ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
DEPARTMENT OF COMPUTER SCIENCE

Morphological Analysis of Ge'ez Verbs Using Memory Based Learning

Yitayal Abate

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

November 07, 2014

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
COLLEGE OF NATURAL SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

Morphological Analysis of Ge'ez Verbs Using Memory Based Learning

Yitayal Abate

Advisor: Yaregal Assabie (PhD)

Co-advisor: Dessie Keleb (Memhr)

SIGNATURE OF THE BOARD OF EXAMINERS FOR APPROVAL

1. Dr. Yaregal Assabie, Advisor _____
2. _____ _____
3. _____ _____

ACKNOWLEDGEMENTS

First and foremost, I would like to thank the almighty God, who favors me to begin and to bring to an end this study. My next and most grave gratitude must go to my advisor Dr. Yaregal Assabie for his continuous support, sharing his interesting idea, experience, and discussion and guidance throughout the various stages of this study.

I would like to thank my co-advisor Memhr Dessie Keleb, Mergeta Abiy Lebeza and Deacon Belsty Abewa, they were with me to analyse the Geʿez verbs. Without them, it was impossible to work on Geʿez language for me. They were with me from the beginning to the completion of this thesis answering each and every query I have related to the language.

Lastly, I also thank everybody who stands on my side to finish this study, especially my classmates and friends who shared me their ideas and helpful comments and suggestions.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Abbreviations and Symbols

Abbreviations:

AI = Artificial Intelligence

ASR = Automatic speech recognition

ATM = Aspect Tense Mode

C = Consonant

CV = Cross-Validation

EOTC = Ethiopian Orthodox Tewahido Church

HMM = Hidden Markov Model

IBl = Instance Based Learning

IB2 = Instance Based two

ID = Inverse-Distance

IE = Information Extraction

IG = Information Gain

IGTREE = Information Gain Tree

IL = Inverse-Linear

ILP = Inductive Logic Programming

IR = Information Retrieval

KNN = K-Nearest Neighbor

LOOCV = Leave-One-Out Cross-Validation

MBL = Memory Based Learning

MBLP = Memory Based Learning Processing

ML = Machine-Learning

MT = Machine Translation

MVDM = Modified Value Difference Metrics

NLP = Natural Language Processing

NLU = Natural Language Understanding

OMS = Object Marker Suffix

POS = Part Of Speech

SMS = Subject Marker Suffix

SVM = Support Vector Machine

TiMBL = Tulburg Memory-Based Learning

TLM = Two-Level Morphology

TRIBL = Tree Instance Based Learning

TRIBL2 = Tree Instance Based Learning Two

WER = Word-Error Rate

WSD = Word Sense Disambiguation

1ppn = First Person Plural Neutral

1psn = First Person Singular Neutral

2ppf = Second Person Plural Feminine

2ppm = Second Person Plural Masculine

2psf = Second Person Singular Feminine

2psm = Second Person Singular Masculine

3ppf = Third Person Plural Feminine

3ppm = Third Person Plural Masculine

3psf = Third Person Singular Feminine

3psm = Third Person Singular Masculine

Symbols:

[ ] What is inside is a reference

/ / What is inside is a transliterated Ge‟ez word.

( ) What is inside is a gloss or abbreviation or explanation

Abstract

Ge‟ez is the classical language of Ethiopia and still used as the litrugical language of EOTC. Many ancient literatures were written in Ge‟ez. The literature includes religious texts and secular writings. The ancient philosophy, tradition, history and knowledge of Ethiopia were being written in Ge‟ez. For automatic analysis of these documents Ge‟ez morphological analysis is needed. Morphological analyzer is one of the most important basic tools in automatic processing of any human language. It analyses the naturally occurring word forms in a sentence and identifies the root word and its features.

In this study, we used MBL to automatically analyze the morphology of Ge‟ez verbs. The system has two components: training and analysis. In the training phase, we identified the annotation process for our dataset in a character based representation of features. Then, these annotated dataset are extracted in a fixed length of instance vectors using windowing method. Next, instances are passed to the memory based learning tool (TiMBL). Finally, the learning model is built. On the other hand, the analysis phase performs instance making by extracting features from the given text to have similar structure of features during comparison. Then the extracted features are passed to the morpheme identification process to be compared with individual instances in memory and stems are extracted with their morpheme functions. Finally, the roots are extracted from the stems.

The system was developed using python where we used TiMBL‟s IB2 and TRIBL2 algorithm for implementation. The performance of the system has been evaluated using 10-fold cross-validation technique. Testing was done using the default and optimized parameter settings. The overall accuracy with optimized parameters using IB2 and TRIBL2 was 93.24% and 92.31%, respectively. Similarly, the overall precision, recall and F-score with optimized parameters using IB2 were 55.6%, 56.3% and 59.95%, respectively. In the same manner the precision, recall and F-score using TRIBL2 were 58.8%, 60.3% and 59.54%, respectively. Moreover, a learning curve was drawn. The graph showed that as the number of training dataset increase, the accuracy on unseen data can be increased. Therefore, IB2 algorithm shows better result than TRIBL2 algorithm for Ge‟ez verb morphology.

Key words:- Ge‟ez Verbs, Ge‟ez Morphology, Morphological Analyzer, Memory Based Learning, Character Based Analysis, Cross Validation, Feature Extraction

Chapter One: Introduction

1.1 Background of the Research

Natural Language Processing (NLP) has been developed in the 1960s, as a subfield of Artificial Intelligence and Linguistics [37]. It is a field of Computer Science that investigates interactions between computers and human languages, which is used for both generating human readable information from computer systems and converting human language into more formal structures that a computer can understand [9]. Therefore, it is important for scientific, economic, social, and cultural reasons. It is experiencing rapid growth as its theories and methods are deployed in a variety of new language technologies. For this reason it is important for a wide range of people to have a working knowledge of NLP. Within the industry, this includes people in human-computer interaction, business information analysis, and web software development. Within academia, it includes people in areas from humanities computing and corpus linguistics through to computer science and artificial intelligence [14].

The field of NLP was originally referred to as Natural Language Understanding (NLU) in the early days of AI. It is well agreed today that while the goal of NLP is true NLU, that goal has not yet been accomplished. A full NLU System would be able to: paraphrase an input text, translate the text into another language, answer questions about the contents of the text and Draw inferences from the text. While NLP has made serious inroads into accomplishing goals one to three, the fact that NLP systems cannot, of themselves, draw inferences from text, NLU still remains the goal of NLP [11]. The aim of NLP is studying problems in the automatic generation and understanding of natural languages. Natural language is understood as a tool that people use to express themselves and has specific properties that reduce the efficiency of textual information retrieval systems. These properties are linguistic variation and ambiguity. NLP studies the problems of automated generation and understanding of natural human languages [29].

While NLP is a relatively recent area of research and application, as compared to other information technology approaches, there have been sufficient successes to date that suggest that NLP-based information access technologies will continue to be a major area of research and development in information systems now and far into the future [11]. Current NLP systems tend

to implement modules to accomplish mainly the lower levels of processing. This is for several reasons. First, the application may not require interpretation at the higher levels. Secondly, the lower levels have been more thoroughly researched and implemented. Thirdly, the lower levels deal with smaller units of analysis, e.g., morphemes, words, and sentences, which are rule-governed, versus the higher levels of language processing which deal with texts and world knowledge, and which are only regularity-governed [11]. Well known problems of NLP are morphological analysis, part of speech tagging, word sense disambiguation, and machine translation [9].

Morphology deals with the inner structure of individual words and the laws concerning the formation of new words from pieces, morphs [6]. Morphologies are motivated by four considerations: (1) the discovery of regularities and redundancies in the lexicon of a language (such as the pattern in *walk, walks, walking; jump, jumps, jumping*); (2) the need to make explicit the relationship between grammatical features (such as nominal number or verbal tense) and the affixes whose function it is to express these features; (3) the need to predict the occurrences of words not found in a training corpus; and (4) the usefulness of breaking words into parts in order to achieve better models for statistical translation, information retrieval, and other tasks that are sensitive to the meaning of text [7].

Furthermore, Morphology is the field within linguistics that studies the [36] identification, analysis and description of the structure of a given language's morphemes and other linguistic units, such as root words, affixes, parts of speech, etc. [18]. Morphemes are the smaller elements of which words are built. Two broad classes of morphemes are stems and affixes. Affixes that are added to the base to denote relations of words are morphemes. Morphemes can either be free (they can stand alone, i.e., they can be words in their own right, are also referred as roots), e.g., dog, or they can be bound (they must occur as part of a word), e.g., the plural suffix –s on dogs [36].

Morphology plays two central roles in language. In its first role, *derivational morphology* (*morphemes)* allows existing words to be used as the base for forming new words with different meanings and different functionality. Example, the noun *judgment* is formed from the verb *judge*, and the adjective *inedible* has a different semantic meaning from its related verb eat. In its second role, *inflectional morphology (morphemes)* deals with syntactic features of the languages such as *person* (I am, you are, he is), *number* (one child, two children), *gender* (actor, actress), *tense* (eat, eats, eating, eaten, ate), *case* (he, him, his), and *degree* (cold, colder, coldest). These syntactic

features, required to varying degrees by different languages, do not change the part of speech of the word (as the verb *eat* becomes the adjective *inedible*) and do not change the underlying meaning of the word (as *cellist* from *cello*) [20].

1.2 Statement of the Problem

Ge‟ez is the classical language of Ethiopia and belongs to the Semitic language family. The other Semitic-Ethiopian languages are Tigre and Tigrinya (known as North Ethiopic), Amharic (the national language of Ethiopia), Argobba, Harari, and Gurage (called South Ethiopic). Although Ge‟ez ceased to be spoken in the twelfth or thirteenth century, it has remained the language of literature and of liturgy. Our knowledge of the language derives from the vast literature written in Ge‟ez. The literature includes religious texts (such as the Bible, Apocrypha, Pseudepigrapha, liturgical literature, homiletic, theological, and magical texts, stories of martyrs and saints, religious poetry, hymns in honor of Christ, the Virgin, the martyrs, the saints, and angels), as well as secular writings (histories and romances, legal, mathematical, and medical texts) [25]. The ancient philosophy, tradition, history and knowledge of Ethiopia was being written in Ge‟ez and also there are different books which are written in this language. To keep and transfer these identities to the next generation, citizens must know the meaning of these written books/documents. If they don‟t know the idea in the documents, they will not give any attention for these heritages. These resources can also be used as sources of philosophy, creativity, knowledge and civilization both to Ethiopia and the rest of the world. Moreover, If someone who wants conduct a research on issues related to the classical custom, history, politics, tradition, and religion of Ethiopia, he/she has to explore the works handed down from the previous generations. So he/she must investigate these literatures. To use these resources, one must know the language itself or else these literatures have to be translated into either of the currently spoken languages manually, which may take a long time. To solve this problem and the previous one, studying the linguistic nature of the language in a scientific approach with the help of Information Technology is necessary. This requires conducting a research to develop a more efficient morphological analyzer for Ge‟ez verbs to minimize the time to study the language and to extract automatically the documents which were written in Ge‟ez language.

