannee, ntaau, ntataa, ntee, nteen, nteeiita, ntooda, ntoonte'eechch, ntoos, ntootinne, ntoou, ntu'nna		
'nnaqqancha, 'nnaqqanchiinii, 'nnaqqant, 'nnaqqantaaga, 'nnaqqantee	n	
nn, nno, nnoommida	r	if word does not start with "***wa***"
ccameenii, ccamii, ccamuha, ccant, ccantaaga, ccantaaha, ccantunta, ccaqqamu, ccaqqanchahaa, ccaqqanchiin, cceekkeet, cci, ccitaa, iccamiinii, nsaamm, nsaammi, nsaammii, nsaammiihu, nseemm, nseemmiru, nseen, nsoomm, nsoommi, nsoommida, nsoommigiin, nsoommiigiin, nsoommogiin, nsoongiin, nsoonsa, nsun, nsunka	s	if word does not start with "***xa***"
chchaan, chchas, chchat, chche, chcheeu, chchessa, chchiyye, chcho, chcho'nneda, chchoda, chchoga, chchoommigiin, chchora, chchossada, ntaamm, ntaammigaa, ntaanse, ntan, ntantaaiita, nteemm, nteeneet, nteeneetba'a, ntit, ntita, nto, ntoni, ntonii, ntoommida, ntun, xxayyoo, xxi	t	
cco	x	
ccano, cci, ccit, jj, jje, jjessa, jjo, jjoe, jjoochch, jjos, jjosidaa, nzaankee, nzan,	z	if word does not start with "***ha***"

APPENDIX III: RULES FOR REMOVING SUFFIXES

Rule for Step 1:

if ((*word ends on aqqansiisaanchiichch****) && (****length of the remaining part is greater than 1****)) {***

*remove the suffix****;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter****;***

}

}

Rule for Step 2:

if ((*word ends on aankiInnehanniichch|eenayyoomakketannee|inaammitanneehaatii|iteenantahanniichch|isaanchiihaankiInne****) && (****length of the remaining part is greater than 1****)) {***

*remove the suffix****;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter****;***

}

}

Rule for Step 3:

if ((*word ends on aankinnehanniichch|aaqqaanchisitannee|ansiissaahaarranii|hshoommitanneehaat|isaanchiihankiInne|isaanchoohankiInne|ittaakkaachchiInne****) && (****length of the remaining part is greater than 1****)) {***

*remove the suffix****;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter****;***

}

}

Rule for Step 4:

if ((*word ends on*

*ammeenniyaarranka|anchaakkaachchina|ansiisaannitannee|aanchitannehaatii|aqqansiis aanchiya|aqqansiisaannuhaa|aqqansiisseenanta|eiseenuInnaeechch|iishshoomaantassa|i noommihanniichch|isaanchiintaInnee|isanotanneehaatii|ishshoommibiichch|isseentaachc hinaa|iteentahanniichch|itoontihanniichch|ittaakkatanneehaa|nteentahanniichch|siisseent aachchii****) && (****length of the remaining part is greater than 1****)) {***

*remove the suffix****;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter****;***

}

}

Rule for Step 5:

if ((*word ends on*

*aakkaahaankiInne|aaqqaanchitannee|aaqqansiisaanchi|eennogiitannehaa|ammohanniich chii|amuInnaachchinne|anchaakkaachchin|anchoontaabaibdo|aqqansiisseenata|aqqantee ntaachch|ateentahanneenin|atteenumbuaaggii|beentahaaniichch|eechchaakkaachch|eena ntariichchii|eesanotanneehaat|goommitanneehaat|hshaakkatanneeha|hshaakkatanneeha ...****) && (****length of the remaining part is greater than 1****)) {***

*remove the suffix****;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter****;***

}

}

Rule for Step 6:

if ((*word ends on*

aakkaatankiInne|aammihanniichch|aankiInnebikkii|ichchitanneehaa|aankiInnetannee|aat ayyoondoonii|aisiishshiyansa|amanotanneehaat|ameemmatanneeha|amumbonnetannee|a nchchaakaachch|annibikkiinkaat|anonnehannigana|ansiisaanchutaa|ansiteentaranka|ans

uInnaachchii|antaasitanneeha|antoohanniichch|antumbutanneeha|aqqansiisaanchi…)
&& (*length of the remaining part is greater than 1)) {*

remove the suffix;

***if** ((length of the remaining word is greater than 4) **&&** (the remaining word ends on double letter)) {*

remove last letter;

