Designing A Rule Based Stemming Algorithm for Kambaata Language Text

Jonathan Samuel *jimmyelove@gmail.com*
Telecom Excellence Academy/ Digital Learning
Ethio Telecom
Addis Ababa, Ethiopia

Solomon Teferra *solomon.teferra@aau.edu.et*
Faculty of Informatics/ School of Information Science
Addis Ababa University
Addis Ababa, Ethiopia

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 strictly suffixing language with a rich morphology and word formations mostly relying on suffixation; even though its word formation involves infixation, compounding, blending and reduplication as well.

The output 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. A test set of 2425 distinct words was used to evaluate the stemmer. The output from the stemmer indicates that out of 2425 words, 2349 words (96.87%) were stemmed correctly, 63 words (2.60%) were over stemmed and 13 words (0.54%) were under stemmed. What is more, a dictionary reduction of 65.86% has also been achieved during evaluation.

The main factor for errors in stemming Kambaata words is the language's rich and complex morphology. Hence several 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: Kambaata Stemmer, Rule-based Stemmer, Stemming Algorithm, Kambaata Language, Information Retrieval.

1. INTRODUCTION

Stemming algorithms are automated programs to reduce all terms with the same root to a common form by eliminating the words' morphological affixes [1]. In today's world, stemmers are commonly applied in different natural language processing applications such as information retrieval, text classification, text summarization, morphological analyzer and automatic machine translation [2]. Therefore, designing a stemming algorithm for Kambaata language has a huge benefit in the development of various natural language processing applications.

Different forms of the same word can be created in Kambaata without changing the word's part of speech through inflectional morphology. These variations are outcomes of changes in person, number, tense and gender [3]. As stated in [4], these kinds of variations do not alter the word's original class.

An example of a stem can be the word “mar” (go - 2male) which is the stem for the variants “marro” (goes - 3male), “marree(u)” (went - 3male), “marimba’a” (didn’t go - 3male), “marano” (will go - 3male), “marayyoo(u)” (is going - 3male), and “marota” (to go - 1sg/3male).

Another technique for word formation in Kambaata is using derivational morphology which results in change of the word’s part of speech [4]. For instance, affix changes a word from adjective to nouns, from verb to nouns, from noun to verbs, and so on. For example, “jaalu” (friend - noun), “jaalloomaan” (friendly - adjective), and “jaalloomata” (friendship - noun).

Kambaata language has very complex morphology [5]. According Treis [6], Kambaata does not make use of prefixes for word formation. Nevertheless, complicated word forms can be created by suffixation, infixation, compounding, blending and reduplication, specifically by full reduplication or by reduplication of portion of the word in Kambaata [5]. The reduplicated section of the syllable is prefixed in Kambaata [5].

Several studies agree that Kambaata is a strictly suffixing language with a rich morphology and complex word formations mostly relying on suffixation; even though its word formation involves infixation, compounding, blending and reduplication as well [5], [6].

This paper describes the design and evaluation of the first rule based, longest match and context sensitive stemmer developed for Kambaata language.

2. LITERATURE REVIEW AND RELATED WORKS

The concepts of conflation techniques, stemming algorithms and stemmer evaluation methods are the basic components in stemming researches.

2.1 Word Conflation

Word conflation is a method of matching semantically related words with different morphological variations [7]. It is performed either manually or automatically by means of software programs [8]. Automated word conflation is carried out through computer programs known as stemmers and those programs eliminate the affixes from words to form their corresponding stems [1].

2.2 Stemming Algorithms

Stemming algorithms are classified as rule-based, table lookup, successor variety, and n-gram based on their strategy of word stemming [7].

Search query words must match the terms in databases or documents for effective retrieval of the required information [9]. There must be a mechanism decide whether the given query word matches the word in the document or not. The straightforward technique is to allow exact matching only; for instance, “stem” would match itself only and a document that contained “stem” but “Stem” would not be recognized as a match.

The properties of stemming algorithms differ depending on whether stem dictionary and suffix lists are being utilized, and also on the purpose for which the stemmer is designed [1], [8]. However, most of stemmers are based on specific rules and techniques [10]. These techniques include removal of a single longest matching suffix or the iterative elimination of numerous simple base suffixes. The motive behind iterative strategy is the fact that suffixes are affixed to stems one after the other in concatenation. In the iterative approach, suffixes are removed from the word in the order of their derivational rules. The suffix removal begins from the end of the word performing in the direction of the word starting. The longest match strategy removes the lengthiest suffix possible at one time using single pass approach [4].