1.3 Objectives

General objective

The general objective of this research is to develop a morphological analyzer for Geʿez verbs using memory based learning approach.

Specific objectives

To achieve the general objective, the following specific objectives will be addressed: -

- Understanding morphological, phonological and orthographic[1] properties of Geʿez verbs and the phonological and morphological processes involved in Geʿez verbs formations and conjugations.
- Assess different techniques and approaches employed so far in morphological analysis tasks and select the ones that are appropriate to the morphological property of Geʿez verbs.
- Organizing training and test corpus data.
- Adopt a machine learning algorithm that is appropriate to the morphological property of Geʿez verbs.
- Develop a prototype of the system.
- Test the effectiveness and appropriateness of the morphological analysis technique adopted.

1.4 Methodology

Literature Review

The first basis of information for this research are scholars of Geʿez in the EOTC (Ethiopia Orthodox Tewahido Church) who have studied Geʿez both scientifically and traditionally. Consultations with these scholars in the area of Geʿez language morphology will be conducted to better understand the morphology of the language and to get information which is helpful for the research. Other sources for the detailed understanding of the language are the Geʿez grammar books written by foreign and traditional scholars such as, Memhr Dessie Keleb [19], Bender [3], Aleka Kidane Wold Kifle [69], and Dillmann [70]. In addition to this, a number of resources such as journal articles, research

[1] Orthographic rules are spelling rules used to model the changes that occur in a word.

reports, manuals, and other published and unpublished thesis will be used in order to achieve the objectives.

Corpus Preparation

A machine learning approach requires a corpus data to develop the morphological analysis. The corpus being electronic text data consisting of list of words such as those found in magazines, journal articles, brochures (həmər, smathʷ̃dḱ, gubaʷ̃e ḱana) and books (Geʷ̃ez grammar books written by foreign and traditional scholars. In addition to this, a number of resources such as research reports, manuals, and other published and unpublished thesis will be used for the preparation of the corpus.

Prototype Development

Based on the experiment result which is obtained from the training and testing data, the prototype will be developed. For this purpose python and C++ programming languages will be used. Therefore, conclusion and recommendation will be drawn from the training and testing data and finally, the prototype result will be reported.

1.5 Scope and Limitations

The scope of this study is developing a morphological analyzer for Geʷ̃ez verbs and their inflected words. It doesnʷ̃t include the inflected words of other POS categories. Most Geʷ̃ez words are derived and formed from the stem or the root of verbs. There are many nouns, adjectives and adverbs which are basically derived from verbs (Table 1.1). The study doesnʷ̃t include the inflected words of these derived words.

Table 1.1: Some Words of other POS Category Derived from a Verb "ቀደሰ"

Verb	Derived word	Gloss	POS category
ቀደሰ /ḱədəsə/	ቀዳሲ /ḱidasi/	Praise (male)	Adjective
	ቀዳስያን /ḱədasiyan/	Praises(they, 3ppm)	Adjective
	ቀዳሲት /ḱədasit/	Praise (fmale)	Adjective
	ቀዳስያት /ḱədasiyat/	Praise (they, 3ppf)	Adjective
	ቅዳሴ /ḱidase/	Praising/thanks	Noun
	ቅድስና /ḱidisina/	The act of praising	Noun
	ቅድስት /ḱidisit/	Praised (fmale, singular)	Adjective
	ቅዱሳት /ḱidusat/	Praised (fmale, plural)	Adjective
	ቅድሳት /ḱidisat/	the act of praising	Noun
	ቅዱስ /ḱidus/	Praised (male, singular)	Adjective
	ቅዱሳን /ḱidusan/	Praised (male, plural)	Adjective

Source: Adopted from [19].

1.6 Application of Results

Morphological analyzer is one of the most important basic tools in automatic processing of any human language. It analyses the naturally occurring word forms in a sentence and identifies the root word and its features. In spite of its significance, some languages do not have any morphological analyzers available. The absence of such a tool for research severely impedes the development of language technologies and applications like natural language interfaces, machine translation, etc. in these languages [36]. Traditionally, morphological analysis has been done manually. It took linguistic experts several months to years in order to describe a single language. More recently, computer algorithms have been applied which can automate and therefore speed-up the process. These algorithms deploy techniques from machine learning to improve their performance with increasing numbers of words or analyzed word examples they have access to. Morphology and its analysis play an important role in many natural language processing tasks. Automatic speech recognition (ASR) is concerned with the identification of spoken words and their transformation into text. Morphologically complex languages are especially challenging due to the combinatorial explosion of possible morpheme structures [13]. Therefore, morphological analyzer will also be an important component of the NLP to be developed for Geez language. Thus, the beneficiaries of this research include researchers who need to take part in achieving the goal of developing efficient NLP system for Ge‟ez language. The research will be used to put linguistically motivated structure of Ge‟ez verbs, to help Ge‟ez learners and to develop higher forms of NLP systems such as automatic dictionary (lexicon) compilation, spell-check, machine translation, speech recognition, POS tagging, automatic sentence construction, morphological synthesizer, etc.

1.7 Organization of the Thesis

This thesis is organized in six chapters. Chapter Two discusses about the basic concepts of morphology, along with the various approaches employed and selected algorithms for this study. In this chapter, we also discuss about morphology of Ge‟ez verbs and their formation process. Related works are presented in Chapter Three. The architecture of Ge‟ez verbs morphological analyzer along with the implementation principle and algorithms used in the implementation is discussed in Chapter Four. Chapter Five discusses the experimental results along with the default and optimized parameters. Conclusions and future works are presented in Chapter Six.

Chapter Two: Literature Review

2.1 Introduction

Morphological analysis is the basic process for any Natural Language Processing task. It is the first step in Natural Language Processing. As discussed in the first chapter, the main objective of this study is to develop morphological analyzer of Ge"ez verbs. Morphology is the study of internal structure of the word. Morphological analysis retrieves the grammatical features and properties of a morphologically inflected word [27]. Morphological analyzer is a computer program which takes a word as input and produces its grammatical structure as output. It will return its root/stem word along with its grammatical information depending upon its word category [37].

2.2 Concepts and Terminologies

The term morphology is generally attributed to the German poet, novelist, playwright, and philosopher Johann Wolfgang von Goethe (1749–1832), who coined it early in the nineteenth century in a biological context. Its etymology is Greek „morph" which means „shape, form", and morphology is the study of form or forms. In linguistics morphology refers to the mental system involved in word formation or to the branch of linguistics that deals with words, their internal structure, and how they are formed [31]. A primary source of information about morphology is formed by the descriptive grammars of individual languages which usually give a description of inflection and word formation. As stated above in Chapter one, morphology is the sub discipline of linguistics that deals with the internal structure of words [22]. It is the study of word formation –how words are built up from smaller pieces. When we do morphological analysis, then, we are asking questions like, what pieces does this word have? What does each of them mean? How are they combined?

2.2.1 Morphemes

It is clear that the words of a language are not totally separate items without any similarities between them or regularities within their structure. It has been traditional within linguistics to describe the regularities in the structure of words by postulating that a single word can be viewed as made up of one or more smaller units called morphemes. These are able to act as words in isolation and bound morphemes can operate only as parts of other words [32]. Morpheme is the

minimal meaningful unit in a word. The concept of word and morpheme are different, a morpheme may or may not stand alone. One or several morphemes compose a word. As stated in [36] there are four types of morphemes:

- *Free morphemes:* like town and dog, can appear with other lexemes (as in town hall or dog house) or they can stand alone, i.e. "free".
- *Bound morphemes:* like "*un-"* appear only together with other morphemes to form a lexeme. Bound morphemes in general tend to be prefixes and suffixes.
- *Derivational morphemes* can be added to a word to create (derive) another word: the addition of *"-ness*" to "*happy*" for example, to give "*happiness*". They carry semantic information.
- *Inflectional morphemes* modify a word"s tense, number, aspect, and so on, without deriving a new word or a word in a new grammatical category (as in the "dog" morpheme if written with the plural marker morpheme "*-s*" becomes "*dogs*"). They carry grammatical information.

Bound morphemes can also be further categorized into two groups as affixes and contracted (clitics) forms [41].

a. Affixes

In languages with a reasonably rich morphology affix ordering is an important topic for morphological analysis. The basic question is how we can account for the ordering in which the different types of morphemes have to appear in multiply complex words [22]. Affixation is a widespread process used in word formation, whereby an affix is prefixed, suffixed, infixed, or circum fixed to some input form. A prefix is an affix that precedes a base or seed[2]. For example in English, *un-, dis-* and *-ir* are prefixes in words untidy, dishonest and irregular respectively. A suffix is an affix which follows the base or the seed. In English *–s, -ed* and *-ize* in words dogs, kicked and nationalize are examples of suffixes. Hyphen (-) indicates the position of the attachments of affixes. Suffixes which are applied to every word of the same POS are called productive suffix [41]. For example, in English the *–ing* for verbs and the *-s* for nouns are gerund and plural indicator productive suffixes respectively.

Infixes are segmental strings that do not attach to the front or back of a word, but rather somewhere in the middle [31]. An infix is an affix which is added within the base or the seed.

[2] Stands for verbs without affixes.

Infixation of different vocalic patterns to the root of the seed results at different verb-forms with varied tense-moods, person, gender, number, etc.

Circumfixes are affixes that come in two parts. One attaches to the front of the word (prefixes), and the other to the back (suffixes). They are controversial because it is possible to analyze them as consisting of a prefix and a suffix that apply to a stem simultaneously [31]. They will bring different meaning from the seed or base word. Similarly Geʿez has such type of affixations (see Chapter Three).

Affixation (prefix, suffix and circumfix) and compounding are referred to as concatenative morphology since their mode of operation is that of concatenating roots, stems, and affixes [22]. However, a morphology which includes infix is said to be non-concatenatative morphology. Compounding is the joining of two or more base forms to form a new word. For instance, two nouns foot and ball can be fused to create football. Geʿez has non-concatinative morphology.

b. Clitic

A clitic is an element that behaves like an affix and a word. However, they are quite complicated in that they are also part of word formation. Unlike other morphological phenomena, clitics occur in a syntactic structure and their attachment to words isnʿt part of the word formation rules like the rest of morphology. We will expand on this in parts. "A clitic is an element that behaves like an affix and a word." For example, English has an obvious clitic, the „***'s***ʿʿ used to denote the possessive (sometimes known as the genitive grammatical case). Linguists call it an *enclitic* which means that it is a clitic that attaches to the right of the word, as does a suffix. However, they are quite complicated in that they are also part of word formation. The „***'s***ʿʿ is attached ("glued on") to the word or phrase to which the possessive is related [35]. But Geʿez doesnʿt have a clitic morpheme.