}

}

Rule for Step 7:

if ((*word ends on anonnehannigan|aanchoomaantas|amisigiitannee|eennossagiihaa|issaahaarrando|atteent assadaa) && (****length of the remaining part is greater than 1)) {*

remove the suffix;

***if** ((length of the remaining word is greater than 4) **&&** (the remaining word ends on double letter)) {*

remove last letter;

}

}

Rule for Step 8:

if ((*word ends on aakkaahaaInne|aakkaahaInnee|aanniichcheet|achchiichchin|eenoochchessa|aakkaantass an|aakkaantaInne|ishshobiihaat|aakkatanneeha|aakkatansinii|aamiichchinne|aanchukkiI nne|aankiInneenku|aanniintakkin|aansaqqachchi|aansiteenanta|aaqqiteenanta|amumboss aarra|anchaakkachch|anchakkaachch|annaakkatanii…) && (****length of the remaining part is greater than 1)) {*

remove the suffix;

***if** ((length of the remaining word is greater than 4) **&&** (the remaining word ends on double letter)) {*

remove last letter;

}

}

Rule for Step 9:

if ((*word ends on aahaankiInne|aahanniichch|aakatansinii|aakkaachchii|iteentaachch|ansiisanseei|aakkaahaansa|aakkaachchin|aakkaachchis|aakkaahaansa|aakkaahaInne|aakkaantasen|aakkaantassa|aakkakkiInne|aakkataaInnu|aammiriichch|aanchiihaans|aanchiihanse|aanchisigiin|aanchitannee|aannihanniin|aanniichchin…****) && (****length of the remaining part is greater than 1****)) {***

*remove the suffix****;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter****;***

}

}

Rule for Step 10:

if ((*word ends on aahankiInne|aahankiInne|aakkaahaans|eennaaggiin|isiisanseei|anqaxeechch|iteentibala|isiisanseei|siisiishsho|isansanossa|aakkaahaans|aakkaahaant|ansitaniyan|eemmotannee|aachcheetii|eennaaggiin|oontantassa|attooiichch|iichchissaa|aakkaahanne|isanobandoo|anaantassaa|aakkaahansa…****) && (****length of the remaining part is greater than 1****)) {***

*remove the suffix****;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter****;***

}

}

Rule for Step 11:

if ((*word ends on aachchinne|aachchissa|aadaakkata|ansiishsho|aqqammiyye|eedaatbaIa|amanoganka|siisanseei|ansiishsho|siisanseei|siisanseei|aakkaachch|aakkaahans|ansiishsho|siisanseei|ansiissara|ansiishsho|aeennosiga|ittaakkata|anoriichch|isubossara|innebiihuu|aahaaInnee|aahankInne|aahannigan…****) && (****length of the remaining part is greater than 1****)) {***

*remove the suffix****;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter****;***

}

}

Rule for Step 12:

if ((*word ends on aachchise|aaggiinii|aahaaInne|amanseei|ammeeIeru|isooindoo|isiishsho|hshotanne|eenumbrra|aanchuhaa|isiishsho|aakkaahaa|ammee'eru|iisanseei|eeminnita|ishshgiin|ayyoondoo|immaanina|amiiechch|ittaakkat|anobandoo|iroIoonta|iininbala|amauobaIa|anchaakka…****) && (****length of the remaining part is greater than 1****)) {***

*remove the suffix****;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter****;***

}

}

Rule for Step 13:

if ((*word ends on aachchii|aachchin|aachchis|oontibii|achchuna|iyyaanta|isahossa|umbuuhuu|isahossa|ootannee|iteenant|aanchiin|amanseei|ooriniin|siishsho|ubossaga|amanseei|amanseei|siishsho|hshooman|ogiichch|amanseei|eematina|eennogaa|eennogii|eenoogii…****) && (****length of the remaining part is greater than 1****)) {***

*remove the suffix****;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter****;***

}

}

Rule for Step 14:

if ((*word ends on*

*aabunku|aachcha|aachche|aachchu|igiihaa|ebiihaa|antunka|tokoont|amanian|amanogu|hshassa|ansussa|aakkata|eemassa|amanona|eennaga|amumbua|aIaannu|attiyan|atosina|asibiit|assabii|aaebaIa|aaggiin|aahaans|aahanne|aahansa…****) && (****length of the remaining part is greater than 1****)) {***

*remove the suffix****;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter****;***