Stemmers can also be classified as context-free and context-sensitive stemmers [2]. In context-free stemming algorithms, any restriction is not applied on the stem and therefore no extra procedures are required to examine exceptional scenarios. Context sensitive rules identify certain conditions by which each suffix could be removed from the word to be stemmed [4]. Studies recommended that better output can be attained by incorporating restrictions to stripping procedures while applying context-sensitive conditions [3], [11].

2.3 Evaluation Methods for Stemmers

The most popular performance analysis techniques to measure the accuracy of stemmers are the manual, vocabulary reduction and Paice's methods [12], [13]. In the manual evaluation technique, it is an individual that makes the decision whether the stem is correct or not for each word stemmed. We have three evaluation parameters in this approach: the number of correctly stemmed words, the number of over stemmed words, and the number of under stemmed words [14]. The word compression is also another mechanism to measure stemmer's effectiveness in terms of reducing duplicate words having same root.

Stemmers are also evaluated using Paice's method [14]. In this technique, measures of under stemming and over stemming decide the level of the stemmer's effectiveness beyond retrieval context. In this method, three measurements are applied to make a qualitative contrast among various stemmers: the over stemming index (OI), the under stemming index (UI), and the stemming weight (SW). The strategy requires 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 [14].

2.4 Related Works

Stemming researches have been conducted to several languages both internationally and in the local context. Locally, stemming algorithm design and development has been attempted for Amharic, Afaan Oromo, Tigrigna, Wolaytta, Silt'e and few others. However, there is no any research carried out to explore stemming technique for Kambaata words and there has never been any attempt done to design a rule-based stemmer for Kambaata language text. Thus, this research is the first of its kind to explore stemming method and for designing an algorithm for stemming Kambaata words.

Lovins Stemmer

Lovins Stemmer is the first popular and effective stemmer which was proposed in 1968 by Julie Beth 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 recoding could make the stemming process fast, the output might not be necessarily accurate.

Dawson Stemmer

Dawson stemmer is an extended version of the Lovins stemmer except that it covers a much more comprehensive list of about 1200 suffixes [15]. Similar to that of Lovins' stemmer, the stemmer is a single pass and hence it is very fast. The suffixes are stored and arranged in the reverse order indexed by their length and last letter. Dawson did not use recoding technique in this algorithm instead used an extension of the partial matching procedure.

Dawson stemmer covers more suffixes compared to Lovins stemmer. It also performs faster than Lovins stemmer. However, the weaknesses of Dawson stemmer are its complexity and lack of standard reusable implementation [16].

Porter Stemmer

The Porter stemmer is one of the most popular stemmers today, which is proposed in 1980 [3]. Since then, the original Porter stemming algorithm have been changed and improved multiple times [17]. The stemmer is based on the idea that the suffixes in the English language are mostly made up of a combination of 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.

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 Stemmer
The Paice/Husk stemmer is an iterative algorithm with one table containing about 120 rules indexed by the last letter of a suffix [18]. 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.

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.

Paice/Husk stemmer has a benefit of its simplicity and each iteration handles both deletion and recoding during the application of the rules. However, the algorithm is very heavy hence over stemming may occur during stemming process.

Krovetz Stemmer
The Krovetz stemmer was developed by Robert Krovetz at the University of Massachusetts in 1993 [19]. 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, the conversion of past to present tense, 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.

The stemmer attempted to enhance the accuracy. However, it is inefficient during stemming complex words and large test data. The other problem of this stemmer is that it is unable to handle words that are not in the lexicon. As a result, its reliability is affected for recall and precision [20].

Amharic Stemmers
The first Amharic stemming algorithm that conflates words for information retrieval was developed by Nega Alemayehu and Peter Willett [4]. Their work was one of the earliest main works in Amharic NLP researches. The stemmer was iterative that removes prefixes and suffixes and also considered letter inconsistency and reiterative verb forms. This algorithm first identifies a set of stop-words and then a set of affixes associated with the remaining content-bearing words. 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. 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, Atelach and Lars developed another Amharic stemmer which is based on table lookup strategy [12]. This 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 large Amharic news corpus and some old-fashioned words from Amharic fiction. 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.