There is a null morpheme which is a morpheme that is realized by a phonologically null affix (an empty string of phonological segments). In simpler terms, a null morpheme is an "invisible" affix. It is also called zero morphemes. The null morpheme is represented as either the figure zero (0), or the empty set symbol Ø [24].

2.2.2 Morphotactics

Natural languages make use of a number of formal means for the formation of complex lexemes: compounding, affixation, reduplication, conversion, stem alternation (also referred to as internal modification), stress, and tone. In compounding two or more lexemes are combined into a new one. Cross linguistically, compounding is one of the most common means for word formation, in particular compounding in which one of the constituents is the head (so-called endocentric compounding). The English word football is a compound, consisting of the two lexemes foot and ball, of which the second functions as the head: a football is a particular kind of ball, not a kind of foot [22]. The morphemes of a word cannot occur in random order. In every language, there are well-defined ways to sequence the morphemes. The morphemes can be divided into a number of classes and the morpheme sequences are normally defined in terms of the sequence of classes. The order in which morphemes follow each other is strictly governed by a set of rules called morphotactics. It is the syntax of morphemes, the smallest linguistic meaning bearing unit of a word. It is concerned with the underlying rules that spell out the order in which affixes are attached with respect to the stem in a word [33]. In Ge‟ez, these rules play a very important role in word construction and derivation as the words are formed by a sequence of morphemes. Rules of morphotactics also serve to disambiguate the morphemes that occur in more than one class of morphemes. The analyzer uses these rules to identify the structure of words [36].

2.2.3 Morphological Classification

Languages may be classified according to the role and nature of morphology in each language [23]. A first dimension of classification is the index of synthesis: languages that do not make use of morphology are called analytic or isolating, languages with a lot of morphology are called synthetic. Hence, languages may be ranked on an index of synthesis. Traditionally, Chinese is referred to as an isolating language because it has almost no inflection. Chinese is not analytic in an absolute sense. The second index on which languages can be ranked is that of poly synthesis: some languages allow the incorporation of stems, leading to relatively complex words. The third dimension of classification is the index of fusion. In fusional languages, one morpheme may express more than one grammatical feature. Latin is such a language. Such languages can be contrasted with agglutinating languages in which each bound morpheme corresponds to one grammatical feature. For instance, case and number in Turkish are expressed by different suffixes, unlike what is the case for Latin. These three indices of morphological complexity are

useful in given a global characterization of the morphology of a language [22]. Like this Ge‟ez language is classified in second index on which languages are ranked in poly synthesis, since Ge‟ez has a complex morphology.

2.2.4 Types of Morphology

As Kumar [36] stated, morphology is traditionally classified into three main divisions: inflection, derivation, and compounding. Inflectional morphology deals with the formation of different forms in the paradigm of a lexeme. In it, words undergo a change in their form to express some grammatical functions but their syntactic category remains unchanged. Many inflectional features appear on words to express agreement (agreement in person, number, and gender) as well as to express case, aspect, mood, and tense.

The derivational morphology is concerned with "the creation of a new lexeme via affixation". In English, the process of word formation through derivation involves two types of affixation: pre-fixation, which means placing a morpheme before a word, e.g. *un-happy*; and suffixation, which means placing a morpheme after a word, e.g. *happi-ness*. Derivation poses a problem to translation in that not all derived words have straight forward compositional translation as derived words. In English, for example, the same meaning can be expressed by different affixes. Moreover, the same affix can have more than one meaning. This can be exemplified by the suffix *-er*. This suffix can be used to express the agent as in *player* and *singer*. But this is not the only meaning it can convey as it can describe instruments as in *mixer* and *cooker*. In this way the affix can have a range of equivalents in the target language and the attempt to have one-to-one correspondences for affixes will be greatly misguided.

Compounding morphology is the process of forming a new word through combining two or more words. Compounding is a process of word formation that involves combining complete word forms into a single compound form; *dogcatcher* is therefore a compound, because both *dog* and *catcher* are complete word forms in their own right before the compounding process has been applied, and are subsequently treated as one form. An important notion in compounding is the notion of head. A compound noun is divided into head and modifier or modifiers. For instance, the compound noun *watchtower* in which watch and tower can be represented as a head and modifier.

2.2.5 Prosodic Morphology

One of the classic linguistic issues is that of providing an account of the non-concatenative morphological system prevailing in most members of the Semitic language family [83]. In non-concatenative morphology, other formal means are involved in the creation of new morphological forms. In the case of internal modification a stem with a different form is created, for instance, by replacing a vowel pattern or a consonant pattern (or both) with another one. Vowel alternations are characteristic of a number of Indo-European languages; in Semitic languages verbal roots may appear in a number of different „binyanim", templates with specific patterns of consonants and vowels, sometimes in combination with a prefix [22]. As described in [83] Prosodic morphology was initiated by McCarthy in1981. The author noted the similarity in the behavior of vowels introduced into consonantal roots by morphological processes and phonological prosodies, such as tone spreading in Arabic language. He hypothesized that the verb in Arabic has elements arranged on three independent tiers at the underlying level of representation in the lexicon, the three tiers being the consonantal tier, also called the root tier, the skeletal tier and the vocalic melody tier. For example the verb ገበራ /gəbɨra/ in Ge"ez will have the structure shown in Figure 2.1.

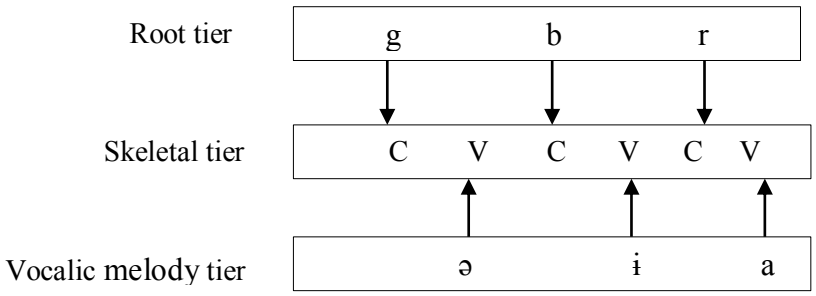

Figure 2.1: CV-based formation of the Ge'ez verb ገበራ /gəbira/

The above three tiers give the Ge"ez verb ገበራ */gəbɨra/* (she did). As McCarthy described in [83], these three tiers are linked together by association lines. The author noted the three universal conventions for making associations between the prosodic templates and the auto-segmentalized strings are as follows:

- The meaning of a verbal lexeme is signaled at the root-tier by the consonantal segments. Usually a verb has three consonants in its underived lexical entry in the lexicon. Thus the root /*g-b-r*/ in Ge"ez represents the lexeme *do*, which is realized by a variety of verb forms.

- The skeletal tier provides a canonical shape that is associated with a particular meaning or grammatical function. The template $C_1V_1C_2V_2C_3V_3$, for example, carries the grammatical meaning perfective.
- The vocalic melody tier provides information analogous to that carried in English by inflectional affixes like tense, aspect, number or derivational affixes. For example, in the above Geʿez verb /gəbɨra/ the vocalic pattern /ə-ɨ-a/ indicates that the tense of the verb is perfective. The last vowel /a/ indicates that the person is third person feminine and the number is singular.

2.2.6 Computational Morphology

Computational morphology deals with developing theories and techniques for computational analysis and synthesis of word forms. By computational analysis of morphology, one can extract any information encoded in a word and bring it out so that later layers of processing can make use of it [10]. It is the branch of computational linguistics concerned with word structure. Two kinds of processing are of interest: morphological analysis, by which a surface word form is analyzed into a lexical representation, consisting of the wordʿs component morphemes or grammatical features, and morphological generation, by which a lexical representation is converted to a surface word form [28].

2.2.7 Morphological Analyzer

Morphological analyzer is essential and foremost one for any type of Natural Language processing work [37]. It is a program for analyzing the morphology of an input word; the analyzer reads the inflected surface form of each word in a text and provides its lexical form [29]. Therefore, the role of morphological analyzer is very significant in the field of natural language processing (NLP) applications like machine translation (MT), information extraction (IE), information retrieval (IR), spell checker, lexicography, etc. So from a serious computational perspective the creation and availability of a morphological analyzer for a language is important. The morphological analyzer maps an inflected word into its stem, parts of speech and feature equations corresponding to inflectional information. The morphological structure of an agglutinative language is unique and capturing its complexity in a machine analyzable and generatable format is a challenging job [26].

Knowledge Required by Morphological Analyzer

Morphological analyzers should have basic knowledge which is common to most analyzers. As it is described in [38], to analyze existing or novel word forms in terms of their components three main types of knowledge are required by the analyzer:

- Knowledge about spelling or phonological changes consequent upon affixation (notice we are only dealing with isolated word forms);
- Knowledge about the syntactic or semantic properties of affixation (i.e., in flexional and derivational morphology), and
- Knowledge about the properties of the stored base forms of words (which in our case are always themselves words, rather than more abstract entities).

Furthermore, the analyzer should have knowledge of the syntactic and semantic properties of the vocalic patterns of the seed, particularly for Semitic words. This is because the vocalic patterns of the seed in Semitic languages can determine the inflectional and derivational category of a word. For example , the two Ge‟ez words **ቀደሰ** /*ḱədəsə*/ of vowel pattern 111 (all first order vowels[3]) and **ቀዳሲ** /*ḱədasi*/ (3psm) of vowel pattern 143 (first, fourth and third order vowels, respectively) are derived from the same root **ቅድስ** /*ḱds*/ but have different syntactical functions due to the different vocalic patterns they have.

2.3 Machine Learning

Machine learning deals with techniques that allow computers to automatically learn and make accurate predictions based on past observations. The major focus of machine learning is to extract information from data automatically, by using computational and statistical methods. Its techniques are being used for solving various tasks of Natural Language Processing. This includes speech recognition, morphological analysis, document categorization, document segmentation, part-of-speech tagging, and word-sense disambiguation, named entity recognition, parsing, machine translation and transliteration [36]. There are two main tasks involved in machine learning; learning/training and prediction. The system is given with a set of examples called training data. The primary goal is to automatically acquire effective and accurate model from the training data. The training data provides the domain knowledge, i.e., characteristics of

[3] See Section 2.6.1 for detail.

the domain from which the examples are drawn. This is a typical task for inductive learning and is usually called concept learning or learning from examples. The larger the amount of training data, usually the better the model will be. The second phase of machine learning is the prediction, wherein a set of inputs is mapped into the corresponding target values. The main challenge of machine learning is to create a model; with good prediction performance on the test data, i.e., model with good generalization on unknown data [36]. Machine learning is a promising alternative to obtain morphological rules. This method can avoid problems such as costly human labor, rule inconsistency and can provide additional statistical information which can be used in morphological analysis procedure. Based on information type employed in the machine learning task, we can usefully obtain two classes: unsupervised learning and supervised learning [41]. Recently, machine learning approaches are found to be dominating the Natural Language Processing field.

