ADDIS ABABA UNIVERSITY
ADDIS ABABA INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

A Deterministic Approach to Tri-Radical Amharic Verb Derivatives Generation

By: Samrawit Kassaye

A Thesis Submitted to the School of Graduate Studies of Addis Ababa University in Partial Fulfillment of Masters of Science in Computer Engineering

Advisor: Yalemzewd Negash (Ph.D.)

February, 2022
Addis Ababa, Ethiopia

A Deterministic Approach to Tri-Radical Amharic Verb Derivatives Generation

By: Samrawit Kassaye

Approved by Board of Examiners

Advisor: Yalemzewd Negash (Ph.D.) Signature: _____

Examiner: Bisrat Derebssa (Ph.D.) Signature: _____

Examiner: Fitsum Assamnew (Ph.D.) Signature: _____

Dean, SECE: Bisrat Derebssa (Ph.D.) Signature: _____

Submitted in Partial Fulfillment of the requirements
for Masters of Science in Computer Engineering
Addis Ababa Institute of Technology
School of Electrical and Computer Engineering
February, 2022

Declaration

This thesis is a presentation of my original research work. Whenever contributions of others are involved, every effort is made to indicate this clearly, with proper citation of sources. I, the undersigned, declare that this thesis has not been presented for a degree in any other university.

Samrawit Kassaye

_____

Acknowledgments

First and foremost, praises and thanks to GOD, for his countless blessings throughout my life and this research.

I would like to express my deep and sincere gratitude to my research advisor, Dr. Yalemzewd Negash, for giving me the opportunity to do this research and providing me invaluable guidance throughout the research work. It was a great privilege and honor to work and study under his guidance with all the constructive ideas and comments which enabled me to gain good research experience.

I am extremely grateful to my parents for their prayers, love and sacrifices for educating and preparing me for my future. They have been my source of strength next to GOD.

Last but not least, I would like to express my heartfelt thanks to all my colleagues and friends for their immense patience, wisdom and professionalism through this journey.

Abstract

Morphological synthesis or generation is a process of returning one or more surface forms from a sequence of underlying (lexical) forms. Today, synthesizers of different kinds have been developed for languages that have relatively wider use internationally. Amharic is the second most populous Semitic languages after Arabic. But it is not exploited in the digital world.

In this research paper, a rule-based approach to morphologically derive or generate Amharic words from tri-radical verbs to finally generate rich Amharic lexicon is elaborated. This work utilizes two data sources namely Amharic word list and tri-radical verbs. The Amharic word list file contains more than 450,000 unique Amharic words. The tri-radical verb data source contains more than 350 unique tri-radical verbs. The proposed method works towards identifying rules from existing Amharic words after analyzing with tri-radical verbs. The new feature identified is applying index changing the letter of tri-radical verbs. Index changing (adding vowel to consonant letters) is one of the approaches used in morphological derivation of Amharic words from root(stem) words.

After index changing of the tri-radical verbs, the index changed words will be searched in the Amharic word list file. If the index changed words are found directly or part of word with prefix and/or suffix, the pattern of the word with respect to the root verb and index changed words will be captured. From the pattern captured morphemes are extracted and rules are identified. 85,115 unique rules are identified. While identifying rules, the frequency of every rule is recorded in order to evaluate the efficiency of each rule. A memory-based machine learning approach applied to evaluate the frequency of the rules.

From the 85,115 rules, the prefix of 29,776 rules and the suffix of 32,401 rules are wrong, and 11,390 rules are discarded by wrong index changing process.

The rules identified showed the accuracy of 0.99, average precision of 0.88 and average recall of 0.85. Based on these rules, a comprehensive set of derivatives for tri-radical Amharic verbs were generated and end up having a rich Amharic Lexicon.

Keywords: *Morphological synthesis, Natural Language processing, Memory-based approach, Tri-radical verbs, Amharic Lexicon, Rule-based approach, Morpheme*.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

ASMA	Amharic Stems Morphological Analyzer
CCC	Consonant- Consonant- Consonant
CLOG	Prolog based Inductive Logic Programming
CSV	Comma Separated Values
FN	False Negative
FP	False Positive
IBl	Instance-Based learning
IDE	Integrated Development Environment
IGtree	Information Gain tree
ILP	Inductive Logic Programming
MBL	Memory Based Learning
MDL	Minimum Description Length
NLP	Natural Language Processing
POS	Part of Speech
SVO	Subject, Verb and Object
TN	True Negative
TP	True Positive

Chapter One: Introduction

1.1 Background

Language is human being only and cultivated method of communicating thoughts, feelings, and needs by using symbols created by elites in a community [1].

Amharic is a Semitic language from the Afro-Asiatic language group. After the most commonly used Arabic language, Amharic is also the second most populous Semitic language. It is the working language of the Ethiopian Federal Government and some regional governments in Ethiopia [2]. According to the census done by the Central Statistics Agency of Ethiopia (it was named Population Census Commission) in 2007, Amharic was a mother-tongue language for more than 21 million people [3].

Natural Language Processing (NLP) is a sub-domain of Artificial Intelligence and Linguistics, focused to enable computers understand the language of human beings. Natural language processing evolved to simplify the task of computer users and to make the communication between the computer users and the computer more effective [4].

The aim of natural language processing is mainly to develop a representation of the text that makes the unstructured natural language into a structured one that can easily be handled by computers, without affecting the rules of linguistics. This structure can be shown in syntactic form like retrieving the grammatical relationships texts or in semantic like obtaining the meaning of the specific text. The main characteristics of natural language processing system contains of integrated components that process an input text in continuously sophisticated processes. In general, the task of each component is to provide structure to the text that can be used to smooth the task of the other processes. At the beginning, the components process tasks that are close to the surface strings of the text, while the components at the end focused on analyzing concepts and relationships. Multiple methods can be used to perform the task of the components; one is rule-based methods for instance regular expressions and finite state automata. And also, statistical and machine learning models are available [5].

Morphology is the structure or nature of words focusing on morphemes. Morphemes are parts of the word that are distinguished as the minimal units of morphology. An example of morpheme, the word “አክብሮት” can be morphologically analyzed into three separate morphemes: the prefix

“አ”, the root “ክብር”, and the suffix “አት”. Stem is part of the word that never changes even when morphologically inflected. For example, *ፈለግ* is the stem for the words *ፈለግኩ, ፈለግሽ, ፈለግን, ፈለግአችሁ*, etc. while root/lemma is citation form of a set of words. Amharic root form is usually a sequence of three consonants known as radicals. For example, ፍልግ is the root form for *ፈለገ, ፍለጋ, መፈለግ, ተፈለገ, ፈላጊ*, etc [4].

Morphological analysis is the process of finding morphemes of a word. It is an important element for spelling correction, machine translation, information retrieval, text generation and other natural language systems. Morphological generation is the process of generating different words from a morpheme.

Words can be formed from morphemes in two ways either in Derivational Morphology or Inflectional Morphology. Derivational Morphology is a morphology concerned with the way in which words are derived from morphemes through processes such as affixation or compounding. This derivation process usually changes the part-of-speech category. Inflectional Morphology is a morphology that deals with the combination of a word with a morpheme, usually resulting in a word of the same class as the original stem, and serving same syntactic function. Inflection can be achieved by marking a word category for person (first, second, third), gender (feminine, neuter, masculine), number (singular, plural), case (subjective/ nominative, objective/accusative/dative, possessive/genitive), definiteness (definite, indefinite), degree (positive, comparative, superlative), tense (past, present, future), aspect (perfective, imperfective/continuous), politeness (impolite, polite), etc [6].

The process of Amharic morphology analyzing is a very complex task. And primarily most of the words in Amharic language are derived from verbs. These verbs can be identified as root verbs having three radicals with pattern [7]. A pattern can possess a set of vowels that are injected between the consonants of a root to produce a stem. Amharic verbal stems can be generated from a ‘root + vowels + template’ combination. For example, the stem verb ፍልግ+ኧኧኧ+ CVCVC forms the root ፈለገ (‘want). For non-concatenated morphological features, to generate inflectional and derivational morpheme, Amharic uses different affixes. These affixes can be injected as prefix, infix, suffix and circumfix.

One of the sophisticated characteristics of Amharic language is a single word can be as meaningful and complete as a sentence comprising all the Subject, Verb and Object. For

example, the word "እፈልጋቸዋለሁ" meaning ("I need them") and possesses all SVO (S-"እኔ", V-"ፈለግ" O-"እነርሱ") in a single word. Amharic verbs are inflected for any combinations of person, gender, number, case, tense/aspect and mood. From these combinations, tens of thousands of verbs (in surface forms) can be generated from a single verbal root [8].

This focuses on deriving words from tri-radical verbs to finally obtain rich Amharic lexicon. Tri-radicals are verbal roots with three consonants in Semitic literature [10].

Number of researches made on identifying or generating Amharic lexicon from tri-radical verbs, but none of those researches provide perfect model or algorithm. That makes this research area open for more study. And this thesis focuses on designing mainly a hybrid morphological model. The model integrates the features of rule-based and memory-based learning morphological approach. This hybrid approach will add value in providing assistance to Amharic NLP, since Morphological synthesizers have significant impact in NLP systems in finding morphemes of a word using Morphological Analyzers. NLP processes such as Spelling Correction, Machine Translation, Information Retrieval, Text Generation and other natural language systems require an accurate morphological analyzer [9].

1.2 Statement of the Problem

Amharic is the second populous Semitic language next to Arabic but there doesn't exist a system or application that completely manages the language well in the digital world. It has a rich verb morphology which is based on triconsonantal roots with vowel variants describing modifications to, or supplementary detail and variants of the root form. A significantly large part of the vocabulary consists of verbs, which exhibit different morphosyntactic properties based on the arrangement of the consonant-vowel patterns [11]. Verbs are morphologically the most complex POS in Amharic, with many inflectional forms; numerous words with other POS are derived primarily from verbs [12].

The complexity of these verb morphological rules has been one of the most difficult tasks to deal with in Amharic NLP [13][14][15]. Birile [14] describes this problem as it relates to speech recognition.

"In Amharic, there are much more than one thousand morphological expansions for each verb. As far as speech recognition is concerned, each of these words are considered distinct and must

be considered separately, raising the complexity of Amharic speech recognition thousands of times more than that of English."

The use of morphological synthesizers for the generation of Amharic lexica was put forward both by Lisanu [13] and Birile [14]. Lisanu [13] explored this idea by using a rule based morphological synthesizer to generate verb derivations. He used a sample 8 root words and 42 derivations to test his idea, getting an overall accuracy of 81.48%. Birile [14], on the other hand used rule-based generation on five root words and 349 different derivations. Birile's thesis was about speech recognition and only used the morphological synthesizer as part of the bigger problem. He deliberately chose derived forms that provide 100% accuracy. Both authors recommended further work on better modeling by expanding the research.

This thesis is proposing to develop an Amharic lexicon by morphologically generating or deriving Amharic words from tri-radical verbs. Various researches are done in generating Amharic words by implementing different kinds of approaches. Researches made based on rule-based approach in the case of generating Amharic words from root/stem words are done but none of the researches can list or define all the rules of Amharic words derivation. It is understandable that with the complex nature of Amharic language, it appears unattainable to identify the derivation rules. Researches are also made based on statistical approaches that mainly apply mathematical techniques on large text corpora to develop generalized models of linguistic phenomena. The statistical approach bases on, the actual examples found on the text corpora without adding linguistic or world knowledge. The statistical approach cannot be an ideal approach for deriving Amharic words from root/stem words because of the unpredictable nature of Amharic language.

In this thesis, a hybrid approach will be applied by combining rule-based and memory-based approach for deriving large plus accurate words from tri-radical verbs. The main stand of this thesis is rules must be found or created in order to have a valid Amharic lexicon.

In this work, the main research area is finding the best methods to develop an Amharic lexicon by deriving Amharic words from tri-radical verbs using rule-based and memory-based approach. With the help of memory-based techniques, rules found from sample documents will be rated. The rating of rules found will be a new method to validate the correctness of the rules.

1.3 Objectives

1.3.1 General Objective

Develop an Amharic lexicon by deriving words from tri-radical verbs applying rule-based and memory-based learning approaches.

1.3.2 Specific Objectives

- Analyze common Amharic tri-radical verbs.
- Review related works in deriving Amharic verbs.
- Design hybrid model for developing large Amharic lexicon by deriving words from tri-radical verbs.
- Design proper algorithm to test the designed model.
- Develop a demo application to demonstrate the designed model.
- Test and evaluate the demo application and the lexicon data generated.

1.4 Research Methodology

In this thesis work, there are four phases to effectively obtain the desired result.

