Hawassa University
ሀዋሳ ዩኒቨርሲቲ

MORPHOLOGICAL ANALYZER AND GENERATOR FOR KAMBAATISSA USING FINITE STATE TRANSDUCER

MSc. THESIS

BY: LIDIYA TADESSE GETISO

HAWASSA UNIVERSITY, HAWASSA, ETHIOPIA

MAY, 2023

MORPHOLOGICAL ANALYZER AND GENERATOR FOR KAMBAATISSA USING FINITE STATE TRANSDUCER

BY: LIDIYA TADESSE GETISO

MAJOR ADVISOR: WONDWOSSEN MULUGETA (PhD)

CO- ADVISOR: KITAW AYELE (MSc.)

A THESIS SUBMITTED TO THE

DEPARTMENT OF COMPUTER SCIENCE,

HAWASSA INSTITUTE OF TECHNOLOGY, SCHOOL OF GRADUATE STUDIES

HAWASSA UNIVERSITY

HAWASSA, ETHIOPIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

MAY, 2023

ACKNOWLEDGMENT

First of all, I would like to praise my God for his mercy on me and for giving me the strength to accomplish this study successfully.

I would like to express my greatest thank to my advisor Dr. Wondwossen Mulugeta staring from teaching Natural Language Processing course, he made me to get a good understanding about the course which inspires me to have an interest in the area and to start this study. He has given me a lot of support by showing direction and giving me valuable comments while conducting this research. In addition to this, I would also like to thank my Co-advisor Mr. Kitaw Ayele for his support and guidance throughout the study.

My heartfelt thanks go to Mr. Temesgen Senbeto, PhD Candidate at Linguistics department, at Addis Ababa University, and Dr. Desalegn, from Hawassa University Linguistics department for their kindly support and giving time for discussions on morphology the of Kambaatissa language and other linguistic information during this study.

I would also like to express my appreciation to Kambaata Xembaaro Zone Education biro, for helping to get student text books. Kambaatissa Department from Wachemo University Durame campus for helping me to access different language experts and to get different Kambaatissa books including student modules. Mr. Dagimlidet Bekele, application software developer who developed an android application of Kambaatissa dictionary for providing me list of words for verb part of speech. My special thanks also go to Wachemo University, my employer for giving me the chance of the master's program and financial support during my study.

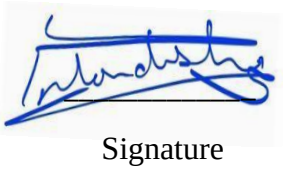
Last but never be the least, I would like to thank my beloved families, for their love, encouragement, prayer and support towards my study.

SCHOOL OF GRADUATE STUDIES

HAWASSA UNIVERSITY

ADVISORS' APPROVAL SHEET

This is to certify that the thesis entitled "***Morphological Analyzer and Generator for Kambaatissa Using Finite-State Transducer***" was submitted in partial fulfillment of the requirements for the degree of Master's Degree with specialization in COMPUTER SCIENCE for, the Graduate Program of the Department COMPUTER SCIENCE, and has been carried out by Lidiya Tadesse (ID NO. GPCoScR/0014/13), under our supervision. Therefore, we recommend that the student has fulfilled the requirements and hence hereby can submit the thesis to the Department.

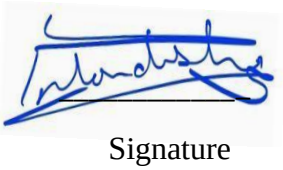
Wondwossen Mulugeta (PhD)		May 19, 2023
Name of Major advisor	Signature	Date
Kitaw Ayele (MSc.)	_____	_____
Name of Co-advisor	Signature	Date

SCHOOL OF GRADUATE STUDIES

HAWASSA UNIVERSITY

EXAMINERS' APPROVAL SHEET

We, the undersigned, members of the Board of Examiners of the final open defense by Lidiya Tadesse Getiso have read and evaluated his thesis entitled "Morphological analyzer and generator for Kambaatissa using Finite State Transducer", and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirements for the degree of Master's Science in Computer Science.

Wondwossen Mulugeta (PhD)		May 19, 2023
Name of Major Advisor	Signature	Date
Kitaw Ayele (MSc.)	_____	_____
Name of Co-Advisor	Signature	Date
_____	_____	_____
Name of Chairperson	Signature	Date
_____	_____	_____
Name of Internal Examiner	Signature	Date
Michael Melese (PhD)		May 16, 2023
Name of External Examiner	Signature	Date
_____	_____	_____
SGS Approval	Signature	Date

DECLARATION

I hereby declare that this MSc thesis is my original work and has not been presented for a degree in any other university, and all sources of material used for this thesis have been duly acknowledged.

Lidiya Tadesse Getiso	_____	_____
Name	Signature	Date

Table of Contents

LIST OF ACRONYMS AND ABBRIVATIONS

NLP	Natural Language Processing
DAWG	Direct Acyclic Word Graph
HEC	Highland East Cushitic
KX	Kambaata Xambaaro
SNNPR	Southern Nations, Nationalities and Peoples Region
POS	Part of Speech
IR	Information Retrieval
MT	Machine Translation
FST	Finite State Transducer
FSA	Finite State Automata
TLM	Two-Level Morphology
SFST	Stuttgart Finite State Transducer
HFST	Helsinki Finite State Transducer
XFST	Xerox Finite State Transducer
OS	Obstruent Sonorant cluster
SO	Sonorant Obstruent cluster
1prSg	First Person Singular
1prPl	First Person Plural
2prSg	Second Person Singular
2prPl	Second Person Plural

2prHon	Second Person Honorific
3prSgM	Third Person Singular Masculine
3prSgF	Third Person Singular Feminine
3prPl	Third Person Plural
3prHon	Third Person Honorific
IPV	Imperfective verbs
PVE	Perfective verbs with characteristic e-vowel
PVO	Perfective verbs with characteristic o-vowel
PROG	Progressive verbs
Neg.	Negation

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Kambaatissa is a Highland East Cushitic Language spoken in the Kambaata Xambaaro zone of South Nation Nationality and People Regional State, Ethiopia. It is a strictly suffixing and morphologically rich language. The language is one of the under-resourced languages in Ethiopia. For languages with complex morphology, nearly all computational work depends on the presence of tools for morphological processing. Many researches have been conducted in morphological analysis extensively for different languages, while this work is the first work in Kambaatissa natural language processing applications. This study focused on a morphological analyzer and generator, which is a lower-level natural language processing application that is used as a base for many higher NLP applications. A finite state transducer is a framework for modeling morphology. In this study, Foma is used as an implementation toolkit and lexc formalism for designing the lexicon. The experiment is done using 860 root verbs. There are seventeen continuous classes and forty different rules in the lexicon and foma file respectively. Result from the experiment shows that 92,020 words are generated among them the transducer gives 95.2% correct Kambaatissa verbs.

Keywords: *Finite state transducer, Kambaatissa, Morphological analyzer and generator*

CHAPTER ONE

1. Introduction

1.1. Background

Technology refers to methods, systems, and devices that are the result of scientific knowledge being used for practical purposes[1]. Language is one of the fundamental aspects of human behavior and it can be either spoken or written. It is the best way of exchanging information between human beings. Use of technology and its advancements for language processing aims at resolving many real world problems for human beings.

Natural language processing is a computational system that uses technology for understanding and generating human language. It is automatic or semi-automatic processing of human language that makes computers to recognize and process NL [2]. NLP has many applications like machine translation, speech recognition, sentiment analysis, part of speech tagging, Information retrieval, etc…

This study is interested in exploring the morphological processing of one of Ethiopia's local languages that deals with the structure of the words. Linguist uses morphological analysis to derive detailed grammatical information about words that enables the user to understand the meaning of the word as well as to produce comprehendible and correct words. Morphology is the study of the shape and the rules of combining morpheme as a subpart of a word that also carries lexical or grammatical meaning. A morpheme can be free or bound, based on the meaning that they give. Free morphemes can stand alone and have lexical meaning. While bound morphemes cannot stand alone and needs free morpheme to exist and change the grammatical meaning of the word. Such morphological information is useful for different Natural Language Processing Applications, like machine translation, speech synthesis, semantic analysis, POS tagger, spelling correction, dictionary development, grammar checking, and information extraction [3]. The computer needs a program to understand the word and related information to support the above tasks.

A Morphological processing studies how the internal structure of words and word formation of a language can be modeled computationally [4]**.** Such analyzer will return a lemma or a stem and grammatical information of the morpheme in a word.

A Morphological generator does the opposite of analysis that is, given a root word and grammatical information as input, it will generate a particular surface form of a word [5]. It constructs a word given its grammatical representation, which includes derivation, inflection, and combining word lexemes. Many pieces of research have been conducted on Morphological analysis and generation to several languages in the world for instance: Telugu, Arabic, Russian, and Ukrainian languages, and some of the research on local languages are Amharic, Afaan Oromo, Tigrigna, Ge'ez, and Wolaytta.

The study is focus on developing a morphological analyzer and generator for the Kambaatissa language.

Kambaatissa is a language that belongs to the Cushitic linguistic family which is Highland East Cushitic (HEC) language. According to the 2007 National census [6], the Kambaatissa language has around 600,000 native speakers but according to the Joshua Project website [7], the current number of speakers is more than one million. The speakers of the language live in the high land areas around the Hambarichcho massif. It is spoken in the Kambaata-Xambaaro-Zone of the Southern Nations, Nationalities, and Peoples Regional State. It is about 300km southwest of Addis Ababa [6]. The Kambaata people refer to their language by the term Kambaatissa, which includes the derivational formative -issata for names of or as Kambaati afoo "the mouth of Kambaata" [8]. The language has a phoneme-based orthography in the Latin script which is developed in 1986 E.C. (1993) [8].

Kambaatissa is a rigorously suffixing language having very productive verb morphology. The stem of a noun consists of a root plus derivational morphemes [9]. It is characterized as agglutinative or concatenative, i.e., words are formed by adding suffixes to the root word in a series. A single word might have many affixes which are attached to the root. It needs a morphological analysis to make it suitable for different applications and use in digital technology. For languages with complex morphology, almost all computational work depends on the existence of tools for morphological processing. The absence of a morphological analyzer for the Kambaatissa language is creating a problem in the use of the language in digital devices and systems. This study investigates Finite State Transducer approach for Kambaatissa morphological analysis and generation.

1.2.Research Motivation

Currently, there are no NLP applications for the Kambaatissa language that makes the language to be used in a digital system. The lack of those applications is resulting in different problems during retrieving documents and writing texts on the computer using the language. A morphological analyzer is one of those applications used to generate surface word forms found in everyday communication, from lexical components. Developing a morphological analyzer is the foundation for the future development of other NLP applications. Some NLP applications use a morphological analyzer's output as an input; For example, a spell checker can use for a suggestion list and information retrieval systems can use it for a searching keyword. Developing the morphological analyzer and generator is one of the contributions to the development of the language and the researcher as the speaker of the language wants to contribute some relevant work to make the language more suitable for other NLP applications.

1.3. Statement of Problem

Many researches have been conducted on morphological analysis and generation of several languages in the world using different techniques. Morphological Analyzer for Telugu Using Support Vector Machine [10], morphological generator for the Tamil language [11][12], Morphological Analyzer and Generator for Russian and Ukrainian Languages[13], Malayalam morphological analysis at the character level using a deep learning approach [14] are some of them. There are also researches on local languages like Amharic [15] [16], Afaan Oromo morphological analyzer using machine learning [17], and A Finite-State Morphological Analyzer for Wolaytta [4] Morphological Analyzer for Sidaamu Afoo [18], morphological analyzer for Ge'ez verbs [19]**.**

Although a morphological analyzer is developed for many different languages around the world and also for local languages using various techniques, it cannot be applied to the Kambaatissa language because each language has its unique orthographic representation and morphological structure which means the word formation is not similar for all languages. The way of combining morphemes in a word such as attaching affixes to the root and the number of affixes used are different from language to language. Even if the languages belong to the same language family there are differences in word formation, concatenation, alternation and other rules. The plural is marked by {-aka-ta} in Kambaatissa like - ama-ta // ama-aka-ta (mothers) and when it comes to Afaan Oromo

there are different ways of marking plural nouns and among which {-ota/o:ta} is more frequent like: - mana // man-o:ta (houses). This implies morphology is language-specific and the model developed for one language might not fit the other.

As far as the knowledge of the researcher is concerned, there is no prior study conducted on developing a morphological analyzer for the Kambaatissa language. Kambaatissa is suffixing language with very productive verbal morphology [21] and this study focuses on verb morphological analysis. The language has no large collection of prepared corpus for natural language processing and this led to development of rule based morphological analyzer.

The absence of a morphological analyzer and generator for the Kambaatissa language is creating a problem in the use of the language in digital devices and systems. Other advanced NLP application may greatly relay in morphological analyzers and it absence will impede the advancement of the language in the digital ecosystem. When someone wants to write Kambaatissa language text using his computer or other electronic devices, he may face the problem of spelling errors and those errors results in a change in the meaning of the word. This problem can be solved using a spelling checker application which needs morphological processing of the language. On the other side, for languages with complex morphology, almost all computational work depends on the existence of tools for morphological processing [10]. Kambaatissa is one of the languages which has complex morphology so the absence of an efficient morphological analyzer and generator creates a negative impact on developing information retrieval systems and other language processing applications. Nowadays many translation works have started from other languages to Kambaatissa language such as Bible translation, proverbs, and other books. So, the absence of a Morphological analyzer affects the machine translation systems by minimizing the quality of translation, and also it may lead to incorrect translation which means the meaning of the translated word will be wrong.

For this reason, the researcher has proposed to put a milestone by developing a morphological analyzer for the language to flourish in the natural language processing area.

1.4. Research Questions

In general, by designing a morphological analyzer and generator the following research questions will be answered:

1. How to approach computational aspects of the verbs and what are the challenges in the language while using FST?
2. To what extent the FST approach enables the morphological analyzer and generator to generate and analyze kambaatissa words correctly?

1.5. Objective

1.4.1. General Objective

The main objective of this research work is to design a morphological analyzer and generator for the Kambaatissa using Finite-State Transducer

1.4.2. Specific Objective

The specific objectives of the research work will be the following:

- To collect documents to identify word-formation structure in the Kambaatissa language.
- To prepare root verbs with their affixes from language experts.
- To design a morphological analyzer and generator for Kambaatissa.
- To develop a prototype of the designed morphological analyzer and generator.
- To evaluate the performance of the developed prototype

1.6. Significance of the Study

The development of a good morphological analyzer and generator for the Kambaatissa helps in the next higher-level applications. Many researches show that languages that have agglutinating nature requires morphological analysis before further processing in NLP applications due to the complex morphology of the words. As is described above Kambaatissa language is agglutinative so it is necessary to have an efficient morphological analyzer and generator. It can be used by other researchers as a component for different higher-level NLP work like part of speech tagger, grammar checker, machine translation, and others.

Kambaatissa language is highly inflectional (morphologically complex) and resource-limited so, the proposed morphological analyzer and generator will be used in machine

translation to reduce data sparseness and improve translation quality. It also solves one of the problems of lexical resource scarceness.

The proposed morphological generator plays a vital role in target language sentence generation in Machine Translation (MT) systems. It is also used in Kambaatissa Information Retrieval (IR) systems in the front-end for query expansion and to increase the recall of search engines.

In the Kambaata-Xambaro zone, there are many employers from other areas of the country and the existence of a morphological analyzer and generator helps them to know the language easily.

1.7.Scope and Limitation of the Study

1.6.1. Scope of the Study

The scope of the study is limited to designing a morphological analyzer and generator for Kambaatissa using a Finite-State Transducer. It focuses particularly on verb word class because the language has rich verbal morphology. The study covers both derivational and inflectional morphology.

1.6.2. Limitation

There are two types of word classes in the Kambaatissa language open word class and closed word class. In the open word class, there are noun, adjectives, ideophones, and interjections and in the closed word class, there are Pronouns, numerals, demonstratives, conjunctions, and adverbs. The above part of speeches are not covered by the study. There is very limited amount of linguistic resource on derivational morphology, this makes the analysis on derivational morphemes difficult.

1.8. Thesis organization

The first chapter contains the background of the study, motivation, the problem to be solved, the research question, the general and specific objectives, the scope of the study, and limitation of this study.

The second chapter is about review of different literatures on concepts of morphology. It also discusses about different approaches to morphological processing. Additionally, related works in local language as well as foreign languages are reviewed. Finally different

researches relevant to this study are properly reviewed, strong sides and gaps are also identified.

The third chapter is overview of Kambaatissa language, in this section the phonological and morphological nature of the language are reviewed. Verb morphology of the language is described in detail.

The fourth chapter is about the materials and methods used to achieve the objective of the study. Data collection, Research method, Implementation tools and performance evaluation are discussed here.

The fifth chapter is about the proposed design of the morphological analyzer. In this chapter the architecture of the proposed morphological analyzer and generator is discussed, and each components of the architecture is explained.

In chapter six, the implementation, result of the experiments, and performance evaluation result are presented.

Finally, chapter seven presents the conclusion of the study, recommendations, future work and contribution of this study.

CHAPTER TWO

2. Literature Review

2.1. Introduction

In this chapter, important literature on morphology and different approaches for morphological analysis and generation are reviewed. Section 2.2 discusses concepts and terminologies of morphology and Section 2.3 discusses types of approaches to morphological analysis and generation that are important to this study and the last section 2.4 is about related works that are conducted on local language as well as foreign languages.

2.2. Concepts and Terminologies

2.2.1. Introduction

The term morphology is made up of the Greek word morph-, which means 'shape/form' and morphology is the study of forms, in particular the forms of words. It is the branch of linguistics that deals with how words are formed and their internal structure [3].

Theoretical morphology deals with the internal structure of words. Computational morphology is intended to perform the task of morphological analysis automatically with the use of computers and computational methods. Generally, the task of computational morphology is surface word analysis and synthesis [19].

2.2.2. Morpheme

A morpheme is minimal meaningful elements that make up a word. It is the smallest unit of a language that has a meaning and it cannot be further divided into smaller meaningful parts. The term morpheme came from two Greek words, **morphos** mean "form or structure" and **eme** means "an element or little piece of something". A morpheme can be the whole word, part of a word, or a single phoneme and a single word can contain one or more morphemes [22]. Allomorph is the variant forms of a given morpheme. When we want to refer to a smallest grammatical form merely as a form, we will use the term morph [23]. Morphemes are classified in to free and bound morpheme. A free morpheme is a

morpheme that can appear as a word by itself which means it can stand alone. Bound morphemes are morphemes that must be attached to a free morpheme to give a meaning. In the English language the word *'eat'* is a free morpheme that is made up of a single morpheme but the word *'eating'* has a morpheme 'eat' which is a free morpheme and the morpheme '–ing' which is a bound morpheme. Words are classified into three classes based on the number and type of morphemes they contain those are: simple, complex, and compound. Simple words contain only one free morpheme, complex words consist of more than one bound morpheme and compound words are words that contain two free morphemes.

2.2.3. Affixes

Affixes are morphemes that are attached to a base word to give grammatical meaning or to form a new word. Words in morphologically complex language have many affixes so they need to be combined correctly to give appropriate meaning. Affixes are classified into four based on the position they attached to the base word [24].

Prefixes

Prefixes are affixes that come at the beginning of the word to produce grammatical meaning. For example: legal = illegal, Countable = uncountable. In the example, the affixes /il-/ and /un-/ are prefixes of the words.

Suffixes

Suffixes are affixes that are attached at the end of the word to form a new word. For example, The English words swim= swimming and computer= computers. Here the affixes /-ing/ and /-s/ are suffixes of the words.

Infixes

Infixes are affixes that occur in the middle of the given word. They can be anywhere except at the beginning and end of the word.

Circumfixes

Circumfixes are affixes that are attached at the beginning and end of the word at the same time.

For example, the Amharic word መጣ [he came] =አልመጣም [he hasn't come]. Here the affixes /አል-/ and /-ም/ occurred as prefix and suffix at the same time so they are called a circumfix.

2.2.4. Morphotactic

Linguists classified languages into different categories based on the word formation properties of the language and the way of combining morphemes in a word [25]. There are three categories of language isolating/analytical, agglutinating and fusional. Isolating languages are language whose words are made of free morphemes and no affixation is allowed. In agglutinative language the words are built from more than one morpheme and each morpheme carries distinct grammatical information but the words in fusional language contain morphemes that carry more than one grammatical information and perform more than one morphosyntactic role [26][24].