Machine learning is a branch of Artificial Intelligence (AI) concerned with the design of algorithms that learn from examples. Machine learning algorithms can be supervised or unsupervised. The input and corresponding output data are used in supervised learning. In unsupervised learning, only input samples are used. The goal of machine learning approach is to use the given examples and find out generalization and classification rules automatically. All the rules including complex spelling rules can be handled by this method. Morphological Analyzer based on machine learning approaches does not require any hand coded morphological rules. It only needs morphologically segmented corpora [36].

2.3.1 Unsupervised Learning

In supervised learning, the aim is to learn a mapping from the input to an output whose correct values are provided by a supervisor. In unsupervised learning, there is no such supervisor and we only have input data. The aim is to find the regularities in the input. There is a structure to the input space such that certain patterns occur more often than others, and we want to see what generally happens and what does not [42].

Unsupervised learning seems much harder: the goal is to have the computer learn how to do something that we don't tell it how to do! There are actually two approaches to unsupervised learning. The first approach is to teach the agent not by giving explicit categorizations, but by using some sort of reward system to indicate success. Note that this type of training will

generally fit into the decision problem framework because the goal is not to produce a classification but to make decisions that maximize rewards. This approach nicely generalizes to the real world, where agents might be rewarded for doing certain actions and punished for doing others. A second type of unsupervised learning is called clustering. In this type of learning, the goal is not to maximize a utility function, but simply to find similarities in the training data. The assumption is often that the clusters discovered will match reasonably well with an intuitive classification [40].

2.3.2 Supervised Learning

Supervised machine learning is the search for algorithms that reason from externally supplied instances to produce general hypotheses, which then make predictions about future instances. In other words, the goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown. Every instance in any dataset used by machine learning algorithms is represented using the same set of features. The features may be continuous, categorical or binary. If instances are given with known labels (the corresponding correct outputs) then the learning is called supervised, in contrast to unsupervised learning, where instances are unlabeled. [45]. It is fairly common in classification problems because the goal is often to get the computer to learn a classification system that we have created.

A supervised learning algorithm takes a set of training examples as input and produces a classifier or predictor as output. The set of training examples provides two kinds of information. First, it tells the learning algorithm the observed output values y_i for various input values x_i. Second, it gives some information about the probability distribution D(x). For example, in optical character recognition for ASCII characters, the training data provides information about the distribution of images of ASCII characters. Non-ASCII characters, such as Greek or Hebrew letters, would not appear in the training data [44].

Supervised learning is the most common technique for training neural networks and decision trees. Both of these techniques are highly dependent on the information given by the pre-determined classifications [40]. Machine learning approaches that use supervised learning

approach includes support vector machine (SVM), inductive logic programming (ILP), hidden Markov model (HMM) and memory based learning (MBL).

Support Vector Machine (SVM): It is a commonly used „eager learning" method. It finds the best separating hyper-plane between two sets of classes in such a way that the distance between the two classes is maximized. Using different kinds of kernel functions, the separating hyper-plane can be found in a space of higher dimensionality than the data itself. It performs especially well with sparse data [46].

Inductive Logic Programming (ILP): It is the process of learning a set of rules from instances (examples in a training set), or more generally speaking, creating a classifier that can be used to generalize from new instances [45]. Inductive logic programming allows for the introduction of domain knowledge in the form of predicate logic expressions as background theories to the learning system, as a way to constrain the search for a model that covers the training examples [30].

Hidden Markov Model (HMM): It is a popular statistical tool for modeling a wide range of time series data. It is a powerful tool for modeling generative sequences that can be characterized by an underlying process generating an observable sequence [47].

Memory Based Learning (MBL): It is a machine-learning method based on the idea that examples can be re-used directly in processing natural language problems. Training examples are stored without modification or abstraction. During the classification process, the most similar examples from the training data are located, and their class is used to classify the new example [30]. Other names that have been used for this kind of learning algorithm are instance-based, exemplar-based, example-based, case-based, analogical, and locally weighted learning or lazy learning based on the k-nearest neighbor classifier, is a supervised inductive learning algorithm for learning classification tasks [1, 48], or it is a class of inductive, supervised machine learning algorithms that learn by storing examples of a task in memory [2]. It treats a set of labeled (pre-classified) training instances as points in a multi-dimensional feature space, and stores them as such in an instance base in memory.

Memory-Based Learning (MBL) is a simple and robust machine-learning method which has been successfully applied to a wide range of NLP tasks. Some of the tasks, with references to some of the work in which memory-based learning has been applied are: Part-Of-Speech

Tagging e.g., [1], Grammatical Relation Finding, e.g., [49], Shallow Parsing (combining memory based tagging, chunking, PP finding, and grammatical relation finding), e.g., [30, 50], Morphological Analysis, e.g., [1, 2, 30, 61], Phoneme-To-Grapheme Conversion, e.g., [51], understanding User Utterances in Human-Computer Spoken Dialogue Systems, e.g., [52], Disfluency Detection in Spoken Language, e.g., [53], Pitch Accent Placement, e.g., [54], Semantic Role Labeling, e.g., [55], Word Sense Disambiguation, e.g., [56], Named Entity Recognition, e.g., [57], PP Attachment Disambiguation, e.g., [58], Co-Reference Resolution, e.g., [59], and Information Extraction, e.g., [60].

An example memory-based learning system is shown below in Figure 2.2. The encoder "γ"maps an input from the input space X into a set of addresses and the decoder "β" maps the set of activated memory locations into an output in the output space Y. The look-up table for MBL systems can be organized as hash tables, trees or full-search tables [8].

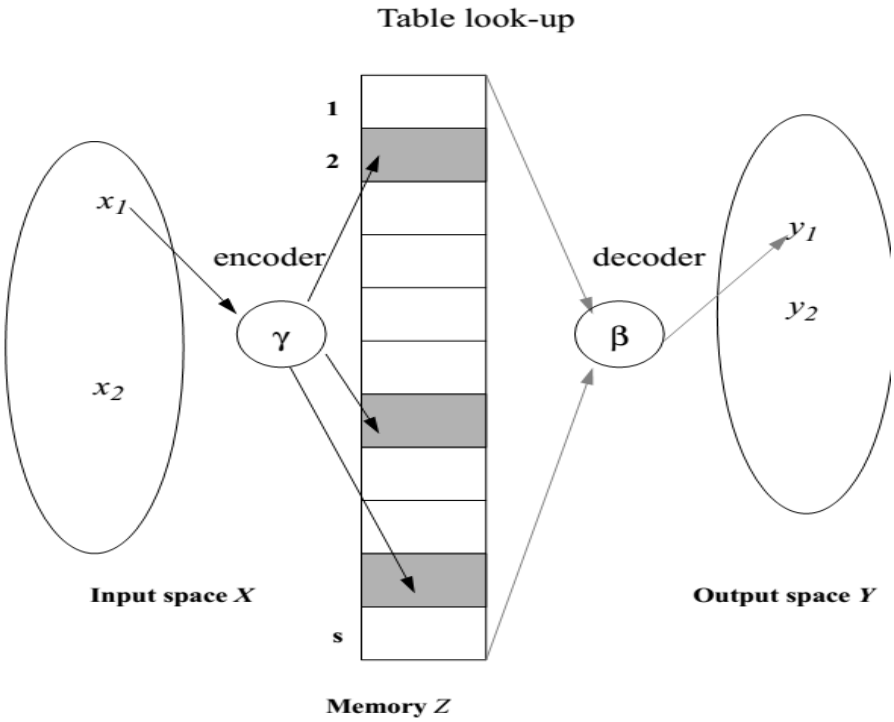

Figure 2.2: An Example Memory-Based Learning System

An instance consists of a fixed-length vector of **n** feature-value pairs, and an information field containing the classification of that particular feature-value vector. After the instance base is stored, new (test) instances are classified by matching them to all instances in the instance base,

and by calculating with each match the distance, given by a distance function (X, Y), between the new instance X and the memory instance Y [1,10].

Memory-based learning is founded on the hypothesis that performance in cognitive tasks is based on reasoning on the basis of similarity of new situations to stored representations of earlier experiences, rather than on the application of mental rules abstracted from earlier experiences (as in rule induction and rule-based processing). The General architecture of an MBL system which is shown in Figure 2.3, contains two components: a learning component which is memory-based (from which MBL borrows its name), and a performance component which is similarity-based. The learning component of MBL is memory-based as it involves adding training instances to memory (the instance base or case base); it is sometimes referred to as „lazy" as memory storage is done without abstraction or restructuring. In the performance component of an MBL system, the product of the learning component is used as a basis for mapping input to output; this usually takes the form of performing classification [10].

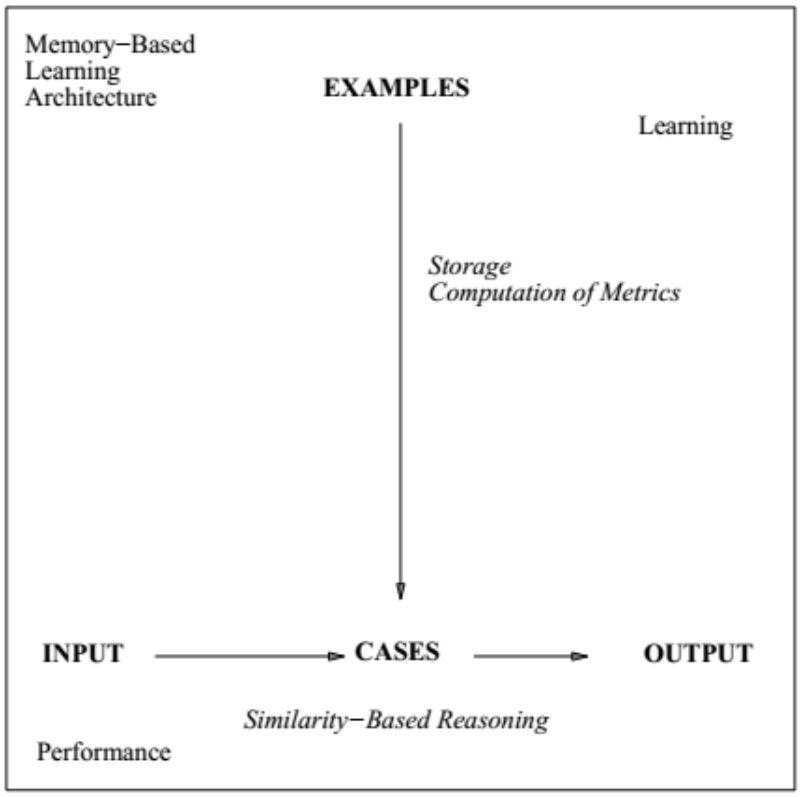

Figure 2.3: General Architecture of an MBL System (Adopted from [10])

Memory-based learning has been the primary machine learning method used in the present work. It has the following advantages in contrast to most other machine learning algorithms [1]: It

presupposes no more linguistic knowledge than explicitly present in the corpus used for training, i.e., it avoids a knowledge-acquisition bottleneck; Learning is automatic and fast; Processing is deterministic, non-recurrent (i.e., it does not retry analysis generation) and fast, and is only linearly related to the length of the word form being processed [2]. Because of these advantages, it is selected to test the morphological analysis of Geʿez verbs.