A. Analysis

- Literature Review: to identify the existing methods developed with their specific pros and cons.
    - Review of Amharic Natural Language Processing
    - Review of Amharic morphology
    - Review of existing Amharic morphological synthesizers and approaches applied.

B. Design

- Design model for deriving Amharic words from tri-radical verbs.
- Formulate an algorithm for the designed model.
- Define steps to entertain every activity involved in generating of derived Amharic words.

C. Implementation

- The designed algorithm will be developed into application. And it will be available for testing and demonstration.

D. Evaluation

- The accuracy and effectiveness of the proposed model and the algorithm designed will be evaluated by measuring the number of words generated and the approval rating of the rules obtained.

1.5 Scope and Limitation of the Study

Scope

- This study focuses on designing of a model for generating derived Amharic words from tri-radical verbs.
- It will use selected number of tri-radical verbs for testing and demonstration.

Limitation

- It will only consider tri-radical verbs.
- It will only focus on identifying derivation rules not grammatical rules.

1.6 Significance of the Study

After the implementation of this model;

- More Amharic lexicon will be available.
- The Amharic lexicon can be a good input for Amharic language related applications like word prediction, spell correction, grammar checking, information retrieval, information extraction, machine translation, question-answering, dialogue systems, text summarizations.
- Researches on Amharic morphological derivational rules will have another point of view.
- The availability of rich Amharic lexicon will be a great milestone for Amharic language in the digital world.

1.7 Organization of the Thesis

This thesis is organized in six chapters. In Chapter 2, basically relevant literatures will be reviewed regarding Amharic morphology; Amharic words derivation rules, Amharic morphology morphological synthesizers. In Chapter 3, the existing researches that are clearly related to this thesis will be covered. In Chapter 4, the design of the model generating/deriving Amharic words from tri-radical verbs will be presented and explained in detail. In Chapter 5, the implementation and evaluation of the design in Chapter 4 will be presented with evaluation results. At the end, in Chapter 6, conclusions points will be presented based on the evaluation results obtained with the summary of the thesis. Future works with regard to the research area of this thesis will be presented that can be researched or tackled in the future.

Chapter Two: Literature Review

2.1 Introduction

Amharic is the second most spoken Semitic language. Amharic can also be identified as Ethiopian Semitic (Ethio-Semitic), sometimes also categorized as African Semitic or branch of South Semitic. From the other Ethiopian Semitic languages, Amharic possesses major Semitic language features, namely inflectional and derivational morphology patterns. Amharic has SOV (Subject-Object-Verb) word order [15].

2.2 Concepts and Terminologies

2.2.1 Amharic Orthography

Amharic is written using a syllabic writing system different from other Semitic languages such as Arabic, Hebrew, and Syriac. The script was initially prepared for the other very older Ethiopian Semitic language Ge'ez and it is also used by other Ethiopian Semitic languages [16]. Amharic used the writing system of the Ge'ez (Ethiopic). Amharic has 27 consonant phonemes including the letter /v/ that is applied for words originated from foreign language. And also has 7 vowels [15].

2.2.2 Amharic Morphology

- Morphology is the study of the structure of words in a specific language.
- Morphemes are the minimal units of morphology, e.g., ሰውነት. Morphemes can be identified as Free and Bound Morphemes.
    - Free morphemes are morphemes that can stand on their own to give meaning. Example, ሰው from ሰውነት is free morpheme since it has meaning by itself
    - Bound morphemes are morphemes that cannot stand on their own as a word. Example, ት from ሰውነት is bound morpheme because it has no meaning by itself or cannot stand alone.

  Morphemes can also be recognized as Roots, Affixes or Combining Forms.

    - Root morphemes are morphemes (within a non-compound word) that make the most precise and concrete contribution to the word's meaning, and is either the sole morpheme or else the only one that is not an affix. Example, ፍለግ in ፈልግ, ፈለገ, ተፈለገች, etc.

- Affixes are bound morphemes that are either precede, follow or are inserted inside the root or stem.
    - Prefix: For example, *የ*-in *የውሰን* is an affix that precedes the root *ውሰን.*
    - Suffix: For example, -*ኦቸው* in *ፈለግኦቸው* is an affix that follows the stem *ፈለግ.*
    - Infix: For example, -*ላ*- in *ትላልቅ* is an affix that is inserted inside the root *ትልቅ.*
    - Circumfix: For example, *አስ…ናቸሁ* in *አስፈለግናቸሁ* is an affix that precedes and follows the stem *ፈለግ.*
  - By Combining Forms, morphemes can be formed from two bound or free-like roots. For example, ***ልብወለድ***; *ላብአደር*, *ሆደሰፊ*, *ሊቀመንበር*, etc.
- Stem is part of word that never changes even when morphologically inflected. For example, *መሰን* is the stem for the words *መሰንኩ*, *መሰንሽ*, *መሰንን*, *መሰንኦቸሁ*
- Root/Lemma is the citation form of a set of words. Amharic root form is usually a sequence of three consonants known as radicals. For example, *ፈለገ* is the root form for *ፈልግ*, *ፈለገ*, *ተፈለገ*, *አፈላለገ*, *ፈልጉ*, etc.
- Part-of-Speech/Lexical Category/Word Class is a linguistic category of words that explains how the word is used in a sentence. Although different languages may have different classification schemes, like English, Amharic words are usually classified into eight lexical categories: noun, pronoun, adjective, verb, adverb, preposition, conjunction and interjection. Morphologically important parts-of-speech in Amharic include: nouns, adjectives and verbs.
- Morphological Analysis is the process of finding morphemes of a word. It is an important component of Spelling Correction, Machine Translation, Information Retrieval, Text Generation and other natural language systems.
- Morphological Generation is the process of generating different words from a morpheme.
- Lemmatization is the process of finding the root/lemma of a word.
- Stemming is the process of finding the stems of a word.

Words can be generated from morphemes in two ways namely Derivational Morphology and Inflectional Morphology. The main difference between inflectional and derivational morphology is a functional difference: derivational (word-formation except compounding) is the kind of

morphology that serves to create new lexemes, while inflectional serve to create different forms of the same lexeme [20, 21].

2.2.3 Amharic Derivational Morphology

- Nouns: Amharic nouns can be derived from:
    - Verbal Roots by infixing vowels between consonants (C)

Table 2.1: Example of Noun Derivation from Verbal roots

Verbal Roots (Examples)	Pattern of Derivation	Derived Noun
ም-ል-ስ	CሾCC	ምሾልስ [መልስ]
ህ-ም-ም	CCሾC	ህእምሾም [ህመም]

    - Adjectives by suffixing bound morphemes
    - Stems by prefixing or suffixing bound morphemes
    - Stem-like Verbs by suffixing the bound morpheme -ታ-
    - Nouns by suffixing bound morphemes
    - Compound Words (sometimes by affixing the vowels ኧ and ኦ)

- Amharic adjectives can be derived from:
    - Verbal Roots by infixing vowels between consonants (C)
    - Nouns by suffixing bound morphemes
    - Stems by suffixing bound morphemes
    - Compound Words of nouns and adjectives by affixing the vowel -ኧ
- Amharic verbal stems (from which various forms of verbs are formed) can be derived from:
    - Verbal Roots by affixing the vowel -ኧ- to produce and also by repeating penultimate consonants and affixing the vowels -ኧ- and -ኣ- to produce
    - Verbal Stems by affixing morphemes

- Compound Words of stems and verbs and also compound words of sub-word and verbs.

2.2.4 Amharic Inflectional Morphology

Inflectional Morphology is a morphology that deals with the combination of a word with a morpheme, usually resulting in a word of the same class as the original stem, and serving same syntactic function. They do not change the part-of-speech category but the grammatical function.

Inflection can be achieved by marking a word category for person (first, second, third), gender (feminine, neuter, masculine), number (singular, plural), case (subjective/nominative, objective/accusative/dative, possessive/genitive), definiteness (definite, indefinite), degree (positive, comparative, superlative), tense (past, present, future), aspect (perfective, imperfective/continuous), politeness (impolite, polite), etc [20,21].

- Amharic nouns can be marked for:
  - Number by affixation of morphemes (and vowel changes) or repetition of words
  - Definiteness by affixation of morphemes or vowels based on number, gender, and/or ending of the noun.
  - Gender by affixation of the morpheme -ኢት
  - Objective case by affixation of the morpheme -ን (subjective case)
  - Possessive case by affixation of morphemes or vowels based on person, number, gender, and/or ending of the noun (personal pronouns by prefixing የ-.
- Amharic adjectives can be marked for:
  - Number by affixation of morphemes or repetition of consonants (and affixing the vowel -ኣ)
  - Definiteness by affixation of morphemes or vowels based on number, gender, and/or ending of the adjective.
  - Gender by affixation of the morpheme -ኢት
  - Case (Objective Case) by affixation of the morpheme -ን

- Amharic verbs are marked for:
    - Person, gender, number, case, and tense/aspect
    - Mood (Completed Action, Command, Request, Negative)

Chapter Three: Related Works

3.1 Design and Development of Automatic Morphological Synthesizer for Amharic Perfective Verb Forms

In this study, Kibur Lisanu [17] tried to develop automatic morphological synthesizer prototype for Amharic, focusing on perfective verb forms. The applied both rule-based and neural network approaches to develop a prototype, referred as Amharic Morphological Synthesizer.

It applied the rule-based approach to generate all the root words. And it also applied the neural network to forecast the type of roots from the test data set claimed an accuracy of 81.48%. In the test process of the research, the test result showed different accuracy for each type of perfective verb forms. For type A perfective verb forms - 80% of accuracy, for type B perfective verb forms - 25% accuracy and type C perfective verb forms - 100% accuracy.

The method used is manually synthesizing words for the development of the prototype and to perform the experiment. The result obtained using the small manually constructed root table will encourage the undertaking of further research in the area, especially with the aim of developing a full-fledged Amharic morphological synthesizer.

For the implementation of the prototype, there exists is database to capture templates, roots, suffixes and phonological variations (palatalization, consonant change and vowel change), which helps the synthesizer to generate different words from an input, are designed. For the neural network part, training data sets and running facts were also availed.

The study concluded that the development of the prototype is adequate with the complex nature of the language, limitation of large corpora in the language annotated with roots with their type. And also performing manually the identification of the root type and derivation of large size of data needs maximum labor, expense and time.

3.2 Learning Morphological Rules for Amharic Verbs Using Inductive Logic Programming

Wondwossen and Michael [18], in this study applied a supervised machine learning approach to perform morphological analysis of Amharic verbs. They used Inductive Logic Programming

(ILP) and CLOG as implementation tool. CLOG is a Prolog based Inductive Logic Programming system that enables to learn first order decision lists (rules) based in of positive examples only. In this, they prepared the training dataset manually to learn the morphological rules with regard to the structure of the background predicates available for the process of learning. To present different grammatical structures, the study limited to only subject prefixes and suffixes with high complex nature of inflectional and derivational verb morphology of Amharic language. While Amharic has more prefixes and suffixes for various morphological features, our system is limited to only subject markers. Combinations of subject and tense-aspect-mood were availed in the training dataset for the training. The study concluded that this approach is not practical if all the prefix and suffixes are going to be included in the learning process. And also, the study accepted the inadequacy of CLOG in the case of using variables as part of body of learning predicates.

From the test case with 1,784 Amharic verbs, 108 stem and 19 root templates were extracted. With the rules generated after combining the various rules generated, the program has been tested using a test set containing 1,784 Amharic verbs. An accuracy of 86.99% achieved from the test results. They tried to demonstrate that with small number of sample datasets, ILP is an adequate method for morphologically complex language like Amharic, to perform morphological rules learning.

In parallel, the study covered learning the root-template extraction and stem-internal alternation rule identification for Amharic.

3.3 Development of Amharic Morphological Analyzer Using Memory-Based Learning

Mesfin and Yaregal [8], used Memory-based learning approach as enabler for the Morphological analysis of Amharic language. Having in mind the complexity nature of Amharic morphology, they applied a supervised data-driven experimental approach that can use for the development of morphological analyzer of Amharic language. To track hidden or unseen classes from, they used a memory-based supervised machine learning method with the memory that records examples. The study showed a unique way in considering morphological analysis as a classification task that can capture properties of morphologically inflected words and the grammatical functions. The study applied thoroughly examining of the morphological structure of words and their

specific representation in the memory-based learning by analyzing words inflected based on vowel.

The performance evaluation of the model of the study is performed using 10-fold cross-validation with IBL algorithm (i.e., the simplest instance-based learning algorithm) with accuracy of 93.6% and IGtree algorithm (i.e., Information Gain tree, alternative approach where instance memory recognized) with accuracy of 82.3%.