}

}

Rule for Step 15:

if ((*word ends on*

*aachch|aachua|aacnhu|aaduma|anseei|iyyata|isussa|eganka|ansano|anseei|anseei|aoonii|eechch|anseei|anseei|eemant|isndoo|aIanna|aaggii|eennas|eenoga|itaaga|attaan|aagiin|aahank|aahans|aaiiha|uInnaa|eeiihu…****) && (****length of the remaining part is greater than 1****)) {***

*remove the suffix****;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter****;***

}

}

Rule for Step 16:

if ((*word ends on*

*aabIa|aagaa|aagan|aagga|aIano|iyyat|amano|ancha|ansat|antee|autaa|ammao|aalaa|aagii|aahaa|aaihu|aaiit|aaita|aakas|aakat|aakka|aalla|aamii|aamit|aamma|aamme|aammi|aammo|aammu|aamua|aamus…****) && (****length of the remaining part is greater than 1****)) {***

*remove the suffix****;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter**;***

}

}

Rule for Step 17:

if ((*word ends on aabe|aabi|aabu|aada|anto|anoo|anog|aaga|aagi|aagu|aaha|aahu|aaii|aait|aame|aami|aamm|aamo|aamu|aana|aani|aank|aanm|aann|aano|aans|aant|aanu|aaqq|aara|aari|aaru|aase…**) && (****length of the remaining part is greater than 1**)) {*

*remove the suffix**;***

if** ((length of the remaining word is greater than 4) **&&** (the remaining word ends on double letter**)) {

*remove last letter**;***

}

}

Rule for Step 18:

if ((*word ends on aai|aam|aan|aas|aat|obo|aan|iin|nto|aIa|aau|aba|ada|aee|aga|aha|akk|amm|ama|ame|ami|aii|amo|amu|ana|ani|ann|ano|ans|ant|anu|aoo|aqi|aqq|aro|asa|ase|asi|ass|asu|ata|ati|ato|att|atu|aua|auu|bai|bat|bba|bot|chi|cho|chu|daa|doo|eba|ebu|eeb|eeh…**) && (****length of the remaining part is greater than 1**)) {*

*remove the suffix**;***

if** ((length of the remaining word is greater than 4) **&&** (the remaining word ends on double letter**)) {

*remove last letter**;***

}

}

Rule for Step 19:

if ((*word ends on aa|ae|ai|ak|am|an|as|at|au|be|bo|bu|ee|ei|en|eo|es|eu|ia|ie|ii|in|is|it|kk|oe|oi|on|oo|os|qi|qo|ra|ro|ru|sa|se|si|so|ss|su|ta|te|to|ua|ue|ui|un|us|ut|uu|yi|yu|ee**) && (****length of the remaining part is greater than 1**)) {*

*remove the suffix**;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter**;***

}

}

Rule for Step 20:

if ((*word ends on a|u|i|e|o|n|s|t****) && (****length of the remaining part is greater than 1****)) {***

*remove the suffix**;***

if *((length of the remaining word is greater than 4)* ***&&*** *(the remaining word ends on double letter****)) {***

*remove last letter**;***

}

}

APPENDIX IV: KAMBAATA WORDS BEFORE STEMMING

Un-stemmed words taken from test set two (TS2) are given below. This is not the whole test data, rather one word for each corresponding stem is selected and presented.