Morphotactic is a rule in a language that is used to control the order in which morphemes are combined to form a meaningful word [27]. It is about the possible sequence of morphemes to form a word that is acceptable in the language. For instance the English word "immunization" is formed from the morphemes 'immune-', '-ize' and '-ation'. The morphemes must be combined in the order of "immune-ize-ation" and exchanging the order of these morphemes not only changes the meaning of the word but also it will make the word meaningless and out of the rule in the language. Morphemes are the smallest unit that carries a meaning so, to get the correct meaning the morphemes have to be arranged according to the rule of the language. In addition, understanding the order of morphemes in the word-formation will make the morphological analyzer easily identify the arrangement of morphemes when it finds a complex word [28].

2.2.5. Types of Morphology

There are two ways of word formation, those are inflectional morphology and derivational morphology [24].

Inflectional Morphology is a way of forming words by combining a stem word with a grammatical morpheme to get a word that has a similar part of speech to the original stem

word [23]**.** The English word ‘wash’ can have the inflection form ‘washes’, ‘washing’, ‘washed’. Here the inflectional morphemes attached to the stem word are ‘-es’, ‘-ing’, and ‘-ed’. These morphemes did not change the word class of the stem word which is a verb. An inflectional morpheme adds some kind of grammatical information to the stem word and this grammatical information can be the case, gender, tense, mood, number, and aspect. In the above example, the inflectional morphemes attached to the verb stem add the grammatical information of third-person singular, present continuous tense, and past tense respectively.

Derivational Morphology is the other technique used in the process of word formation. It deals with combining a grammatical morpheme with a stem word to get a word in different parts of speech. It is a way to create a new lexeme from a given word by adding a suffix to it [17]. Derivational morphemes are affixes that change the part of speech of a word when attached to a word stem. In the English language, the word ‘friend’ is classified in the noun word class but when the derivational morpheme ‘-ly’ is attached to it the noun will become an adjective ‘friend-ly’.

2.3. Approaches to Morphological Processing

Computational morphology is a branch of computational linguistics that deals with word structure. It is concerned with designing a software that analyzes or generate words as complex objects that are identified by linguists [32]. It is a way of performing the task of morphological processing using computer programs. Morphological analysis is the action of breaking down the surface word in to its component morpheme and analyze their grammatical features. It accepts the surface form of the word as an input and create the lexical form of it as an output. The morphological generator performs the inverse of the analyzer. For instance, the word working is a surface form in English language, the morphological analyzer breaks it into the morpheme ‘work-’ and ‘-ing’ which is the lexical output. Morphological generator takes the lexical form ‘work-ing’ as an input and generate the surface form ‘working’ as output.

2.3.1. Morphological analyzer and generator

Morphological analysis is the process of segmenting a word into its component or morphemes. It analyzes the internal structure of words. It deals with retrieving the syntactic and morphological properties of a complex word. A morphological analyzer is a tool that

identifies the internal structure of a given word and returns the morphological and grammatical information related to the word. It takes a word as input and produces the stem word with its grammatical information as an output. A morphological generator is the reverse of an analyzer. It is the process of making a word from its lexeme and grammatical representation. Morphological analysis is the first step for all Natural language processing.

NLP research groups have developed many approaches and algorithms for word processing. Overall survey on different approaches applied to morphological analysis and generation includes the following.

a) Finite State Automata

This approach is used to accept or reject a string in a given language [33]. It describes a class of models of computation that are characterized by having a finite number of states. It is a model of behavior composed of state, transitions, and actions. It represents information using its state. State changes in response to inputs. Transitions are rules that tell how the state changes in response to inputs [20]. It starts working from an initial state and when it reaches any one of the final states it accepts its input or stops working. It is defined as 5 tuples: A= (Q, q0, ∑, δ, F)

- Q - set of finite states including start state, final states, and normal states
- q0 - is the start state
- ∑ - is a set of a finite number of inputs
- δ - is the transition function that shows when an automaton moves from one state to the next state
- F - is the finite state of final states

b) Two-level morphology

Two-level morphology is a general computational model for word-form recognition and production. This approach is developed by a Finnish computer scientist known as Kimmo Koskenniemi in 1983 [33]. It is the first general model in the history of computational linguistics for the analysis and generation of morphologically complex language. It assists the arrangement of rules that relate pairs of surface strings through systematic rules [34]. The two basic components are finite-state automata components in which rules are encoded and dictionary components in which Roots and affixes are listed [20].

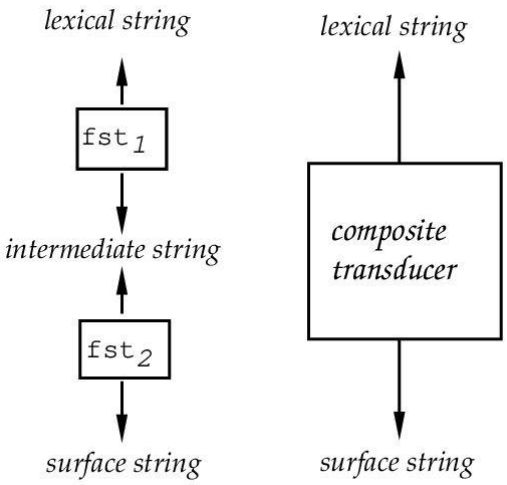


Figure 1: Two level morphology adapted from (L. Karttunen, R. M. Kaplan, and A. Zaenen) [35]

c) Finite State Transducer (FST)

FST is the advanced version of finite-state automata with two tapes one input tape and an output tape. It reads the input from the input tape and writes the output to the output table and generates the relation between the input string and output string and vice versa. It is bidirectional and able to simultaneously generate all possible word forms for a given stem and parse given inflected word forms [36].

The result of traversing FST during morphological analysis is not only an output character sequence, representing the root of the word but a set of feature-value pairs that represent the grammatical structure of the input word. The output in the generation direction is surface word form corresponding to the input root and grammatical structure [5]. FST relates the surface and lexical levels directly, as in two-level morphology, and because of the reversibility of FSTs, it is a simple matter to convert an analysis FST to one that performs generation [5].

Formally, a finite state transducer is defined as the collection of 7 tuples:

(Q, q0, ∑, Λ, δ, λ, and F) where:

- Q - is a set of finite state
- q0 - is the initial state
- ∑ - is a set of the input alphabet
- Λ - is a set of output alphabets

- δ - is a set of transition functions (Qx∑->Q) that map two states and an input alphabet to state.
- λ - is a transduction function (Qx∑-> Λ) that maps state and input alphabet to output alphabet.
- F – is a set of the final state

FST is an improved version of the finite-state automation model based on the two-level morphology principles. It has input: output pairs on labels which are also called input and output tapes. The transducer transition is accompanied by (T= {a:a, b:b, c:c,…})

d) Direct Acyclic Word Graph (DAWG):

In this approach, the process of analysis/synthesis is reduced to a search in the graph. It is a data structure that keeps sorted characters of words using edges and nodes. The edges of the graph represent the characters of the words and nodes represent the branching point or terminal. The words can be separated into the base and affix parts to represent on the graph. This approach is language-independent and does not employ any morphophonemic rules or any other special linguistic information [37]. It minimizes the usage of memory space of the words which have similar affixes in a common. It can compactly store finite strings of information, taking advantage of common prefixes and suffixes in the strings [38]. It is an efficient data structure for lexicon representation and fast string matching. There are two types of DAWG namely deterministic and non-deterministic. The deterministic DAWG has no more than one outgoing link from the node with the same symbol but the non-deterministic DAWG can have more than one outgoing link So, it can perform both morphological analysis and generation [20].

e) Paradigm approach

This approach classifies each word class in the given language into a specific type of paradigm and it develops morphological analysis and generator model based on their morphophonemic behavior [1]. It uses tables that contain word forms that cover the words in the language and this table is prepared by a language expert. The table depicts the relationship between stem and all word forms. The root word follows a certain pattern (paradigm) to generate word form and let to differentiate its categories [39].

f) Rule-based approach

The rule-based approach is used to incorporate domain knowledge into linguistic knowledge which provides highly accurate results [40]. It is based on a set of linguistic rules and a dictionary that contains roots and morphemes. The morphological analyzer performs matching between the input word and the rule which is already defined and if the match happens, then the system will give back the root and grammatical structure of the word from the dictionary. The advantage of this approach is that, Even if there are limited or few resources, the morphological analysis can be done with reasonable accuracy. It highly relies on linguistic theories so, systems developed using this approach, are often efficient and produce better quality outputs. If the rules are correctly defined it achieves the highest accuracy. The drawback of this approach is that if the input word is not present in the dictionary it fails and the other one is every rule depends on the previous rule So that the failure in one rule will affect the entire rule that follows it.

g) Corpus-based approach

Corpus is a massive collection of written text that belongs to a particular language [33]. Machine learning algorithms train the corpus and generate rules that are used for morphological analysis. They extract statistical information and other necessary features from the corpus [37]. This approach doesn't require any hand-coded morphological rules but it needs only the corpora with linguistic information. The time consumption for corpus preparation is one of the disadvantages of this approach. The other drawback is that the performance of the analyzer highly depends on the size of the corpus[20]. Additionally the corpus based approach has the disadvantage that large volume of data needs to be processed [41].

h) Suffix stripping

This approach is used when the suffix is identified well, the stem word can be obtained by removing that suffix and applying proper orthographic rules [37]. The suffix stripping algorithm-based morphological analyzer and generator can be implemented using a dictionary for storing stem and suffix morphemes and using morphotactics. In this approach, it is easy to analyze words if the suffixes are known well [42]. It is also called stemming based approach. It follows the technique of dividing the word to its corresponding stems and the suffix or prefix and then process them. It needs stem

dictionary to analyze the words. This approach may successfully analyze regular words but it cannot analyze irregular words [41].

2.4. Related work

2.4.1. Local Languages

Morphological analyzer for Amharic, Oromo and Tigrinya

Gasser [5] developed a HornMorpho (a set of Python programs to analyze and generate words in Amharic, Tigrinya, and Oromo). The study is conducted using the Finite State Transducer approach The researchers collected 200 Tigrinya verbs, 200 Amharic, and 200 Amharic nouns and adjectives to evaluate the performance of the developed analyzers and the evaluation of the results was done by a human reader familiar with the language. Finally, they have got an accuracy of 96% and 99% for Tigrinya verbs and Amharic verbs respectively and 95.5% for Amharic nouns and adjectives. The system was designed to generate 10 to 25 verbs depending on the range of forms using a total of 330 tests and the system attain accuracy of 100% and 93% for Amharic and Tigrinya respectively. The study has not incorporated noun analysis and generation for the Tigrinya language.

Finite State Morphological analyzer for Wolaytta

The first complete morphological analyzer for Wolaytta language was developed by using Finite State Transducer [43]. The study explains that Finite State transducers are the most efficient approach for developing morphological analyzers for highly inflectional languages. The system is implemented using the Helsinki Finite-State Transducer toolkit (HFST). The study used lex formalism to implement the morphotactic of the language and twol formalism for the implementation of morphophonological rules. The researchers got a precision of 94.85% and a recall of 94.11% by evaluating the transducer over a manually verified test set.

Morphological analyzer for Afaan Oromoo

Moyka Degefa from Addis Ababa university conducted research on Morphological Analyzer for Afaan Oromoo using a machine learning approach [17]. He proposed a novel Morphological analyzer based on memory-based learning algorithms IB1 and IGTREE. The researcher developed a morphological database, which consists of grammatical information on Afaan Oromoo nouns, verbs, and adjectives to train the model. The

morphological database contains 2270 annotated words. Finally, the researcher obtained a maximum generalization accuracy of 98.86% from IB1with interleaving and 94.36% from IGTREE with feature selection and interleaving.

Automatic Morphological analyzer for Ge'ez Verb

This study is conducted by Desta Berihu Weldegiorgis to develop Morphological analyzer for one the typical language of Ethiopia that is Ge'ez [19]. The researcher used rule-based approaches specifically CV-based and Two-Level Morphology (TLM). The study enlightens that a morphological analyzer is an important component for all natural languages. The developed prototype was tested using verbs which are extracted from New Testament books of the Ethiopic version Bible. The researcher checks the accuracy of the developed analyzer by comparing the analyzer's output with the analysis performed by language expert and he has obtained 92.05% at feature level and 73.98% at verb level.

Morphological Generator and Analyzer for Sidaamu Afoo verbs

A morphological generator and analyzer for Sidaamu Afoo, which is the official language of Sidaama Regional State is designed by Kitaw Ayele [42]. The system was developed by using the Finite State Transducer approach through Foma implementation software. To test the developed generator and analyzer the researcher collected 840 root verb words then the system generated total of 74, 934 words and out of those words 94.8% are correctly generated and analyzed. The system wrongly generated and analyzed 5.2% of the total generated words.

Amharic Morphological Analyzer

Mesfin Abate and Yaregal Assabie developed a morphological analyzer for one of the highly inflectional languageAmharic [15]. The researcher used a memory-based supervised machine learning approach. It considers the morphological analysis as a classification problem. The performance of the analyzer was evaluated by using a corpus that contains 1022 words and out of the total words 841 are verbs and 181 are nouns. The researchers applied IB1 and IGtree algorithms using 10-fold cross-validation. Finally the obtained accuracy was 93.6% in IB1 and 82.3% in IGtree.

2.4.2. Foreign languages

Morphological analyzer and generator for Tamil

Kengatharaiyer Sarveswaran, Gihan Dias and Miriam Butt developed ThamizhiMorph which is a morphological parser for the Tamil language particularly for noun and verb word classes [12]. The approach used to develop the morphological analyzer and generator is Finite State Transducer and implemented using Foma toolkit. The reason for choosing this approach is that, FST has given good success in the previous researches on morphologically rich south Asian languages and it can be used for morphological generation too. The researchers assessed the performance of the system (ThamizhiMorph) by conducting comparative evaluation first by using texts from Tamil text book and the second evaluation was Tamil universal dependency Treebank version 2.5. Finally, the researchers have got 84.7% successfully analysed words. The researchers tried to list the reason for errors and the main reason is that the presence of out of vocabulary words make some words to be analyzed incorrectly.

Morphological generator for Tamil

A morphological generator for Tamil nouns and verbs [44] is developed by Mena S., Vijay Sundarram, and Sobha Lalithadevi. The researchers used the Finte State Automata approach to develop the morphological generator. The developed generator was evaluated using the parameters Precision and Recall. The results from the experiments conducted show that a finite-state-based morphological generator is well-suited for inflectional languages and the researchers got a precision of 99.7% for both noun and verb roots.

Morphological analyzer for Paraguayan Guarani

Anastasia Kuznetsova and Francis M. Tyers developed a morphological analyzer for an agglutinative language, Paraguayan Guarani that is spoken in south America [45]. As the researchers explained, machine learning approach needs large amount of data for training and in the case of supervised approaches, those data should be annotated by trained experts. Because of the above issues and the absence of large amount of data leads the researchers to apply Finite State Transducer approach which is based on the formal linguistic description. FST approach is state-of-the-art for such type of languages. The Morphological analyzer was implemented by HFST toolkit and the study used lex formalism for implementing the morphotactic and twol formalism to the phonological rules

of the language. The performance of morphological analyzer was evaluated by using two publicly available corpora and finally they have got an accuracy between 86% and 91%.

Morphological Analyzer for Hindi Language

Deepak Kumar, Manjeet Singh, and Seema Shukla developed a morphological analyzer for Hindi language [41]. The approach used in this study is FST which is the most efficient approach to develop morphological analyzer for highly inflectional languages. In highly inflectional languages we can generate more than one hundred words from a single root so the presence of Morphological analyzer helps to understand the words in the language. The study used SFST (Stuttgart Finite State Transducer) as an implementation software. The researchers tested the morphological analyzer by using 400 inflectional, derivational and compound words. The developed morphological analyzer is also used in part-of-speech tagger which is tested on 100 sentences.

Finite-state-based morphological analyzer for Turkish

This study develops a morphological analyzer for Turkish language and the system is called TRMOR [46]. The study used a finite state transducer method and SFST as implementation toolkit. The study used a finite state transducer method and SFST as implementation toolkit. The system used a lexicon file that contains 36,903 entries and it has performed 94.12% precision.

Morphological Analyzer for Malayalam

A morphological analyzer for Malayalam, highly agglutinative and inflectional language, is developed by Santhosh Thottingal using Finite State Transducer [28]. The developed morphological analyzer is called as Mlmorph. It is implemented using SFST (Sttutgart Finite State Transducer) formalism and Helsinki FST technology as a toolkit. The researcher mentioned that the developed analyzer is fast and efficient to deal with the morphology of the language. The system performs 85% accuracy for analysis of words in the language.

The following table summarizes the above related work review

Table 1: Summary of Literature review

Author(s)	**Year**	**Title (Topic)**	**Language**	**Method**	**Result**	**Gap**
Michael Gasser	2019	HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya	Amharic, Oromo, and Tigrinya	Finite state transducer	96% for Tigrinya verbs, 99% for Amharic verbs and 95.5% for Amharic noun and adjective	-Limited number of roots for Tigrinya - The study has not incorporated noun analysis for Tigrinya
Tewodros A. Gebreselassie, Jonathan N. Washington, Michael Gasser, and Baye Yimam	2018	A Finite-State Morphological Analyzer for Wolaytta	Wolaytta	Finite state transducer (HFST)	Precision 94.85% and recall 94.11%	-The system analyzes only native Wolaytta words (loan words not included)
Moyka Degefa Mosa	2020	Morphological Analyzer for Afaan Oromoo Using Machine Learning	Afaan Oromoo	Memory Based Learning algorithms IG1 and	98.86% IB1 and 96.36% from IGTRE	-The system performs only analysis

				IGTREE	E	and compound words are not included
Mesfin Abate and Yaregal Assabie	2014	Development of Amharic Morphological Analyzer Using Memory-Based Learning	Amharic	Memory-Based Learning	93.6% with IB1 and 82.3% with IGTREE	-Less amount of corpus is used
Kengatharaiyer S. , Gihan D. and Miriam Butt	2021	ThamizhiMorph: A morphological parser for the Tamil language	Tamil	Finite-State Transducer	84.7%	-The system failed to guess lemma associated with 12 words - Error due to out-of vocabulary items
Anastasia Kuznetsova and Francis M. Tyers	2021	A finite-state morphological analyzer for Paraguayan Guaraní	Paraguayan Guarani	Finite state transducer	naive coverage between 86% and 91%	-The analyzer did not perform nasalization

CHAPTER THREE

3. Overview of Kambaatissa Language

3.1. Phonological Overview

Kambaatissa represents 28 consonant phonemes and 5 vowels which are short and long [47].

Vowel Inventory

Kambaatissa language has a characteristic Cushitic five-vowel system. Vowel quantity is contrastive. Long vowels are bi-phonemic, i.e. represented by combinations of two short vowels. Vowel length is indicated by the use of double letters, e.g. /i:/ = /ii/. The vowel phonemes occur word-medially and word-finally after any consonant.

Table 2: Vowel inventory

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

Consonant Inventory

Kambaatissa language has 28 phonemes in its consonant inventory. These phonemes are listed in the following table with their place and mode of articulation.

Table 3: Consonant phonemes and their orthographic representation adapted from (B. Desta) [19]

		Labial	Alveolar	Palato-alvolar	Velar	Glottal
Stops	Voiceless	<p>⊗	t	<ch>	k	(<’>)
	Voiced	b	d	<j>	g	
	Glottalic	<ph>	<x>	<c>	<q>	
Fricative	Voiceless	f	s			
	Voiced	<v>⊗	z	<sh>		h
	Glottalic		<ts>⊗	<zh>		
Nasals		m	n	<ny>		
Vibrant	Plain		r			
	Glottalized		<’r>			
Laterals	Plain		l			
	glottalized		<’l>			
Glides		w		y		

3.2.Morphophonemic

Morphophonology deals with the phonological structure, properties, and relations of morphologically complex words [30]. It deals with the phonemic alterations when a morpheme is inflected with stems. The sound alternation is situated at the interface between phonology and morphology. The morphophonemic process helps to act regarding the syllable constraints conveyed in the language. It is highly noticeable in the verbal domain and it affects the consonants that meet at the boundary between verbal stem and inflection (the first person marker). It is used to avoid illegal consonant clusters resulting from the suffixation of inflectional morpheme to the verbal stem. Kambaatissa doesn’t allow a word-initial or final cluster of consonants. The only possible consonant cluster is in

the word medial position which is a cluster of two consonants and both consonants have to be identical or in which one consonant is a sonorant [31][30]. The morphophonemic process is triggered by /t/- and /n/- initial personal markers. Kambaatissa has two types of morphophonological processes. The first one is morphophonological process-I and it is observed in any verbal paradigm and is classified into Epenthesis, metathesis, and assimilation. These morphophonemic processes are used by all the Highest East Cushitic languages. The second is morphophonological process-II. It is applied to 1PrSg and 3M forms of perfective paradigms and classified into gemination and palatalization.