The study concluded that the performance or accuracy of the model depends on the size of the training data; the more training data, the more accuracy.

3.4 Automatic Morphological Analyzer for Amharic: An Experiment Employing Unsupervised Learning and Auto-segmental Analysis Approaches

Tesfaye Bayu [19], tried to study and demonstrate the way of automatically analyzing the morphology of Amharic language by applying unsupervised learning and auto-segmental analysis approaches.

The study applied the modified version of Harris's Algorithm of Successor Frequency to identify the morphemes (the study named them word break points). The study also applied extra heuristics to advance the word breaks generated. The study took stem, suffixes and signature (i.e., the relation or association of stem and suffixes). To approve the validity of a signature to be part of the Amharic language morphology, the study used Minimum Description Length (MDL) test as a standard.

The study also applied an approach that is based on the principle of auto-segmental. This approach sound features of a word in different levels and used lines to maintain their associations. This approach finally is an enabler for extraction and representation of stem components.

Amharic Stems Morphological Analyzer (ASMA) is the name of the prototype developed to test the algorithms. The main processes of the study are tested by two systems namely Linguistica2001 and ASMA. Linguistica2001 provide identified stems and ASMA took the results from identified stems as input and perform further extraction and representation of stem components.

After implementing the above-mentioned systems and with the corpuses made by the researcher, the first system (Linguistica2001) showed an accuracy of 87% while the other one (ASMA) showed an accuracy 94%. It must be noted that both systems tested on number of words not more than 500.

3.5 Hybrid Inflectional Stemmer and Rule-based Derivational Stemmer for Gujarati

Kartik et al [23], in their paper introduced two approaches the first approach is a hybrid approach using a lightweight inflectional stemmer and the second approach is a rule-based approach using a heavyweight derivational stemmer. In the case of the lightweight stemming, in order to handle the stems and suffixes they applied unsupervised learning. To enhance the accuracy of the stems and suffixes, they had a module stemming based on POS (Part of Speech) and another module that manages the task changing letters. They reported that the modules they used helped in enhancing the accuracy of the inflectional stemmer ends up attaining 90.7% accuracy. The rule-based approach for derivational stemmer also achieved 70.7 accuracy. They revealed that orthographic rules, substitution and suffix-stripping play a significant role in their achievement of higher accuracy.

3.6 Chapter Summary

In the Sections 3.1, 3.2, 3.3 and 3.4, it is appeared that Amharic language is not yet completely processed in the digital world. These papers showed just fraction of the big share and most of them only tested their proposed ideas in a very small training data. Knowing the richness of Amharic language, we seen a gap in the previous researches, most of them never generated a large data that can be used utilized for further study. Another gap is also none of these researches come up a methodology to figure out the existing rules in deriving words from tri-radical verbs.

Chapter Four: Methodology

In this chapter, the technical design of the proposed research work is explained. This chapter presents how the proposed methodology works in two sections. The chapter begins by explaining the general overview of the proposed methodology. The second section describes and shows how each component of the proposed methodology function by explaining the functionality of each component and the interaction among the components.

4.1 Overview

As introduced in section 1.3, the proposed thesis work collects types of data sources for its general objectives. The first one is the list of Amharic tri-radical verbs that are root verbs with C-C-C. And the second data source is large collection of unique Amharic words. These two data sources are key parts of this thesis work since one is going to be the root verb which number of derived words will be generated from. The other one is going to be used as source of rules in deriving words and verifying the validity of the derived/generated words. In order to process these data sources, they are stored in the Data Repository.

The tri-radical verb data source possesses more than 350 uniquely identified Amharic tri-radical verbs and the reference data source contains more than 475,000 uniquely identified Amharic words that are found from crawling the Internet. And also, this reference data indicates the number of times the word is used online at the time of the crawling process is done. This thesis work does not have contribution in preparation of these data sources rather than using them.

After capturing and storing these data sources, the proposed thesis work searches for the tri-radical verbs from the Amharic words collection data source file. The searching process not-only searches the tri-radical verbs by themselves but also searches with possible changes in the vowel part of the tri-radical verbs. For example, let's take the tri-radical verb root felege [f-e-l-e-g-e] and also changing the vowels [f-e-l-e-g-u]. These two combinations can help us identifying more rules and generating more words. With 'f-e-l-e-g-e', we can find words like felege, felegech, asfelege ...etc. With 'f-e-l-e-g-u' we can find words like felegu, felegut, asfelegut, mefelegu …etc. After the completion of the searching process, the results found will be recorded in the data repository for future reference. The results obtained will help us to extract the morphemes. In other words, the result obtained shows their difference from the root verbs with prefix and

suffix they have before and/or after the root verb respectively. And also shows the difference in the vowel and consonant combination. By focusing on the differences between the root and the results found, we can identify the morphemes from the results found. Those differences can help us setup rules. Every rule identified will be checked whether the rule is new or not. If it is a new rule, it will be added to the rules list. If the rule identified is already existed in the rules list, its rating or frequency will be added by one. This shows the rule is used the number of times on different tri-radical verbs as seen in its rating/frequency.

The lists of rules with their specific rating/frequency are recorded under Rules Repository. From this level, it appears that there is list of rules found and recorded from the process of searching tri-radical verbs and changing of vowels of these words. These rules are listed from all tri-radical verbs. To generate valid words from the rules listed, all the rules will be applied to all tri-radical verbs. These processes enable us to generate a rich Amharic word list. The validity of the generated words can be examined with two parameters. The first parameter is checking whether the rule applied is more frequent or not. Since the nature of tri-radical verbs is quite similar, sharing commonly used rules is valid. The second parameter is checking whether the word generated based the rules listed is found in the Amharic word list (reference file) or not. The second parameter is not a precise way to evaluate the validity of the word generated but if found, it is will be an indicator that the rule is authentic. If the generated words fulfill both parameters, then it will be automatically valid. If the generated word fulfills only the first parameter, how frequent is the rule used will be the question. Since we identify all the rules based on the searching of the derivatives of tri-radical root verbs from the data source (reference file, i.e. Amharic Words List), it is impossible that a generated word can fulfill the second parameter but not the first parameter. To advance the validation process of the rules identified and words generated, the frequency of the words indicated in the data source (reference file, i.e., Amharic Words List for the words will be another option. The more frequently the word, the more the word is valid and the more the rule obtain from that specific word is valid.

In the case of identifying the validity of the rules extracted from the Amharic word list data source with in accordance to the tri-radical verbs, memory-based learning approach is applied.

The final goal of this thesis work is to generate rich Amharic lexicon. In order to achieve this goal, the above-mentioned tasks must be done. And to evaluate the validity of the words to be

added to the lexicon, this proposed thesis work applied multiple evaluation processes as discussed above. Since all things are done automatically after data sources are provided, with unpredictable nature of Amharic language, it is necessary to entertain small amount of human intervention to further evaluate the words generated in accordance to the rules identified whether they are acceptable or not.

4.2 Components of the Proposed Work

In order to achieve the objectives of this proposed work, there exist 10 main and supportive components.

1. Tri-radical verbs source
2. Reference Data Source
3. Source Data Repository
4. Word Finder
5. Morpheme Extractor
6. Rules Identifier/Generator
7. Rules Repository
8. Rules Rater
9. Word Generator
10. Lexicon Repository

In addition to the aforementioned components, the proposed work used various number of built-in and freely importable/reusable development tools used for the implementation of the execution of the program to demonstrate how the proposed thesis work is functioning.

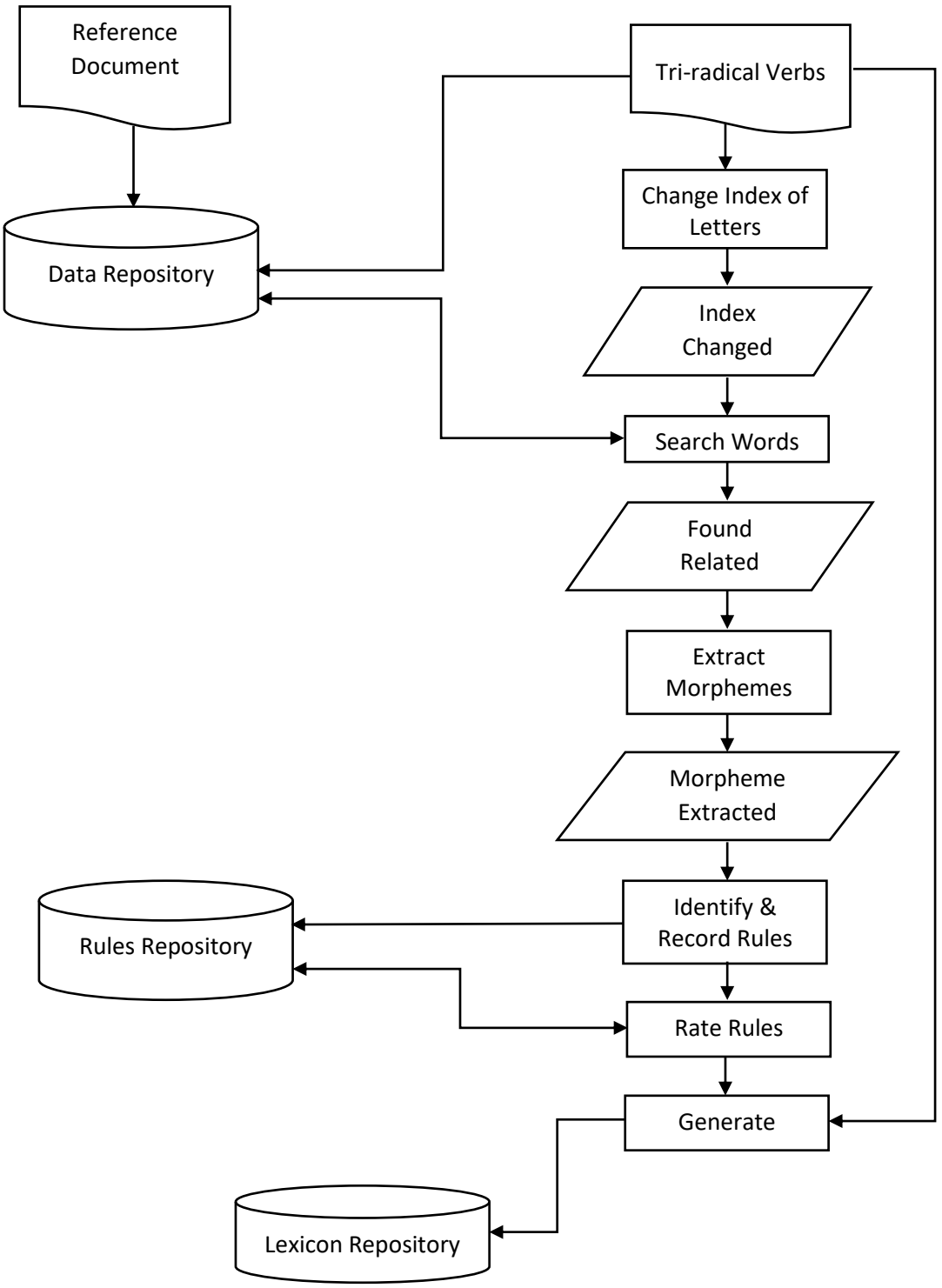

Figure 4.1 Architecture of Amharic Word Generation from Tri-Radical Verbs

4.2.1 Tri-radical Verbs Source

The proposed work performs the derivation of Amharic words from Amharic tri-radical verbs. In this thesis work, a file containing list of tri-radical verbs is availed. The tri-radical verbs inside the file are presented in root form (C-e-C-e-C-e) where 'C' denotes any consonant and 'e' denotes the specific vowel 'e'. For instance, wesede (W-e-S-e-De), felege (F-e-L-e-G-e), etc.

One Amharic consonant can be combined at least with seven vowels. Every combination will make a shape and phonological change on the letters.