Afaan Oromo Stemmers

The first rule based Afaan Oromo stemmer was developed by M. Wakshum [21]. 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 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 [13] to improve weakness of stemmer developed by M. Wakshum which had no rules to stem irregular and duplicated words. This stemmer 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. The output from the stemmer indicates, out of 5000 words 38 words (0.77%) were under stemmed and 220 words (4.39 %) were over stemmed. Totally this stemmer generated 258 words (5.16 %) wrongly stemmed words. As a result, the accuracy of the stemmer was 94.84%.

2.5 Comparative Evaluation of Related Works

Table 1 summarizes the key properties of each of the popular English stemmers.

Language	Researcher/s	Conflation Technique	Context Sensitive?	Advantage	Disadvantage
English	Lovins	Rule-based (longest match)	Yes	Fast	Not all suffixes are available
English	Porter	Rule-based (iterative)	Yes	Most accurate	Sometimes produce invalid stems
English	Dawson	Rule-based (longest match)	Yes	Faster	Very complex
English	Paice/Husk	Rule-based (iterative)	No	Simple form	Over stemming may occur
English	Krovetz	Rule-based and Table Lookup	No	Light stemmer	Poor recall & precision

TABLE 1: Summary of comparative analysis of English stemming algorithms. Source: [22].

As discussed in section 2.4 above, stemming algorithms have been designed for different local languages. However, there is no any research carried out to design stemming algorithm to stem Kambaata words for several natural language processing applications. Thus, the Kambaata stemmer is the first of its kind developed for the first time to stem Kambaata text using rule-based approach.

The Kambaata stemmer is a single pass, context sensitive and rule based longest match algorithm. Its general working procedure looks like the Lovins stemmer. However, its detail working scenario is completely different. Even though, stemming algorithms are different for each language and difficult to compare one with the one designed for another language, the Kambaata stemmer is developed based on large corpus and its performance is far better when compared to other local stemmers developed for Ethiopian languages.

Table 2 summarizes the key properties of some of the local stemmers.

Language	Primary Researcher	Conflation Technique	Context Sensitive?	Accuracy
Amharic	Nega Alemayehu	Rule-based (Iterative)	Yes	95.90%
Amharic	Atelach Alemu	Affix removal & Dictionary Based	No	75%
Afaan Oromo	Mekonnen Wakshum	Rule-based (Longest-match)	Yes	92.52%
Afaan Oromo	Debela Tesfaye	Rule-based (Iterative)	Yes	94.84%
Kambaata	Jonathan Samuel	Rule-based (Longest-match)	Yes	96.87%

TABLE 2: Summary of comparative analysis of local stemming algorithms.

3. MORPHOLOGY OF KAMBAATA LANGUAGE

3.1 Overview of Kambaata Language

Kambaata is the name of the people that speak the Kambaata language and the name of the language that they communicate [5]. It is called “Kambaati afoo” in Kambaata language, literally means ‘the mouth of Kambaata’. At present, the language is estimated to be spoken by more than a million people [23]. Currently, it serves as a medium of instruction in the primary schools as well as taught as a subject in the junior, secondary high schools and preparatory schools. Additionally, Kambaata language is vastly spoken oral language [24].

Kambaata belongs to the Highland East Cushitic, part of the Cushitic and the much bigger Afro - Asiatic language group [5]. The language is mainly spoken and institutionalized in Kambaata and Tambaaro Zone, located at 250 km south west of Addis Ababa, Ethiopia’s capital and situated at northeastern part of Southern Nations, Nationalities, and Peoples Region of Ethiopia. The language is also spoken by Kambaata migrants in other parts of the country and abroad.

The Kambaata people’s name and the language is available in numerous spellings in the literary works. The most frequent ones include Kambaata, Kambata, Kambatta, Kembata, Kembatta, Cambata, Cambatta [5], [23]. The people of Kambaata call their language by the name “Kambaatissata” or “Kambaatissa”. It is also called “Kambaatigna or Kambaatinya” (in Amharic-Latin script) or ከምባትኛ (in Amharic - Ge’ez script), and sometimes Kambatic (in English, just like the ‘ic’ ending of “Amharic or Arabic”) [5].

Kambaata dialects with their lexical similarity are Tambaaro (95%), Alaaba (81%) and Kabeena (81%) [23]. Kambaata also has higher lexical similarity with other Highland East Cushitic groups, i.e. Sidaamo (62%), Libido (57%), Hadiyya (56%), and Gedeo (54%) [23].