3.2.1. Epenthesis

It is observed when a sound is inserted into a word [30]. When a stem that ends with a consonant cluster connects with a consonant initial personal marker then the insertion of the epenthetic vowel /i/ helps to avoid the cluster of three consonants and to keep the linguistic rule. It is represented in the following way:

(i) CC + t ⟶ CC-i-t

(ii) CC + n ⟶ CC-i-n

Examples:

Verb	**Verb stem**	**CC- C**	**Glosses**
toll- t-áanti	toll-	tollitáanti	‘you will stretch out’
fint- n-óommi	fint-	fintinóommi	‘we listed’

The word “tolltáanti” is not allowed in the language because three consonant clusters /llt/ is illegal according to the rule in the language. Therefore the epenthetic (i) need to be inserted between the verb stem /toll-/ and the 2nd person singular marker /-t-áanti/.

Condition 1:- When a consonant-initial suffix is attached to a verbal stem that ends in pre-glottalized sonorants:

Verb	**Verb stem**	**CC- C**	**Glosses**
’aa ’l- n-áammi	’aa ’l-	’aa ’lináammi	‘we will wash’

Condition 2:- When a C-initial suffix is attached to a verbal stem that ends in a geminate glottal stop:

Verb	Verb stem	CC- C	Glosses
ga’’- n-áam	ga’’-	ga’’ináam	‘we will call’
ga’’- t-áa’u	ga’’-	ga’’itáa’u	‘they will call’

Condition 3:- Epenthesis is also used to avoid an unlicensed word-initial cluster in conditions where C-initial personal markers are attached to mono-consonantal stems such as: /sh-/ ‘kill’ and /y-/ ‘say’.

Verb	Verb stem	CC- C	Glosses
sh- teen-óta	sh-	shiteenóta	‘in order to kill’
y- t-áau	y-	yitáau	‘they will say’

Epenthesis is a well-known morphophonemic process in Kambaatissa and it is not restricted to verbal inflection but it always takes place when a sequence of more than two consonants come together in the morpheme boundary.

Epenthesis in Nominal domain

Epenthesis is also on noun inflections in the condition where a nominal stem which ends with an obstruent connected to the singulative suffix /- chchú/

Noun	Noun stem	CC- C	Glosses
ga’- chchú	ga’-á	ga‘ichchú	‘wild cat’

3.2.2. Metathesis

Metathesis is a morphophonemic process that transposes the position of consonant sounds. In Kambaatissa language when a verb stem that ends in an obstruent consonant (except glottal stop) is attached with a sonorant initial morpheme then metathesis is activated and then the position of the obstruent and sonorant is exchanged [8].

Metathesis is represented in two ways:

1. Metathesis I

Metathesis that changes an *Obstruent.Sonorant (OS) cluster, which is not permitted in the linguistic rule of the language to the correct Sonorant.Obstruent (SO) cluster.

O + S → S.O (if O ≠ ’)

Example:

Verb	**Verb stem**	**CC- C**	**Glosses**
biix- n-aammi	biix-	bíinxaammi	‘we will break’
kaas- n-unta	kaas-	káansunta	‘so that we plant’

2. Metathesis II

Metathesis that exchanges the position of a Sonorant (S) and glottal stop (’) when an initial glottal stop morpheme is attached to a verb stem that ends a sonorant. It creates an authorized ’.S cluster

S + ’ → ’.S

Example:

Verb	**Verb stem**	**CC- C**	**Glosses**
wok’k’artáni -’e	wok’k’artáni	wok’k’artá’ne	‘will you beat me?’
‘t’uundániyani -’ne	t’uundániyani	t’uundániya’ne	‘(we) watching you (PL)?’

3.2.3. Assimilation

Assimilation is a condition where there is a sound change in the direction of the nearby sound for simplicity of articulation.

Assimilation-I

It is observed when a /t/ initial suffix meets a verbal stem that ends in an obstruent then, /t/ takes all features of the obstruent and its position will be occupied by the stem’s final obstruent.

Obstruent + -t → Obstruent. Obstruent

For example, Assimilation-I is applied when a stem that ends with obstruent meets a /t/ initial negative imperative singular morpheme /-tooti/.

“sut-”/ ‘insert’ → “súttooti”/ ‘don’t insert’

“nak’-”/ ‘hit’ → “nak’k’ooti”/ ‘don’t hit’

Assimilation-II

Kambaatissa doesn't permit a cluster of non-identical obstruents and also cluster of non-identical sonorants is not allowed in the language. Assimilation is observed when a sonorant initial suffix and sonorant-final verbal stem meet together and if the stem final sonorant is alveolar then the assimilation is called anticipatory otherwise it is known as perseverative [8].

The final /r/ and /l/ of the verbal stem assimilate completely to the /n/ of the following morpheme. For example:

"mur-" / 'cut' + "-no" will be assimilated to "munnó" / 'let's cut!'

"dul-" / 'slaughter' + "-núnka" will be assimilated to "dunnúnka" / 'we should not slaughter'

The non-alveolar sonorant, the bilabial nasal /m/, does not assimilate to the following /n/-initial suffix. But, the initial suffix /n/ assimilates to the verb stem. For example:

"tum-" / 'pound' + "-náyyoommi" → "tummáyyoommi" / 'we are pounding'

Morphophonological process-II is observed only on perfective and imperative verb forms. In the o-perfective *(*1PSG /-óommi*/,* 3PM /´-o/*),* e-perfective *(*1PSG /-eemmi/, 3M /-eeu/), perfective converb *(*1PSG /3PSGM /-i/*)* and imperative *(*2PPL /-é/, /-iyyé/*).* It is classified in to Gemination and palatalization.

3.2.4. Gemination

It is triggered when the above vowel initial suffixes are attached to the stem final consonant. The final consonant of the verb stem performs gemination when it meets the vowel suffix '-o'. For instance:

"hun-" + "-o" → "hú**nn**o" / 'he run away'

"leel-" + "-o" → "léé**ll**o" / 'it cracks'

"k'áag-" + "-o" → "k'áa**gg**o" / 'he remembers'

3.2.5. Palatalization

It is observed when the stem-final consonant is an alveolar obstruent and it attaches to vowel initial suffix. Here the alveolar consonant will be converted to palatal sound and it forms geminate consonant. Example: *o*-perfective main verbs of the third person:

“baad-” + “-o” → “báajj-o” / ‘crawl’

“luus-” + “-o” → “lúushsh-o” / ‘make a mistake’

“kot-” + “-o” → “kochch-o” / ‘not suffice’

3.3. Morphological Overview of Verbs

A verb stem in Kambaatissa consists of the root and derivational morpheme and all inflectional morphemes are found next to the verbal stem. An affirmative declarative main verb consists of two slots for subject agreement morphemes. An aspect morphemes are placed in between the two subject agreement morphemes.

Table 4: The Structure of affirmative declarative main verb

		Inflection			
Stem		Subject agreement	Aspect	Subject agreement	Object suffix
Root	(Derivation)				

Kambaatissa is primarily an aspect marking language and it mainly differentiates the verb forms into perfective and imperfective paradigms [29]**.** It do not mark tense on the verb inflection but instead it is overlaid on aspect marking. In Kambaatisaa the main verb is the last constituent of a sentence this makes it to be a head-final language [48].

Kambaatissa verbs are inflected for aspect (imperfective vs. perfective), mood (indicative, imperative-jussive, and preventive), and subordination (main/final vs. subordinate/non-final verb) as well person, gender, number, and social status of the subject in portmanteau morphemes. Kambaatissa classifies verbs in to main and subordinate verbs.

3.4.1. Main verbs

Main (final) verbs complete a sentence and it has the most elegant inflectional potential i.e. might have more divisions for subject agreement, aspect and, mood [29]. There are four aspect marking on main verb those are imperfective (IPV), perfective with a characteristic e-vowel (PVE), perfective with a characteristic o-vowel (PVO), and progressive (PROG).

Table 5: The Structure of affirmative declarative main verb

Stem: Root(+ derivation)	Inflection			Object suffix
	Subject agreement	Aspect	Subject agreement	
	1Sg: -ø 1Pl: -n 2Sg: -t 2Pl: -t-een 2Hon: -t-een 3SgM: - ø 3SgF: -t 3Pl: -t 3Hon: -een	IPV: *-a(a)* PVE: *-e(e)* PVO: *-o(o)* PROG: *-ayyoo*	1Sg: -m(m) 1Pl: -m(m) 2Sg: -nt 2Pl: -nta(a'u) 2Hon: -nta(a'u) 3SgM: -var 3SgF: -(-'V) 3Pl: -(-'V) 3Hon: -var	1Sg: -'e 1Pl: -nne 2Sg: -kke 2Pl: -(kki)'nne 2Hon: -(kki)'nne 3SgM: -s 3SgF: -se 3Pl: -ssa 3Hon: -ssa

3.2.1.1. Aspect marking on main verbs

I. Imperfective (IPV)

It is formed by inserting an aspect vowel 'a(a)' between the subject agreement morphemes. It marks an event with incomplete action because of the event is habitual, a general truth, events happening at the speech time or expected to happen in the future.

Table 6: The Structure of affirmative declarative main verb

	Subject agreement morpheme	**Imperfective (IPV)**	**Subject agreement morpheme**
1Sg	-ø	-a(a)	-m(m)
1Pl	-n	-a(a)	-m(m)
2Sg	-t	-a(a)	-nt

2Pl	-t-een	-a	-nta
2Hon	-t-een	-a	-nta
3SgM	-ø	-a	-no
3SgF	-t	-a(a)	- (’a)
3Pl	-t	-a(a)	- (’a)
3Hon	-een	-	-no

Example:

Verb stem	**IPV**
“hujat-” / ‘work’	“hujat-áa-mm” / ‘I work habitually/ I will work’
“qeel-” / ‘win’	“qeel-t-áa-nt” / ‘you win/ you will win’
“mar-” / ‘go’	“mar-a-no” / ‘he goes/ he will go’

II. Progressive (PROG)

It is formed by inserting an aspect morpheme ‘ayyoo’ between the subject agreement morphemes. It indicates durative event which is in the process of happening at the reference time.

Table 7: The Structure of affirmative declarative main verb

	Agreement morpheme	Progressive (PROG)	Agreement morpheme
1SG	-ø	-ayyoo	-m(m)
1PL	-n	-ayyoo	-m(m)
2SG	-t	-ayyoo	-nt
2PL	-t-een	-ayyoo	-nta
2HON	-t-een	-ayyoo	-nta

3SGM	-ø	-ayyoo	- (’u)
3SGF	-t	-ayyoo	- (’u)
3PL	-t	-ayyoo	- (’u)
3HON	-een	-ayyoo	-mma

Example:

Verb stem	**PROG**
“he’-” / ‘live’	“he’-t-áyyoo-nt” / ‘you (SG) are living’
“mar-” / ‘go’	“mar-ayyoo-’u” / ‘he is going’
“hujat-” / ‘work’	“hujat-ayyoom” / ‘I am working’

III. Perfective (PVO)

It is formed by inserting an aspect vowel ‘o(o)’ between the subject agreement morphemes except for 2Pl/ Hon and 3Hon. The 1Sg and 3SgM perfective verbs are formed only when the verb stem ends with a single consonant otherwise (ends with double consonant) they can’t be classified as PVO or PVE. A morphophonological change called palatalization and germination have to be performed for 1Sg and 3SgM.

Table 8: The Structure of affirmative declarative main verb

	Agreement morpheme	Perfective (PVO)	Agreement morpheme
1SG	-ø	-oo	-m(m)
1PL	-n	-oo	-m(m)
2SG	-t	-oo	-nt
2PL	-t-een	-ø	-nta(a’a)
2HON	-t-een	-ø	-nta(a’a)

3SGM	-ø	-o	-
3SGF	-t	-oo	- (’u)
3PL	-t	-oo	- (’u)
3HON	-een	-ø	-mma

Example:

Verb stem	**PVO**
“qeel-” / ‘win’	“qeel-t-oo-nt” / ‘you (SG) won’
“mar-” / ‘go’	“mar-oo” / ‘he went’

IV. Perfect (PVE)

It is formed by inserting a characteristic aspect vowel ‘e(e)’ between the subject agreement morphemes except in 3Hon and 2Hon. It describes actions that are completed but whose resulting state continues to the reference time [49]. In 1Sg and 3SgM the perfect verb is formed by applying morphophonological change called palatalization and germination to the root verb. In addition 3Hon and 2Pl/Hon forms shared with the perfective paradigm.

Table 9: The Structure of affirmative declarative main verb

	Agreement morpheme	Perfect (PVE)	Agreement morpheme
1SG	-ø	-ee	-m(m)
1PL	-n	-ee	-m(m)
2SG	-t	-ee	-nt
2PL	-t-een	-ø	-nta(a’a)
2HON	-t-een	-ø	-nta(a’a)
3SGM	-ø	-ee	- (’u)

3SGF	-t	-ee	- (’u)
3PL	-t	-ee	- (’u)
3HON	-ee	-ø	-maa’a

For example:

Verb stem	**PVE**
“qeel-” / ‘win’	“qeel-ee-mm” / ‘I won’
“mar-” / ‘go’	“mar-t-ee-nt” / ‘you went’

Mood is encoded in main verbs as imperative, indicative and jussive. Imperative is applied to a verb to express entreaties, requests, instruction and invitations. The singular imperative verbs are indicated by -i and plurals are indicated by the morpheme -é / -iyyé. The negative form is described by adding the morphemes -oot in the singular and -oochche in the plural form. For example,

Imperative positive	Imperative negative
“waal-i” / ‘you (singular) come’	“waal-toot” / ‘you (singular) don’t come’
“mar-ré” / ‘you (plural) go’	“mar-téen-ochche” / ‘you (plural) don’t come’

Preventive verb is also an imperative form but expresses a very strict commands and warnings. Its negative form is marked by ókkoo such as mar-tókkoont.

Negation on Main verbs

Negation on Imperfective main verb is indicated by adding the morpheme ba’a in the affirmative form of the verb. The morpheme ba’a is also called the standard negative marker [50]**.** For Example:

Verb stem	**IPV**	**NegIPV**
qeel-	“qeel-táant” / ‘you win’	“qeel-táant-**ba’a**” / ‘you don’t win’
mar-	“mar-ano” / ‘you don’t win’	“mar-ano-**ba’a**” / ‘he doesn’t go’

Perfective main verbs and progressive main verbs are changed to negative by adding the non-imperfective aspect marker -im after first subject agreement morpheme and followed by the negative marker -ba(’a) [49][29]. For example:

Perfective affirmative	**Perfective Negative**
“marr-ee-mm” / ‘I went’	“mar-**im-ba**” / ‘I did not go’
Progressive affirmative	**Progressive Negative**
“mar-ayyoo-mm” / ‘I am going’	“mar-**im-ba**” / ‘I am not going’

3.4.2. Subordinate verbs

Subordinate (non-final) verbs commonly need the presence of a superordinate main verb form and have reduced inflection potential [51]. They are found in sentence medially and it is divided into relative verbs, purposive verbs, and converbs. Subordinate verbs have limited number of aspects these are perfective and imperfective. Person values are restricted from seven to five classes (1sg/3m, 2sg/3f/3pl, 3hon, 1pl and 2pl) and there is no inflection for mood.

Relative verbs are subordinate verbs that are marked by final accent. Example:

Verb stem	**IPV Relative verb**
“mar-” / ‘go’	“mar-eennó” / ‘they go’

Converbs are subordinate verbs that consists of the perfective and imperfective paradigms.

Table 10: The Structure of affirmative declarative main verb

	perfective converb	Imperfective converb
1Sg	C- palatalization and germination -ø CC- -ø -i-	-ø-an
1Pl	-n-	-n-an-
2Sg	-t-	-t-an-

2Pl	-teen-	-teen-an-
2Hon	-teen-	-teen-an-
3SgM	C- palatalization and germination -ø CC- -ø -i-	-ø-an
3SgF	-t-	-t-an-
3Pl	-t-	-t-an-
3Hon	-een-	-een-an-

Negation on Subordinate verbs

Relative verbs mark negation by using the negative morpheme **-umb**. For instance the verb kul is relative affirmative verb and it is marked as negative **kul-umb-u**.

Table 11: Negation on Relative verbs

Person	Negation morpheme
1Sg	-ø-umb-u
1Pl	-n-umb-u
2Sg	-t-umb-u
2Pl	-teen-umb-u
2Hon	-teen-umb-u
3SgM	-ø -umb-u
3SgF	-t-umb-u
3Pl	-t-umb-u
3Hon	-een-umb-u

Converbs mark negation using the negative morphemes -u'nna, -u'nnaan, -u'nnaachch. For instance

- "hujat- u'nna" / '(He) without work'
- "waal-u'nna" / '(He) without come'

Jussive verbs mark negation using - (n)ka For instance:

- "hujat - un-ka" / 'don't let him work'
- "waal-un-ka" / 'don't let him come'

Table 12: Tense inflection morphemes

Person	**Tense**			
	IPV(present/future)	PVO (present perfect)	PVE(simple past)	PROG
1pSg	aa	oo	ee	ayyoo
1pPl	aa	oo	ee	ayyoo
2pSg	aa	oo	ee	ayyoo
2pPl	a	-	-	ayyoo
2pHon	a	-	-	ayyoo
3pSgM	a	o	ee	ayyoo
3pSgF	aa	oo	ee	ayyoo
3pPl	aa	oo	ee	ayyoo
3Hon	-	-	-	ayyoo

3.4.3. Verb derivation

Kambaatissa derivational morphology is basically described by the following derivational morphemes.

1. Causatives

The causative can be short which is represented by adding suffix *-s* (CAUS_I) or it can be long which is by adding the suffix '-siis' (CAUS_II)

Short causative (CAUS1)

The addition of the morpheme '-s' to an intransitive verb will create CAUS_I

Example:

Verb root	**CAUS_I**
"bajig" / 'become happy'	"bajig-is" / 'make happy'
"fayy" / 'become healthy'	"fayy-is" / 'make healthy'
"baab" / 'get worried'	"baab-is" / 'make worried'
"fa'" / 'be saved'	"fa'-is" / 'save'
"waal" / 'come'	"waash-sh" / 'bring'

Double causative (CAUS_II)

The addition of the morpheme '-siis' to a transitive verb will create CAUS_II

Example:

Verb root	**CAUS_II**
"qorab" / 'wait (for) take care'	"qorab-siis" / 'make wait (for)'
"wajj" / 'fear'	"wajj-siis" / 'make fear'
"ass" / 'do'	"ass-siis" / 'make do'
"hab" / 'forget'	"hab-siis" / 'make forget'
"shol" / 'cook, prepare food'	"shol-siis" / 'make cook'

2. Passive (PASS)

The derivational morpheme that indicates passive is '-am'

Example:

Verb root	**PASS**
“hiir” / ‘translate, solve’	“hiir-amm” / ‘translated, solved’
“fan” / ‘open’	“fan-am” / ‘be opened’
“mar” / ‘go’	“mar-am” / ‘walk’
“lall” / ‘become visible’	“lall-am” / ‘become famous’

3. Middle (MID)

A middle causative has two phonologically conditioned allomorphs /-’/ and /-aqq/. The allomorph /-’/ is added when the verb root ends with a single consonant. If the root final consonant is sonorant then /-’/ is infixed.

Example:

Verb root	**MID**
“wor” / ‘put into’	“wo’rr” / ‘put into for one’s benefit’
“min” / ‘build a house’	“mi’nn” / ‘build a house for one’s self’
“im” / ‘dig’	“i’mm” / ‘dig for one’s benefit’

If the consonant is non-ejective obstruent the glottal stop assimilates in place to it and triggers its ejectivization.

Example:

Verb root	**MID**
“dag” / ‘find’	“daq-q” / ‘find for one’s benefit’
“saad” / ‘praise’	“saax-x” / ‘praise oneself’
“huf” / ‘comb’	“huph-ph” / ‘comb oneself’
“has” / ‘look for, want’	“hac-c” / ‘look for, want for one’s benefit’

The morpheme /-aqq/ can be added at the end of a cluster or monoconsonant and it is also found after a single plosive.

Example:

Verb root	**MID**
“sull” / ‘hang’	“sull-aqq” / ‘hang oneself’
“quss” / ‘massage’	“quss-aqq” / ‘massage oneself’
“eeb” / ‘bring’	“eeb-aqq” / ‘bring for oneself’
“maax” / ‘hide’	“maax-aaq” / ‘hide for oneself’

In Kambatissa the derivation of verbs from other word class is not common but some verbs can be derived from adjectives by eliminating the adjectival suffixes.