Table 4.1 Amharic Scripts Consonant & Vowels Combination [22]

Amharic Consonants		**Amharic Vowels**											
		ä/e	u	I	A	ē	ə	O	ʷä/ue	ʷi/ui	ʷa/ua	ʷē/uē	ʷə
		[ə]					[ɨ], Ø		[ʷə]				[ʷɨ/ū]
		Consonants X Vowels											
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
h	/h/	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ					
l	/l/	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ			ሏ		
ḥ	/h/	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ			ሗ		
m	/m/	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ			ሟ		
ś	/s/	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ			ሧ		
r	/r/	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ			ሯ		
s	/s/	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ			ሷ		
š	/ʃ/	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ			ሿ		
q	/k'/	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቈ	ቍ	ቋ	ቌ	ቍ
b	/b/	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ			ቧ		
v	/β/	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ			ቯ		
t	/t/	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ			ቷ		
č	/tʃ/	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ			ቿ		
ẖ	/h/	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኈ	ኍ	ኋ	ኌ	ኍ
n	/n/	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ			ኗ		

ñ	/ɲ/	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ			ኟ		
’	/ʔ/	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ			ኧ		
k	/k/	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኰ	ኵ	ኳ	ኴ	ኸ
x	/h/	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ	ዀ	ዅ	ዃ	ዄ	ዅ
w	/w/	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ					
ʿ	/ʔ/	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ					
z	/z/	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ			ዟ		
ž	/ʒ/	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ			ዧ		
y	/j/	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ					
d	/d/	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ			ዷ		
ǧ	/dʒ/	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ			ጇ		
g	/g/	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጐ	ጕ	ጓ	ጔ	ጕ
ṭ	/t’/	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ			ጧ		
č̣	/tʃ’/	ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ			ጯ		
p̣	/p’/	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ			ጷ		
ṣ	/ts’/	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ			ጿ		
ṣ́	/ts’/	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ					
f	/f/	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ			ፏ		
p	/p/	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ			ፗ		

In the above table, we have assigned index number to the consonant X vowel combinations. When the letters of all tri-radical verbs are in index-0, then the verb is considered as tri-radical root verb. And we also figured out, when the letters of all tri-radical verbs are in index-5, then the verb is considered as tri-radical stem verb. Based on the indexing we assigned, it also makes the process of converting the root word into stem word and vice versa. And this converting process works precisely as long as the verbs in the conversion process are tri-radical verbs.

The indexing task performed has significant impact finding derived words.

Randomly selected sample tri-radical words: 'ሰለለ', 'ሰለመ', 'ሰለቀ', 'ሰለበ', 'ሰለቸ', 'ሰመረ', 'ሰመጠ', 'ሰረቀ', 'ሰረዘ', 'ሰረገ', 'ሰረጸ', 'ሰቀለ', 'ሰቀዘ', 'ሰበረ', 'ሰበቀ', 'ሰበከ', 'ሰነቀ', 'ሰነፈ', 'ሰከረ', 'ሰከነ', 'ሰወረ', 'ሰየመ', 'ሰደበ', 'ሰደደ',

'ሰገደ', 'ሰጠመ', 'ሰፈረ', 'ሰፈነ', 'ሰፈፈ'. And these words are tri-radical root verbs where all the letters in each word are with index-0.

4.2.2 Reference Data Source

In order to identify derivatives of the tri-radical verbs words and also identifying for the rules applied in parallel, the strategy we used is searching the tri-radical verbs and their related derivatives from an existing data source comprising more than 475,000 unique Amharic words. Since, we are looking for the derivatives of tri-radical words, we only use words with three or more number of letters.

One of the significances of the file we are using as reference document is the file not only have the list of Amharic words but also shows the frequency of each specific word and the string length of the word. The file we are using as reference document is a file generated after crawling Amharic words from the Internet.

Word	Frequency	Str_Length
ውስጥ	80072	3
ነበር	69074	3
ነገር	57927	3
አንድ	50442	3
ደግሞ	47468	3
ሰዎች	39746	3
እንደ	38077	3
ወይም	34721	3
በዚህ	33239	3
በኋላ	31865	3
ቅዱስ	31392	3
ናቸው	29819	3
ያለው	29030	3
እንዲህ	27678	4
ሲሆን	27515	3
እንጂ	27159	3
ሁኔታ	26643	3
በላይ	26241	3
መሆኑን	24496	4
መንገድ	24393	4
ማለት	24235	3

Figure 4.2 Screenshot showing how the Reference Document looked

4.2.3 Source Data Repository

As illustrated in section 4.2.1 and 4.2.2, to simplify the process of generating Amharic words from tri-radical verbs, this thesis work utilizes two files. The first file is the one containing list of tri-radical verbs and the other one is a file comprising more than 475,000 unique Amharic words. Both files are converted and are ready in CSV (Comma-Separated Values) file formats since CSV files are platform independent and can be easily read or extracted and ease the task of searching words. And these files are stored in the Source Data Repository for future reference.

4.2.4 Word Finder

In order to find the derivatives of a specific tri-radical verb from the reference file in the source data repository, we applied the following approaches:

- Without changing the index of the letters for the tri-radical root verb, then searching the words from the reference file with the criteria:
    - Words that contain the tri-radical word

Example: Tri-radical root verb: ፈለገ

የፈለገውን	ስለፈለገ	የፈለገበት	ለተፈለገው	የፈለጎች	በፈለገዉ
ከተፈለገ	ያልተፈለገ	በፈለገበት	ወደተፈለገው	ከተፈለገም	ስለፈለገው
በሚፈለገው	ያስፈለገበት	ለሚፈለገው	የፈለጎችው	የፈለገበትን	ከተፈለገና
የፈለገው	እየፈለገ	ለፈለገው	ያስፈለገበትን	የሚያስፈለገው	እንደፈለገና
የፈለገ	በፈለገ	የተፈለገበት	ባላስፈለገ	ስላስፈለገ	እስከፈለገ
የሚፈለገውን	ካስፈለገም	ካልፈለገ	የፈለገዉን	እንዳልፈለገ	የማይፈለገው
ከፈለገ	ወዳልተፈለገ	እየተፈለገ	ካልተፈለገ	ከፈለገም	እንደፈለጎችው
አስፈለገ	አስፈለገው	ወደሚፈለገው	ያስፈለገውም	እየፈለጎች	ካስፈለገን
ፈለገ	ፈለጎች	ያልፈለገው	ባስፈለገው	የፈለገውንና	አስፈለገኝ
የሚፈለገው	እንዳስፈለገ	ያልፈለገ	አላስፈለገኝም	ወደፈለገበት	እንደፈለገን
አልፈለገም	ላልተፈለገ	ከፈለጎች	እንዳስፈለገው	የሚፈለገዉን	ስላልፈለጎች
እንደፈለገ	ተፈለገ	የተፈለገ	ፈለገው	ባልተፈለገ	ያስፈለገኝ
በፈለገው	ከሚፈለገው	አልተፈለገም	ስለፈለጎች	የሚያስፈለገውን	ያልፈለጎች
ያስፈለገው	አልተፈለገ	ስለተፈለገ	የፈለገዉ	የተፈለገውም	እንደሚፈለገው
የተፈለገው	ስላልፈለገ	በተፈለገ	ተፈለገው	በፈለጎችው	ባላስፈለገም
የተፈለገውን	እንደተፈለገ	እንደፈለጎች	ወደፈለገው	በተፈለገበት	ከፈለገና
ካስፈለገ	አላስፈለገውም	ለፈለገ	ባስፈለገ	ፈለገና	ካልተፈለገለት
በተፈለገው	የፈለጎችውን	ያልፈለገውን	ካስፈለገው	ካልፈለጎች	እንደፈለገዉ
እንደፈለገው	አልፈለጎችም	እንደተፈለገው	ከፈለገው	ያስፈለገን	የሚፈለገውም

ኣስፈለገ	የፈለገንን	የምንፈለገው·	እስከፈለገኝ	ያስፈለገዉ	ወዳስፈለገበት
ኣላስፈለገም	እንዳስፈለገም	ባስፈለገን	ካላስፈለገ	እየፈለገው·	ስላልፈለገችና
ወደአልተፈለገ	ወደማይፈለገው·	ያላስፈለገው·	ከፈለገኝ	እስከፈለገው·	ለሚፈለገውና
ባላስፈለገን	በፈለገችበት	እየተፈለገላቸው·	ስላልፈለገው·	አስፈለገዉ	አልፈለገኩም
አስፈለገን	አልፈለገው·ም	የፈለገውንም	ያልተፈለገውን	ባልተፈለገም	ከሚያስፈለገው·
የምፈለገው·	የተፈለገዉ	ፈለገቻቸው·	የምፈለገዉን	ስለአስፈለገኝ	እንደማይፈለገው·
የፈለገም	ከአስፈለገ	የፈለገውና	ካልፈለገው·	ካፈለገ	ወደተፈለገበት
የፈለገን	ፈለገም	ፈለገችው·	ከፈለገ	ከተፈለገው·	ኣላስፈለገህ
ኣላስፈለገንም	ባላስፈለገው·ም	በሚፈለገዉ	ካልፈለገችህ	እስከተፈለገው·	እሚፈለገው·
ባላስፈለገው·	አልፈለገምና	ከፈለገችው·	ባልፈለገው·	የፈለገወም	ያልፈለገዉ
ስለፈለገና	ያልፈለገችው·	ስለፈለገችው·	የተፈለገውንም	ስላስፈለገኝ	ያስፈለገባቸው·
ካስፈለገህ	ፈለገን	አየፈለገ	የፈለገችው·ም	ስላልፈለገሽ	እየፈለገና
በፈለገውና	የሚፈለገውንና	ባፈለገው·	ካስፈለገው·ም	ፈለገው·ም	ያስፈለገህ
እያስፈለገው·	እንደፈለገህ	እተፈለገ	እንዳልፈለገው·	ኣያስፈለገው·ም	እንዳስፈለገዉና
ሚፈለገው·	ያልፈለገበት	ከሚፈለገውና	ስላስፈለገበት	ያልተፈለገለት	እንፈለገዋለን
እንደፈለገኝ	ካሰፈለገ	ካልፈለገችው·	የሚፈለገ	ካልፈለገውና	አትፈለገም
ስላስፈለገው·	የፈለገው·ም	ባስፈለገበት	አልተፈለገና	እስካልፈለገ	የማይፈለገን
የማይፈለገውን	እንደፈለገም	አሰፈለገ	እንደፈለገውና	ያላስፈለገውን	በፈለገችዉ
እንዳስፈለገና	ከፈለገን	ባልተፈለገበት	አይፈለገም	ያስፈለገውን	በፈለገሰላም
የተፈለገች	ያልተፈለገው·	ካስፈለገኝ	ያልፈለገበትን	የፈለገውስ	ስልፈለገ
እንዳስፈለገህ	የፈለገችበት	ፈለገሰላምን	ካልፈለገም	ያስፈለገውስ	ከፈለገዉ
ኣልፈለገም	እያስፈለገ	ፈለገሰላም	አስፈለገዎት	ያስፈለገባቸው·ም	እየፈለገኝ
ያስፈለገበትም	እንዳልተፈለገ	ካስፈለገማ	ስላልተፈለገ	ሲፈለገብኝ	

- By changing the index of the letters of the tri-radical root (combining the consonant letters of the tri-radical words with possible vowels). Let's name these words Index Changed Words. Then searching the words from the reference file with the criteria:
    - Words that contain the Index Changed Words

    Example: With the Index Changed Word: ፈለገ → ፈለግ

መፈለግ	ሲፈለግ	እየፈለግን	ካለመፈለግ	የሚፈለግብንን	እየፈለግህ
ፈለግ	መፈለግና	ስለፈለግሁ	በፈለግኸው	የፈለግከው	የፈለግንበት
ለመፈለግ	የፈለግነው	ፈለግኩ	የፈለግኸውን	በመፈለግና	የፈለግሽውን
ከፈለግን	አለመፈለግ	ይፈለግ	የሚፈለግባቸውን	እንዲፈለግ	ስትፈለግ
በመፈለግ	አልፈለግሁም	እንደፈለግን	በሚፈለግበት	መፈለግህ	እንደመፈለግ
ከፈለግህ	በፈለግነው	እንደሚፈለግ	እንደፈለግሁ	ስላልፈለግሁ	ከፈለግኸው
ከመፈለግ	አልፈለግኩም	ከፈለግሁ	እንደፈለግህ	ቢፈለግም	እየፈለግሁ
ፈለግሁ	የማይፈለግ	ለመፈለግና	እንደፈለግነው	የመፈለግና	እንደፈለግኩ
የመፈለግ	ያለመፈለግ	ከፈለግሽ	አልፈለግንም	እንዲፈለግለት	ሊፈለግ
የፈለግሁት	ቢፈለግ	የሚፈለግበት	ፈለግህ	ሊፈለግለት	እስከፈለግን
የሚፈለግ	የፈለግኩት	የፈለግኩትን	ሳይፈለግ	ይፈለግበታል	ስላልፈለግን
የፈለግነውን	ፈለግን	የፈለግሁትን	መፈለግን	የፈለግኸው	እንደማይፈለግ
አይፈለግም	ባለመፈለግ	የፈለግከውን	ፈለግና	ካልፈለግን	ከፈለግኩ
ከፈለግከ	የሚፈለግበትን	ስለፈለግን	ስለሚፈለግ	ስለፈለግኩ	አያስፈለግም