3.2 Kambaata Morphological System and Word Formation

Kambaata is strictly suffixing language with a rich nominal and verbal morphology [24]. It is an agglutinative language, where almost all derivational morphology and all inflectional morphology involve affixation. It has been emphasized in [24] that Kambaata is exclusively suffixing language and that there are no prefixes in the language. However, this study has discovered very few loanwords with prefix affixation. For instance, the Kambaata loan word “xaaf” ‘write’ from Amharic word “ፃፍ” ‘write’ can have multiple affixations including prefixes. By applying Kambaata morphological rules, different word forms can be created as indicated in Table 3.

Loan Word	Stem	Prefix	Suffix	Amharic	English
ma-xaaf-f-aachch	xaaf	ma-	-f-aachch	ከመፃሕፍት	From books
ma-xaaf-a	xaaf	ma-	-a	መጽሐፍ	Book (n)

TABLE 3: Prefix formation for loan word in Kambaata.

Inflectional affixes change stems with grammatical markers for things such as 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) [24]. Ideophones and interjections are morphologically invariant in Kambaata [5]; prepositions and conjunctions are totally unproductive for natural language processing purposes. This is also the case for adverbs which are negligible in number [24]. Therefore, the discussion of derivational and inflectional morphology concentrates on the main three parts of speech, namely verbs, nouns and adjectives in which the rules are constructed.

In Kambaata, nouns are inflected for genders and cases [5]. Verbs are inflected for tense aspect mood, person, gender, number and social status. Adjectives are inflected for genders and cases like nouns. Other word classes are morphologically invariant or not important for the application of NLP [24].

Word formation in Kambaata involves concatenated suffixes and irregularities through infixation, compounding, blending, and reduplication of affixes which is the main challenge for stemming Kambaata words.

4. KAMBAATA STEMMER

4.1 Corpus

Various text documents that contain the Kambaata words has been compiled. The corpus that is 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 the test data. The test data was collected from separate corpus to examine the algorithm from another corpus.

4.2 Normalization and Tokenization

All punctuation marks apart from ‘apostrophe’ (’), which is used as glottal sound marker ‘i’ (e.g. “asi’m” “look at”) and also used to separate successively occurring similar vowels in words (e.g. “ga’aa” ‘for tomorrow’), control characters, numbers and special characters are removed from the text before the data is processed. After punctuation marks and special characters apart from apostrophe have been changed to spaces, which is used to mark a word splitting-up border, all words were put in to separate lines in the tokenization process.

4.3 Compilation of Affixes

The known affixes of Kambaata language are suffixes, infixes, reduplication, compounding and blending. 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 be able to remove suffixes, but also remove infixes and transform irregular words to their stems as well. Without removing all these affixes, the stemmer cannot be effectively used to stem Kambaata documents.

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. Thus, collecting large data for affix analysis is considered as the ideal choice to compile the largest 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 irregular word formations that require context sensitive and substitution rules have been identified in the study.

The suffix collection ranges from simple suffixes; for instance, “ii”, “ikke”, “aan”, “indo” to concatenated suffixes; for example, “anniichchisin”, “eemmahanniichch”, “iishshoomaantassa”.

Occasionally, “-n-” and “-m-” are infixed in Kambaata words [25]. The “-uu-” is also the other infix known 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 most literatures that Kambaata is exclusively suffixing language [26]. However, the researcher believes that the existence of this prefix might be because of the loan word ‘xaaf’ which is the stem of the semantically related words to this stem.

4.4 Kambaata Stemmer Rules

The Kambaata stemmer has got two major components, the context-sensitive component and suffix removal component. The stemmer has 20 groups of suffix removal rules that remove 6299 suffixes with longest match first remove fashion. The stemmer is also context sensitive having 255 context sensitive and recoding rules to handle other word formations including irregular words.

Context Sensitive Rules

The three types of context sensitive actions defined and applied in the stemmer are:

Action 1: Don’t perform affix removal
Action 2: Transform or substitute the word with others partly or completely as specified in the rules.
Action 3: Remove affixes.

The three types of conditions are the following:

Condition 1: Check characters at the beginning and at the end of the word, compare with the rules if they match. If a match found, the stemmer transforms the word as per the specification. This is to avoid the removal of non-genuine affixes.

```
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";
}
```

FIGURE 1: Algorithm for condition 1.

The algorithm in FIGURE 1 is one typical example that replaces “ntaa” with “m” for a word “giphpha-ntaa” but this is not the case for words like “gaan-taa” and “gix-antaa” which start with the same letter ‘g’. The action taken for this condition is Action 2 when the condition is satisfied.