Adjective	**Derived verb**
“annann-a (-ta)” / ‘different’	“annann-” / ‘be (come) different’
“ros-aan-ch-u (-ta)” / ‘student’	“ros-” / ‘learn’
“ros-is-aan-ch-u (-ta)” / ‘teacher’	“ros-is-” / ‘teach’

3.4.4. Verb Inflection

The person is represented as first, second and third with gender and number inflections. Kambatissa uses **‘ani/eesi’** to represent first person singular and **‘na’ooti/na’oota’** for plural. The first person singular do not identify gender as feminine and masculine and both are represented by **‘ani/eesi’** and the plural is neutral gender. Second person singular and plural are represented by **‘ati/keesi’** and **‘ano’ooti/ki’ne’eeta’** respectively. There is also honorific **‘a’nu/ki’neta’** which represents social status of the person. The second person singular **‘ati/keesi’**, as the first person singular it used for both genders and the plural is also neutral gender. Third person singular is **‘isi/isu’** to indicate masculine gender and **‘ise/ise’ta’** for feminine. The plural is **‘isso’ooti/isso’oota’**, which is neutral gender as the prior plural person classes and honorific is indicated by **‘issa/issata’** [52].

The above pronoun words are present in Kambaatissa language words in the form of suffixes attached to the stem of the verb in a sentence. For example:

Person	**verb inflection**	**glosses**
1prSg	kul-aam(m)	I will tell
1prPl	kun-naam(m)	we will tell
2prSg	kul-taant	you will tell

2prPl/Hon	kul-teenanta	you will tell
3prM	kul-ano	he will tell
3prF/Pl	kul-taau	she will tell/ they will tell
3Hon	kul-eenno	he/ she (hon) will tell

Gender Suffix

Gender is indicated as masculine or feminine values and words may contain suffix to indicate a certain gender. In Kambaatissa verbs gender is not differentiated in the first and second person. The third person singular differentiates gender and it has different suffixes to indicate masculine and feminine gender. For example:

Person	**gendersuffix**	**glosses**
1pr.	Kul-aam	I tell /I will tell
2pr.	Kul-taant	you tell /you will tell
	Kull-teenanta	you tell /you will tell
3pr.M	kul-ano	he tell /he will tell
3pr.F	kul-taau	she tell /she will tell

Number suffix

Number is expressed as singular and plural. Verbs have suffixes to indicate the number inflection and the verbal suffixes is differentiated in connection with pronouns. In kambatissa the third person masculine is zero-marked (it has no overt suffix). Examples of gender suffixes are listed below.

Person	**Stem**	**singular**	**plural**
1pr	qell-	“qell-om” / ‘I won’	“qenn-o-m” / ‘we won’
2pr	qell-	“qell-t-o-nt” / ‘you won’	“qel-t-ee-nta” / ‘you won’
3pr	qell-	“qell-o/ k’ell-t-o” / ‘he (she) won’	“qel-t-o” / ‘they won’

CHAPTER FOUR

4. Materials and methods

4.1. Introduction

This chapter discusses the materials and methods used in the design and implementation of the Kambaatissa morphological analyzer and generator. After reviewing many literatures we have chosen a suitable research method and techniques to implement the analyzer and generator. Section 4.2 discusses the source and methods of data collection, Section 4.3 presents the chosen research method for the study and Section 4.4 discusses the technique used for the implementation of the system as well as the hardware and software tools.

4.2. Data Collection

This study applied both primary and secondary sources for data collection. The primary are language experts and the secondary data sources are different linguistic books of the Kambaatissa language, those books help to get information about how words are formed in the language and which grammatical features are carried by the morphemes. As the study is conducted using a rule-based approach it needs a deep understanding of the language. To get detailed knowledge about word formation in the language plenty of time is invested with language experts. At the time different language-specific rules are identified to use in the morphological analysis. The study also used different linguistic books written on the morphology and grammar of the language to identify the morphophonological changes during word formation. We collected word lexicon from different written materials some of them are: Kambaatissa student textbooks from primary, secondary, and preparatory schools by contacting Kambaata-Xambaro-Zone education biro, the other one is Wachemo University Kambaatissa department, student modules of different courses by contacting the head of the department and Kambaatissa-Amharic-English dictionary is also used. The above documents are collected in the form of softcopy by contacting the respective body in person. The collected documents are tokenized into words using the NLTK python library. The tokenized lists of words are given to language experts to select verbs and nouns from the tokenized words. After that, the experts performed morpheme segmentation for the selected verbs and nouns. For each word, they differentiated the root word and the attached suffixes with the grammatical information carried on it. Finally, 860 root verbs are identified from the above documents

and the Kambaatissa-Amharic-English dictionary. These root words are used as input for the morphological generator and analyzer implementation.

4.3. Research Method

The research method includes all the techniques and methods used by a researcher in a research process to find the solution to a research problem [53]. We tried to review different articles on the subject of the study to design the research. The research design that is suitable for this study is experimental that is, the raw experimental output of a single test to the entire lexicon implementation. The experimental results were analyzed and presented using a numerical value which is the concept in the quantitative method. The data collected through quantitative research methods are usually in numerical form [54]. Therefore, the study used both quantitative and experimental.

4.4. Implementation technique and tools

The Kambaatissa morphological analyzer and generator is designed using the technique called Finite State Transducer. It is often used to develop successful morphological analyzers for different languages in the world, especially for morphologically rich languages [55] [12] when there are no sufficient corpus to deploy a machine learning approach. It performs the relation between two regular languages that are upper side and generation lower side regular languages. It is an improved version of the finite-state automation model based on the two-level morphology principles.

The greater advantage of using FST is it performs not only analysis but also generation with a bidirectional processing model. A single FST can perform an analysis in the upward direction and generation in the downward direction. This perfectly aligns with language processing where people are interested to analyze or understand a written word as well as produce or generate a word. The input of the system is a surface word and internally the system looks for morphemes. The information related to each morpheme is grasped by the system and determined based on the rule and the system generates a lexicon with detailed information about the morphemes. Performing morphological analysis requires the following three pieces of information [56].

1. Lexicon

A lexicon contains a list of roots or stems of a word in a particular category. Those root words will later be combined with grammatical morphemes to form surface word forms.

2. Morphotactics

Morphotactics deals with restrictions on the order and class of morphemes that make up a word in a given category. For example, a verb in the Kambaatissa language has to follow the following order of morphemes:

Subject agreement-1, aspect/ mood, subject agreement-2, object suffix (optional), Negation

3. Alternation rules

The rules of alternation are responsible for the variation of the forms that morphemes take in the presence of other morphemes. For example, in Kambaatissa verb/noun if a consonant initial suffix morpheme connects to a root that ends in a double consonant then Epenthesis (i-insertion) rule will be applied to remove the unauthorized consonant cluster.

There are different toolkits used to implement FST. These include XFST, OpenFST, HFST, and Foma. SFST is a toolbox for the implementation of morphological analyzers and other tools which are based on finite state transducer technology. OpenFst is a toolkit for constructing and combining weighted finite-state transducers. Foma is also a compiler and programming language for constructing finite-state automata and transducers to use in different language applications. It has specific support for many NLP applications like morphological analyzers.

XFST is a closed-source and proprietary tool but Foma is open-source software and can be easily extended to web applications [12][56]. In this study, the FST is developed using the Foma implementation toolkit which is a popular framework for developing morphological analyzers.

Foma is a free and open-source finite-state toolkit that has facilities for a variety of natural language processing applications. It can be considered as a programming language that has its compiler to create and use automata and transducers. It is used to develop FSTs for different Natural language applications and a morphological analyzer is one of them.

This toolkit consists of two components the first one is a list of morpheme component that contains the lexicon and morphotactics. The second one is the alteration rule component which contains morphophonological and orthographic rules [12].

The lexicon component contains the root words and ordering restrictions of the root and its morphs then it maps them to an intermediate or final form. The alternation rule component contains a morphophonological and orthographic change that takes place in the word formation. Here we can define many rules then concatenate them and apply the rules to the output of the lexicon component to get the final forms. The lexicon file will be written in text-editor and saved in ***.lexc*** file format and the set of defined rules will also be written in text editor but saved in ***.foma*** file format. The final transducer will be the composition of the FSTs defined in the two files.

4.5. Hardware tools

For the successful completion of this study, different hardware tools are used. Laptop computer with 64-bit operating system and Intel(R) Core i7 2.70GHz processor, USB flash disc etc.

4.6. Software tools

The software tools used for the study are Oracle VM VirtualBox 7.0 with Ubuntu 22.01 for foma implementation, Microsoft office word 2016 for preparing the word document, and Microsoft Office PowerPoint to prepare the presentation.

4.7. Performance Evaluation

The FST-based morphological analyzer can be evaluated by using corpora that contain different word forms in the language. Here the performance is calculated by testing how many word forms from the given corpus are covered by the proposed analyzer [57]. Different researchers used the above method for evaluating their analyzer [58] and some of them also uses precision and recall to calculate the average accuracy of the analyzer [59]. Some previous works [5] mentioned that evaluating a rule-based morphological analyzer is meticulous because it needs a language expert or someone familiar with the language to check the analyzer's output carefully. This study evaluates the efficiency of the system by comparing generated words with the words that exist in the collected corpus and to check the correctness of some word forms which are not found in the collected corpus, the output

of the analyzer is given to language experts. Analysis efficiency is measured by checking the tag sets which are used in the lexicon file and the correct meaning they give when combined. Accuracy refers to the amount of correct analysis out of the analyzer's output. The experimental result presented in the term of accuracy obtained from the difference in total generated words (TW) and incorrect words (IW) out of 100%

Accuracy = Total generated words – Incorrect words / Total generated words *100

Accuracy = TW - IW / TW*100

CHAPTER FIVE

5. Design of Proposed Morphological analyzer and generator

5.1. Introduction

This chapter discusses the research design and implementation approach deployed for this study. In the first part, the architectural designing issues are dealt with which is followed by the list of morphological rules of different categories used for analysis and generation of Kambaatissa language. As discussed in the previous chapter, a finite state transducer is the better approach for the implementation of a morphological analyzer and generator, especially for languages with limited linguistic resources and computational systems. It has a design technique based on in-depth knowledge of the language's word formation characteristics. The design has two parts. That are the lexicon/morphotactic design and the morphophonological alternation rules design. These design segments are put in place to handle the lexicon based morphological features and the changes that occur due to the intercalation of morphemes that cause surface form alternation respectively. The design process of this system is based on the morphological properties of the language presented in chapter three and the assumptions and approaches discussed in chapter four.

5.2. Architecture of the proposed morphological analyzer and generator

The proposed morphological analyzer and generator has two basic components namely, ***morphotactics*** and ***morphophonological alteration*** components. Root verb is the input to the system and based on the underlining FST architecture, the engine works in two directions. The upward direction performs analysis and the down ward direction deals with generation. The following diagram shows the architecture of the proposed analyzer and generator for Kambaatissa language. The components of the architecture are explained below next to the diagram.

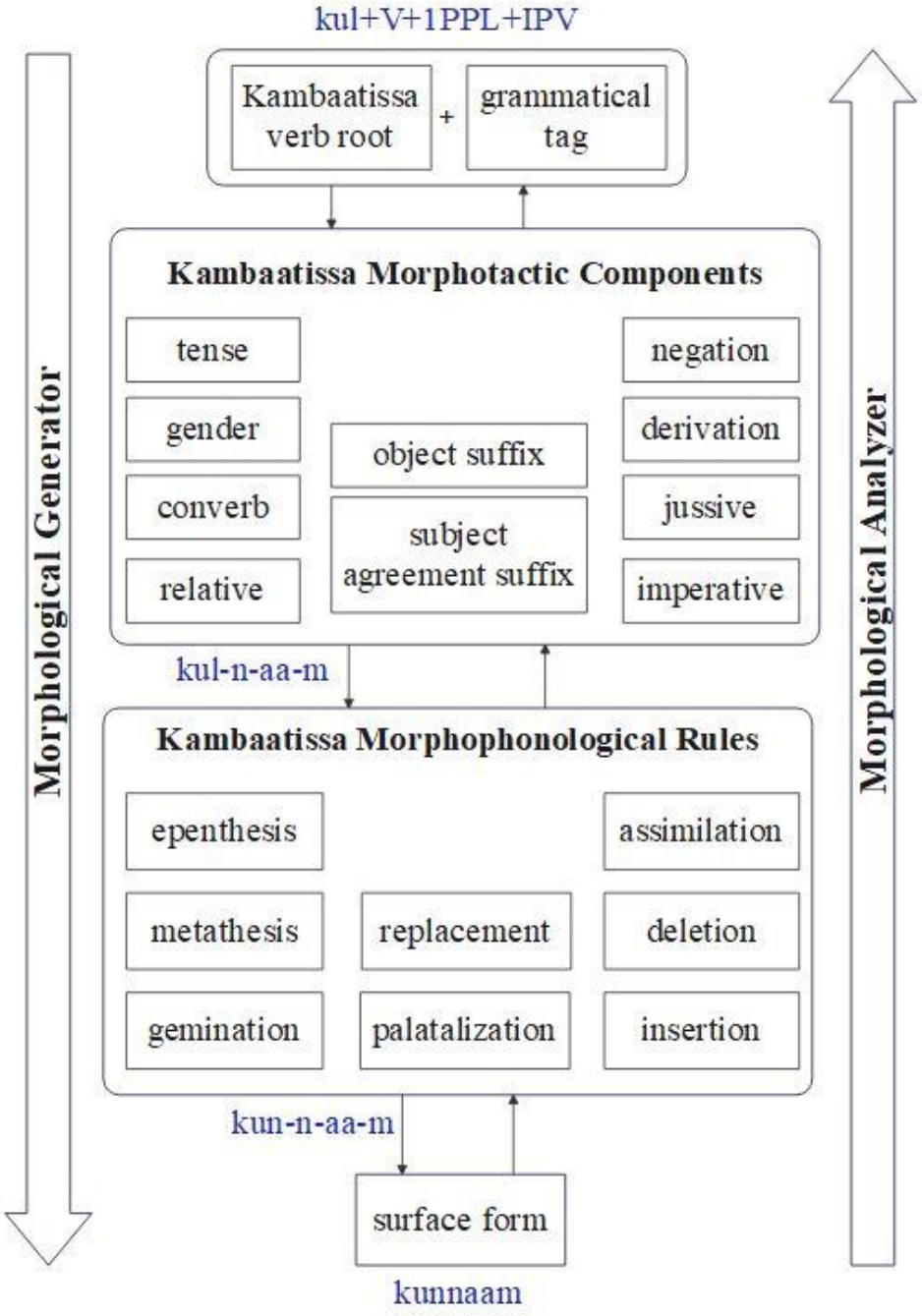


Figure 2: Architecture of FST based Kambaatissa morphological analyzer and generator

The above architecture clearly represents each component of the proposed morphological analyzer and generator. The next sub-section contains the detail explanation of those major components of the architecture.

5.2.1. Root verb and Grammatical tag

Root verb

A root word is a free morpheme that does not have any derivational or inflectional affix that is attached to it. The lexicon file contains the root words for Kambaatissa verb word classes. This file is used as a dictionary of the root forms of the language and can be expanded as needed by including more verb roots.

Grammatical tags

Tags are symbols that represent the grammatical feature carried by the corresponding morphemes within the given word form. The study, by using language experts' judgement and the linguistics knowledge of Kambaatissa, incorporated the following tags in the design of the morphological analyzer and generator.

Tags that are used in the Verb word class are:

- ✓ V → to represent the root verb
- ✓ 1pr → to represent the first person.
- ✓ 2pr → to represent the second person.
- ✓ 3pr → to represent the third person.
- ✓ 3prSgM → to represent the third person singular with masculine gender.
- ✓ 3prSgF → to represent the third person singular with the feminine gender.
- ✓ Sg → to represent singular form.
- ✓ Pl → to represent plural form.
- ✓ F → to represent the feminine gender
- ✓ M → to represent the masculine gender
- ✓ Hon → to represent honorific form.
- ✓ IPV → represents present/future (imperfective verb).
- ✓ PVE → represents present perfect (perfective verb with vowel E).
- ✓ PVO → to represent simple past (perfective verb with vowel O).
- ✓ PROG → represents present continuous (progressive verb).
- ✓ IMP → represents an imperative verb.
- ✓ JUSS → to represent jussive verb.
- ✓ REL → represents a relative verb.
- ✓ CON → to represent converb.
- ✓ NEG → to represent negation form.

The above tag sets are used to perform the FST implementation and to represent the associated grammatical features that are exhibited by Kambaatissa verbs.

5.2.2. Morphotactic Component

The morphotactic component designates the morphotactic that controls the order and class of the morphemes that make up a word within a particular word class [56]. This component contains different continuation classes which contain suffixes that marks different grammatical features.

Continuation class

The inflection of words includes numerous suffixes, each of which carries a unique piece of grammatical information. These suffix sequences are attributed to the appropriate classes. Numerous continuous classes found in lexical scripting serve as the foundation for the analysis task. The classes that were captured and used for the experiment are listed below.

Continuous classes in the Verb lexicon:

- Subject agreement class
- Tense class
- Imperative class
- Derivation class
- Imperative class
- Converb class
- Jussive class
- Relative class
- Negation class

Figure 3: Continuous classes in the verb lexicon

Subject agreement class includes all suffixes that identify a person, number, and gender marker in one. The Kambaatissa verb marks person, number, and gender using a single suffix. Subject agreement morphemes are placed in two positions. In the language, there are 1st person, 2nd person, and 3rd person in their singular and plural forms. The 3rd person is expressed in the masculine and feminine gender. There is also honorific which is used to

represent the social status of the person. In this design, there are two classes to represent those two positions namely Subject agreement 1 and Subject agreement 2.

a) Subject agreement class 1 (SubAgr1)

It is the first place to put a person, number, and gender marker. In the singular form, there is no suffix for 1st person, /-t-/ for 2nd person and 3rd person with masculine gender has no suffix but /-t-/ is used in its feminine gender. In the plural form /-n-/ for 1st person, /-teen-/ for 2nd person and /-t-/ for 3rd person. In addition to singular and plural, there is also honorific in the second and third person which is represented by the suffixes /-t-een/ and /-een/ respectively.

b) Subject agreement class 2 (SubAgr2)

It is the second place to put a person, number, and gender marker. In the first person /-m(m)/ for singular and plural. For the second person the suffix /-nt/ in the singular, /-nta/ in the plural, and honorific. Finally /-'u/ is used in the third person singular both for masculine and feminine gender but there is no suffix for honorific.

c) Tense class (TEN)

The tense class contains four basic classes simple past tense (PVO), present/future tense (IPV), present perfect tense (PVE) and present continuous tense (PROG) suffixes. It is also described in a table on chapter three.

d) Imperative class (IMP)

It contains the Imperative verb suffixes of 2nd person singular and 2nd person plural and honorific. The suffixes are /-i/ for singular and /-e/, /-iyye/ for plural and honorific.

e) Derivation class (DRV)

It contains different suffixes that marks the verb derivation. The first one is causative derivation which can be classified into causative I (Caus_1) and causative II (Caus_2). Causative I is marked by /-s-/ suffix and causative II is marked by /-siis-/ suffix. The other derivation is middle causative (MID) which is marked by the suffix /-'/ and /-aqq/. The last derivation, Passive (PASS) is marked by the derivational morpheme /-am/.

f) Negation suffix class

This class contains negation morphemes for different verb types such as the negation of imperfective verbs, negation of jussive, negation of imperative, and negation of converb. Negation of imperfective verbs is marked using the standard negation morpheme /*-ba'a/*. Perfective main verb and the progressive verb use the negation suffix /-imba(a)/. Relative verbs mark negation using the suffix /–umb/, jussive verbs by /-unka/ and converb uses the negation marker /-u'nnachch/.