እንደሚፈለግና	ፈለግነው	ከፈለግንና	እንዲፈለግም	እንዲፈለግላቸው	ስላልፈለግነው
እስከመፈለግ	መፈለግም	ስለፈለግነው	በመፈለግም	ካልፈለግክ	የፈለግክበትን
ለፈለግሁት	አልፈለግክም	ስለፈለግህ	የሚያስፈልግበት	መፈለግዎን	አልፈለግኩትም
ይፈለግለታል	እንደማልፈለግ	የፈለግን	የማትፈለግ	ባለመፈለግም	የፈለግኩበትን
የማይፈለግበት	የፈለግነውም	ያለመፈለግን	ስለፈለግሁት	ባለመፈለግህ	በፈለግሽበት
ማይፈለግ	ፈለግሽ	መፈለግህን	ከፈለግህም	የሚፈለግን	እየፈለግንም
ፈለግሁና	ፈለግኩኝ	እንደምፈለግ	መፈለግስ	የፈለግናቸውን	ያልፈለግሁበትን
ስለማይፈለግ	የፈለግነውንና	ካለመፈለግና	አንፈለግም	የሚፈለግና	የፈለግክበት
አልፈለግም	እንደፈለግክ	እንደሚያስፈለግ	የመፈለግን	እንደፈለግናት	ከፈለግከው
በፈለግከው	በፈለግኩት	ለመፈለግም	ከመፈለግና	ወደፈለግህበት	እየፈለግከው
ፈለግክ	ያልፈለግነው	አልፈለግህም	አለማስፈለግ	አልፈለግነዉም	በፈለግሁት
ወደፈለግንበት	ወደሚፈለግበት	ለማይፈለግ	ካሰፈለግ	የፈለግኽውን	የተፈለግክበት
በማይፈለግበት	ሊፈለግላቸው	ከፈለግክም	በፈለግሽው	የፈለግሁ	ስለፈለግኩም
ይፈለግብናል	የምትፈለግ	ስለመፈለግ	ፈለግሁኝ	እንደፈለግሁት	የፈለግሁባቸው
ከፈለግ	ላለመፈለግ	የፈለግህበትን	ወደፈለግኩት	አለመፈለግን	ይፈለግባቸው
የሚፈለግባቸው	የፈለግሁበት	እንደሚፈለግም	የፈለግኩበት	አትፈለግም	ስለፈለግኩበት
የፈለግሽው	እንደፈለግከው	ይፈለግበት	የፈለግኹትን	ለፈለግነው	የማልፈለግና
ስላልፈለግኩ	ካልፈለግህ	ወደፈለግክበት	የፈለግኹት	አልፈለግነውምና	አለመፈለግም
ወደፈለግነው	እንደሚፈለግባቸው	ወዳልተፈለግ	የሚፈለግብን	በመፈለግህ	እንደተፈለግን
በፈለግን	በሚፈለግባቸው	መፈለግንም	እንዳልፈለግሁ	ከፈለግንስ	በፈለግኩበት
ይፈለግለት	ወደማይፈለግ	የፈለግነዉ	ከፈለግንም	እየፈለግኩ	በሚፈለግበትም
እንደፈለግሽ	ከፈለግነው	የሚፈለግብህ	ባልፈለግነውና	ፈለግሁት	

Example: With the Index Changed Word: ፈለገ → ፈልግ

የሚፈልግ	ለሚፈልግ	አያስፈልግህም	እንድንፈልግ	እንደሚፈልግም	ልንፈልግ
አልፈልግም	ስለሚፈልግ	ሊያስፈልግ	ብንፈልግ	ሳንፈልግ	በሚያስፈልግህ
አያስፈልግም	አትፈልግም	የሚያስፈልግህን	ብትፈልግም	ባይፈልግም	ስንፈልግህ
እንደሚያስፈልግ	ስፈልግ	ሳይፈልግ	ስለምትፈልግ	ከሚፈልግ	ያስፈልግዎታል
አይፈልግም	ፈልግ	እንደማያስፈልግ	ወደሚያስፈልግበት	ቢያስፈልግም	ባትፈልግም
እንደሚፈልግ	እንደምንፈልግ	የሚፈልግበት	እንደማልፈልግ	ባይፈልግ	አልፈልግሽም
አንፈልግም	እንፈልግ	ቢያስፈልግ	ሳልፈልግ	እንደሚፈልግና	ኣያስፈልግም
የሚያስፈልግ	እንደምፈልግ	እንደማትፈልግ	ባልፈልግም	ልፈልግ	የምፈልግ
ይፈልግ	ብትፈልግ	እንደሚያስፈልግህ	ብንፈልግም	ባንፈልግም	የማይፈልግበት
ሲፈልግ	እንደሚያስፈልግም	በሚያስፈልግበት	ሳትፈልግ	በሚፈልግ	በሚያስፈልግባቸው
እንደምትፈልግ	ስለሚያስፈልግ	እንዲፈልግ	ስለማንፈልግ	አይፈልግ	እንደምትፈልግና
የማይፈልግ	የሚያስፈልግበት	ስለማልፈልግ	በምትፈልግበት	የሚፈልግን	የምፈልግበት
ያስፈልግሃል	ያስፈልግህ	እንደማንፈልግ	አንፈልግህም	አትፈልግ	ስለማትፈልግ
ያስፈልግ	ስለምፈልግ	በሚፈልግበት	ያስፈልግሻል	ሲያስፈልግም	ሲፈልግም
ቢፈልግ	የማያስፈልግ	የሚያስፈልግህ	የማትፈልግ	በምንፈልግበት	አይስፈልግም
የምትፈልግ	ስለምንፈልግ	የምንፈልግበት	እንደምትፈልግም	አያስፈልግሽም	እንፈልግሃለን
የምንፈልግ	ቢፈልግም	ሊያስፈልግህ	የማንፈልግ	ይፈልግብናል	ሊፈልግና
ትፈልግ	ስንፈልግ	እንደሚያስፈልግና	የሚፈልግም	የሚፈልግና	በሚያስፈልግ
እንደማይፈልግ	ሲያስፈልግ	ብፈልግም	ባያስፈልግም	ሲፈልግና	እንዲፈልግላቸው
እፈልግ	ስትፈልግ	ሊፈልግ	የምትፈልግበት	ላያስፈልግ	አይፈልግምና
ሳያስፈልግ	ስለማይፈልግ	ብፈልግ	አልፈልግህም	አታስፈልግም	እንድትፈልግ

ልንፈልግለት	የሚፈልግብን	የሚያስፈልግና	ማያስፈልግ	ሊፈልግበት	አጥፈልግም
ላይፈልግም	የሚፈልግብህን	የሚፈልግበትን	ለማስፈልግ	ሊፈልግባት	ስለሚፈልግና
በምፈልግበት	እንደሚፈልግባቸው	አልፈልግ	ወደማያስፈልግ	የሚፈልግለት	ስለሚአስፈልግ
ፈልግና	እስኪፈልግ	የሚያሰፈልግ	የማያስፈልግም	ስለምንፈልግና	እንዳትፈልግ
የማይፈልግና	ሳያስፈልግህ	በማይፈልግ	ከማያስፈልግበት	እፈልግና	የማያስፈልግህን
ላያስፈልግ	እንደማይፈልግና	ላያስፈልግምም	ላይፈልግምም	እንደምንፈልግና	ላያስፈልግህ
ላያስፈልግምና	የሚፈልግባቸው	እንዳንፈልግ	ለመፈልግ	ላያስፈልግዎትም	በማያስፈልግህ
የሚፈልግብንን	ሊፈልግለት	እንደሚያስፈልግ	እንደምፈልግና	ላያሰፈልግም	የምንፈልግባቸው
የሚፈልግባቸውን	አልፈልግማ	እንደሚፈልግብን	ስፈልግህ	ከማያስፈልግ	የምትፈልግበትን
እንደማያስፈልግህ	ፈልግም	እንደሚፈልግብኝ	ምንፈልግበት	የማያስፈልግና	በማንፈልግበት
እንድፈልግ	የሚያስፈልግበትን	እንደሚፈልግበት	የሚያስፈልግውን	እንደሚፈልግብህ	የምናስፈልግበት
አልፈልግምና	ከምንፈልግበት	ወደሚያስፈልግባቸው	የምትፈልግና	ስለሚፈልግባቸው	ፈልግንም
አንደሚያስፈልግ	ስለማያስፈልግ	እሚፈልግ	የሚያስፈልግን	በሚያስፈልግን	የምፈልግና
ይፈልግሃል	እንደማትፈልግና	እንፈልግና	በመፈልግ	እንዲሚያስፈልግ	በሚያፈልግበት
ወደሚፈልግበት	ይፈልግልን	እንደሚያስፈልግሽ	እንዳይፈልግ	ስለምፈልግና	ከምትፈልግበት
የማልፈልግ	እፈልግሃለሁ	ሚፈልግ	ስለማይፈልግና	የምትፈልግለት	ባያስፈልግህም
በማይፈልግበት	ይፈልግሻል	የሚያስፈልግባቸው	አትፈልግህም	ባልፈልግ	እንደምንፈልግም
እንድንፈልግና	እንደማትፈልግህ	ሳንፈልግም	የማትፈልግህ	ሊፈልግባቸው	የምፈልግዉ
እንፈልግለት	እንደማያስፈልግና	የማትፈልግበትን	ላልፈልግም	ቢፈልግና	ልፈልግልሽ
ያስፈልግል	ለምንፈልግ	እንፈልግሻለን	የአልፈልግሽም	ባይፈልግስ	ቢፈልግለት
በማያስፈልግ	የአልፈልግህም	የሚያስፈልግሽ	የሚፈልግሽ	እንደማንፈልግና	ወደምንፈልግበት
የምትፈልግባቸውን	ፈልግልኝ	ያስፈልግና	ሚያስፈልግ	እንዲያስፈልግ	በሚፈልግብኝ
ስለሚያስፈልግህ	የማልፈልግበት	የማትፈልግበት	እፈልግሻለሁ	እንፈልግዎታለን	
ይፈልግና	የሚያስፈልግም	ሲያስፈልግህ	ላያስፈልግህ	ብትፈልግስ	
አንፈልግምና	ባያስፈልግ	በሚፈልግባቸው	እንደሚስፈልግ	ያስፈልግሽ	
ቢያስፈልግህ	እንዲፈልግለት	ቢፈልግማ	ለማያስፈልግ	ይፈልግለትና	
ልትፈልግ	ለማይፈልግ	አይፈልግማ	ፈልግልን	ይፈልግባቸዋልን	

These are some of the Index Changed Words from the tri-radical verb 'ፈለገ'. We will search the possible derived words from the Reference file based on these Index Changed Words

ፈለጉ	ፈላጊ	ፈልጋ	ፋላጎ	ፍላገ	ፍሌጎ
ፈለጊ	ፈላግ	ፈልጌ	ፋልጉ	ፍላጉ	ፍልገ
ፈለጋ	ፈላጎ	ፈልግ	ፋልጋ	ፍላጊ	ፍልጉ
ፈለጌ	ፈላጐ	ፈልጎ	ፍለጊ	ፍላጋ	ፍልጋ
ፈለግ	ፈሌጉ	ፈልጐ	ፍለጋ	ፍላግ	ፍልጎ
ፈለጎ	ፈልገ	ፈልጓ	ፍለግ	ፍላጎ	
ፈለጓ	ፈልጉ	ፋለጋ	ፍለጎ	ፍላጐ	
ፈላገ	ፈልጊ	ፋላጊ	ፍሊጎ	ፍላጓ	

It is vital to understand that there may exist invalid words from the words found based on the tri-radical root verb and Index Changed Words.