The main disadvantage of memory-based learning, compared to most rival approaches, is its slow classification speed. Its worst-case complexity of classification is O(nf), where **n** is the number of memorized examples, and f is the number of features; each new example needs to be compared against all of the memorized examples, each time involving a comparison of all features [63].

Another disadvantage of the memory-based learning approach that it shares with other discriminative classifiers is that its strength is in classification tasks with relatively low dimensionality in the class space. In the larger context of NLP tasks with structured output specifications, such as parsing or machine translation, it is widely recognized that discriminative classification alone is not enough to perform these global tasks, as the class spaces that would cover entire sequences, or large subsequences, would be too high-dimensional, thus too sparse to allow for sufficient amounts of examples per class. Even memory-based learning, with its insensitivity towards the number of classes, suffers directly from such sparseness [63].

2.4 Memory-Based Language Processing

Memory-Based Language Processing system is based on the idea that learning and processing/performance are two sides of the same coin [30]. The learning component is memory-based as it involves storing examples in memory without abstraction, selection, or restructuring. In the performance component of a MBLP system the stored examples are used as a basis for mapping input to output; input instances are classified by assigning them an output label. During classification, a previously unseen test instance is presented to the system. The class of this instance is determined on the basis of an extrapolation from the most similar example(s) in memory. There are different ways in which this approach can be operationalized [63].

Memory-based learning (MBL) is an approach to NLP based on a symbolic machine learning method. Memory-based learning is based on the assumptions that in learning a cognitive task from experience people do not extract rules or other abstract representations from their

experience, but reuse their memory of that experience directly. Memory based learning and problem solving incorporates two principles [64]: learning is the simple storage of a representation of experiences in memory, and solving a new problem is achieved by reusing solutions from similar previously solved problems. This simple idea has appeared in many variations in works in artificial intelligence, psychology, statistical pattern recognition, and linguistics [30]. Generally, the central claim of the MBL paradigm is that decisions about new facts are based on re-use of stored past experiences. Therefore, the main goal of an MBL model is to extrapolate the class of new exemplars based on their similarity to stored exemplars. As described in [30, 48, 62, 63], memory-based learning has a great future potential to analyzing Natural language processing (NLP) tasks from spelling error correction to machine translation and automatic extraction of knowledge from text.

All experiments with memory-based learning (e.g. Section 2.3.2) described here were carried out using the Tilburg Memory-based Learner, or TiMBL[4]. This software package is developed and maintained by the Induction of linguistic knowledge group at Tilburg University, and is a very efficient feature rich implementation. It can be run either as a "one-shot" command line classifier or as a server, and there is also a C++ API that makes it possible to integrate the memory-based learning functionality into other programs such as python [63]. For example, python-TiMBL is a Python extension module wrapping the full TiMBL C++ programming interface. With this module, all functionality is exposed through the C++ interface that is also available to Python scripts. Being able to access the API from Python greatly facilitates prototyping TiMBL-based applications [34].

As described in Chapter one, TiMBL is an open source software package implementing several memory-based learning algorithms such as IGTree, IBl, IBl-IG, TRIBl, TRIBL2, IB2, C4.5, etc., among which **IB2** and **TRIBL2** are used in this study**. IB2** is an implementation of *k*-nearest neighbor classification with feature weighting suitable for symbolic feature spaces, since it is an extension of IB1, and **TRIBL2** is a hybrid combination of IGTREE and IB1. IGTREE is a decision-tree approximation of IB1. These two algorithms rely on *k*-nearest neighbor (K-NN) classifier. All implemented algorithms have in common that they store some representation of the training set explicitly in memory. During testing, new cases are classified by extrapolation from the most similar stored cases [17, 30].

[4] Freely Available tool at Http://ILK.uvl.nl.

KNN:

The k-Nearest-Neighbors (k-NN) is a non-parametric classification method, which is simple but effective in many cases [65]. It is a case-based learning method, which keeps all the training data for classification. Being a lazy learning method prohibits it in many applications such as dynamic web mining for a large repository. For a data record **t** to be classified, its k-nearest neighbors are retrieved, and this forms a neighborhood of **t**. Majority voting among the data records in the neighborhood is usually used to decide the classification for **t** with or without consideration of distance-based weighting. However, to apply k-NN we need to choose an appropriate value for **k**, and the success of classification is very much dependent on this value. In a sense, the k-NN method is biased by **k**. There are many ways of choosing the **k** value, but a simple one is to run the algorithm many times with different k-values and choose the one with the best performance [66].

In the performance component of an MBL system, the product of the learning component is used as a basis for mapping input to output; this usually takes the form of performing classification. During classification, a previously unseen test example is presented to the system. The similarity between the new instance X and all examples Y in memory is computed using some distance metric Δ(X, Y). The extrapolation is done by assigning the most frequent category within the found set of most similar example(s) (the k-nearest neighbors) as the category of the new test example [10]. The distance between two instances is calculated using Equation 2.1 [10] (simple overlap metrics equation) which simply sum of the differences between the features [8, 10, 30]. A feature in morphological analysis is individual characters or classes that represent the complex morphological analysis.

$$\Delta(X, Y) = \sum_{i=1}^{n} \delta(xi, yi) \qquad (2.1)$$

$$\text{Where: } \delta(xi, yi) = \begin{cases} abs\left(\frac{xi-yi}{maxi-mini}\right) & \text{if numeric, else} \\ 0 & \text{if xi = yi} \\ 1 & \text{if xi} \neq yi \end{cases}$$

Where Δ(X, Y) is the distance between instances X and Y, represented by n features, and δ is the distance per feature.

The value of **k** refers to k-nearest distances rather than k-nearest examples. With **k** = 1, for instance, TiMBL"s nearest neighbor set can contain several instances that are equally distant to the test instance [10]. In order for k-NN to be less dependent on the choice of **k**, look at multiple sets of nearest neighbors rather than just one set of k-nearest neighbors [65]. K-NN kernel could therefore be called k-nearest distances classification [10].

Even if k-NN has its own drawbacks its low efficiency - being a lazy learning method prohibits it in many applications such as dynamic web mining for a large repository and its dependency on the selection of a "good value" for k [65]. Daelemans and Bosch stated in [63], the k-nearest neighbor classifier has a number of advantages that make memory-based learning the method of choice in certain particular situations, compared to other rival discriminative supervised machine-learning algorithms:

1. The basic version of the k-NN classifier that uses the overlap metric is insensitive to the number of class labels, both in terms of efficiency in training and in classification. This makes memory-based learning suited for classification tasks with very large numbers of classes, such as word prediction or machine translation;
2. Memory-based learning is able to reproduce the classification of training data flawlessly, as long as there are no identical training instances in memory with different class labels. This advantage, an important component of the "forgetting exceptions is harmful" tenet, is especially useful in NLP tasks in which much of the training data can be expected to recur in new data, such as in word pronunciation, where a typical lexicon used for training will already contain the pronunciation of approximately 95% of all words in a new text;
3. Memory-based learning allows for incremental learning at no cost, or with little cost if the similarity function uses weighting functions; this is practical in situations in which training examples become available over time, and the classifier needs to be retrained preferably with the availability of each new training example. Also, the algorithm is equally easily decremental, allowing for fast leave-one-out testing.
4. It has been shown that the 1-nearest neighbor classifier has an attractive error upper bound: as the amount of data approaches infinity, it is guaranteed to yield an error rate no worse than twice the Bayes error rate.

However, the impact of these non-parametric statistical methods (k-NN) on the development of systems for solving practical problems has remained limited because of a number of short comings: they were computationally expensive in storage and processing; intolerant of attribute noise and irrelevant attributes; sensitive to the similarity metric used; and the Euclidean distance metaphor for similarity breaks down with non-numeric and missing feature values [64]. To solve (some of) the problems with k-NN listed before, recently, the intuitive appeal of the nearest neighbor approach has been adopted in artificial intelligence in many variations on the basic nearest neighbor modeling idea, using names such as memory based learning which has IBl, IGTREE, IB2 and TRIBl2 algorithms implemented in TiMBL.

IBl:

The classification function of IB1 computes the similarity between a new instance and all stored instances, and returns the class label of the most similar instance. It uses information gain weights in the overlap function to define similarity. In IBl the instance base is reorganized (by compression rather than by pruning) in such a way that access to relevant instances is faster, and no generalization performance is lost [16].

In IBl, Instance-based learning algorithm, similarity of a new case to stored cases is used to find the nearest neighbors of the new case. The class of the new case is predicted on the basis of the classes associated with the nearest-neighbor cases and the frequency of their occurrences. All features are assigned the same relevance. The similarity function in lazy learning consisted simply of multiplying, when comparing two feature vectors, the similarity between the values for each feature with the corresponding information gain, or in case of features with different numbers of values the gain ratio, for that feature [67].

IGTREE:

IGTREE combines two algorithms: one for constructing decision trees, and one for retrieving classification information from these trees. During the construction of IGTREE decision trees, instances are stored as paths of connected nodes. All nodes contain a test (based on one of the features) and a class label (representing the default class at that node). Nodes are connected via arcs denoting the outcomes for the test (feature values). Information gain is used to determine the order in which instance features are used as tests in the tree. This order is fixed in advance, so the maximal depth of the tree is always equal to the number of features, and at the same level of the tree, all nodes have the same test [16].The reasoning behind this compression is that when the

computation of information gain points to one feature clearly being the most important in classification, search can be restricted to matching a test instance to those memory instances that have the same feature-value as the test instance at that feature. Instead of indexing all memory instances only once on this feature, the instance memory can then be optimized further by examining the second most important feature, followed by the third most important feature, etc. Again, compression is obtained as similar instances share partial paths [10, 16].