aagaqqanchu aalota aaqqantee aassammeessaarra aazeenii abbaachch afaakka afeehaarra afuu'lleeu agga agissaau agudanua alaphphitaau aleenin amaa'llancha amaarssaga amanta amma'nnamussaa ammanaatuta anabbabamanooha angassa ankari annanna annunku asi'mmu assamii awwa'nnaqqanchu azuta azzazosigiin baa'yyaatta baabbu'nnaachch baaddaachch baasa baissaau bakkani barcumaa bargammo barraa baru beehancha beetii bifa biixaakkaan birena bizzeenni bokku bonata booradisuhuu booyyeekkeehaa boqoon bullaakkaachch bushagan caakkee calcalat ceemmit cu'mmisiisi cumuqus daaddaabbita daddaabbee daftaraankassa dagamano danaamita dandaamba'a daqqinaamm dibeemma digiirammaa dikkai diraamaan doo'rr dooi dullo duuhagiin duuni eebbaaga egedo enkee'nniin eraantas ereeransanua etaru faareisanossa faduu faisano fanneena fanqalaansit farrat fidallaan fintant foolooccat fulii fushsheennondoo ga'aata gaantuntaa gaazeexxataa gabbancho gaffarsiisiin gagi gajaajjaakkaahaa galaxxisanonne galte gann gardabbaan gare gashshaanchuta geemmaruse gi'iizi gidaanii giphphammat gobaanchuta gonsamanosi gooca gooffoogan goolliteentaachch goonchuu gorruu gotu haabdoollichchut haasaawwaa habantaa hadata hagaakkaachchis hagara haliin hamaam hammaakkaan hanqafamu hansawwiichch haraaraga hasammoru hawwanka hayyaakkaan he'eeraan heellisu heessa'aau hegeegiihans hezzeettoot higgaa higgo hiiramm hinca'ano hiqamanseei hiramanian hogobo hooggo hoolamas hoolchutaans horanka hosisaassa hossootannee hujachchessa hundaanka hunt ichchannebiihuu ihanora iillanosba'a iitta ilansano illeen injiijjeen inkiiliin insantoo jaallaakkaahansa jeechchona kaa'llanossa kaamebiihuu kaardaakkat kaasanseei kajjeelloda kalu kanteenoochchessa karammo kasaakkaanii kee'mmubossaga keenatoon keisansii keu kichcheehchi kifilii kirrahaa kodataa kulammo kulisantoos kulliyyessa kuushebii laagaakka laalloga lagaakkaa lallabaanchoomaantas lamaqqat lehiqqi leinii lenqeeqqissan luusa ma'nnaakkaan maabaras maanana maao maarrata maashata maaxamii maccat maccooccaannus madu makkeeii malaakkaa malaltan malatt maleesiseemmada manchi mannaakkaachch maranoba'a mashka'a mataqqat maxaaffa maxammo meemmaaraantas meexxoomataa meguta mereeri meseleet mexxita midanoo miine miinjaakkaan minaadabi mini misili mooshshaqqi muramanseei

naqaashshooman oIayyoou oddaqqian oli onteet onxane oodami oogattaau ororreesu qaada qaaggishshata qaanqaakkata qaarsi qachchooneet qalamuhaa qanso qasanseei qaxi qegi qeraa'rrimana qexeesaanchisi qiraareen qixxammeeii qocichchuu qomesira qoobaakka qooccano qoodamaanni qoorsua qorabamanua qormiin qoxara quImmai quuxaakkata raadoonaantas rehi reshaan rosaanchut saaliichch saamuhuu sadaakka salakkit saqalu saseenta satatteemant sawwaitti seeli seeraamua sereegg serekketii sha'llu shalana shallaggin shashimata shee sholeemma shoohamanua shoolita shoosaawwaakkata shu shuunnantaau sila'antaada sirisi sohinnta sou su'mmahaa ta'mmaanniichcheet taaphanneena tamissaa tashsha'anna teeltoou teeppi televizhiinaan tishshiteehaarra tochchaanchoonin toliin tophpheechch toqqansiinii toroshammeeu tunsuta tuuteeccanseemmada ubiin usheexxanoba'a uullataa waabbaakka waajjeen waalano wiimaa wixi wogi wolii wollaakkataa womaashsha woqqaan worro woyyaaggii wozana xa'mmaakkaa xaaccu xaawaataaga xaaxxisu xaazz xabbaii xalla xaphphattaan xaqqu xeata xeffittaakkatanneehaa xinkuta xishshimaan xooffoogan xophphi xufi xummiininbala xuudamanora yaa'aakkat yaadii yaaniin yaarano yamee yoo you zabaakkata zahicciin zahisoon zakkaanchoonin ziiqut zirranno

APPENDIX V: KAMBAATA WORDS AFTER STEMMING

Stemmed words taken from test set two are given below. The words contain only unique stems of the output of the test set two (TS2) of the improved stemmer.