Algorithm 4.1: Algorithm to Generate Indexed Changed Words

```
Input: Tri-Radical Root Verb, Fidels[]
       // Tri-Radical Verb: one Tri-radical root verb
       // Fidels[]: Indexed Array representing Every Amharic letter
Task: Generate Indexed Changed Words
Start
    Declare string variable rootVerb
    rootVerb ← Tri-radical verbs
    Declare integer variable rowLetter1 with initial value 0
    Declare integer variable rowLetter2 with initial value 0
    Declare integer variable rowLetter3 with initial value 0
    for each lettersRow in Fidels do
       for each letter in letterRow do
             if first letter of rootVerb is same as letter then
                   rowLetter1 ← Row Number of the Letter in Fidels
             if second letter of rootVerb is same as letter then
                   rowLetter2 ← Row Number of the Letter in Fidels
             if third letter of rootVerb is same as letter then
                   rowLetter3 ← Row Number of the Letter in Fidels
       end for
    end for
    Declare integer variable i with initial value 0
    Declare integer variable j with initial value 0
    Declare integer variable k with initial value 0
    Declare string variable indexedWord
    Declare string array indexedWords

    for each i in the list 0,3,5 do
       for j←0 to 12 do
             for k←0 to 12 do
                   if Not fidels[rowLetter1][i] is empty or
                         fidels[rowLetter2][j] is empty or
                         fidels[rowLetter3][k] is empty then
```

```
                        indexedWord← fidels[rowLetter1][i] +
fidels[rowLetter2][j] + fidels[rowLetter3][k]
                        indexedWords[] ← Append [indexedWord,i,j,k]
            end for
      end for
   end for
Return indexWords[]
End
```

Algorithm 4.2: Algorithm to Word Finder

```
Input: indexedWords[], rootVerb, Reference File
      // indexedWords: words generated from Tri-radical root verb by changing
index of each letter
      // rootVerb: tri-radical root verb
Task: Find Related/Derivative Words
Start
   Declare string array foundWords[][]
   for each indexedWord in indexedWords do
      Open Reference File
      for each row in Reference File do
            if word in row is contains the indexed word then
                  foundWords[][] ← Append word in row, indexedWord
                  Perform Morpheme Extraction (Algorithm 4.3)
      end for
      close Reference File
   end for
return foundWords[]
End
```

4.2.5 Morphemes Extractor

In order to identify the rules applied in derivation of words from the tri-radical root verbs, it is vital to understand the structure change made from the tri-radical root verbs.

For example,

If the tri-radical root verb is ‘ፈለገ’ and the found word is ‘አስፈለጋቸው’. From this we can understand that

- ‘ፈለገ’ is index changed to ‘ፈለጋ’ in other words from index 0-0-0 to index 0-0-3.
- ‘አስ’ is prefix that is added to the left of the root/index changed word.
- ‘ቸው’ is suffix that is added to the right of the root/index changed word.

Therefore, the morphemes are ‘አስ’, ‘ፈለጋ’, ‘ቸው’. Identifying the morphemes and their difference with the tri-radical root verb enables the Rules Identifier and Rules Generator perform their specific task.

Algorithm 4.3: Algorithm for Morphemes Extractor

```
Input: foundWords[], indexedWords[], rootVerb
       // foundWords: words found from Reference File containing the tri-radical root verb
       // rootVerb: tri-radical root verb
       // indexedWords: words generated by changing the index of each letter of the tri-radical root verb
Task: Extract Morphemes
Start
    Declare string array morphemedWords[][]
    Declare string variable prefix
    Declare string variable suffix
    Declare string variable rootSub
    Declare integer variable indexLetter1
    Declare integer variable indexLetter2
    Declare integer variable indexLetter3

    for each word in foundWords[][] do
       prefix ← letters before the indexedWord in foundWord
       suffix ← letters after the indexedWord in foundWord
       rootSub ← indexWord
```

```
        indexLetter1 ← index of Letter1 in indexedWord
        indexLetter2 ← index of Letter2 in indexedWord
        indexLetter3 ← index of Letter3 in indexedWord
        morphemedWords[][] ← [rootVerb, rootSub, prefix, suffix, indexLetter1,
indexLetter2, indexLetter3]
    end for
return morphemedWords[][]
End
```

4.2.6 Rules Identifier/Generator

After the completion of morpheme extraction, identifying the rule from the extracted morphemes is mandatory. The morpheme extraction process generates morphemes possessing tri-radical root verb, the index changed word (if changed), prefix (if any), suffix (if any), index of changed word letter 1, index of changed word letter 2, index of changed word letter 3. The rules identifier, then captures the output of morpheme extractor, and creates a rule pattern and records the rule pattern in the rules repository if the rules is not found in the repository.

Example:

Tri-radical Root Verb: ፈለገ Index 0-0-0

Index Changed Word: ፈልግ Index 0-5-5

Related/Derivative Word found from the Reference File: የምትፈልግባቸውን

After Morpheme Extraction:

Prefix: የምት-

Suffix: -ባቸውን

Rule Pattern: የምት- 0-5-5-ግባቸውን

After rule pattern is settled, it will be checked whether it existed in the rule's repository. If the rule is not found in the rule's repository, it will be recorded to the rule's repository with frequency value of 1. If the rule already existed in the rule's repository, it will add the frequency value of the rule in the rule's repository by 1.

Algorithm 4.4: Algorithm for Rules Identifier

```
Input: morphemedWords[][]
       // morphemedWords: is output of Morpheme Extractor Process and possesses rootVerb, rootSub, prefix, suffix, indexLetter1, indexLetter2, indexLetter3
Task: Rule Pattern Identifier
Start
    Declare string variable ruleFound
    for each morphemedWord in morphemedWords do
       ruleFound ← append [prefix, indexLetter1, indexLetter2, indexLetter3, suffix ]
       Open Rules Repository File
       new <- 1
       for each rule in rules_list do
             if rule is same as ruleFound then
                   rule's frequency ← rule's frequency + 1
                   new ← 0
                   end for
       end for
       if new is equals to 1 then,
             rules_list ← append ruleFound
             save
       close file
    end for
End
```

4.2.7 Rules Repository

In this thesis work, we have a repository named Rules Repository. This repository possesses a tabular data representing the rule pattern of Amharic tri-radical verb derivatives. The tabular data combines the following data to setup, rate, and verify the rule pattern.

- Prefix: extracted letters found before the rootVerb/indexChangedWord within the found possible derivative.
- Index of Letter 1: Index of the letter 1 according to the rootVerb.
- Index of Letter 2: Index of the letter 2 according to the rootVerb.

- Index of Letter 3: Index of the letter 3 according to the rootVerb.
- Suffix: extracted letters found after the rootVerb/indexChangedWord within the found possible derivative.
- Frequency: The number of times a specific rule is used by the words found in the reference file.
- Status: After general rules' rating is done, the status can be Y/N (Yes/No) based on the rating it obtains.

Table 4.2: Examples of Rules Repository

prefix	indexLetter1	indexLetter2	indexLetter3	suffix	Frequency	status
እንደተ	0	0	0		55	Y
እየ	0	0	3	ችሁ	16	Y
እንዲ	0	5	1	ት	25	Y
ሊ	0	5	5	በት	14	Y
እንዲ	0	5	5	ብን	11	Y

4.2.8 Rules Rater

To generate a lexicon possessing accurate words, the rules applied to derive words from the tri-radical verbs must be valid. There comes a need to validate or prove the accuracy of the rules. In this thesis work, to verify the accuracy of the rules used to generate word derivatives from tri-radical verbs, rating based on the frequency of the rules used from the words found in the resource file is applied. The more frequently used rules can automatically be approved to be applied for derivation of words from possible tri-radical root verbs. Since the nature of all tri-radical verbs is similar, a rule applied for number of tri-radical verbs can be applied to all tri-radical verbs.

It is difficult to automatically decide whether the rules rated low are incorrect or not. As long as the Amharic words in the reference source file are error free, the rules generated after finding words related to the tri-radical root verbs will have none or minimum number of errors while generating derivatives.

Algorithm 4.5: Algorithm for Rules Rater

```
Input: rules_list[][]
       // rules_list: is list of rules to generate Amharic words from tri-
radical verbs
Task: Rating Rule based on the number of types the rule is applied
Start
    Declare string array prefixes[][]
    Declare integer array index1[][]
    Declare integer array index2[][]
    Declare integer array index3[][]
    Declare string array suffixes[][]
    Declare integer variable flag
    for each rule in rules_list do:
       // frequency of prefix
       flag ←0
       for each prefix in prefixes do:
             if rules[0] is same as prefix then:
                   prefix[frequency] ← prefix[frequency] + 1
                   flag ← 1
                   end for
       end for
       if flag is equals to 0 then:
             prefixes ← Append rule[0],1

    // frequency of index of letter 1
       flag ←0
       for each index in index1 do:
             if rules[1] is same as index then:
                   index1[frequency] ← index1[frequency] + 1
                   flag ← 1
                   end for
       end for
       if flag is equals to 0 then:
             index1 ← Append rule[1],1
```

```
    // frequency of index of letter 2
    flag ←0
    for each index in index2 do:
          if rules[2] is same as index then:
                index2[frequency] ← index2[frequency] + 1
                flag ← 1
                end for
    end for
    if flag is equals to 0 then:
          index2 ← Append rule[2],1
    // frequency of index of letter 3
    flag ←0
    for each index in index3 do:
          if rules[3] is same as index then:
                index3[frequency] ← index3[frequency] + 1
                flag ← 1
                end for
    end for
    if flag is equals to 0 then:
          index3 ← Append rule[3],1

    // frequency of index of suffix
    flag ←0
    for each suffix in suffixes do:
          if rules[4] is same as suffix then:
                suffix[frequency] ← suffix[frequency] + 1
                flag ← 1
                end for
    end for
    if flag is equals to 0 then:
          suffixes ← Append rule[4],1

for each rule in rules_list do:
```

```
      if     frequency(rule[0])>10     and     frequency(rule[1])>10     and
frequency(rule[2])>10  and  frequency(rule[3])>10  and  frequency(rule[4])>10
then:
            rule[5] ← "Y"
      else
            rule[5] ← "N"
   end for
End
```

4.2.9 Word Generator

In order to generate words based on the rules captured from the reference file that are proper derivatives of the tri-radical verbs, the rules that are highly rated or valid will be applied to all valid tri-radical root verbs. A specific rule contains 5 attributes that can make inflectional and derivational changes to the tri-radical root verbs.

- Prefix: if and only if the rule is valid and rated highly, prefix will be added or concatenated to the left of the tri-radical verb. The value in the prefix attribute can be null or verified prefix morpheme.
- Index of Letter 1: tri-radical verbs only have 3 letters that are consonants. If and only if the rule is valid and rated highly, the index specified in the Index of Letter 1 attribute will be applied to the first letter of the tri-radical verb. The index may or may not have a change on the letter.
- Index of Letter 2: tri-radical verbs only have 3 letters that are consonants. If and only if the rule is valid and rated highly, the index specified in the Index of Letter 2 attribute will be applied to the second letter of the tri-radical verb. The index may or may not have a change on the letter.
- Index of Letter 3: tri-radical verbs only have 3 letters that are consonants. If and only if the rule is valid and rated highly, the index specified in the Index of Letter 3 attribute will be applied to the third letter of the tri-radical verb. The index may or may not have a change on the letter.

- Suffix: if and only if the rule is valid and rated highly, suffix will be added or concatenated to the right of the tri-radical verb. The value in the suffix attribute can be null or verified suffix morpheme.

One of the decisive attributes a rule that is mentioned above is status, the status has two possible values. At the time of generating words based on the rules, only rules with status "Y" will be applied. Those rules with status "N" will not applied, until status changed to "Y".

The process of generating words by applying the verified rules to the tri-radical verbs is a time taking process since having large number rules and tri-radical verbs.

Table 4.3: Examples of Word Generation

Rules						Sample	
Prefix	Index Letter 1	Index Letter 2	Index Letter 3	Suffix	Status	Tri-radical verb	Result
እንደተ	0	0	0		Y	ፈለገ	እንደተፈለገ
እየ	0	0	3	ችሁ	Y	ሰበረ	እየሰበራችሁ
እንዲ	0	5	1	ት	Y	ደመረ	እንዲደምሩት
ሊ	0	5	5	በት	Y	ለመነ	ሊለምንበት
እንዲ	0	5	5	ብን	Y	ከበደ	እንዲከብድብን

Algorithm 4.6: Algorithm for Word Generator

```
Input: rules_list[][], tri_radical_verbs[] as trvs
       // rules_list: is list of rules to generate Amharic words from tri-radical verbs
       // trv: is list of Amharic tri-radical root verbs
Task: Generate words combining rules X root verbs
Start
    Declare string array Words_List[][]
    for each verb in trvs do:
       for each rule in rules_list do:
             if rule[status] is same as "Y" then:
```

```
                    Words_List ← append rule, verb, concatenation of (
rule[prefix],    letterIndexChangeto(Letter1    of    verb    into    Index    of
letterIndex1),     letterIndexChangeto(Letter2    of    verb    into    Index    of
letterIndex2),    letterIndexChangeto(Letter3    of    verb    into    Index    of
letterIndex3), rule[suffix])
       end for
    end for
return Words_List[]]]
End
```

4.2.10 Lexicon Repository

The main objective of this thesis work is to generate lexicon containing vast number of Amharic words generated from tri-radical verbs. After generation Amharic words by applying acceptable rules to the tri-radical verbs, the words generated with their source form (i.e., root form) and the rules applied will records as lexicon in the Lexicon Repository.