Because IGTREE makes a heuristic approximation of nearest neighbor search by a top down traversal of the tree in the order of feature relevance, we no longer need to store all the paths. The idea is that it is not necessary to fully store those feature-values of the instance that have lower Information Gain than those features which already fully disambiguate the instance classification [16, 67].

IB2:

The IBI algorithm stores all the instances of the training data in memory. This implies that all instances, even those that fall well within the borders of the target description, are saved in memory. Instances like these have no effect during classification, as they do not affect the generalization accuracy of the system; they only confirm the positions of the already existing target descriptions. It will therefore make sense to restrict the instances in memory to those that will have an effect (i.e., the instances that lie beyond or on the boundaries of the target description) on the classification of other instances [10]. This is the basis of the operation of the IB2 algorithm. The operation of IB2 is therefore identical to the operation of IBl, except that IB2 only saves misclassified instances in the training data (77). IB2 starts with an instance base containing only a small portion of the available training instances. This initial number of training instances in memory can be set by the user. IB2 then adds instances into memory only when they are misclassified by the k-NN algorithm, on the basis of the instances already in memory at that point.

Additional instances are added because they do not form part of the concept description of the instances in memory at that particular time. They are included to expand the concept description because we can assume that they represent a part of the instance space where there is not enough instances contained in memory for accurate classification [10].

IB2 does not lower or improve the generalization performance of IB1, but it lowers the computational burden of the system since there are less training instances stored in memory. The negative aspect of IB2 is that it is sensitive to noisy training data, since it has a strategy of only storing misclassified instances [77].

TRIBL2:

It was designed as a combination between IB1 and IGTREE with the aim to exploit the trade-off between the search speed of IGTREE and the generalization accuracy of' IB1. it do that by splitting the classification of new instances into a quick decision-tree (IGTREE) traversal based on the first (most important and most class-disambiguating) features, followed by a slow but relatively accurate k-NN (IB1) classification based on the remaining less important features. During search, the normal IGTREE search algorithm is used, until a feature having an IG value below the average IG for all the features is reached. This point represents the shift from IGTREE to IB [10].

TRIBL2 does not employ a fixed switching point like TRIBL algorithm. During the classification of an instance it continues to use IGTREE as long as it finds matching feature values in the weighting-governed feature ordering. It only reverts to IB1 classification when a mismatch is found. The reasoning behind this mismatch-based switching is that the switch to IB1 is only invoked when mismatching occurs, being the typical point in which IB1 can improve on decision-tree-style classification [10, 17].

A choice in algorithms is between using IB2 and TRIBL2. In the trade-off between generalization accuracy, incremental and efficiency, IB2 performs best when as much as possible of the training set was used for bootstrapping and only a few instances were incrementally added, which suggests that the strategy was not very successful or that the data set was too small. Its accuracy is depending on the parameter value of **–b**. IB2 storage requirement smaller than TRIBL2. This reduction in storage requirements is most dramatic when none of the training instances are noisy. Its accuracy will be decreases quickly as the level of noise increases. This occurs because noisy instances are, naturally, almost always misclassified. Since IB2 saves only a small percentage of the non-noisy training instances, its saved noisy instances are more often used to generate classification decisions. This is the huge disadvantage of the IB2 algorithm. In TRIBL2 algorithm, If the first mismatch occurs very early on in the tree, the probability that it will find a different classification than IB1 is low (zero when a mismatch

occurs on the first feature). The same goes for test instances that mismatch very late, because the more matches are found, the lower the probability that there is a combination of features in other parts of the tree that yields a smaller distance. Mismatches occurring in between will have the highest probability of causing different classification behavior. The more consecutive matches the faster classification will be, as each additional match decreases the size of the sub tree that has to be searched. Accuracy and speed seem to be inversely related [84]. Therefore, TRIBL2 will have the disadvantage of both algorithms depend on the mismatch point.

In sum, it can be said that in the trade-off between incremental editing and reduction storage memory requirements, the IB2 approach chooses to invest more time in organizing the instance base using Information Gain and compression, to obtain simplified and faster processing during classification, as compared to TRIBL2 algorithm [10, 16, 84]. Storing too many instances can result in large memory requirements and slow execution speed, and can cause an over sensitivity to noise. One advantage to an IB2 is that if instances are made available later, after training is complete, they can continue to be added according to the same criteria. Another advantage of IB2 algorithms is that it can be faster and use less storage during learning than non-incremental algorithms, since it can ignore some of the discarded instances when adding others. Thus instead of $O(n^2)$ time and O(n) storage during the learning phase, it can use O(n s) time and O(s) storage, where ***n*** is the number of training instances and ***s*** is the number of instances retained in the subset [85]. Therefore, Because of these advantages, IB2 algorithm techniques were selected for this study to design the morphological analysis of Ge‟ez verbs.

2.5 Morphology of Ge'ez Verbs

The main verbs in Ge‟ez are perfect and imperfect. Perfect is usually past or completed action. It includes past perfect, past continuous, past participle with the relative pronouns ዘ (of). The imperfect one is usually present, continuous and future action. The end of all perfect verbs is the first order while all imperfect verbs have in the end the six orders. This is under the pronoun ውእቱ (he). The other main verbs are subjunctive (with ከመ, ኀ, በ or without), imperative, infinitive and gerundive [3, 19]. As described in the first Chapter, verbs are the most important part of the Ge‟ez language which serves as a basis for almost all other POS categories. To illustrate the importance of the verb some scholars [74, 19] say that the verb is the language.

Geʿez is studied by local and foreign scholars. Among local scholars Aleka Kidane Wold Kifle [69], Like Hruyan Belay Mekonen [70], Memhr Dessie Keleb [19] and Memhr Zeradawit Adhana [74], and among foreign scholars Dillmann [70], Bender [3], Leslau [25], and Lambdin [71] are mentioned. The analysis of these scholars is used for this study to understande the language. Nowaday, the language doesn't have native speakers, and the main resources are books. Moreover, the linguistic work of Muluken Andualem [12] is also used to understand the way of classification of Geʿez verbs in the traditional schools of ቅኔ /*ḳ̇ine*/ found under the EOTC.

2.5.1 Ge'ez Writing System

A study of the Geʿez writing systems is essential to understanding the history of Ethiopia and the evolution and modern usage of the language. The only language in Ethiopia which has its own alphabets is Geʿez. Other languages[5] like Amharic and Tigrigna adopted these alphabets fully from Geʿez [3, 70]. An alphabet or *fidəl* of a language represents its sound. They are also called hohyat /ሆሀያት፡ in Amharic/. Geʿez sounds can be studied by dividing them into simple-sounds and complex-sounds [74]. Simple-sounds are represented with 182 alphabets. From these, seven of them represent vowel sounds: ኀ/ə/, ኁ/u/, ኂ/i/, ኃ/a/, ኄ/e/, ኅ/ɨ / and ኆ/o/. Whereas, others represents consonant sounds.

Prior to the modification performed by the first Ethiopian Bishop, *Abba Fremɨnatus*, the total number of Geʿez alphabets were twenty six each with vowel sound ኀ/ə/. After the modification, however, every consonant was combined with other six vowel sound alphabets to produce six more derived alphabets [4, 74]. For example, the combination of the original alphabet ሀ/*hə*/ with the six vowel sound alphabets ኁ/u/, ኂ/i/, ኃ/a/, ኄ/e/, ኅ/ɨ/ and ኆ/o/ yields six more derived alphabets as: ሁ/*hu*/, ሂ/*hi*/, ሃ/*ha*/,ሄ/*he*/, ህ/*hɨ*/ and ሆ/*ho*/ respectively.

Generally, Ge'ez has essentially 26 main syllographs or alphabets, all consonants, and each with six more derivations while the rest are essentially those with additional strokes and modifications added on to the main forms to indicate a vowel sound associated with it or to make aural adjustments in the basic consonant sound [75]. Hence, it has a total of 182 syllographs, with twenty six alphabets and seven vowels (26×7 = 182 alphabets). Doing the same derivation for

[5] Ethiopia has more than eighty languages.

every twenty six alphabets, a matrix of 26×7 size is produced. Each of the columns are labeled as **ግእዝ** /geʿez/ (first order), **ካእብ** /*kaʔb*/ (second-order), **ሳልስ** /*salɨs*/ (third-order), **ራብእ** /*rabɨʔ*/ (fourth-order), **ሓምስ** /*hamɨs*/ (fifth-order), **ሳድስ** /*sadɨs*/ (sixth-order), and **ሳብእ** /*sabɨʔ*/ (seventh-order) of alphabets. The orders represent the sound of each of the vowels. For example, if we take the verb **ኖለወ** /*noləwə*/ (he kept), the alphabet **ኖ**/*no*/ is read as **ንእ**/*no*/, where **እ**/o/ is a vowel with seven-order sound. Hence, it is sorted under the seventh column /seventh -order/ category. However, the alphabets **ለ** /*lə*/ and **ወ** /*wə*/ are read as **ልእ** /*lɨʔə*/ and **ውእ** /*wɨʔə*/ respectively, where **እ** /ə/ is a vowel with first -order sound. Hence, both are sorted under the first-order group.

The complex-sounds are represented with twenty letters. The alphabets are four in number. These are: **ኩ** /$k^{w}ə$/, **ጉ** /$g^{w}ə$/, **ቈ** /$k'^{w}ə$/, and **ኍ** /$x^{w}ə$/. Alphabets representing complex-sounds have only four derived alphabets, which are produced after the combination of the simple-sound alphabets **ከ** /*kə*/, **ገ** /*gə*/, **ቀ** /*k'ə*/ and **ኀ** /*xə*/ with the vowels **እ** /*ə*/ and **ኢ** /ɨ/ and the semi-vowels **ወ** /*wə*/ and **የ** /*yə*/ in different patterns. For instance, the alphabets **ጉ** /$g^{w}ə$/, **ጕ** /$g^{w}ɨ$/, **ጉ** /$g^{w}i$/, **ጓ** /$g^{w}a$/ and **ጔ** /$g^{w}e$/ are derived due to the combination of the vowel and the semi-vowel sounds with the consonant **ገ** /*gə*/ [74]. E.g: **ገ** /*gə*/ +**እ** /ə/ +**ወ** /*wə*/ = **ጉ** /$g^{w}ə$/

The Geʿez alphabets adopted and used in this study and detailed discussion on Geʿez writing system is found in [19, 69, 70, 74].

2.5.2 Formation of Ge'ez Verbs

As described in [70], a verb in Geʿez must pass through three stages of formation. These are: seed formation, tense-mood formation and person-gender-number formation.