aag aal aaqq aass aaz abb af afoo afuuIll agg ag agud alaphph al amaa'll amaar am amma'n amman nibaab ang ankar annann ann asi'm ass awwan azut azzaz baa'yyaat baab baad baas bai bakkan barcum barg barr bar beeh beet biix bir bizz bokk bon boorad booyyees boq bull busha caakk calcal ceem cu'mm cumuq daaddaab daddaab daftar dag dan dand daqq dib digiir dikkis diraam door dooi dull duuh duun eebb eged enkeen era ereer etar faar fad fai fann fanqashsh farr fidal fint foolooc ful fushsh ga'aa gaan gaazeex gabba gaffar gag gajaaj galax gal gann gardab gar gashsh geemmar gi'iiz gidaan giphph gob gons gooc goof gool goon gorr got haabdool haasaaw hab had hag hagar hal ham hamm hanqaf hansaw haraar has haww hayyut hei heel hees hegeeg hezzeet hig higg hiir hinc hiq hir hogob hoog hoolam hoolch hor hos hoss hujat hund hun it ih iill iitt il ill injiij inkiil ins jaal jaat kaa'll kaam kaard kaas kajjeel kal kam kar kas keeIm keen kei keu kichchei kifil kirr kod kul kulis kull kuush laaga laal lag lallab lam leh lei lenqeeq luus ma'n maabar maan maai maar maash maax macc maccooc mad makk malah malal mal malees manch mann mar mashk mat xaaf max meemaar meex meg mereer mesel mexx mid miin miinj minaadab min misil mooshsh mur naqqas oi oddis oli ont onxan ood oogat oror qaad qaag qaanq qaar qachch qal qan qas qax qeg qeraa'r qexees qiraar qixx qoc qom qoob qooc qood qoor qorab qorm qoxar quIm quux raadoon reh resh ros saal saam sad salak saqal sas satat saww seel seer sereeg serekket sha'l shal shallag shash shi shol shooh shool shoosaaw shu shuun silai sir soh sou su'm ta'm taaphan tam tashsh teel teep televizhiin tishsh tochch tol toll tophph toqq torosh tuns tuuteecc ub usheex uul waab waaj waal wiim wix wog wol woll wom woqq wor woyy woz xa'mm xaaz xah xaax xaazz xabb xall xaphph xaqq xei xeff xink xishsh xoof xophph xuf xumm xuud yaai yaad yi yaar yamaa yoo you zab zahic zah zakk ziiq zirr

APPENDIX VI: KAMBAATA VERB INFLECTION EXAMPLE

TABLE VI-1: KAMBAATA VERB INFLECTION FOR A VERB "***KUL***" 'TELL'

Person	**Perfective PVO**	**Perfective PVE**	**Negation of PVE**	**SS Purposive**	**DS Purposive**	**Imperfective (IPV)**	**Jussive**	**Conjunct forms**	**Progressive (PROG)**
1SG	Kul-*l-oom(m)*	Kul-l-*eem(m)*	Kul-*imba'a*	Kul-o-ta /Kul-*ii(ha)/*	Kul-*un-ta*	Kul-*aam(m)*	Kul- *Ø*	Kul-*l*	Kul-*ayyoom(m)*
2SG	Kul-*toont*	Kul-*teent*	Kul-*timba'a*	Kul-*t-ota*	Kul-*t-un-ta*	Kul-*taant*	Kul- *Ø*	Kul-*t*	Kul-*tayyoont*
3M	Kul-*l-o*	Kul-*l-ee(u)*	Kul-*imba'a*	Kul-*o-ta*	Kul-*un-ta*	Kul-*ano*	Kul-*un*	Kul-*l*	Kul-*ayyoo(u)*
3F/PL	Kul-*too(u)*	Kul-*tee(u)*	Kul-*timba'a*	Kul-*t-ota*	Kul-*t-un-ta*	Kul-*taau*	Kul-*tun*	Kul-*t*	Kul-*tayyoo(u)*
1PL	Ku***n***-*noom(m)*	Ku***n***-*neem(m)*	Ku***n***-*n-imba'a*	Ku***n***-*n-ota*	Ku***n***-*n-un-ta*	Ku***n***-*naam(m)*	Ku***n***-*nun* Ku***nn***-o	Ku***n***-*n*	Ku***n***-*nayyoom(m)*
2P/HON	Kul-*teenta(a/u/)*		Kul-*teen-imba'a*	Kul-*teen-o-ta*	Kul-*teen-un-ta*	Kul-*teenanta*	Kul-*l-e*	Kul-*teen*	Kul-*teenayyoonta*
3HON	Kul-*eemma(a/u/)*		Kul-*een-imba'a*	Kul-*een-o-ta*	Kul-*een-un-ta*	Kul-*eenno*	Kul-*een-un*	Kul-*een*	Kul-*eenayyoomma*

APPENDIX VII: SAMPLE WORD STEM AND ITS VARIOUS WORD FORMS

The following 207 distinct words are variations that are formed by inflection and derivation of a verb stem "***kul***" (tell). This list is one typical scenario to see how Kambaata language has complex morphology. However, the following is not the complete list of word formation for the verb stem "***kul***".