5.2.3. Morphophonological rules

Morphophonological rules are also called alternation rules which are responsible for the change of forms that morphemes take in the presence of other morphemes [56]. There are different morphophonological changes that takes place when an affix with a particular grammatical information is attached to a root word to form new word. For instance, in the verb paradigm a root final sound /-t/ changes to /-ch/ when it connects with e/o initial suffix. Example: "su**t** /su**chch**ee". Sometimes -r changes to -n, Example: "maraam / mannaam" / 'I will go / we will go'

Here different rules are stated to control the morphophonological alternations of the word formation in the language. The complete rule design of the system is presented at the end of this document in Appendix 2 but some of them are listed below:

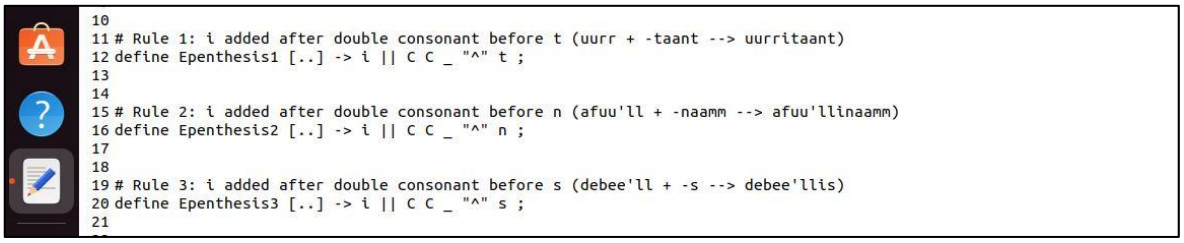
- Epenthesis
- Metathesis
- Gemination
- Assimilation
- Replacement
- Palatalization
- Deletion

Figure 4: Morphophonological rules

a) Epenthesis (i- insertion)

In Kambaatissa cluster of three consonants is not permitted. The epenthesis rule deals with the insertion of the vowel /i/ when there is the probability of being a cluster of three consonants which is not permitted in the language. The rule is applied when a consonant initial suffix connects with a verb stem that ends with a double consonant and here the

vowel /i/ is inserted between the stem ending double consonant and the consonant initial suffix to ensure the rule of the language. The following Epenthesis rules are used in Kambaatissa morphological analyzer.



```
10
11 # Rule 1: i added after double consonant before t (uurr + -taant --> uurritaant)
12 define Epenthesis1 [..] -> i || C C _ "^" t ;
13
14
15 # Rule 2: i added after double consonant before n (afuu'll + -naamm --> afuu'llinaamm)
16 define Epenthesis2 [..] -> i || C C _ "^" n ;
17
18
19 # Rule 3: i added after double consonant before s (debee'll + -s --> debee'llis)
20 define Epenthesis3 [..] -> i || C C _ "^" s ;
21
```

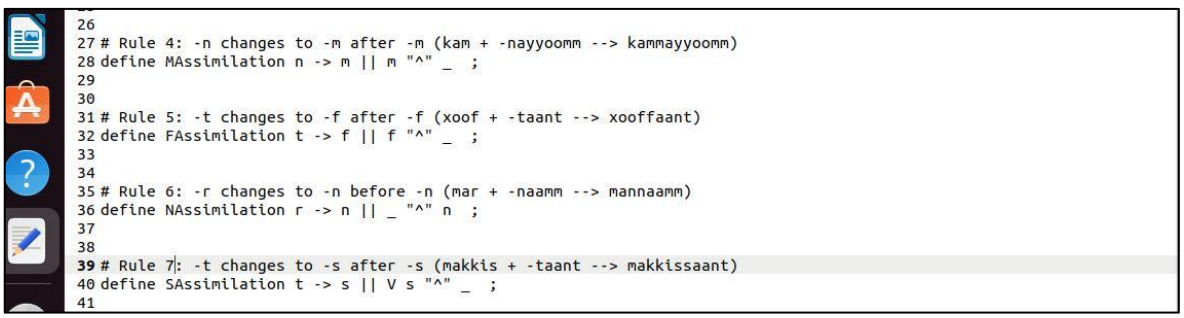
Figure 5: Sample for Epenthesis rules

b) Metathesis

It is a morphophonological rule that works by exchanging the position of two consonant sounds when the form unpermitted cluster. For instance, when the verb root that ends with obstruent consonants such as, /-f/, /-s/, /-g/, /-t/ connects with an /-n/ initial suffix then the metathesis rule is applied by changing the position of the above stem ending consonants with /-n/.

c) Assimilation

The language has different assimilation rules that deal with the change of sounds in the direction of the nearest consonant. Most of the time when a /-t/ initial suffixes are attached to root verbs that end with an obstruent consonant here, the suffix initial /-t/ will be assimilated to the sound of the root ending consonants. It is also applied when the verb root that ends with /-l/, /-r/ connects with an /-n/ initial suffix then the stem ending consonants will be assimilated to the sound /-n/. The following are some of the rules which are used in the analyzer.



```
26
27 # Rule 4: -n changes to -m after -m (kam + -nayyoomm --> kammayyoomm)
28 define MAssimilation n -> m || m "^" _  ;
29
30
31 # Rule 5: -t changes to -f after -f (xoof + -taant --> xooffaant)
32 define FAssimilation t -> f || f "^" _  ;
33
34
35 # Rule 6: -r changes to -n before -n (mar + -naamm --> mannaamm)
36 define NAssimilation r -> n || _ "^" n  ;
37
38
39 # Rule 7: -t changes to -s after -s (makkis + -taant --> makkissaant)
40 define SAssimilation t -> s || V s "^" _  ;
41
```

Figure 6: Sample for Assimilation rules

d) Replacement

This component contains different types of replacement rules that are applied during word formation. When a root verb that ends with /-t/ connects to a vowel initial suffix then the root ending consonant /-t/ will be replaced by /-chch / and this rule is stated as TReplacement. When a verb root ends with /-s/ and if it connects with a vowel initial suffix then the final /-s/ will be replaced by /-shsh/ according to the rule Sreplacemecnt.

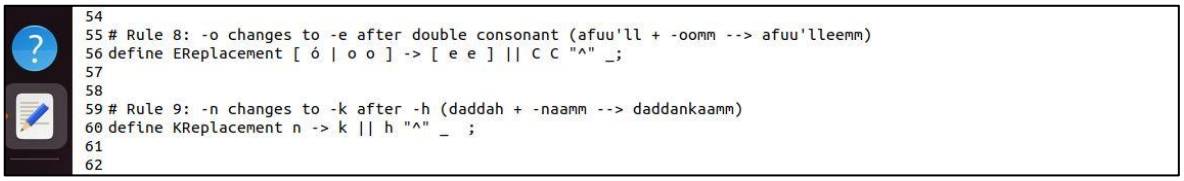


Figure 7: Sample for Replacement rues

e) Deletion

There are also deletion rules that are used to remove some letters in the word formation process. Some words that ends with /-h/ apples deletion rule when it connects with a suffix that starts with the consonant /-t/

f) Palatalization

The rule of palatalization deals with changing an alveolar consonant to a palatal consonant. It is applied when a root verb that ends with alveolar consonants such as /t/, /s/, /d/, /x/, /z/ and a vowel initial suffix is connected to it then the above alveolar consonants will be changed to their palatal counterparts like /ch/, /sh/, /jj/, /cc/, and /zh/ respectively. The vowel initial suffixes can be /-e/, /-o/ and /-i/. The following are possible palatalization rules in the morphological analyzer.

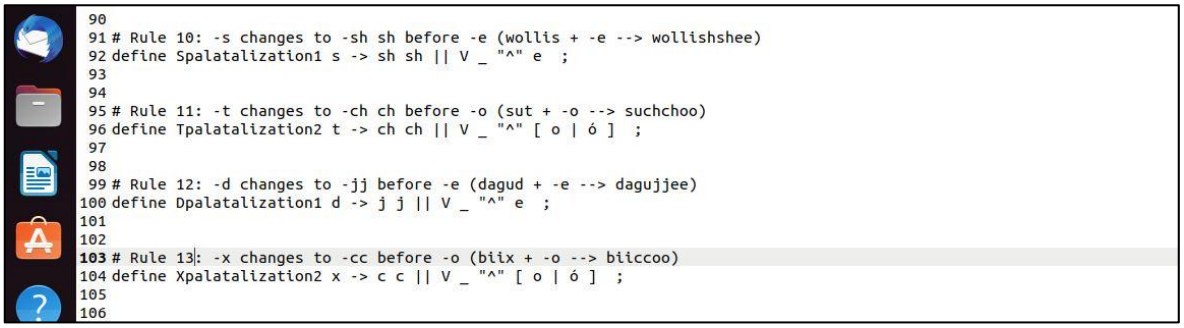


Figure 8: Sample for palatalization rules

g) Gemination

Consonant gemination is a cluster of consonants that occurs when a noun or a verb stem connects with vowel-initial suffixes. These vowel initial suffixes are suffixes that start with either in /-e/ or /-o/. Here the addition of these vowels makes the stem final consonant to be doubled.

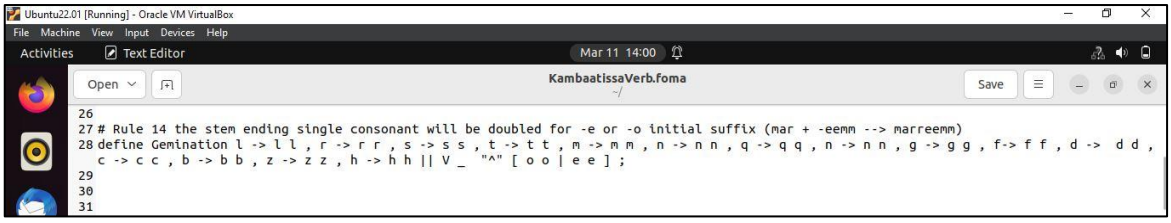


Figure 9: Sample for gemination rules

5.2.4. Lexicon Design

The system takes a lexicon as input. It processes the given word based on the parts of the input word and analyze the word into corresponding grammatical information and parts. The morphological generator operates in the opposite direction by combining the appropriate morpheme based on the root verb form and the required grammatical feature. The system generates various word forms after the lemma/root is provided with the appropriate tag sets that designates the grammatical features. The experiment tried to includes different kinds of affixes that show various grammatical features.

Sample lexicon script

The lexicon script is written in the text editor. In the lexicon file the first step is declaring multicharacter symbols. Then we declare the lexicon root, base word in which different affixes are attached to it. In this case the lexicon root is a verb. The following figure shows the declaration of root verb and multicharacter symbols in the lexicon file KambaatissaVerb.lexc.

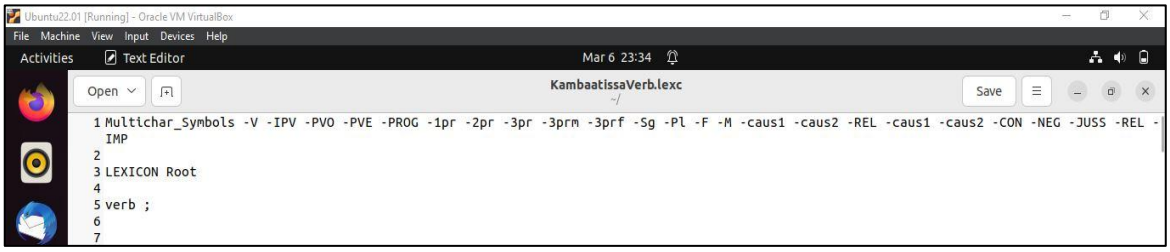


Figure 10: Multi-character strings declaration

The next step is defining the lexicon which is declared as a root. It contains lists of valid stem (root) in the language. Each root verb is defined using the variable **BASE**, which is a lexicon class with list of continuous classes that contains possible suffixes of the given root. The figure below shows some list of root verbs defined in the above lexicon file.

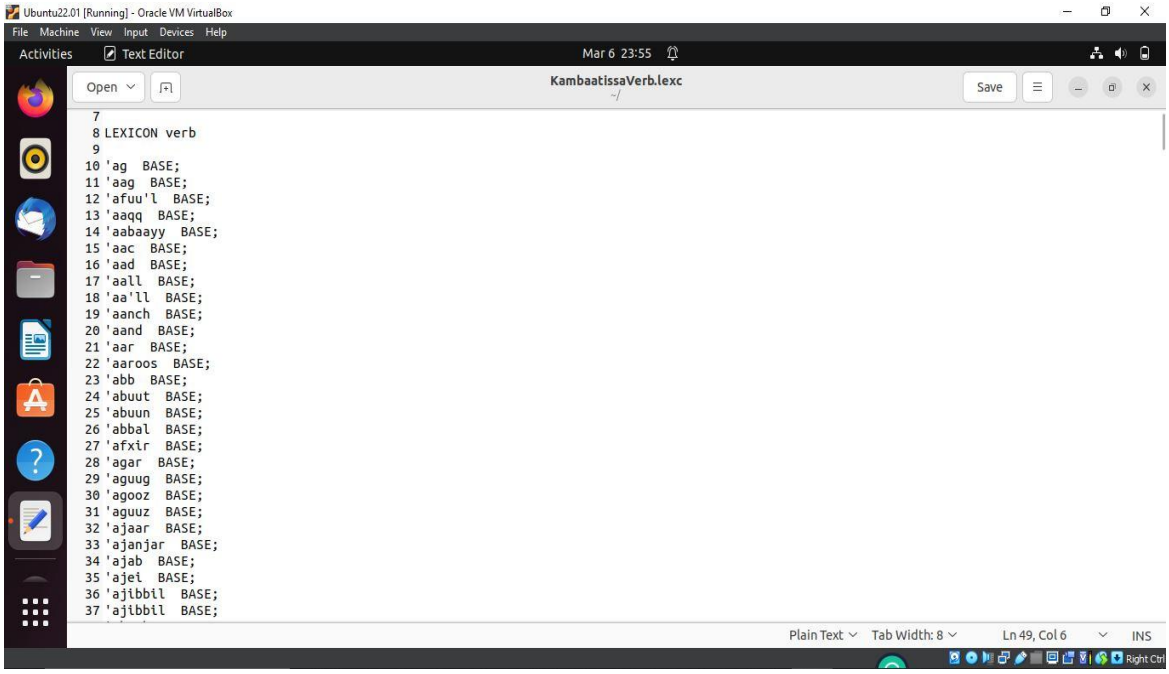


Figure 11: Root verb declaration

The entries of the class ***BASE*** are continuous classes. Each continuous class are further defined with their possible type of suffixes for each person and gender.

Figure 12: Continuous class declaration

The continuous classes which are declared above are defined with their suffixes. The figure below shows Jussive, which contains jussive verb marking suffixes. The suffixes are listed for each person (1st person, 2nd person and 3rd person), number (singular and plural) and

gender (masculine and feminine). It also displays the negation marking suffixes for jussive verbs.

Figure 13: Jussive verb class definition

After designing the above two files called the lexicon and alteration rules then the Foma toolkit is initiated to run the above files in the Ubuntu terminal. The following figure shows how to start the foma to build the lexicon file.

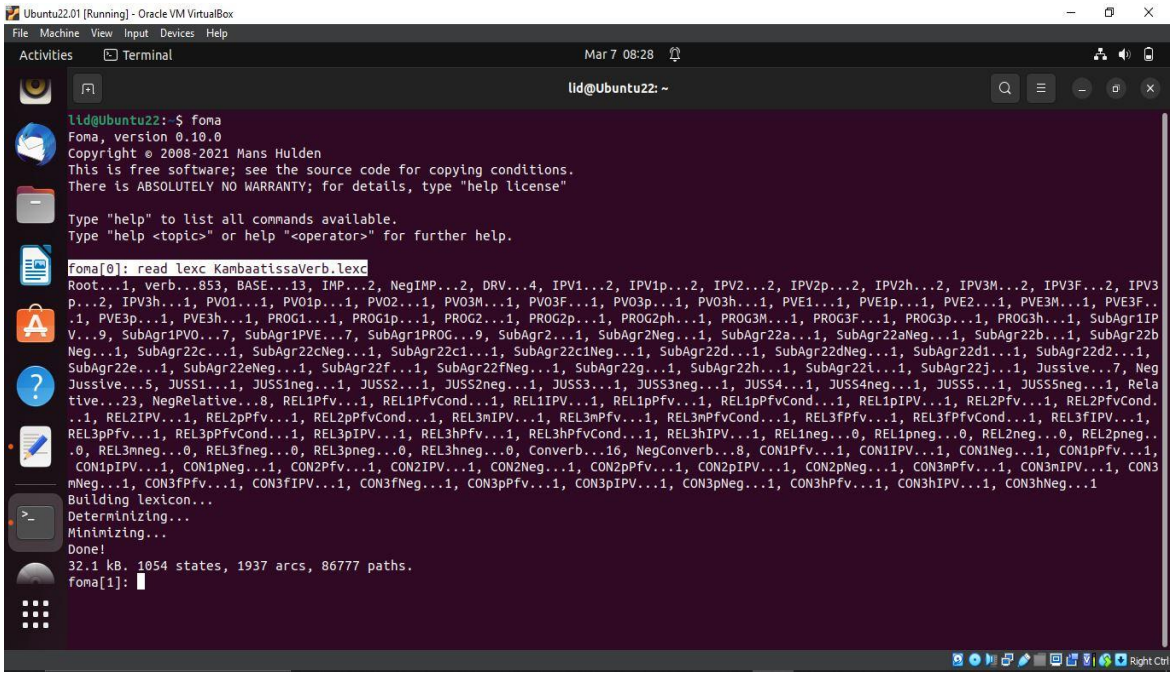


Figure 14: Starting foma

Running the .foma file (source KambaatissaVerb.foma) compiles the lexicon and morphophonological rule by joining them by composition. The figure below is a screen shoot from foma terminal to present this concept.

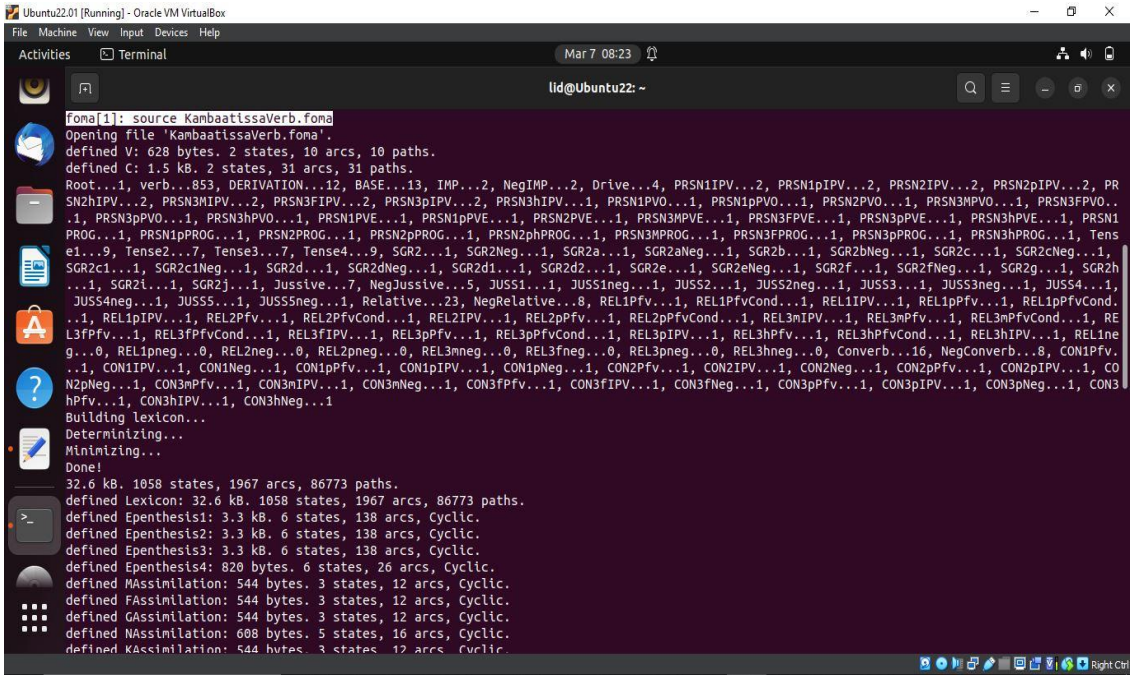


Figure 15: Compiling lexicon

The detail working of the morphological analysis and generation is presented in the next chapter.

CHAPTER SIX

6. Implementation and Result

6.1. Introduction

This chapter discusses the implementation of the proposed morphological analyzer and generator. It also presents the experimental result of the analyzer. The implementation of the morphological analyzer is performed based on the methodology discussed in chapter four and the design of the analyzer which is discussed in chapter five are strictly followed. In this section the implementation of the morphological analyzer and generator will be discussed in detail.

6.2. Implementation Environment

The proposed Morphological analyzer and generator for Kambaatissa is implemented using Foma, C library and compiler that is used for developing different natural language applications especially for constructing finite state automata and transducers. In the case of other NLP toolkit for morphological analysis like XFST, a large amount of lexicon file is built using regular expression but, the Foma toolkit simplifies the process of building a lexicon file because it uses the ***lexc*** language.

6.2.1. Foma

Foma is an open-source application that is used to compile finite state transducers. It is installed on the Linux operating system. In this work, we used the Ubuntu Linux operating system and Oracle VM VirtualBox machine to run the operating system.

To begin with foma software first it has to be installed on the Ubuntu terminal using the commands ***sudo apt-get update*** and ***sudo apt-get install foma-bin***.