Chapter Five: Experiments and Evaluations

In this chapter, how the designed model can be implemented and the results obtained from running the demonstration code will be evaluated. The results obtained can determine the strength and weakness of the proposed/demonstrated solution for generating Amharic words from tri-radical verbs. To verify whether the objectives of this thesis works is met or not, different experiments are conducted. To effectively evaluate the results obtained, the experiments are divided into three parts. The first part of the experiment focuses on evaluating the effectiveness of the indexing process done to manipulate the vowels of the root verbs so that new words related to the root verbs can easily be found from the reference source file. The second part of the experiment focuses on evaluating the efficiency of the process of morpheme extraction and rule extraction so that rules extracted from the found related words of the tri-radical verbs are good enough to be applied. The third part of the experiment is to test the effectiveness of the rating process applied to verify the rules found. And this last experiment also bases its evaluation on the words generated by the rules approved by the rating process.

5.1 Overview

In the implementation and demonstration of the proposed work – this thesis work begins its task by availing mandatory source datasets that are vital to the processes ahead. These sources datasets are reference file containing list of thousands of unique Amharic words, list of Amharic tri-radical verbs, and indexed Amharic letters list. After generating index changed words from the root verbs then those index-changed words will be searched in the reference file with a condition of containing the word in any form. If found, it will be captured and morphemes will be extracted. From the morphemes and the changed-index, rules applied will be identified. The rules identified will be rated in order to validate. the rules found. The rating processes decide whether the specific rule identified is good enough to be applied to tri-radical verbs.

5.2 Development Environment

In order to test and demonstrate the main functionalities of the proposed thesis work, various tools are used. List of programming languages, coding editors, libraries, dataset related tools, Operating System and others are presented below.

- **Python**

Python[1] is a very powerful high-level programming language that is considered as easy to learn. Its high-level data structures are effective, simple and also are highly recommended for object-oriented programming. As a high-level programming language Python has Python interpreter with extensive standard library. Python has also a large development community contributing in the development of more capabilities to its functional and non-functional abilities. of is a that is processed by a python interpreter to produce results.

In this thesis work, Python is selected as the best development tool because:

- It minimizes the development time as compared to other programming languages such as C++, Java and C#.
- It possesses large number of standard and user-defined/open-to-use libraries and packages.
- It is highly recommended by experts in machine learning and artificial intelligence.
- It can work in any platform.
- Previous works related to this thesis work are developed by Python.
- It makes things easier to process Unicode texts like Amharic.

The python version used in development demonstration for this thesis work is Python 3.9 stable version.

- **PyCharm**

In this thesis work, we used PyCharm[2] as our Python code editor. It is an integrated development environment (IDE), primarily dedicated to Python programming. It provided various functionalities such as boosting code qualities, supporting code analysis, enabling integration with version control systems, possessing a graphical debugger and an integrated unit tester, and supporting advance front-end and back-end developments like web development, and also supporting data science.

PyCharm works on all operating systems such as Windows, macOS and Linux versions. Its community edition is free to use. And its professional edition need fee for proprietary license and it possesses extra features not available on the community edition.

[1] https://www.python.org
[2] https://www.jetbrains.com/pycharm

In this thesis work, the community edition is used since it adequate for the features demanded.

- **CSV Library**

To make the file processing easier, CSV (Comma Separated Values) format is used since it is platform-independent and relatively small in file size and that makes the process of reading, writing and manipulating faster. Source files that are mandatory to this thesis work are converted to CSV format. And also results from different components of the proposed work are generated in CSV. Python's CSV library[3] is used to read, write and manipulate the CSV files processed in this thesis work.

- **Windows**

In order to demonstrate how this thesis work, the program developed is running on Windows operating system knowing the fact that it is one of the most commonly used Operating System in the world. Windows 10 is the version used.

5.3 Data collection

For this thesis, we used two datasets to test the validity of the designed model. The first one is a data having a list of tri-radical verbs. And the other one is a dataset containing list of Amharic words crawled from the Internet. And this dataset not only contains the list of Amharic words but also the number of frequencies the words is used.

The first dataset containing the Amharic tri-radical verbs is organized in one column table data of CSV file to easily access the data. And the second dataset is also organized in CSV file but has three columns namely the word, word length and frequency.

5.4 Configuration of the Proposed Work

To obtain a better or accurate result in derivation of Amharic words from tri-radical verbs, the following must be fulfilled.

- List of Amharic tri-radical verbs are needed.
    - It is recorded in CSV file.
- List of Amharic Words used for reference

[3] https://docs.python.org/3/library/csv.html

    - It is recorded in CSV file.
- Python setup is needed.
- IDE for Python setup is needed.

5.5 Implementation Details

In the previous section, Section 5.4, the mandatory tools and data components setup is described. In this thesis work, the implementation detail of the components is presented below in Table 5.1.

Table 5.1 Implementation detail of the components

Component Name	Implementation Detail
Tri-radical Verbs Source	▪ Avails list of Amharic tri-radical verbs
Reference Data Source	• Avails list of Amharic Words to be used for reference
Source Data Repository	• Stores both Tri-radical verbs and Reference Data source files
Lexicon Repository	• Stores newly generated Amharic lexicon
Rules Repository	• Stores list of rules identified
Index Changer	• Select Word from the tri-radical verbs • Change the index of each letter of the tri-radical • Generate list of index-changed words • Save the index-changed words
Word Finder	• Open the Reference Data Source file • Pick word from the index-changed words • Search the word from the Reference Data Source file with the parameter “contains word” • If found, pass the word from the Reference Data Source file to the Morpheme Extractor • Save the word found. • Iterate until done for all words.
Morpheme Extractor	• Accept word found from Reference Data Source File

Component Name	Implementation Detail
	and Index Changed Word • Extracts the prefix, the root, the suffix from the Word Found as compared to the Index Changed Word. • Segregate the root, prefix, suffix, index-changed word and Transfer to Rules Identifier • Save segregated elements of a word (root, prefix, suffix, index-changed
Rules Identifier/ Generator	• Accept segregated elements of a word (root, prefix, suffix, index-changed) • Check the pattern of the word with rules from the Rules Repository • If found, add the frequency value of the rule by one. • If not found, add the segregated elements of a word (root, prefix, suffix, index-changed) to the rules repository and assign the frequency value of the rule to one. • Save the changed in the Rules Repository.
Rules Rater	• Open the Rules Repository. • Check the frequency of each Rule • Increment the frequency value of the rule by one each time the rule found being used in a word • Save Changed on the Rules Repository
Word Generator	• Open the Rules Repository • Open the Tri-radical verbs source file • Select the highly rated rules from the repository • Apply the rules selected to each of the tri-radical verbs • Save the newly generated words on the lexicon Repository

5.6 Evaluation Metrics

In this thesis, to evaluate the results obtained from the model developed, the results obtained are verified manually by humans.

Accuracy

In the case of this thesis, accuracy measures the ratio of correctly identified rules to the whole number of rules identified.

Accuracy is the most spontaneous one. Accuracy answers the question that how many rules are correctly labeled out of the general number of rules identified.

$$\text{Accuracy} = (TP + TN) / (TP + FP + FN + TN)$$

Numerator: all correctly identified rules (All trues)

Denominator: all rules identified

Precision

Precision is the ratio of the correctly identified rules by the model developed to all identified as correct ones whether they are correct or not.

Precision answers the question that how many of those who selected as correct rules are actually the right rules.

$$\text{Precision} = TP/(TP + FP)$$

Numerator: All correctly selected rules.

Denominator: All identified rules as correct ones whether they are correct or not in reality.

Recall (Sensitivity)

Recall is the ratio of the correctly identified rules by the model developed to all rules that are correct in reality.

Recall answers the question that from all correct rules, how many of those the model developed correctly predicts.

$$\text{Recall} = TP/(TP + FN)$$

Numerator: All correctly selected rules.

Denominator: All rules found

5.7 Experimental Setup

The testing of the designed model for morphological derivation of Amharic words from tri-radical verbs is done in a personal computer with the capacity shown below.

Table 5.2 Specifications of the personal computer used for testing.

Manufacturer	Lenovo
Installed Physical Memory	8GB
Processor	Intel® Core™ i7-4510U CPU @ 2.00GHz, 2601Mhz, 2 Core(s), 4 Logical Processor(s)
Operating System	Windows 10 Pro

5.8 Baseline Experimental

Since there is no specific research made on generating words from tri-radical but the closest research that has morpheme extraction functionality is the one by (Mesfin and Yaregal, 2014). They divided their data into training and testing. Their corpus comprises 1022 words, from those words 181 of them are nouns/adjectives and the remaining 841 are verbs. The model they developed extracted 1356 instances from the 181 nouns/adjectives. And from the 841 words, 6719 instances are extracted.

5.9 Experimental Scenarios

To measure the efficiency of this study, the following scenarios are tested.

- **Scenario 1**: To check the accuracy of the Index Changer component. The Index Changer component changes the index of the tri-radical root verbs into possible combination of letters. The Index Changer changes the first letter of the tri-radical verb into three forms (0, 3, 5). For example, ‘ፈ’ can be changed into only ‘ፈ’, ‘ፋ’, and ‘ፍ’. The decision made to only use the three letter forms, is after changing all forms and found out the letter from except the three mentioned above (for the remaining 4 indexes) does have a meaning at all. And the remaining two letters of the tri-radical verb can be index changed into all letter forms.

Cartesian product of possible letter combination: 7 x 7 x 7 = 343 possible index-changing for one tri-radical word. With the issue mentioned above, the first letter for 3 possible letter forms: 3 x 7 x 7 = 147 possible index changed letters combination will be considered for the upcoming processes of the design model. The perfect accuracy of the Index changer component can only be identified when it is combined with or without prefix and/or suffix.

- **Scenario 2:** To check the accuracy of Word Finder component. When searching words from the reference data source (Amharic Word List), the words that are less than 3 in length will be discarded to minimize the effort of searching process. The searching process applies "contains" method, i.e., words containing the index changed words either in left, right, or middle of the word will be considered for the upcoming processes.
  The accuracy of the result of the Word Finder Component will be evaluated when the Morpheme Extractor figures out which part of the word found is prefix, root, or suffix. And which prefixes or suffixes are wrong or correct.

- **Scenario 3:** To check the accuracy of the Morpheme Extractor component. When the Word Finder component identifies the words based on the index-changed words, the letter(s) before the index-changed word will be considered as prefix; whereas the letter(s) after the index-changed word will be considered as suffix. In this component, the correctness of the prefixes and the suffixes will be evaluated. If one of the morphemes in a word is wrong, the rule extracted from the word is also wrong.

- **Scenario 4:** To verify the accuracy of the Rules Identifier/Generator. From the above scenarios mentioned, rules in generating Amharic words from tri-radical verbs can be identified in different ways. Changing the index of the words, adding prefixes and/or suffixes.

5.10 Results

According to the experimental scenarios mentioned in Section 5.9, the following results are obtained.

Scenario 1: Evaluating the accuracy of the Index Changer. One of the many approaches in Amharic language to derive words from tri-radical verbs is changing the order (in our case index). In this thesis work, index changing performed to identify the possible derivative of the tri-radical verbs.

- The first letter of the tri-radical can be 1st order (index-0), 4th order (index-3) or 6th order (index-5). This is taken to consideration in the source code of the demo program created this minimizes the task of identifying wrong rules. And this also protects the machine from processing unnecessary data.
- The second and third letters of the tri-radical verbs were given full index (0-7) for index changing and those results in wrongly index words especially in the second letter of the tri-radical verb. And from the results obtained the following are noticed:
    - All the rules generated by index changing the second letter of the tri-radical verbs into second order (index-1), third order(index-2), fifth order (index-4), and seven order (index-6) are totally unacceptable since there is no such word in Amharic that has such rule.
        - Number of rules generated by wrongly consider the second letter of the tri-radical verbs: 11,390 rules from the total 85,115 rules (>=13%).
    - All the rules generated by index changing with the second letter of the tri-radical verbs into first order (index-0), fourth order(index-3), and six order (index-5) are acceptable and must be good enough to combine with other parts of the word and finally make a meaningful word.
    - Furthermore, it can be concluded that 3x7x7 approach of index changed words instances generation is not going to be the perfect approach. The approach should be 3x3x7 approach that can reduce the time spent to process and the number wrongly identified rules.

Scenario 2: The second scenario to consider is figuring out the accuracy of the program prepared for demonstration purpose in the case of finding the exact match of the index changed words from the reference data source.