kulaanchchii
kulaanchchiihaa
kulaanchi
kulaanchiin
kulaanchina
kulaanchisi
kulaanchisibiinku
kulaanchisina
kulaanchisitannee
kulaanchisitanneeha
kulaanchisitanneehaa
kulaanchitannee
kulaanchitanneeha
kulaanchitanneehaa
kulaanchoon
kulaanchu
kulaanchuhuu
kulaanchunku
kulaanchunkus
kulaanchus
kulaanchusii
kulaanchut
kulaaniichch
kulaaniichchis
kulaaniin
kulaannibii
kulaannii
kulaanniichch
kulaanniichchis
kulaanniichchisii
kulaanniichchisin
kulaanniiha
kulaanniihaa
kulaanniihaans
kulaanniihaansii
kulaannisi
kulaannisitannee
kulaannu
kulaannuha
kulaannuhaa
kulaannuhuu
kulaannunku
kulaannunkus
kulaannus
kulaannusii
kulaanoon
kulaanoontanne
kulammee
kulammeehaa
kulammeeii
kulammeeiiha
kulammeeiihaa
kulammosi
kulamumbu
kulamumbuha
kulamumbuhaa
kulan
kulaneen
kulanian
kulanians
kulaniyaan
kulaniyaans
kulano
kulanoba'a
kulanobikkii
kulanoga
kulanohannii
kulanohannitannee
kulanohannitanneeha
kulanohannitanneehaa
kulanondoo
kulanoo
kulanosiga
kulanosigu
kulanotannee
kulanotanneeha
kulanotanneehaa
kulanotanneehaat
kulanotanneehaatii
kulanteenumbutannee
kulanteenumbutanneeha
kulanteenumbutanneehaat
kulanteenumbuu
kulanteenumbuuha
kulanteenumbuuhaa
kulantoo
kulantumbu
kulantumbuta
kulantumbutaa
kulanua
kuli
kuliga
kuliichchis
kuliiha
kuliihans
kuliin

kull
kullanchisibii
kullee
kulleehaa
kulleei
kulleeii
kulleeiiha
kulleeiihaa
kulleeikke
kulleeikkeeraan
kulleeu
kullo
kulloga
kullogga
kulloonku
kullos
kullosi
kullosihannii
kulota
kulsiisheeikke
kulsiisheeikkeeraan
kulsiisii
kult
kultaa
kultaaga
kultaagaa
kultaahaa
kultaahaagga
kultaahaarra
kultaai
kultaara
kultaarii
kultaariiha
kultaariihaa
kultaariineet
kultaaru
kultaaruu
kultaassarii
kultaassaru
kultaatannee
kultaatanneeha
kultaatanneehaa
kultaau
kultan
kultanian
kultanians
kultaniyaan
kultaniyaans
kultee
kultee’nna
kultee’nnaachch
kultee’nnaachchii
kultee’nnaachchiis
kultee’nnaachchiisin
kulteehaa
kulteehaagga
kulteehaando
kulteehaandoo
kulteeikkeeraan
kulteeindo
kulteeindoo
kulteent
kulteenumbuunka
kulteenunta
kultii
kultoo
kultoobaan
kultooga
kultoogaa
kultoohaagga
kultoohaarra
kultoohanneen
kultoohanniin
kultooi
kultooii
kultooiiha
kultooiihaa
kultooiihu
kultooiihuu
kultooiinku
kultoont
kultoora
kultoorii
kultooriiha
kultooriihaa
kultooriineet
kultooru
kultooruu
kultoossarii
kultoossaru
kultootannee
kultootanneeha
kultootanneehaa
kultoou
kultota
kultun
kultuntta
kultunttaa
kultunttaat
kulu
kuluhuu
kulumbusi
kulumbusii
kulumbussa
kulunta
kunn
kunnaamm
kunnaammii
kunnaneen
kunnota
kunnun

DECLARATION

I, the undersigned, declare that this thesis is my original work, prepared under the guidance of **Dr. Solomon Teferra.** All sources of materials used for the thesis have been duly acknowledged. I further confirm that the thesis has not been submitted either in part or in full to any other higher learning institution for the purpose of earning any degree.

Jonathan Samuel _____

Name Signature

St. Mary's University, Addis Ababa **March, 2018**

ENDORSEMENT

This thesis has been submitted to St. Mary's University, School of Graduate Studies for examination with my approval as a university advisor.

Solomon Teferra (PhD) _____

Advisor Signature

St. Mary's University, Addis Ababa **March, 2018**