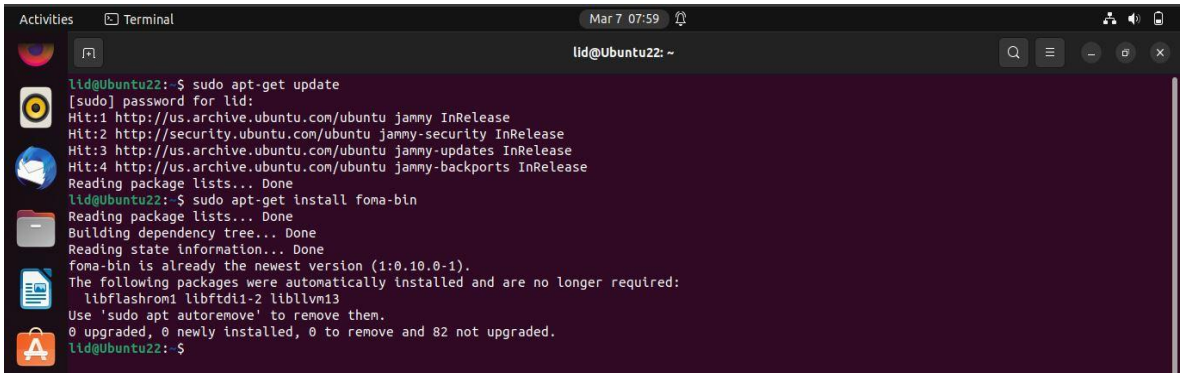


Figure 16: Commands to start Foma

The lexicon file which contains list of all root verbs with their morphotactic (continuous class) is prepared on a text editor and saved in the form of ***filename.lexc***. The filename can be any name that will be given by the researcher and ***.lexc*** is the file extension. The figure below is a sample of lexicon file with file name *KambaatissaVerb.lexc*

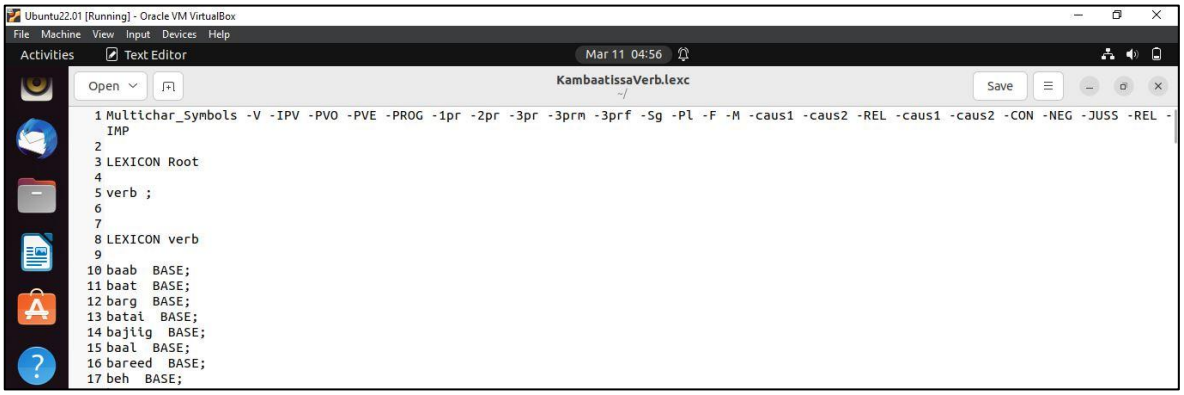


Figure 17: Sample Lexicon *KambaatissaVerb.lexc*

The rule-defining component that contains list of alteration rules also scripted on a text editor and saved in the form of ***filename.foma***. Here also any filename can be used and the file extension is ***.foma***. This component uses the lexicon file because the defined rules are applied to it so, the lexicon file called inside the rule component using the command ***read lexc filename.lexc***. The following figure shows sample of rule component with a file name *KambaatissaVerb.foma*

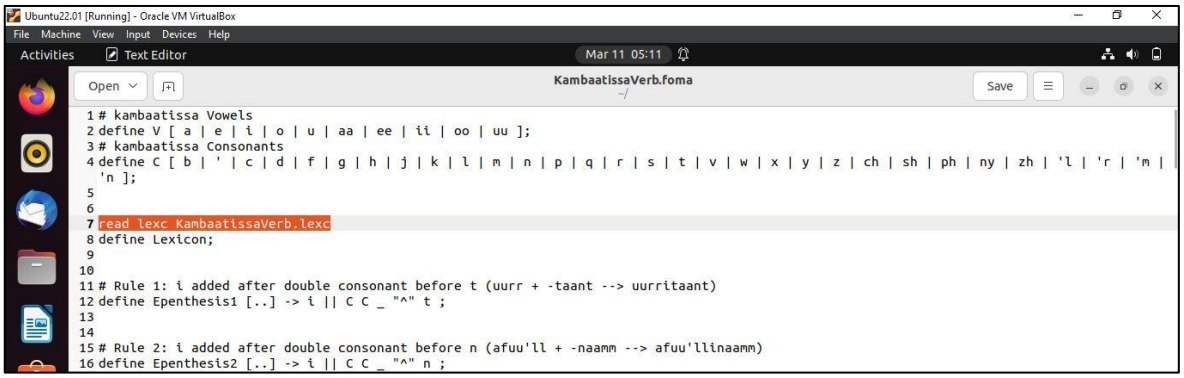


Figure 18: Sample Rule component *KambaatissaVerb.foma*

6.3. Experiment result for word formation by gemination

A verb root that ends with a single consonant can form a perfective verb by doubling its last consonant. Perfective verb marking suffixes are morphemes that start with the vowel -e

or -o so the root final consonant is doubled when it meets -e or -o initial perfective suffixes. In the experiment this rule is applied for all root verbs in the lexicon file that ends with a single consonant except alveolar consonants (d, t, s, x, z) because those root ending consonants do not follow this rule instead when they meet perfective suffixes, they perform palatalization. In the lexicon file, there are 550 root verbs that fulfill the above condition and there are 12,100 possibilities of forming perfective verbs, among those possible perfective verb formations 11,770 words are correctly generated verbs, and 330 verbs are wrongly generated. This because of some root verbs that ends with a glottal stop consonant which leads to the generation of wrong words when they perform gemination. Finally, the system generates 97.27% words correctly and the remaining 2.73% are wrong words. The following table presents the experiment result.

Table 13: Experiment result for gemination

Root verb that ends with a single consonant	Suffixes	words generated by consonant germination	Number of correct words	Number of wrongly generated words
550	-e	8,800	8,582	218
	-o	3,300	3,188	112

The following figure shows an FST graph formed from gemination.

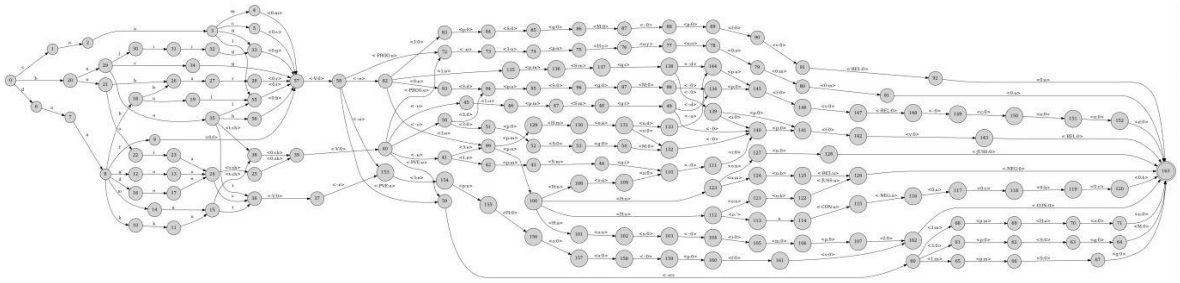


Figure 19: Sample FST graph for gemination

6.4. Experiment result for word formation by Palatalization

The process of converting the root ending alveolar consonant to corresponding palatal consonant to form perfective verbs is called palatalization. Alveolar consonants are (d, s, t, x and z) those alveolar consonants changed to their respective palatal consonants (jj, shsh, chch, cc and zhzh) when they meet a suffix that starts with -e or -o vowel and they form perfective verbs. In the lexicon file, some of the verb root ends with one of the above alveolar consonants. Verb root that ends with single consonant d =30, s =60, t =30, x =18,

z =10. In the lexicon file, 17 suffix morphemes start with -e sound, and 5 suffix morphemes that start with -o. totally there are 21 possible suffix morphemes start with -e and -o. For each consonant, 57.2% of correct words are generated and 42.8% are wrongly generated. The system generates correct words for most of the roots except when the third person honorific marking suffixes are attached to the root because this suffix morpheme has the same pattern with perfective verb markers and this results in the generation of the wrong words. The following table 14 shows the detail of the results.

Table 14: Experiment result for palatalization

Verb ending consonant	Number of words that ends with the consonant	Generated words	Correct words	Wrongly generated words
Root verb that ends with single consonant -d which have to be replaced by –jj	30	630	360	270
Root verb that ends with single consonant -s which have to be replaced by –shsh	60	1,260	720	540
Root verb that ends with single consonant -t which have to be replaced by -chch.	30	630	360	270
Root verb that ends with single consonant -x which have to be replaced by -cc.	18	378	216	162
Root verb that ends with single consonant -z which have to be replaced by -zhzh.	10	210	120	90

The Finite state transducer graphical view of the above word formation by palatalization is presented in the figure below

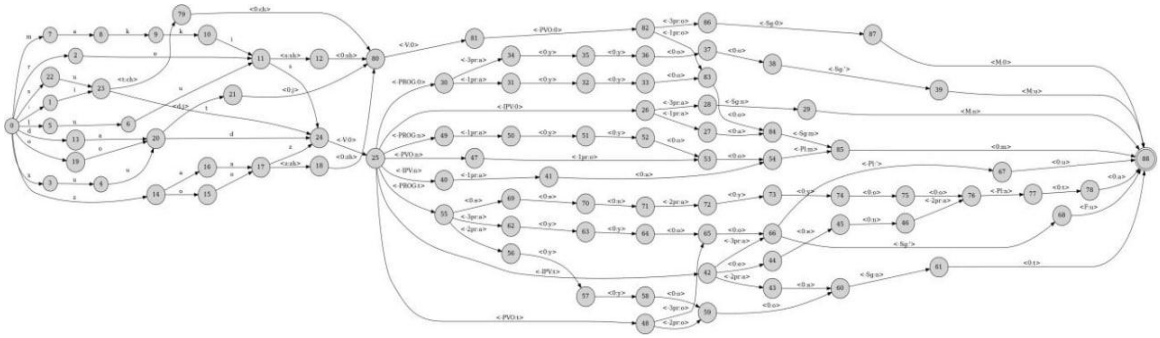


Figure 20: Sample FST graph for palatalization

6.5. Experiment result for Converb

Converbs are subordinate verbs that can be represented in two paradigms called perfective and imperfective. The experiment shows that the proposed analyzer and generator gives correct result for most of the words in the lexicon file but some words are wrongly generated because in the first and third-person singular forms, there is a distinction between the added grammatical feature when the root ends with a single consonant and when it ends in a double consonant. So this will create a conflict with the rule for their equivalent imperative forms. The result shows that the system generated 20,640 imperative verbs out them 19,285 which is 93.4% are correct and 1,355, which is 6.56% are wrongly generated results. Table 15 represents the detailed description of the result.

Table 15: Experiment result for converb

Total words	Paradigm	Suffix	Correct words	Wrongly generated words
860	Perfective	-i	1,674	46
		-n, -t, -teen, -een,	4,945	215
	Imperfective	-an, -nan, -tan, -eenan, -teenan	6,650	230
	Negation	-u’nnaachch	6,016	864

6.6. Experimental result of Imperative verb

Imperative verb takes the suffix morpheme *-i* for the singular and *-iye* for the plural in the second person. The morpheme -i in the second person singular is added after the root ending consonant but here the sound is devoiced because it is not audible. So, it does not be represented by adding to the root verb but instead, the root verb by itself is considered

as a second-person singular imperative form. It also has a negation form which is represented by the morphemes *-toot* and *-teenochche*. Table 15 below lists the amount of correct and incorrect words that are generated from the root words in the lexicon file. The result shows that the analyzer gives 3,440 results for imperative verbs and among them, 3,291 (95.7%) words are correctly generated and 140 (4.3%) are wrong results

Table 16: Experiment result of imperative verbs

Total words	Person	Suffix	Correct words	Wrongly generated words
860	Second person singular	-	860	0
		Negation (-toot)	802	58
	Second person plural	-iye	827	33
		Negation (-teenochche)	802	58

6.7. Experimental result of Jussive verb

Jussive verbs are generated by adding different suffix morphemes for each person and there is also negation morpheme to form negative jussive verbs. Table 15 describes the result using the number of correctly generated words and wrongly generated words. According to the result the system gives correct result for all the roots to form jussive verbs.

Table 17: Experiment result for jussive verb

Total words	Person	Suffix	Correct words	Wrong words
860	1prSg	-u	860	0
	1prPl	-no	860	0
	3prSgM	-un	860	0
	3prSgF/ 3prPl	-tun	860	0
	3prHon	-eenun	860	0
	Negation	-unka	860	0

6.8. Experiment result for word formation by Assimilation

In the word formation process, some consonants are replaced by their neighboring consonant because the sound of one consonant dominates the other. Root verbs that have a single obstruent consonant at the end follows the assimilation rule when they meet -t initial suffix and the obstruent replaces the position of the sound -t (t → (k, b, d, g, f, s, h, ch, j, c, q, x , ph, p, sh and z)). In the lexicon file, 63 suffix morphemes start with -t sound and this is 52% of the total suffix morphemes in the lexicon. The suffix morphemes that start with -n are 14 in number. Table 17 presents the result for each replacement rule. The result shows that the analyzer gives the correct result for all roots in the lexicon.

Table 18: Experiment result for Assimilation

Root ending consonant	Number of words that end with the consonant	Correct words	Wrongly generated words
The root ending g replaces the attached suffix initial t (t→g)	27	1,701	0
The root ending f replaces the attached suffix initial t (t →f)	29	1,827	0
The root ending s replaces the attached suffix initial t (t →s)	60	3,780	0
The root ending b replaces the attached suffix initial t (t →b)	27	1,701	0
The root ending d replaces the attached suffix initial t (t →d)	30	1,890	0
The suffix initial n replaces the root ending l in which it is attaching (l →n)	66	924	0
The suffix initial n replaces the root ending r in which it is attaching (r →n)	84	1,176	0
The root ending m replaces the attached suffix initial n (n →m)	23	322	0

6.9.Experiment Result for Derivation

There are four types of derivation marking morphemes those are causative 1(-s), Causative2 (-siis), Mid(-am) and Passive(-aqq/ ’). The result shows that more number of wrong words are generated in the passive derivational morphemes relatively from others and the reason behind this is it has two types of morphemes. The following table represents the result in detail.

Derivational morphemes	Correctly generated words	Wrongly generated words
Causative1(-s)	650	210
Causative2(-siis)	860	-
Mid(-am)	687	173
Passive(-aqq / ’)	538	322

6.10. Morphological analyzer

The figure below is an output screen that presents the output of the morphological analyzer using foma implementation environment. The foma command ***‘print pairs’*** is used to display pair of words that is the analysis of a given lexicon with its generated word form. The command ***‘up’*** is used to print the analysis of a word given the word form.

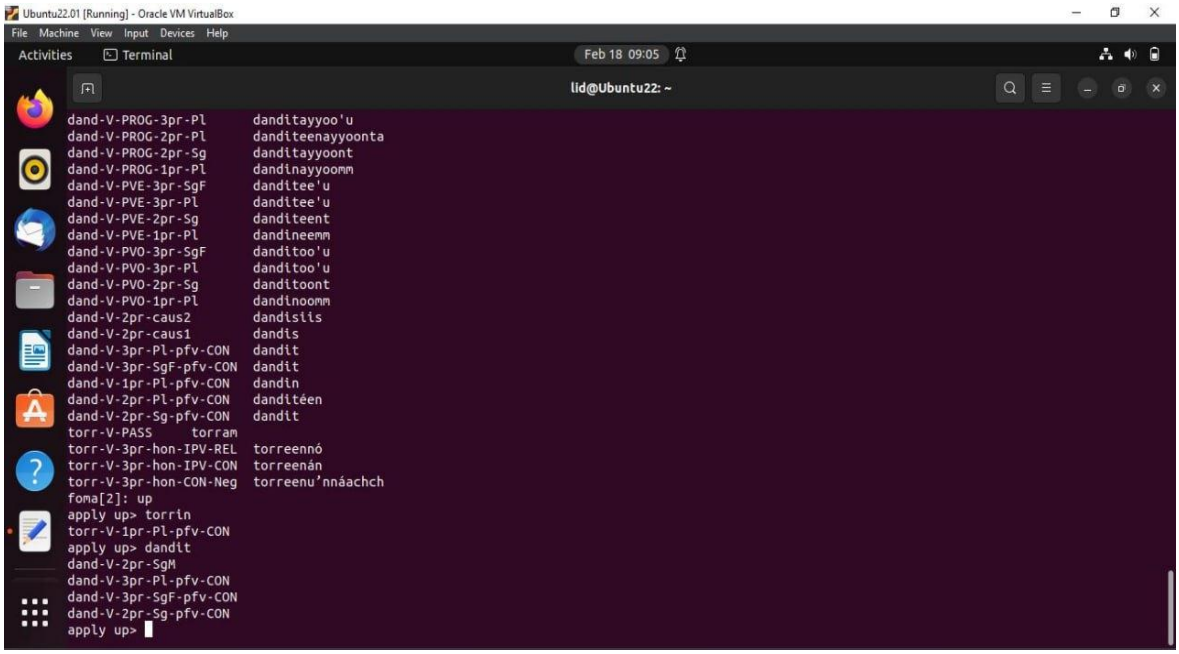


Figure 21: FST Morphological analyzer

In the figure above the morphological analyzer displays the analysis result for the root verb **“dand”** / *‘tolerate/ can’* with its inflectional and derivational morphemes. Here a word form can have more than one analysis result such as, in the above figure for the word **“dandit”** the analyzer prints four possible analysis result.

The analyzer can also print the overall analysis of words which is called a list of upper words. The foma command ‘***print upper words’*** gives us analysis results for all words in the lexicon file. The following figure displays the analysis result for the root verb **“noqorb”**/ *‘adultrate’*

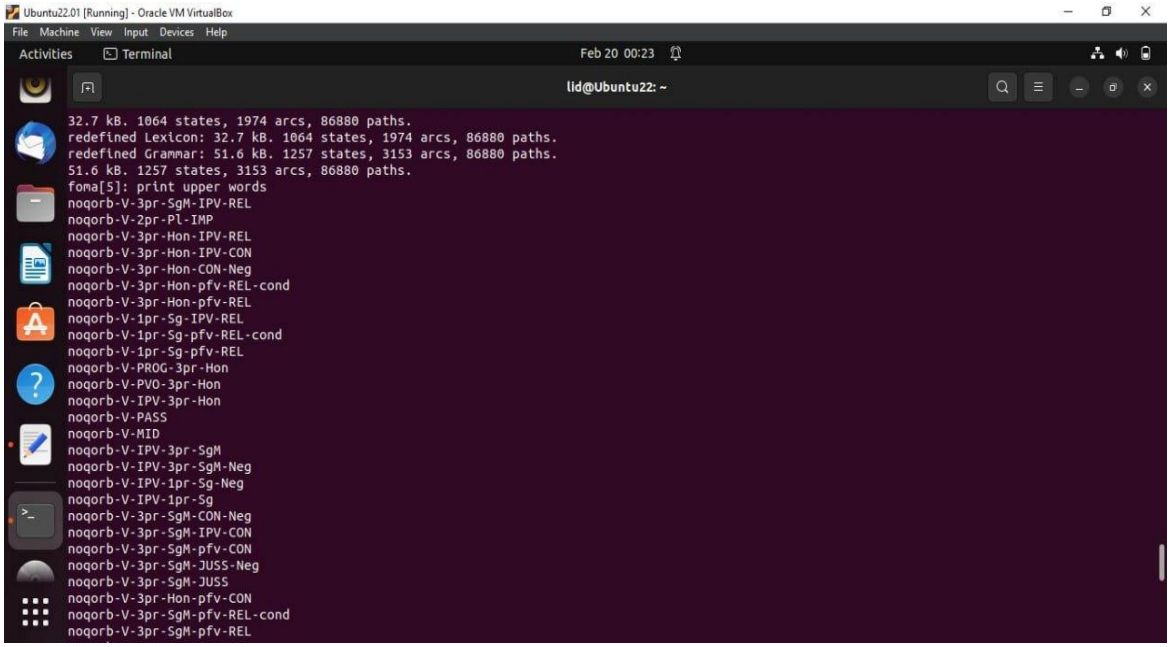


Figure 22: The command *print upper words*

6.11. Morphological generator

A morphological generator contains generated words. The following figure is a foma output screen that shows the list of generated word forms given a root verb with its grammatical features. The command ***‘down’*** is used to print the word form given a root word and its possible suffixes.