Semitic language verbs are produced as the result of the intercalation of vowels and roots in a certain template. This fact is valid for Geʿez verbs too. Such intercalation of vowels and consonants in a given template produces what is called seed[6]. The produced seed grows up to a more natural verb form by attaching itself to tense -mood and person-gender-number marker affixes. Each seed has its own unique template. This essentially implies that verbs do have different templates. Their basic difference lies on the type of vowels participating during

[6] Root + vocalism + template.

the formation process. More precisely, the initial, medial, final alphabets of these seeds results at one true difference among verbs of Ge"ez. The following sub-sections describe thoroughly the aforesaid difference [4].

Initial Orders of Ge'ez Verbs

As discussed in Section 2.6.1, a consonant sound combines with each of the seven vowel sounds of the language to form /*ge'ez*/, /*kaʔb*/, /*salɨs*/, /*rabɨʔ*/, /*hamɨs*/, /*sadɨs*/, and /*sabɨʔ*/. Among these orders, /*kaʔb*/ and /*salɨs*/ do not occur in initial position of verbs. The other five can occur in initial position as recognized by many of the Ethiopian scholars [69, 74]. The system of these scholars takes CV -template as a basic unit to determine the category of verbs. Accordingly, the orders that occur in initial position include for example, **ለ** /lə/, **ላ** /la/, **ሌ** /le/, **ል** /lɨ/ and **ሎ** /lo/.

The Ethiopian scholars believe that the base for any verb is what they call „relative seed"[7] or „father seed"[8]. According to them, a verb is derived either from its father seed or its relative seed. The father form is an infinitive form, like **ፈቂድ** /*fəqid*/ (to like) [4].

As Kidane Wold Kifle described [69], a verb is formed only from a father seed, while modern scholars argue that the formation process begins from the root, which is purely consonantal. Muluken Andualem [12] argued that the „seed" and „root" are more or less the same. For him, the difference of these two terms is in the vowels of the seeds. Further, he argued that if we remove the vowels in a seed, we get the root. The researcher also agreed with the argument of Muluken and would prefer to call the prime mover of a verb as „root" and the immediate result of the intercalation of the root and the vowels as „seed" [4].

Terminal Orders of Ge'ez Verbs

Some of Ethiopian scholars believe that the terminal orders in Ge"ez verbs are only /*ge'ez*/ and /*hamɨs*/. According to them, the only verb that ends with /*hamɨs*/ is **ይቤ** /*yɨbe*/[9]. On the other hand, the verb /*yɨbe*/ is arguable that many scholars said differently about it. For example, Zeradawit said that /*yɨbe*/ has the perfective form **በሀለ** /*bəhalə*/ (he said) whereas Kidane

[7] It is **ዘመድ ዘርኣ** and it is the same as verbal noun.
[8] It is **አባት ዘርኣ** and it is the infinitive form of a verb which is considered as the base for other forms.
[9] It is a verb with a different morphology when compared to all other verbs of Ge"ez.

Wold Kifle argued that /*yɨbe*/ is the imperfective form of **ቤለ** /*belə*/ (he said) and its causative is **አበለ** /*ʔəbələ*/ (he caused someone to say).

Foreign scholars such as Lambdin, Dillmann, and Leslau consider /*yɨbe*/ as derived from the root /*b-h-l*/. Lambdin said that /*yɨbe*/ has **ብህለ** /*bɨhɨlə*/ (he said) as its perfective form, which seems to be the same as what Zəradawit and Dillmann have said. However, all scholars agreed that all perfective verbs end with a first-order consonant. Their only difference is on the terminal of /*yɨbe*/.

2.5.3 Classification of Ge'ez Verbs

Foreign scholars classify verbs based on the stem vowel and the root consonants, and classification of modern linguists is based on root consonants and patterns of vowels in stem formation [12]. However, classification of Ethiopian scholars on the basis of gemination, non-gemination, number and position of radicals as well as the positions of gutturals and semi-vowels in written verbs and the conjugation[10] pattern of the verbs.

The basic concepts of classification of verbs employed by all scholars are used in this study. As a result, the classification criteria is summarized into three: based on object indication, based on number of radicals and based on „heads" and „troops"[11].

Based on Object Indication

Based on Ethiopian scholars, Ge"ez verbs are grouped into three types derived from their tendency to indicate the object in a sentence. These are copula, transitive, and intransitive verbs [12, 19].

Transitive verbs occur with an object in a sentence. For example, the verbs **ቀደሰ** */ḱədəsə*/ (he consecrated), **ገበረ** /*gəbɨrə*/ (he did), **አንበረ** /*ʔənɨbərə*/ (he sat something), **ጸፍአ** /*ṣəfɨʔə*/ (he hit), **አቅረበ** /*ʔəḱɨrəbə*/ (he brought), **ባረከ** /*barəkə*/ (he blessed), **ሰበረ** /*səbərə*/ (he broke), and **ቀተለ** /*ḱətələ*/ (he killed) are transitive verbs. These all cannot be used without mentioning the action receiver (the object) of the verb. For example, we can have sentences like, **ሰበረ ዮሴፍ መንበረ ዝንቱ** (yosef broke this chair), where chair is the object which receives the action.

[10] It means to give various inflectional endings of a verb.
[11] See detailed in Section 2.6.3.

Intransitive verbs, conversely, are other types of verbs which occur without an object in a sentence. For example, the verbs ኀጸረ /*xəṣərə*/ (he become short), መጽአ /*məṣiʔə*/ (he came), ሞተ /*motə*/ (he died), ነበረ /*nəbərə*/ (he sat), ሖረ /*ħorə*/ (he went), ነደደ /*nədədə*/ (it was fired) and ቆመ /*ḱomə*/ (he stand) are intransitive verbs. For example, in the sentence **ሖረ ኢየሱስ እምገሊላ ኀበ ዮሐንስ** (Juses went from gelila to John), the verb **ሖረ** is used without an object.

The third type of verbs that occurs together with main verbs in a clause is called copula. **ውእቱ** /*wiʔtu*/ (Am, is, are, was, were), **ሀሎ** /*həlo*/ (He existed, presented, there was), **ኮነ** /*konə*/ (It was done), and **አኀዘ** /*ʔəxəzə*/ (He held, he began, he started) are examples of Copula verbs.

Conjugation means to give various inflectional endings of a verb, i.e. voice, mood, tense, number and person. For example, in the sentences **ኮነ ይገብር ግብረ ብረት፡** (he was wont to fabricate implements of brass) and **ወሀሎ ዮሐንስ ያጠምቅ በገዳም** (John baptizingin the wilderness), **ሀሎ** /*həlo*/ and **ኮነ** /*konə*/ is copula verbs.

Based on Number of Radicals of a Root

Ge‟ez verbs can be also classified into three depending on the number of alphabets their roots have as tri-literal, quadri-literal, and multi-literal [25, 70]. Tri-literal roots are composed of three consonants and are those which best answer to the Semitic root forming tendency [70]. Table 2.1 shows the formation of verbs from tri -radical roots.

Table 2.1: Formation of Verbs from Tri-Radical Root

Root	Template	Verb	Gloss
ḱ-t-l	$C_1əC_2əC_3ə$	/*ḱətələ*/	He killed
	$C_1əC_2əC_3u$	/*ḱətəlu*/	They killed (2ppm)
	$C_1əC_2aC_3i$	/*ḱətali*/	Somebody who killed (male)
ħ-ṣ-ṣ	$C_1əC_2əC_3ə$	/*ħəṣəṣə*/	He becomes small
	$C_1əC_2əC_3a$	/*ħəṣəṣa*/	They become small (2ppf)
	$C_{1i}C_2uC_3$	/*ħiṣuṣ*/	Somebody who is small (male)

As shown in Table 2.1, a root usually consists of three radicals. However, roots possessing only two consonants like ሐጸ /*ħəṣə*/ (he becomes small), ጎየ /*goyə*/, (he flee), መነ /*mənə*/ (he excommunicated himself from the world), ሐመ /*ħəmə*/ (he ill), ሐገ /*ħəgə*/ (he formulated a

law), and **ነደ** /*nədə*/ (he/it was fired). Nevertheless, these roots are products of the later time influences on the language. The original nature of these roots was tri-radical as **ሐጸጸ**, **ጎየየ**, **መነነ**, **ሐመመ**, **ሐገገ**, and **ነደደ** respectively [70]. Dillmann continue to argue that every original root of Ge'ez comprises of three firm letters. The researcher also believes that bi-radical roots are only found in the latest literatures and are results of the influence of the later writers who are not native to the language. For example, the original shapes of the verbs **ቦአ** /*boʔa*/, **ሖረ** /*ħorə*/, **ቆመ** /*ḳomə*/, and **ኤለ** /*ʔelə*/ were **በወአ**[12] /*bəwəʔə*/, **ሐወረ** /*ħəwərə*/, **ቀወመ** /*ḳəwəmə*/, and **አየለ** /*ʔəyələ*/ respectively. Furthermore, the verbal noun **ሐዋርያ** /*ħəwarɨya*/ (apostle) could only be a derivation of a verb only if the verb is /*ħəwərə*/, not /*ħorə*/. As the result, bi-radical[13] roots are not at all common and natural to Ge"ez language [4].

Quadri-literal[14] roots, on the other hand, are roots with four radicals. For example, the verbs **ተንበለ** /*tənɨbələ*/ (he begged), **ማህረከ** /*mahɨrəkə*/ (he took captive), **አእመረ** /*ʔəʔmərə*/ (he knew)and **ደንገፀ** /*dənɨgəʂə*/ (he become terrified) are verbs of Quadri-literal roots. The occurrence of such verbs is too small compared to the tri-radical verbs. Multi-literals are also roots of more than four letters. For example, the root of the verbs **አመንተወ** /*ʔəmənɨtəwə*/, **ደለቅለቀ** /*dələḳɨləḳə*/ (he made something to shake) are a quni-literal (five-radical) root. Nevertheless, roots with more than four letters are not common in Ge"ez at all [70].

Even if the tri-radical verbs are dominant in the language [73], the focus of this study is on all type of verbs formulated from any radical roots.

Based on Heads and Troops

Ethiopian and foreign scholars agreed on the grouping of Ge"ez verbs as „heads" and „troops" taking their CV-template as a basic unit. Heads are those verbs which can represent other verbs of their type. Troops are any other verbs which can be sorted under either of the identified heads because of their pattern (template) similarity with the head. However, these scholars do not agree on the number of heads.

[12] When a vowel appears independently in a verb, it is counted as a radical.
[13] Roots with two radicals.
[14] Roots with four radicals.