- From the sample data for this thesis, the word-finder component searches the entire word instance it obtained from the Index Changer module without analyzing all the meaning, structure of the word, or POS it is in. For that specific reason, the accuracy of the word-finder will be considered perfect. The main issue is the performance of the designed model that can be resolved partially by selecting or applying the best algorithm.

Scenario 3: The third and vital scenario is evaluating the accuracy of the morphemes extracted. The morphemes extracted after finding the index changed words in any part of word listed in the reference document. In morpheme extraction, identifying the prefix, the stem/root and suffix/

- In the case of prefixes:

  10920 unique prefixes were extracted and from these unique prefixes, the following results are also identified.

  - Out of the 10920 unique prefixes, 10656 of them are labeled wrong prefixes automatically. And from these 10656 prefixes 63 of them are wrong labeled but truly are correct prefixes. The remaining 10593 prefixes from the 10656 are totally wrong prefixes.
  - Out of the 10920 unique prefixes, 264 of them are labeled correct prefixes automatically. And from these 264 prefixes 40 of them are correctly labeled but truly are wrong prefixes. The remaining 224 prefixes from the 264 are totally correct prefixes.

  $$\text{Accuracy} = (TP + TN)\ /(TP + FP + FN + TN)$$

  $$= (224 + 10593)\ /(224 + 40 + 63 + 10593)$$

  $$= 0.99$$

  $$\text{Precision} = TP/(TP + FP)$$

  $$= 224/(224 + 40)$$

= 0.84

Recall = $TP/(TP + FN)$

$= 224/(224 + 63)$

= 0.78

- In the case of suffixes:

  14802 unique suffixes were extracted and from these unique suffixes, the following results are also identified.

  - Out of the 14802 unique suffixes, 14114 of them are labeled wrong suffixes automatically. And from these 14114 suffixes 110 of them are wrong labeled but truly are correct suffixes. The remaining 14004 prefixes out of the 14114 suffixes are totally wrong suffixes.
  - Out of the 14802 unique suffixes, 687 of them are labeled correct suffixes automatically. And from these 687 suffixes 77 of them are correct labeled but truly are wrong suffixes. The remaining 610 suffixes from the 687 are totally correct suffixes.

  Accuracy = $(TP + TN)/(TP + FP + FN + TN)$

  $= (610 + 14004)/(610 + 77 + 110 + 14004)$

  = 0.99

  Precision = $TP/(TP + FP)$

  $= 610/(610 + 77) = 0.88$

  Recall= $TP/(TP + FN) = 610/(610 + 110) = 0.85$

Scenario 4: The fourth and main scenario is evaluating the results obtained in the process of identifying and generation of rules. All of the above scenarios have direct impact on the results of rules identified and/or generated.

Generally, 85,115 rules are identified.

- Out of the 85,115 rules in 55,339 rules, the prefixes are well placed. In another words the prefixes are correct by themselves without considering the status of the remaining morphemes (parts of word). But in the remaining 29,776 rules, the prefixes are wrong by themselves and make the rules also incorrect.
- Out of the 85,115 rules in 52,714 rules, the suffixes are well placed. In another words the suffixes are correct by themselves without considering the status of the remaining morphemes (parts of word). But in the remaining 32,401 rules, the suffixes are wrong by themselves and make the rules also incorrect.
- The index changer component also makes 11,390 rules unfunctional as mentioned in scenario 1.

Finally, after interconnecting the wrongly identified rules based on the above scenarios, 25,509 unique rules are identified.

In order to enhance the process of identifying the correct rules, instance-based learning algorithms (also called memory-based learning algorithms) are applied. For instance, to decide the more accurate rules, the statistical or frequency of the rules in the reference document are considered. In simple words and real example:

- 25,509 unique rules will be applied to all 373 tri-radical verbs. And Lexicon with 9,514,857 (373x25,509) Amharic words will be generated. When the frequency of the rules is not an evaluation approach.
- 9,222 unique rules will be applied to all 373 tri-radical verbs. And Lexicon with 3,439,806 (373x9,222) Amharic words will be generated. When the frequency of the rules is mandatory and must be above one.
- 5.935 unique rules will be applied to all 373 tri-radical verbs. And Lexicon with 2,213,755 (373x5,935) Amharic words will be generated. When the frequency of the rules is mandatory and must be above two.
- 2,990 unique rules will be applied to all 373 tri-radical verbs. And Lexicon with 1,115,270 (373x2,990) Amharic words will be generated. When the frequency of the rules is mandatory and must be above five.

- 1,701 unique rules will be applied to all 373 tri-radical verbs. And Lexicon with 634,473 (373x1,701) Amharic words will be generated. When the frequency of the rules is mandatory and must be above ten.
- 76 unique rules will be applied to all 373 tri-radical verbs. And Lexicon with 28,348 (373x76) Amharic words will be generated. When the frequency of the rules is mandatory and must be above one hundred.

The more we rely on the frequency of the rules, the more the rules are perfect.

5.11 Chapter Summary

Morphological synthesis systems are useful in many areas of NLP for Amharic. Natural Language processing plays a significant role in increasing computers' ability to understand natural languages. Morphological synthesis is one aspect of the task of understanding natural language, the language by which most human knowledge is recorded.

There have been researches on processing of Amharic language in different aspects. Many of tech researchers contributed to enhance the machine processing the language, but it appeared that the language is not completely exploited in the Natural language processing world. This thesis developed a hybrid approach to generate Amharic words from tri-radical verbs and end up having a rich Amharic lexicon. In this research, the hybrid approach comprises a rule-based incorporated with memory-based learning methodology.

This work utilizes two source data (tri-radical verbs and Amharic word list). Changing the index of the letter in the tri-radical verbs and search in the Amharic word list, if found capture the pattern, identify the morphemes, and identify, and generate the rule. If the rules are acceptable, apply to the whole tri-radical verbs at the end. In order to verify the acceptability of the rules, memory-based learning approach is applied.

The approach of finding the rules by searching the index changed format of the tri-radical from the reference document result is very acceptable accuracy.

Chapter Six: Conclusion and Future Work

6.1 Conclusion

In this thesis, the gaps observed in natural process language processing of Amharic language are elaborated in manner where there is no such rich Amharic lexicon.

The thesis work, starts by identifying tri-radical verbs and their nature with the mission to target these tri-radical verbs as the source for generating thousands of derived words from each of them. Amharic is so powerful and rich, there is no such reference that can explicitly inform the exact number of words can be derived from an Amharic root/stem word. In this thesis, we use existing Amharic word list (reference) documents and few hundreds of tri-radical verbs to identify existing rules that can shape the derivation process. We change the index/order of the tri-radical verbs and search for word containing that index changed words. When the index changed words are found in the reference document, the pattern of the word will be taken to further processing of morpheme extraction. At the time of the morpheme extraction, the prefix part, the root/stem part and the suffix part of the word will be captured. After the morpheme extraction is done, the pattern of the morphemes will be taken as rule. If the rule is highly rated and has no issues like wrong prefix, wrong suffix, and wrong index change, it will used for generation words from the tri-radical verbs. Those generated words will be added to the Amharic lexicon.

In order to accomplish the above-mentioned tasks various components are used. Repositories were used to keep source files (both tri-radical file, reference document) and result files (rules and lexicon). And also processing components having specific tasks namely, Index Changer, Word Finder, Morpheme Extractor, Rules Identifier/Generator, Rules Rater, and Word Generator were implemented. In every part of the components the algorithms applied are considered based on the accuracy they can deliver.

Generating words from Amharic root/stem verbs is not easy. What this thesis accomplished is great with its limited scope of only using tri-radical verbs. It showed that Amharic is so powerful and rich that more than 20,000 words can be generated from just a single word. More than 200 prefixes and more than 600 suffixes can be used.

As a result, this thesis can be a stepping stone for more Amharic morphological related researches.

6.2 Contribution

The contributions of this thesis work are as follows,

- Identifying that when changing the index/order of letters in tri-radical verbs, the first two letters of the tri-radical verbs can be changed into three indexes/orders.
- Without having any number of rules, finding derivation rules from by matching the tri-radical verbs and their index changed forms with existing list of Amharic word list in a reference document.
- Algorithms to easily extract morphemes from Amharic tri-radical stem/root verbs.
- Algorithms to verify the validity every deliverable component with result.
- List of rules to derive more words from Amharic tri-radical stem/root verbs.
- List of Amharic words generated from tri-radical verbs (Amharic Lexicon)

6.3 Future Works

As mentioned in the previous sections of this thesis work, Amharic is a very rich and powerful language and needs to be further studied.

- Morphological derivation from all forms of verbs not only tri-radical verbs.
- Generating of All-in-one Amharic Lexicon.
- Implementation of Amharic Language Processes

References

[1] ***Edward Sapir***, Language: An Introduction to the Study of Speech, 1921

[2] ***Robert Hetzron***., The Semitic Languages, Routledge Taylor and Francis Group, 2005.

[3] ***Federal Democratic Republic of Ethiopia***, Population Census Commission, Summary and Statistical Report of the 2007 Population and Housing Census, December 2008.

[4] ***Diksha Khurana, Aditya Koli, Kiran Khatter and Sukhdev Singh***, Natural Language Processing: State of The Art, Current Trends and Challenges, retrieved from https://www.researchgate.net/publication/319164243, last accessed on February 1, 2021.

[5] ***Karin Verspoor, Kevin Bretonnel Cohen***, Natural Language Processing, retrieved from https://www.researchgate.net/publication/291179558, last accessed on February 1, 2021.

[6] ***Eka Hardiyanti Bugis, Heri Yudityo, and Wa Ode Tika Rizky***, Morphology & Syntax - Morpheme Forms, retrieved from https://www.academia.edu/18431538, Last accessed on January 27, 2021.

[7] ***Saba Amsalu and Dafydd Gibbon***, Finite state morphology of Amharic, retrieved from https://www.researchgate.net/publication/228676251, Last accessed on January 16, 2021.

[8] ***Mesfin Abate and Yaregal Assabie***, Development of Amharic Morphological Analyzer Using Memory-Based Learning, retrieved from https://www.researchgate.net/publication /300023701, last accessed on January 16, 2021.

[9] ***Neha Yadav***, Applications Associated with Morphological Analysis and Generation in Natural Language Processing. Retrieved from https://www.ijstr.org/final-print/aug2017, last accessed on December 27, 2021.

[10] ***Jochen Trommer***, A Feature-Geometric Approach to Amharic Verb Classes, retrieved from http://home.uni-leipzig.de, Last accessed on January 16, 2021.

[11] ***A. Alemu, L. Asker, and M. Getachew***, Natural Language Processing for Amharic: Overview and Suggestions for a Way Forward, 2003 In Proceedings of the 2nd Workshop on Treebanks and Linguistic Theories, Vaxjo University, Sweden.

[12] ***S. Amsalu and D. Gibbon***, Finite State Morphology of Amharic, Fakult¨at f¨ur Linguistik und Literaturwissenschaft, Universit¨at Bielefeld, Germany.

[13] K. ***Lisanu***, Design and Development of Automatic Morphological Synthesizer for Amharic Perfective Verb Forms (Master's thesis), June 2002. School of Graduate Studies, School of Information Studies for Africa, Addis Ababa.

[14] ***M. Birile***, Synthetic Speech Trained Large Vocabulary Speech Recognition System (Master's thesis), July 2008. School of Graduate Studies, Faculty of Technology, Addis Ababa University, Addis Ababa.

[15] ***Imed Zitouni***, Natural Language Processing of Semitic Languages, 2014

[16] ***Rogers H***, Writing systems: a linguistic approach. Blackwell Publishing, Malden, 2005

[17] ***Kibur Lisanu***, Design and Development of Automatic Morphological Synthesizer for Amharic Perfective Verb Forms, 2002.

[18] ***Wondwossen Mulugeta and Michael Gasser***, Learning Morphological Rules for Amharic Verbs Using Inductive Logic Programming, Workshop on Language Technology for Normalization of Less-Resourced Languages, 2012.

[19] ***Tesfaye Bayu***, Automatic Morphological Analyzer for Amharic: An Experiment Employing Unsupervised Learning and Auto-segmental Analysis Approaches, 2002.

[20] ***Christian Lehmann et al***, Morphology: An International Handbook on Inflection and Word-Formation, 2002.

[21] ***Baye Yimam, YeAmaregna Sewasew***, 2nd edition, 2008.

[22] ***Daniels et al***, "Ethiopic Writing", The World's Writing Systems, 1996.

[23] ***Kartik et al***, Hybrid Inflectional Stemmer and Rule-based Derivational Stemmer, 2011.