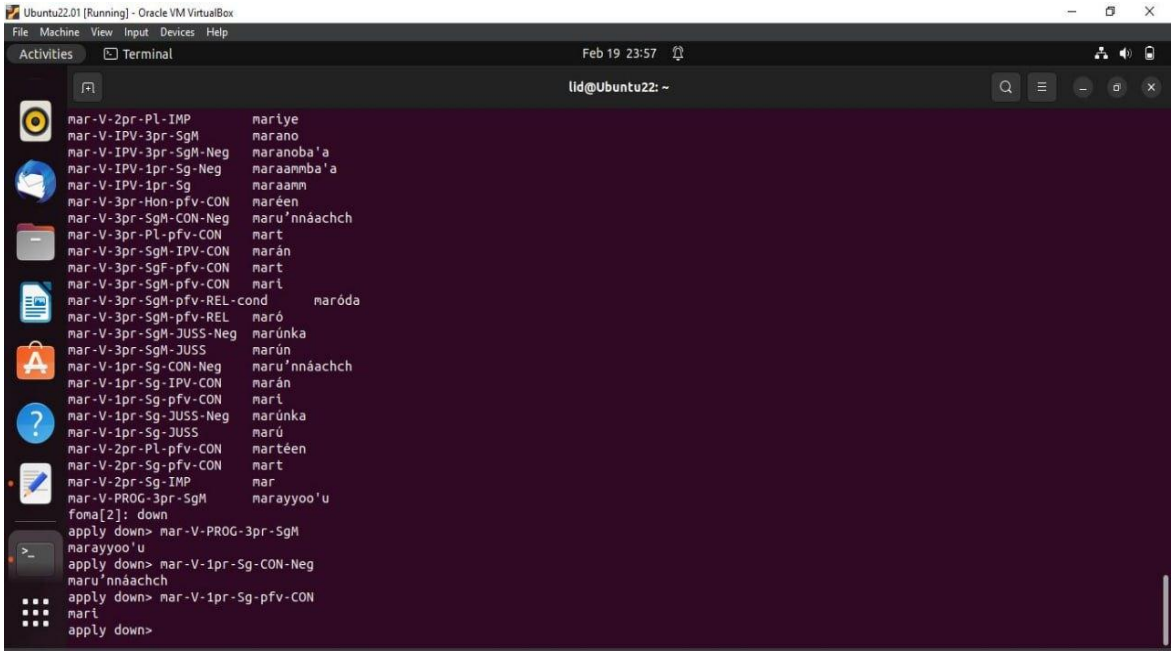


Figure 23: FST Morphological generator

In the figure above the root verb **"mar"**/ 'go' generates different word forms by taking different suffix morphemes that carry different grammatical information. The first word generated by the system "marayyoo'u" / *'He is going'* which is formed from the root verb "mar" / *'go'* and the grammatical tags for third person singular with masculine gender and the tense marker PROG (present continuous tense). By writing the analysis form of the target wordform the system gives the generated word form.

To get the overall generated words by the system which are called lower words, we use the foma command ***'print lower words'***. The figure below displays the generated word forms from the root verb "noqorb"/ *'adultrate'*

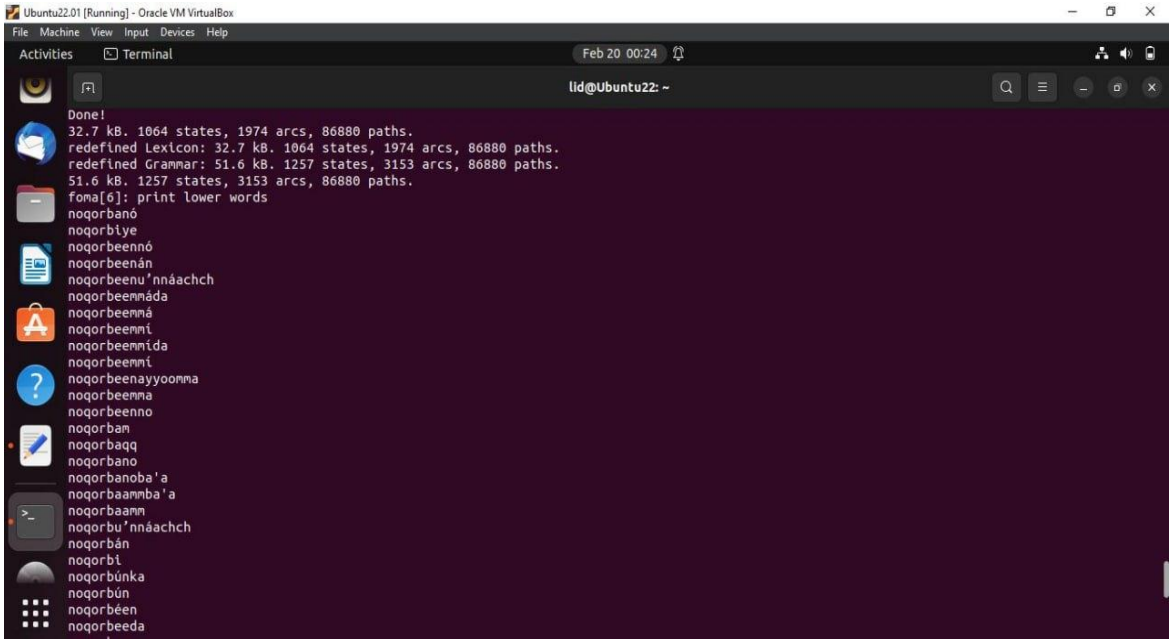


Figure 24: The command print lower words

Generally, the working mechanism of the morphological analyzer and generator is presented using a snap shoot from the Foma interface.

The first figure shows how the generator works; for a given root verb and its grammatical tag the system displays the generated word form and the first figure shows how the analyzer works; for a given word form, the system displays the root and grammatical tags representing the suffixes.

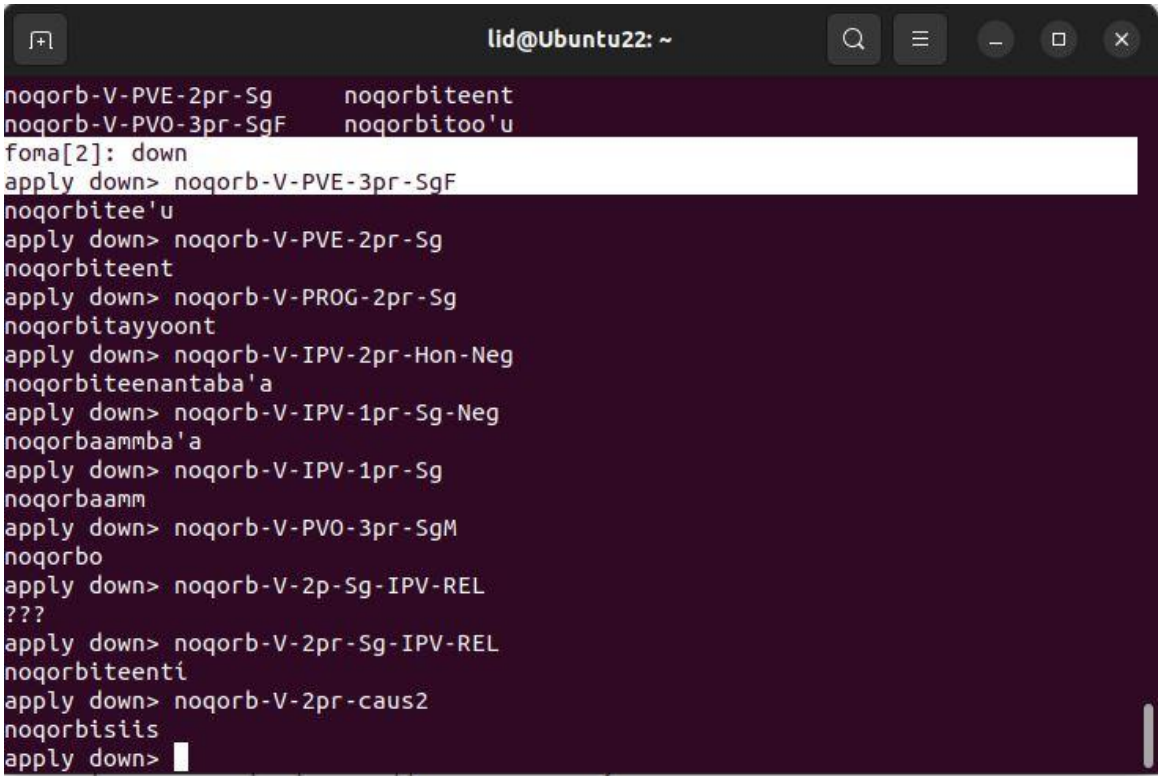


Figure 25: Sample for morphological generator given a root and grammatical tag

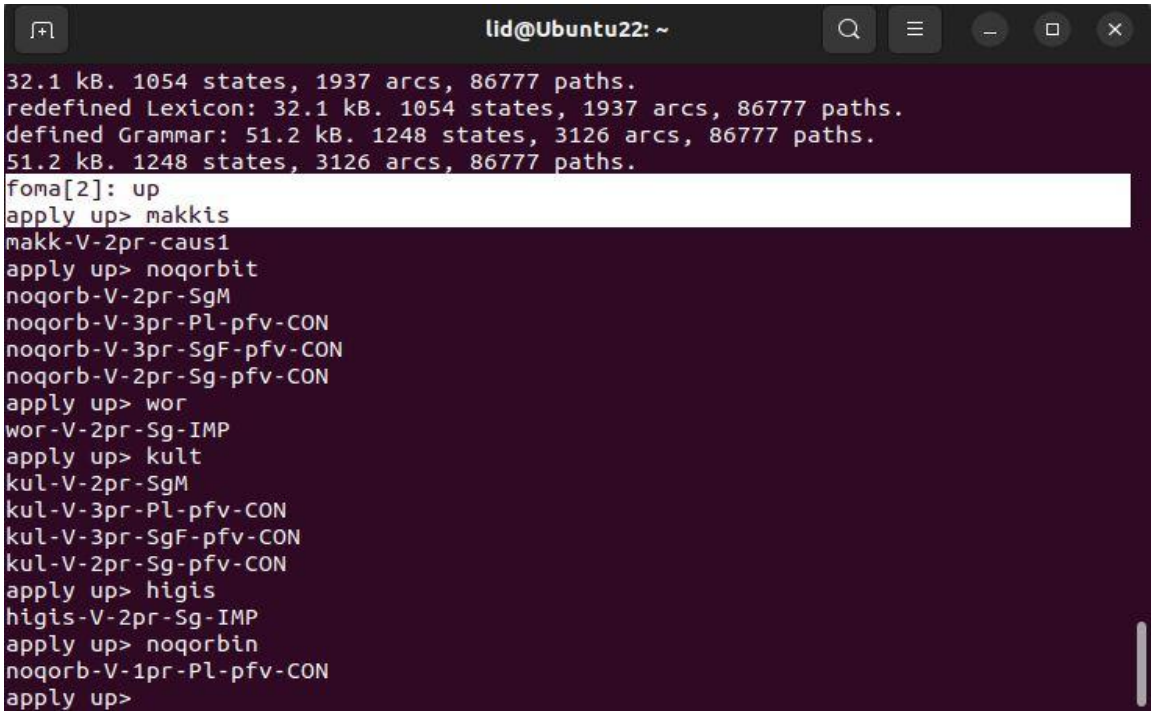


Figure 26: Sample for morphological analyzer given a word

6.12. Evaluation Result

The Kambaatissa morphological analyzer and generator generate a total of 92,020 words and out of them, 87,645 words are correctly generated. It is developed by using 860 root verbs of Kambaatissa language. The performance of the analyzer and generator is evaluated by comparing the words generated by the system with words collected from different Kambaatissa language textbooks and student modules for Wachemo University Kambaatissa department. But many word forms are not directly found in those documents. Therefore, we have used two language experts to evaluate whether these word forms are correct or not. Based on the judgement of the language experts the incorrect words cover 4.8% of the total words generated by the system.

Generally, this study was started by raising the research questions stated in the first chapter. The first question was How to approach computational aspects of the verbs and what are the challenges in the language while using FST? The language is under-resourced that means there are very limited linguistic resources especially on morphology of the language. In the language there is a consonant named glottal stop (’) and words are marked for accent using an accent marker (diacritic marker). Especially suffixes that are used for marking jussive and relative verbs are marked for accent and such symbols are difficult when used in different tools for implementation so, handling such issue is very important to get an efficient NLP applications for the language. The second question lies on the performance of FST approach in generating and analyzing the words in the given language. From this study we can understand that the FST approach performs well in analyzing and generating Kambaatissa words. Even if the verbs in Kambaatissa are very productive the method uses morphotactic, the order in which the suffix morphemes are arranged and morphophonological alteration rule, that guides the sound changes during morpheme combination. Finally, after conducting an experiment using the above components an accuracy of 95.2% is obtained from the system.

CHAPTER SEVEN

7. Conclusion, Recommendation and Future work

This chapter discusses the conclusion of the study on designing a morphological analyzer and generator for the Kambaatissa language using a finite-state transducer. It also presents recommendations and future works suggested to others who are interested in this research area.

7.1. Conclusion

Kambaatissa is one of the Ethiopian languages with rich morphology and low computational resource. A word in the language can have many affix morphemes attached to the root, and the order in which those affixes are combined can change the word's meaning. This nature makes the language to be difficult for computational tasks. Morphological analyzers are useful tools that make the language easier for other computational works. Kambaatissa morphological analyzer and generator is the first step that can be a base for the next higher-level natural language processing applications

We proposed a morphological analyzer and generator to perform morphological processing for the Kambaatissa language. It is implemented using a rule-based approach called a Finite state transducer. It performs morphological analysis, which is identifying root verbs and morphemes that carry grammatical information, and morphological generation, which is combining the root with its affixes to generate a word form. For the implementation, we have collected a list of root verbs and possible suffixes of the root to form different words. In addition to this, the pattern in which the root and affixes are combined is studied carefully with the help of different written books, articles, and language experts who are engaged in different research related to the language. The collected root verbs and affixes with their correct pattern are used as input for the finite state transducer. An experiment is done to get the correct analysis and generation of words. Finally, the performance of the system is evaluated.

The finite state transducer has two components namely, a lexicon which contains a root with its morphotactic, and the rule component which contains morphophonological

alteration rules that transform the intermediate form to the final form. It performs analysis and generation based on the linguistic rules defined in the rule component.

The study achieved good result by passing different challenges. Some of the challenges are the lack of structured linguistic resources especially on the morphotactic arrangement of the word and morphophonological rules in the language. This makes us to invest more time in studying the nature of the word and it needs to take more time with linguistic experts. The other challenge was spelling errors in the documents collected but later it was solved with the help of language experts.

In the study 860 verb roots are used for building lexicon for the experiment and the system generates 92,020 word forms with their analysis. Finally, the system generates 95.2% of the word correctly.

7.2. Recommendation

Language is the most effective way of communication between human beings. In Ethiopia, there are more than 86 languages spoken by the people. But, most of the languages are not used in the computational tasks and Kambaatissa is one of them. There are few works in this language regarding NLP applications so we recommends that other works like Part of speech tagger, Spell-checker, grammar checker, machine translation, and others can be done using this Morphological analyzer.

7.3. Future work

From the knowledge obtained during this study and from the observed gap during reviewing

different related works, for those who have an interest to work in this area in the future and as this study progresses in future, the work to be added in the future is suggested as follows

In the very beginning of the research we planned to develop a morphological analyzer and generator for both verb and noun word classes but when we study the nature of Kambaatissa noun it is vast and hard to cover using rule-based approach. This is because the case marking suffixes are eight in number and they have eighteen declinations. For each declination there are different case marking vowels. We tried to discuss with linguistic experts and can't find general rule for the case marker suffixes so, in this study

the analyzer only works for the verb word class. Developing morphological analyzer for noun, adjective and the remaining word classes will be future work.

In this study we used a rule-based method called an FST for implementation but a machine learning (corpus based), deep learning methods can also be used in future works.

7.4. Contribution

In this study, we studied the morphology of Kambaatissa verbs and applied finite state transducer for morphological analysis and generation. The main contributions of the study are:

- Developing a set of resources like verb root lexicon and list of generated verbs from the morphological generator.
- The analyzer can further be used as a primary step for attaining better output in different language processing applications such as, spell checker, machine translation, etc…

References

[1] J. Piskorski, A. Yli-Jyr a, and B. W. Watson, *Finite-State Methods and Natural Language Processing: Post-Proceedings*, vol. 191. 2009.

[2] B. Abera Hordofa and S. Dechasa Degefa, "A Review of Natural Language Processing Techniques: Application to Afan Oromo," *Int. J. Comput. Appl. Technol. Res.*, vol. 10, no. 03, pp. 051–054, 2021, doi: 10.7753/ijcatr1003.1001.

[3] V. Dhanalakshmi, M. Anandkumar, R. U. Rekha, C. Arunkumar, K. P. Soman, and S. Rajendran, "Morphological analyzer for agglutinative languages using machine learning approaches," *ARTCom 2009 - Int. Conf. Adv. Recent Technol. Commun. Comput.*, pp. 433–435, 2009, doi: 10.1109/ARTCom.2009.184.

[4] H. M. Hussien and Y. N. Shiferaw, *Information and Communication Technology for Development for Africa*, vol. 244. 2018.

[5] M. Gasser, "HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya," *Conf. Hum. Lang. Technol. Dev.*, no. August, pp. 94–99, 2011.

[6] Y. Treis, "Form and function of case marking in Kambaata," *Afrikanistik online*, pp. 1–22, 2006, [Online]. Available: http://www.afrikanistik-online.de/archiv/2006/379.

[7] Joshua, "Kambata in Ethiopia people group profile _ Joshua Project." 2023, [Online]. Available: https://joshuaproject.net/people_groups/12468/ET.

[8] Y. Treis, *A Grammar of Kambaata (Ethiopia), part 1: Phonology,Nominal Morphology and Non-verbal Prediction*. 2008.

[9] Y. Treis, "Cardinal Numerals in Kambaata," *VIVA Africa 2007. Proc. IInd Int. Conf. African Stud. Pilsen, 27-28 April*, pp. 55–70, 2007.

[10] G. Sai Kiranmai, K. Mallika, M. Anand Kumar, V. Dhanalakshmi, and K. P. Soman, "Morphological Analyzer for Telugu Using Support Vector Machine," *Commun. Comput. Inf. Sci.*, vol. 101, pp. 430–433, 2010, doi: 10.1007/978-3-642-15766-0_68.

[11] R. Kannan and F. Andres, *Data Engineering and Management*. 2010.

[12] K. Sarveswaran, G. Dias, and M. Butt, *ThamizhiMorph: A morphological parser for the Tamil language*, vol. 35, no. 1. 2021.

[13] M. Korobov, "Morphological analyzer and generator for Russian and Ukrainian languages," *Commun. Comput. Inf. Sci.*, vol. 542, pp. 330–342, 2015, doi: 10.1007/978-3-319-26123-2_31.

[14] B. Premjith, K. P. Soman, and M. A. Kumar, "A deep learning approach for Malayalam morphological analysis at character level," *Procedia Comput. Sci.*, vol. 132, pp. 47–54, 2018, doi: 10.1016/j.procs.2018.05.058.

[15] L. Noce, A. Zamberletti, I. Gallo, G. Piccoli, and J. A. Rodriguez, *Automatic prediction of future business conditions*, vol. 8686. 2014.

[16] G. De Pauw *et al.*, "Learning Morphological Rules for Amharic Verbs Using Inductive Logic Programming," *Proc. Work. Lang. Technol. Norm. Less-Resourced Lang.*, pp. 6–12, 2012.

[17] M. Degefa, "Morphological Analyzer for Afaan Oromoo Using Machine Learning," 2020.

[18] D. L. Lalego, "College of Natural Sciences Department of Computer Science Development of Morphological Analyzer for Sidaamu Afoo Desta Legese Lalego Advisor : Yaregal Assabie (PhD)," no. June, 2020.

[19] B. Desta, "Design and Implementation of Automatic Morphological Analyzer for Ge ' ez Verbs," *Master's Thesis*, pp. 1–144, 2010, [Online]. Available: aau.etd.edu.et.

[20] R. Sivan, "Study on Morphological Analyzer and Generator for Malayalam Remya Sivan," *Int. J. Eng. Sci. Invent.*, vol. 8, no. 03, pp. 73–77, 2019.

[21] Y. Treis, Y. Treis, and Z. Jan, "Cardinal numerals in Kambaata To cite this version :," 2012.

[22] F. Fabbro, "Introduction to language and cerebellum," *J. Neurolinguistics*, vol. 13, no. 2–3, pp. 83–94, 2000, doi: 10.1016/S0911-6044(00)00005-1.

[23] W. English and C. Other, “5 Morphology and Word Formation key concepts,” pp. 121–146.

[24] J. Hana and L. Typology, “Intro to Linguistics – Morphology,” no. 1, pp. 1–8, 2011, [Online]. Available: Retrieved 9/14/12 from http://www.ling.upenn.edu/courses/ling001/language_change.html.