Condition 2: Words with character lengths of 2 or 3 should directly be taken as stems. These are words directly taken as output stems from test corpus if exist.

```
if (length of Word is 2 or 3) {
        return Word as stem;
}
```

FIGURE 2: Algorithm for condition 2.

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

Condition 3: A minimum stem length should be greater or equal to two characters. The stemmer removes the matching suffix if and only if the length of the remaining word is greater or equal to 2. This is to maintain the minimum stem length of the word in the language. In Kambaata meaningful word or stem has a minimum length of 2.

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

FIGURE 3: Algorithm for condition 3.

The algorithm in FIGURE 3, controls the suffix removal while removing matching suffixes.

Recoding Rules
Substitution rules are defined to handle some of the affixes individually. The algorithm in FIGURE 4 and example given in TABLE 4 are very few typical examples of this kind.

```
if ((word ends on ccano|jjo) && (if word starts with xaa)) {
        replace "can" or "jjo" by "z";
}
```

FIGURE 4: Algorithm for substitution.

For example, the pseudocode for the algorithm in FIGURE 4 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	if word stars with 'xaa'

TABLE 4: Substitution rule example 1.

However, if the word starting is changed to 'xuu', these endings will no more be replaced by the same character 'z'. Rather these ending will be replaced by 'd' to form a stem 'xuud' "see" which has completely different meaning from the previous word 'xaaz' "add".

Ending	Replaced by	Word	Stem	Condition
jjo	d	xuu-jjo	xuud	if word stars with 'xuu'

TABLE 5: Substitution rule example 2.

Suffix Removal Rules
To deal with each suffix individually, 20 groups of suffix removal rules are defined. The rules begin with stemming the longest suffix first and the smallest suffix last together with other conditions. Only one rule is introduced as an example in pseudo-code in Table 6 as follows.

```
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;
                }
}
```

FIGURE 5: Algorithm for suffix removal with suffixes of length two.