[25] J. Manker, “Morphological Typology What is Typology ?,” no. February, pp. 1–34, 2016.

[26] H. Chimhundu, “Computational Modeling of Agglutinative Languages: The Challenge for Southern Bantu Languages,” *Arusha Work. Pap. African Lang.*, vol. 3, no. 1, pp. 52–81, 2021.

[27] E. Bender, D. Flickinger, J. Good, and I. Sag, “Montage: Leveraging advances in grammar engineering, linguistic ontologies, and mark-up for the documentation of underdescribed languages,” *SALTMIL Work. Lr. 2004 First Steps Lang. Doc. Minor. Lang.*, pp. 36–39, 2004.

[28] S. Thottingal, “Finite State Transducer based Morphology analysis for {M}alayalam Language,” *Proc. 2nd Work. Technol. MT Low Resour. Lang.*, pp. 1–5, 2019, [Online]. Available: https://www.aclweb.org/anthology/W19-6801.

[29] J. S. SUMAMO, “DESIGNING A STEMMING ALGORITHM FOR KAMBAATA TEXT: A RULE BASED APPROACH,” *St. Mary’s Univ. Addis Ababa*, vol. 11, no. 1, pp. 1–5, 2018, [Online]. Available: http://197.156.93.91/bitstream/123456789/4455/1/Designing a Stemming Algorithm for Kambaata Text - A Rule Based Approach_Print Version3.pdf.

[30] J. Grieve, *Dialect variation*. 2015.

[31] S. S. Review, “Some Comparative Phonology and Morphology of Kambata and Oromo Tilahun Negash Mekuria Department of English Language and Literature Debre Markos University Ethiopia,” vol. 6, no. 1, 2020.

[32] K. Gorman, “Computational morphology Early work.”

[33] a P. Report, M. O. F. Technology, and C. Engineering, “MORPHOLOGICAL ANALYZER FOR MALAYALAM VERBS CB206CN019 in partial fulfillment for

the award of the degree," no. July, 2008.

[34] I. Zitouni, *Natural language processing of semitic languages*. 2014.

[35] L. Karttunen, R. M. Kaplan, and A. Zaenen, "Center for the Study of language and Information StanJbrd University lexical level surface level lexical level surface level," pp. 23–28, 1992.

[36] S. Moeller, G. Kazeminejad, A. Cowell, and M. Hulden, "A neural morphological analyzer for Arapaho verbs learned from a finite state transducer," *Proc. Work. Comput. Model. Polysynthetic Lang.*, pp. 12–20, 2018, [Online]. Available: https://www.aclweb.org/anthology/W18-4802.

[37] A. P. J and S. K. P, "Computational Morphology and Natural Language Parsing for Indian Languages: A Literature Survey," *Int. J. Sci. Eng. Res.*, vol. 3, no. 3, pp. 136–146, 2012, [Online]. Available: http://www.ijser.org.

[38] K. Sgarbas, N. Fakotakis, and G. Kokkinakis, "A straightforward approach to morphological analysis and synthesis," *Comlex 2000*, pp. 31–34, 2000.

[39] J. Baxi, P. Patel, and B. Bhatt, "Morphological Analyzer for Gujarati using Paradigm based approach with Knowledge based and Statistical Methods," *Proc. 12th Int. Conf. Nat. Lang. Process.*, no. December, pp. 178–182, 2015, [Online]. Available: https://www.aclweb.org/anthology/W15-5927.

[40] K. Shaalan, "Rule-based Approach in Arabic Natural Language Processing," *Int. J. Inf. Commun. Technol.*, vol. 3, no. 3, p. 11{\textendash}19, 2010, [Online]. Available: http://www.ieee.ma/IJICT/IJICT-SI-Bouzoubaa-3.3/3 - Khaledl.pdf.

[41] D. Kumar, M. Singh, and S. Shukla, "FST Based Morphological Analyzer for Hindi Language," *Int. J. Comput. Sci. Issues*, vol. 9, no. 4, pp. 349–353, 2012.

[42] Kitaw Ayele, "DESIGN OF A MORPHOLOGICAL GENERATOR AND ANALYZER FOR SIDAAMU," 2021.

[43] T. A. Gebreselassie, J. N. Washington, M. Gasser, and B. Yimam, *A finite-state morphological analyzer for wolaytta*, vol. 244. Springer International Publishing, 2018.

[44] S. Menaka, V. S. Ram, and S. L. Devi, "Morphological generator for tamil," *Proc.*

Knowl. Shar. event Morphol. Anal. Gener., no. January, pp. 82–96, 2010.

[45] A. Kuznetsova and F. M. Tyers, "A finite-state morphological analyser for Paraguayan Guaraní," *Proc. 1st Work. Nat. Lang. Process. Indig. Lang. Am. Am. 2021*, pp. 81–89, 2021, doi: 10.18653/v1/2021.americasnlp-1.9.

[46] Ö. Kiliç, "Turkish Journal of Electrical Engineering and Computer Sciences TRMOR : a finite-state-based morphological analyzer for Turkish," vol. 27, no. 5, 2019, doi: 10.3906/elk-1902-125.

[47] Y. Treis, Y. Treis, C. Form, and Y. Treis, "The apprehensive in Kambaata (Cushitic): Form , meaning and origin To cite this version : HAL Id : hal-03095794 The apprehensive in Kambaata (Cushitic): Form , meaning and origin," 2021.

[48] Y. Treis, "Expressing Future Time Reference in Kambaata," vol. 20, no. 2, pp. 132–149, 2011.

[49] Y. Treis and Y. Treis, "Past tense in Kambaata (Cushitic) To cite this version : HAL Id : hal-01989628 Past tense in Kambaata (Cushitic)," 2019.

[50] Y. Treis, Y. Treis, K. Cushitic, M. Miestamo, L. Veselinova, and Y. Treis, "Negation in Kambaata (Cushitic) To cite this version : HAL Id : hal-02332852 Negation in Kambaata (Cushitic)," 2020.

[51] Y. Treis, "Motion events in Kambaata," *Annu. Publ. African Linguist.*, vol. 5, no. July, p. (?), 2007.

[52] T. Wachemo, "Causative Construction in Kambaata." 2013.

[53] S. Goundar, M. Universities, and B. Technologies, "Chapter 3 – Research Methodology and Research Method," no. May, 2019.

[54] S. Demeyer, "Research methods in computer science," in *IEEE International Conference on Software Maintenance, ICSM*, 2011, no. March, p. 600, doi: 10.1109/ICSM.2011.6080841.

[55] K. Beesley and L. Karttunen, "Finite State Morphology Homepage," *Comput. Linguist.*, vol. 30, no. 2, pp. 3–5, 2003, [Online]. Available: http://acl.ldc.upenn.edu/J/J04/J04-2006.pdf.

[56] A. Chandía, “FST Morphological Analyser and Generator for Mapud\"ungun,” 2021, [Online]. Available: http://arxiv.org/abs/2109.09176.

[57] D. Zeman, “A Morphological Analyzer for Shipibo-Konibo,” pp. 131–139, 2018.

[58] H. Dolatian, D. Swanson, and J. Washington, “A Free / Open-Source Morphological Transducer for Western Armenian,” vol. 1, no. June, pp. 1–7, 2022.

[59] F. M. Tyers, J. N. Washington, A. Bayyr-ool, and A. Salchak, “A Finite-State Morphological Analyser for Tuvan,” pp. 2562–2567, 2013.

Appendixes

Appendix 1: List of Verb roots used for the experiment

'ass
'aass
'awwan
'a'l
'aansh
'agur
'aancai
'amu'rr
'afaafur
'ajuuj
'akaak
'akkar
'akkai
'akkis
'alallaas
'albajj
'albul
'alceenn
'alees
'allaa'll
'allaas
'allaat
'allaggis
'alliis
'aloott
'anabbab
'argis
'ajiiff
'alooyy
'alaaww
'amaad
'ajiif
'ajuuj
'ajanjars
'ajab
bai
bajig
bakkaar
balees
balallees
bandees
baraar
baraat
barbad
barcheis
bareed
beh
birs

'ag

'aag

baab

baat

barg

batai

bajiig

baal

baalagees

baad

baadees

baaf

ba'am

ba'ans

baancaar

baan

baaqbaall

baaragaar

baas

babbars

badar

bag

biix

booras

bobar

bogg

buss

bahi'r

'ajei

'ajibbil

'ajibbil

'abaabun

'abbaakan

'abbaas

'abbal

'abooyy

'abunn

'acaaqur

'aceeqal

'achche

'agamas

'aphph

'adiss

'adaraar

'adab

'addal

'adees

'adoob

'ajibbil

'anj

'agud

'af

cacar

caccaf

cafal

cafalam

caff

cakk

calallaaq

call

camb

canc

canf

cann

caraq

canc

caakk

ceem

cebb

ceecaar

ceem

ce'rram

bahit

caakk

caal

caam

caancuf

caac

daadees

daadfatt

daaf

daagoos

daakkut

daamat

dab

dabaabull

dabaabur

dabal

dabars

daddah

dad

dag

dand

dagud

da'll

daqq

dakaak

dalas

dam

dans

dambalaaq

dambaq

damm

damuum

dan
dansaqqam
dans
dand
dandeeh
dandeeqq
dandees
dannab
danq
daqqans
daqqansiis
darab
darbai
darg
darsh
das
dayy
debee'll
deedil
deegar
deemmam
deffe'll
degelesh
dei
dereer
desh
dib
dimb
di'n
dirr
digin
door
dor
dun
duud
duub
dul
dawwo'l
'ecer
'ed
'edel
'eeb
'ees
'eec
'eel
'ee'nn
'effel
'eged
'eger
'ejebber
'ekk
'elees
'e'll
'ellaww
'elm
'emeleel
'en
'encer
'en
'encer
'endebeer
'enkee'nn
'er
'ereer
'erg
'erkes
'eta'rr
'ez
'ezù
faal
faangayy
fanqal
fa'
farr
faarei
faat
faaxuul
fad
faf
fakkaakk
fal
fan
fandalal
fanfan
fandalal
fanfan
fang
fanxiiz
far
farshimm
fatt
fayy
feeg
feettall
feffeeg
feneqqel
fennegett
fenq
feqqe'nn
fenqe'nnù
fereffe
ferefferù
ficiniin
ficiriiq

fidiicc
fiffiid
fiicoos
fiiff
fiikk
fiil
fiiq
fikkaan
finc
findiliiq
firiix
fint
ful
ga''
gal
gaffar
gaf
ga'mm
gajaajj
galabax
gaamaat
gaabb
gaaceer
gaagei
gaajj
gaaggab
gaaggaal
gaajool
gaam
gaamaar
gaa'nn
gaanguyy
gaaxax
gaaz
gabaax
gabal
gabbooxx
gabbar
gacai
gaf
gaffar
gafuuc
gagaam
gaggab
gagga'rr
gai
gajaajj
gal
galaxx
galb
gallaac
gallaall
galmai
gam
gamam
gamax
gambabb
gambax
ga'mm
gammis
gan
geeq
gei
gis
gib
gifaat
gic
gid
gifaat
giggl
goshooshsh
gomb
gob
gorax
gug
haam
haasaaww
hab
hangaar
hanqaf
haww
hei
he'mm
hig
hiil
hir
hiir
hiliq
hinci
hiq
hoog
hog
hogai
hogob
hos
huf
hun
hulut
hurur
'iib
'ibiib
'ibaad
'ichchim
'idaad
kill
kot
kotoot
kul
lall
naqqas
nubaaphph
nugguss
'onxah
'onkolo'll
qulaat
qulf
quss
raffad
reh
'ub
'uucc
'uurr
'uujj
'usheexx

wez
wiim
wollis
wor
woqqar
xab
xaaf
xahaaqq
xa'mm
xanq
xoof

xuud
xuf
zaanool
zaass
zab
zaggab
zeezaar
zegeell
zikk
zilq
zogiir
zokkor
zooz

zug
zunn
zeez
zei
ze'ees
zirr
zuug
zuuq
zebb
zeel
zeenzar

Appendix 2: Alternation Rules used for the experiment

```
# kambaatissa Vowels

define V [ a | e | i | o | u | aa | ee | ii | oo | uu ];

# kambaatissa Consonants

define C [ b | ' | c | d | f | g | h | j | k | l | m | n | p | q | r | s | t | v | w | x | y | z | ch | sh | ph | ny | zh | 'l | 'r | 'm | 'n ];

read lexc last.lexc

define Lexicon;

# Rule 1: Adding i after double consonant before t (uurr + -taant --> uurritaant)
```

```
define Epenthesis1 [..] -> i || C C _ "^" t ;

# Rule 2: Adding i after double consonant before n (afuu'll + -naamm --> afuu'llinaamm)

define Epenthesis2 [..] -> i || C C _ "^" n ;

# Rule 3: Adding i after double consonant before s (debee'll + -s --> debee'llis)

define Epenthesis3 [..] -> i || C C _ "^" s ;

# Rule 4: Adding i after double consonant before s (debee'll + -s --> debee'llis)

define Epenthesis4 [..] -> i || s h _ "^" t ;

# Rule 5: Replace -n with -m after the consonant -m (kam + -nayyoomm --> kammayyoomm)

define MAssimilation n -> m || m "^" _  ;

# Rule 6: Replace -t with -f after the consonant -f (xoof + -taant --> xooffaant)

define FAssimilation t -> f || f "^" _  ;

# Rule 7: Replace -t with -g after the consonant -g (hig + -taant --> higgaant)

define GAssimilation t -> g || g "^" _  ;

# Rule 8: Replace -r with -n before the consonant -n (mar + -naamm --> mannaamm)

define NAssimilation r -> n || _ "^" n  ;

# Rule 9: Replace -t with -k after the consonant -k (dakaak + -taant --> dakaakkaant)

define KAssimilation t -> k || k "^" _  ;

# Rule 10: Replace -t with -s after the consonant -s (makkis + -taant --> makkissaant)

define SAssimilation t -> s || V s "^" _  ;

# Rule 11: Replace -t with -d after the consonant -d (xuud + -taant --> xuuddaant)

define DAssimilation t -> d || V d "^" _  ;

# Rule 12: Replace –o with -e after the double consonant (afuu'll + -oomm --> afuu'lleemm)

define EReplacement [ ó | o o ] -> [ e e ] || C C "^" _;

# Rule 13: Replace -l with -n before the consonant -n (kul + -naamm --> kunnaamm)

define LReplacement l -> n || _ "^" n  ;

# Rule 14: Replace -t with -b after the consonant -b (ibiib + -tayyoont --> ibiibbayyoont)

define BReplacement t -> b || b "^" _  ;

# Rule 15: Replace -t with -b after the consonant -b (ibiib + -tayyoont --> ibiibbayyoont)

define QReplacement t -> q || q "^" _  ;
```

Rule 16: Replace -t with -b after the consonant -b (ibiib + -tayyoont --> ibiibbayyoont)

define ZReplacement t -> z || z "^" _ ;

Rule 17: Replace -n changes to -k after the consonant -h (daddah + -naamm --> daddankaamm)

define KReplacement n -> k || h "^" _ ;

Rule 18: Replace -m changes to -n before -t (daddah + -naamm --> daddankaamm)

define NReplacement m -> n || _ "^" t ;

Rule 19: Replace -x after -x (daddah + -naamm --> daddankaamm)

define XReplacement t -> x || x "^" _ ;

Rule 20: Replace -t changes to -c after -c (daddah + -naamm --> daddankaamm)

define CReplacement t -> c || c "^" _ ;

Rule 21: S palatalization: -s changes to -sh sh before -e (wollis + -e --> wollishshee)

define Spalatalization1 s -> sh sh || V _ "^" e ;

Rule 22: S palatalization: -s changes to -sh sh before -o (wollis + -o --> wollishshoo)

define Spalatalization2 s -> sh sh || V _ "^" [o | ó] ;

Rule 23: S palatalization: -s changes to -sh sh before -e (wollis + -e --> wollishshee)

define Spalatalization3 s -> sh sh || V _ "^" i ;

Rule 24: T palatalization: -t changes to -ch ch before -e (sut + -e --> suchchee)

define Tpalatalization1 t -> ch ch || V _ "^" e ;

Rule 25: T palatalization: -t changes to -ch ch before -o (sut + -o --> suchchoo)

define Tpalatalization2 t -> ch ch || V _ "^" [o | ó] ;

Rule 26: T palatalization: -t changes to -ch ch before -e (sut + -e --> suchchee)

define Tpalatalization3 t -> ch ch || V _ "^" i ;

Rule 27: D palatalization: -d changes to -jj before -e (dagud + -e --> dagujjee)

define Dpalatalization1 d -> j j || V _ "^" e ;

Rule 28: D palatalization: -d changes to -jj before -o (dagud + -o --> dagujjoo)

define Dpalatalization2 d -> j j || V _ "^" [o | ó] ;

Rule 29: D palatalization: -d changes to -jj before -o (dagud + -o --> dagujjoo)

define Dpalatalization3 d -> j j || V _ "^" i ;

Rule 30: X palatalization: -x changes to -cc before -e (biix + -e --> biiccee)

```
define Xpalatalization1 x -> c c || V _ "^" [e | e e ]  ;
```

Rule 31: X palatalization: -x changes to -cc before -o (biix + -o --> biiccoo)

```
define Xpalatalization2 x -> c c || V _ "^" [ o | ó ]  ;
```

Rule 32: X palatalization: -x changes to -cc before -e (biix + -e --> biiccee)

```
define Xpalatalization3 x -> c c || V _ "^" i y e  ;
```

Rule 33: Z palatalization: -z changes to -zh before -e (wez + -e --> wezhzhee)

```
define Zpalatalization1 z -> zh zh || V _ "^" e  ;
```

Rule 34: Z palatalization: -z changes to -zh before -o (wez + -o --> wezhzhoo)

```
define Zpalatalization2 z -> zh zh || V _ "^" [ o | ó ] ;
```

Rule 35: Z palatalization: -z changes to -zh before -o (wez + -o --> wezhzhoo)

```
define Zpalatalization3 z -> zh zh || V _ "^" i y e ;
```

Rule 36: is replacement -l changes to -shsh before -s (waal + -is --> waashsh)

```
define CAUS1replace1 l -> sh sh || V _ "^" s ;
```

Rule 37: ' replacement: -' changes to -aqq after double consonant ('iitt + -' --> iittaqq)

```
define MidDrive1 ' -> aqq || C C "^" _ ;
```

Rule 38: ' insertion ' added before single consonant before n (kul + -' --> ku'll)

```
define MidDrive2 [..] -> ' || V _ C "^" ',,
            aqq -> 0 || C _      ,,
            '   -> 0 || C _ ;
```

Rule 39: b and f replacement:-b changes to -phph and -f changes to -phph before ' (dib + -' --> di'phph) | (xuuf + -' --> xuu'phph)

```
define MidDrive3  b -> phph || V _ "^" ' ,,
                        f -> phph || V _ "^" ' ;
```

Rule 40: s and z replacement:-s changes to -cc and -z changes to -cc before ' (makkis + -' --> makki'cc) | (wez + -' --> we'cc)

```
define MidDrive4  s -> cc   || V _ "^" ' ,,
             z -> cc   || V _ "^" ' ;
```

```
# define Metathesis b -> n, n -> b || _ "^" n  ;
```

Cleanup: remove morpheme boundaries

```
define Cleanup "^" -> 0;
```

```
read lexc last.lexc
```

```
define Lexicon
define Grammar Lexicon    .o.
        Epenthesis1       .o.
        Epenthesis2       .o.
        Epenthesis3       .o.
        Epenthesis4       .o.
        MAssimilation     .o.
        FAssimilation     .o.
        GAssimilation     .o.
        NAssimilation     .o.
        KAssimilation     .o.
        SAssimilation     .o.
        DAssimilation     .o.
        EReplacement      .o.
        LReplacement      .o.
        BReplacement      .o.
        QReplacement      .o.
        ZReplacement      .o.
        KReplacement      .o.
        NReplacement      .o.
        XReplacement      .o.
        CReplacement      .o.
        Spalatalization1  .o.
        Spalatalization2  .o.
        Spalatalization3  .o.
        Tpalatalization1  .o.
        Tpalatalization2  .o.
        Tpalatalization3  .o.
        Dpalatalization1  .o.
        Dpalatalization2  .o.
```

Dpalatalization3 .o.

Xpalatalization1 .o.

Xpalatalization2 .o.

Xpalatalization3 .o.

Zpalatalization1 .o.

Zpalatalization2 .o.

Zpalatalization3 .o.

CAUS1replace1 .o.

MidDrive1 .o.

MidDrive2 .o.

MidDrive3 .o.

MidDrive4 .o.

Cleanup;

regex Grammar;