4.5 The Kambaata 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
```

FIGURE 6: Kambaata stemming algorithm.

The designed Kambaata stemming algorithm provided in FIGURE 5 works in the following way. First, a text file containing test data is opened and each word is 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 or 3 characters, the word is returned as a stem without going through the stemming process. If not (i.e. if the length of the word is greater or equal to 4), it adheres the context sensitive and recoding procedures according to the specific rules provided. If the word satisfies the conditions, it is transformed and recoded. Word to be stemmed is provided to suffix removal rules finally when it does not 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 (defined in step 1) to the shortest one-letter suffixes (defined in step 20) sequentially. The minimum stem length controlling 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 remaining words (after suffix removal) with character length of greater than 4 and if the last letter is double. The procedure continues reading next word until end of file (EOF) is reached. If EOF not reached, the process continues until all words are stemmed.

4.6 Evaluation of the Stemmer

To examine the performance of the stemmer, evaluations has been carried out. A separate evaluation data set was taken out randomly from the test corpus which was not utilized for the affix analysis. The evaluation test set contained 2425 distinct words. The corpus from which the rules of the stemmer derived was totally different from the test data. This is done deliberately in order to predict the efficiency of the stemmer in real scenario.

The result of the evaluation of the stemmer indicates that, out of 2425 words, 2349 words (96.87%) were stemmed correctly, 63 words (2.60%) were over stemmed and 13 words (0.54%) were under stemmed. The total errors account for 3.13% (76 words). As a result, the accuracy of the stemmer is 96.87% on the evaluation.

Word Compression

The stemmer is as well evaluated in terms of word compression ratio. For determining the word compression rate (C), or reduction of dictionary is calculated using the formula [27]:

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

Where,

C - is the compression value (in percentage)

W - is the number of the total words

S - is a distinct stem after conflation

The percentage of compression for Kambaata words based on the test set text for the stemmer is 100 * (2425- 828) / 2425 = 65.86%. From this result, it can be understood that the stemmer can reduce the morphological variants of words and the size of the file by 65.86% which is very significant reduction.

Over stemming and under stemming errors:

The main reason for errors in stemming Kambaata words is the language's rich morphology. This is due to complex morphology that makes use of concatenated suffixes, irregularities through infixation, compounding, blending, and reduplication of affixes.

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

- It is difficult to bring the full list of affixes mainly because of the rich morphological behavior of the Kambaata language.
- It is hard to define comprehensive list of context sensitive conditions and/or rules.
- Loan words such as forograammata (programme), tiraatiri (theater) and aksuumaakka (Aksumite) are not conflated correctly.

5. IMPACT AND SIGNIFICANCE OF THE STUDY

This research introduced a stemmer for Kambaata language words that helps the language's speakers to discover information of their need 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 and morphological analysis tools for Kambaata language.

Kambaata word processing tools could also need stemming algorithm that functions together with spell checker software to enhance the efficiency of spelling checking [28]. Kambaata word stemmer could also give an advantage of reducing the size of documents [12]. Because an individual stem usually corresponds to several complete terms, by storing stems rather than words, a data compression rate of 65.86 percent is also attained by this research work.

In Kambaata, a word has got quite large variants and conflating all these variants increases performance of the retrieval [10]. It also decreases storage space needed for index documents [29]. Moreover, the stemming algorithm could also provide advantages of designing tools such as term frequency counter and is used to reduce size of documents by decreasing word variations.

6. CONSLUSION & RECOMMENDATION FOR FUTURE WORK

6.1 Conclusion

Analysis of the morphology of Kambaata words shows that the language is rich morphologically (see Appendix). The types of affixations such as suffixes, infixes, reduplication, blending, compounding and concatenation of suffixes in the language contribute a lot in generating rich morphological variants and make the word formation process complicated. Therefore, attempting to conflate Kambaata words manually is very tedious and extremely difficult. For this reason, applying automated conflation procedure such as stemmer is very important for NLP applications.

In this study, a context-sensitive, longest match stemmer is designed using a rule-based approach for stemming Kambaata text. To apply the longest match technique, all possible long suffixes and basic suffixes were collected from the corpus. This stemmer does not just remove suffixes, but also takes exceptional scenarios into consideration and stems them by applying substitution and context sensitive rules.

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

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 other challenge is that the language is little explored regarding its linguistic feature, most importantly, its morphology.

In general, in this research, word conflation technique for Kambaata words has been explored. Rules have been defined; algorithm has been designed and implemented. The algorithm has also been tested and reported that it is effective and very fast by stemming 330 words per second.

6.2 Recommendation

In this research, stemming algorithm is attempted for stemming Kambaata words for first time. A significant move for future enhancement of stemming Kambaata words could be a study on stemming techniques for reduplicated, blended and compound words.

The researcher also recommends the following potential open research areas.

- Designing other NLP tools like morphological analyzer, machine translation, and automatic text summarization tools for Kambaata language using this stemmer.
- Research could also be conducted on designing Kambaata information retrieval system making use of this stemming algorithm.
- NLP applications need standard corpus preparation. Hence, preparing the standard corpus for Kambaata NLP researches could also be another research opportunity in the field.

7. REFERENCES

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

[2] Y. Fisseha, “Development of Stemming Algorism for Tigrigna Text,” Master’s Thesis, Addis Ababa University, Addis Ababa, June 2011, unpublished.

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

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

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

[6] 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.

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

[8] L. Lessa, “Development of stemming algorithm for Wolaytta text,” Master’s Thesis, Addis Ababa University, Addis Ababa, July 2003, unpublished.

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

[10] 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.

[11] 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.

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

[13] D. Tesfaye, and E. Abebe, “Designing a Rule Based Stemmer for Afaan Oromo Text,” *International journal of computational linguistics (IJCL)*, vol. 1, no. 2, October 2010.

[14] 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.

[15] 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.

[16] Rani, SP Ruba, B. Ramesh, M. Anusha, and J. G. R. Sathiaseelan, “Evaluation of Stemming Techniques for Text Classification,” *International Journal of Computer Science and Mobile Computing,* vol. 4, no. 3, pp. 165-171, 2015.

[17] P. Willett, “The Porter stemming algorithm: then and now,” *Program,* vol. 40, no. 3, pp. 219-223, 2006.

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

[19] 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.

[20] A. Ismailov, M.M. Abdul Jalil, Z. Abdullah and N.H. Rahim, “A Comparative Study of Stemming Algorithms for Use with the Uzbek Language,” *In proceedings of the 3rd International Conference on Computer and Information Sciences (ICCOINS)*, 2016.

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

[22] Anjali Ganesh Jivani et al, “A Comparative Study of Stemming Algorithms,” *Int. J. Comp. Tech. Appl.*, vol. 2, no. 6, pp. 1930-1938.

[23] “Ethnologue: Languages of the World,” *Ethnologue*, 2017. [Online]. Available: https://www.ethnologue.com.country/ET [Accessed: 12- Dec- 2017.

[24] Y. Treis, “Kambaata Numerals and Denumerals Revisited,” LLACAN.

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

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

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

[28] 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.

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