The Ethiopian scholars differ each other in the number of heads. According to *Zeradawit* Adhana [74], the heads are eight in number. These are **ቀተለ** /ḱətələ/ (he killed), **ቀደሰ** /ḱəddəsə/ (he consecrated), **ገብረ** /gəbɨrə/ (He did), **ተንበለ** /tənɨbələ/ (he begged), **ባረከ** /barəkə/ (he blessed), **ኤለ** /ʔelə/ (he rounded), **ክህለ** /kɨhɨlə/ (he abled), and **ዖደ** /ʕodə/ (he wake circular). However, According to Kidane Wold Kifle [69], there are only seven heads. These are **ቀተለ** /ḱətələ/ (he killed), **ቀደሰ** /ḱəddəsə/ (he consecrated), **ባረከ** /barəkə/ (he blessed), **ዴገነ** /degənə/ (he followed), **ኖለወ** /noləwə/ (he kept), **ደንገፀ** /dənɨgəḍə/ (he terrified), and **ማህረከ** /mahɨrəkə/ (he took captive). Memhr Dessie Keleb [19], as he stated, he believe that verbs classified either in eight or seven. Verbs which classified into eight heads include **ቀተለ** /ḱətələ/, **ቀደሰ** /ḱəddəsə/, **ባረከ** /barəkə/, **አእመረ** /ʔəʔmərə/ (he knew), **ቆመ** /ḱomə/ (he stand), **ሣመ** /śemə/ (he sat), **ብህለ** /bɨhɨlə/ (he said) and **ገብረ** /gəbɨrə/. But on the other hand, he also argued that the heads are eleven: **ቀተለ** /ḱətələ/, **ቀደሰ** /ḱəddəsə/, **ባረከ** /barəkə/, **ደንገፀ** /dənɨgəḍə/, **ማህረከ** /mahɨrəkə/, **ዴገነ** /degənə/ (he followed), **ኖለወ** /noləwə/, **ቆመ** /ḱomə/, **ሣመ** /śemə/, **ገብረ** /gəbɨrə/ and **ብህለ** /bɨhɨlə/. In the classification process, the main issue which arguable, is on the number of the heads but not on the sameness of the verbs considered as heads.

The foreign scholars have also classified the verbs differently from each other and from the Ethiopian scholars. Ludolf [72] has recognized three verbs as heads: **ገብረ** /gəbɨrə/ (he did), **ገበረ** /gəbərə/ (he did) and **ጋብረ** /gabɨrə/ (he did). While, Dillmann [70] believe that the heads are four: **ገብረ** /gəbɨrə/ (he did), **ነገረ** /nəgərə/ (he told), **ፈጸመ** /fəṣəmə/ (he completed), and **ባረከ** /barəkə/ (he blessed). However, Lambdine [71] has said that the heads are two: **ቀተለ** /ḱətələ/ and **ቀደሰ** /ḱədəsə/.

The classification given by Memhr Dessie Keleb [19], with eight categories is adopted for this study. Table 2.2 illustrates the „head" verbs and their CV-template.

Table 2.2: „Heads" of Ge"ez Verbs Together with Their Template

No	Head	Template	Number of radicals and vocalic patterns	Gloss
1	ቀተለ /*ḱətələ*/	$C_1əC_2əC_3ə$	Tri-radical, 1-1-1 order of vocalic pattern	He killed
2	ቀደሰ /*ḱəddəsə*/	$C_1əC_2C_2əC_3ə$	Tri-radical, 1-1-1 order of vocalic pattern and geminated at its middle alphabet.	He consecrated
3	ባረከ /*bərəkə*/	$C_1aC_2əC_3ə$	Tri-radical, 4-1-1 order of vocalic pattern	He blessed
4	አእመረ /ʔəʔmərə/	$C_1əC_2əC_3ə$	Quadric-radical, 1-6-1-1 order of vocalic pattern	He knew
5	ቆመ /*ḱomə*/	$C_1oC_2ə$	Bi-radical, 7-1 order of vocalic pattern	He stand
6	ሴመ/*śemə*/	$C_1eC_2ə$	Bi-radical, 5-1 order of vocalic pattern	He sat
7	ብህለ/*bɨhɨlə*/	$C_1ɨC_2ɨC_3ə$	Tri-radical, 6-6-1 order of vocalic pattern	He said
8	ገብረ /*gəbɨrə*/	$C_1əC_2ɨC_3ə$	Tri-radical, 1-6-1 order of vocalic pattern	He did

As Muluken Andualem described in [12], classification of verbs into eight heads is also supported by some scholars of traditional school of ቅኔ /*ḱɨne*/ like washera[15]. The focus of this study lies on these head verbs and their troops including both transitive and intransitive verbs.

2.5.4 Affixation in Ge'ez Verbs

Affixation realizes the two next stage of verb formation: person-gender-number and tense-mood formation. Therefore, verbs attain their maximum growth level through affixation and are verbs of this stage which are the most common and natural verbs of Ge"ez [4]. It has all types of the affixes: infixes, prefixes, suffixes and circumfixes. Infixation is realized that the internal modification of the seed, which means forming seeds from root via intercalating of the vocalic patterns with the consonants. The next sections discuss about the details of the remaining three affixes.

Pre-fixation

As described in [4], prefixes of Ge"ez language can be categorized into two as negation marker and affirmative marker prefixes. The affirmative marker prefixes can further be divided into two as verbal-stem-marker and person-marker prefixes. A given affirmative verb can have a prefix merely from among the affirmative marker prefixes. Only negative verbs can have the possibility of having two prefixes: the negation marker being followed by one of the positive prefixes. That is, negation marker of a verb should always be at the very beginning of the verb. Following is the detail of the three types of prefixes in Ge"ez verbs.

[15] Washera is a traditional schools of Qene under the EOTC.

a. Verbal-Stem-Marker Prefixes: This type of prefixes are attached to the front of the base-stem form of a verb to form four more derived stems namely causative, causative-reciprocal, reflexive and reciprocal. For instance: the base-stem verb **ቀተለ** /*ḱətələ*/ (he killed) has **አቅተለ** /*ʔəḱɨtələ*/ (he caused somebody to killed), **አስተቃተለ** /*ʔəsɨtəḱatələ*/ (he caused others to be killed each other), **ተቀትለ** /*təḱətɨlə*/ (he is killed by), **ተቃተለ** /*təḱatələ*/ (he get killed with somebody) and as its causative, causative-reciprocal, reflexive and reciprocal stem forms respectively.

b. Person Marker Prefixes: These are **አ-** /ʔə-/, **ነ-** /nə-/, **ተ-** /tə-/ and **የ-** /yə-/[16] which are prefixes attached in front of the indicative, subjunctive and jussive mood verbs to indicate the subject of the action of the verb. They can also be used in their second, fourth and sixth orders depending on the formation of the verb. With some exceptional cases like in verbs with gutturals, the usual occurrences of these prefixes are in their sixth order forms. For example, the verb **ቀተለ** /*ḱətələ*/ (he killed) has its indicative, subjunctive and jussive forms as **ይቀትል** /*yɨḱətɨl*/ (he will kill), **ይቅትል** /*yɨḱɨtɨl*/ (he must kill) and **ይቅትል** /*yɨḱɨtɨl*/ (for him to kill). These all are person indicators. Table 2.3 illustrates which of the above discussed prefixes indicate which person.

Table 2.3: Prefixes of Indicative, Subjunctive, and Jussive Verbs

Indicative	Subjunctive	Jussive	Person referred
አ- /ʔə-/	**አ-**/ʔə-/	**አ-** /*ʔə-*/	**አነ** /*ʔənə*/ (I, 1psn[17])
ነ- /nə-/	**ነ-**/nə-/	**ነ-** /*nə-*/	**ንሕነ** /*nɨhɨnə*/ (We, 1ppn[18])
ተ- /tə-/	-	**ተ-** /*tə-*/	**አንተ** /*ʔənɨtə*/ (You, 2psm)
ተ- /tə-/	-	**ተ-** /*tə-*/	**አንትሙ** /*ʔənɨtɨmu*/ (You, 2ppm)
ተ- /tə-/	-	**ተ-** /*tə-*/	**አንቲ** /*ʔənɨti*/ (You, 2psf)
ተ- /tə-/	-	**ተ-** /*tə-*/	**አንትን** /*ʔənɨtɨn*/ (You, 2ppf)
ተ- /tə-/	**ተ-** /tə-/	**ተ-** /*tə-*/	**ይእቲ** /*yɨʔti*/ (She, 3psf)
የ- /yə-/	**የ-** /yə-/	**የ-** /*yə-*/	**ውእቱ** /*wɨʔtu*/ (He, 3psm)
የ- /yə-/	**የ-** /yə-/	**የ-** /*yə-*/	**ውእቶሙ** /*wɨʔtomu*/ (They, 3ppm)
የ- /yə-/	**የ-** /yə-/	**የ-** /*yə-*/	**ውእቶን** /*wɨʔton*/ (They, 3ppf)

Source: Adopted from [4].

[16] According to the Ethiopian scholars, these prefixes are called as **አስራው ግሳት**.
[17] First person singular neutral.
[18] First person plural neutral.

As shown in Table 2.3, the prefixes /ʔə-/ and /nə-/ are unique to the 1psn and 1ppn respectively. Whereas, the prefix /tə-/ is common to all second persons and to the 3psf. On the other hand, the prefix /yə-/ is working only to the 3psm, 3ppm and 3ppf. The hyphen (-) with each of prefixes above indicates the place of attachment. From the table (the last column) can be also noticed that Geʿez, as many of the other Semitic languages, has ten pronouns.

c. Negation Marker Prefix: - In Geʿez, affirmative verb is converted into its negative form by attaching the prefix **ኢ-** /ʔi-/ in front of it. For example, the intransitive verbs **ተቃተለ** /təḳatələ/ (he get killed with somebody) and **ተቀትለ** /təḳətɨlə/ (he is killed by) and their transitive form **ቀተለ** /ḳətələ/ (he killed) can be prefixed with /ʔi-/ to produce their corresponding negative verbs **ኢተቃተለ** /ʔitəḳatələ/ (he is not get killed with somebody), **ኢተቀትለ** /ʔitəḳətələ/ (he is not killed by), **ኢቀተለ** /ʔiḳətələ/ (he didn't kill) respectively.

Generally, all the possible prefixes of Ge'ez verbs along with their grammatical function are summarized as shown in Table 2.4.

Table 2.4: List of Geʿez verbs prefixes along with their syntactical functions

Prefixes	Syntactical function
አ- /ʔə-/	Causative Stem Marker
እ- /ʔ-/	Indicative, Subjunctive and Jussive Moods Marker
አስተ- /ʔəsitə-/	Causative-Reciprocal Stem Marker
ኢ- /ʔi-/	Negation Marker
ና- /na-/	Causative Stem Marker
ናስተ- /nasitə-/	Causative-Reciprocal Stem Marker
ን- /ni-/	Indicative, Subjunctive and Jussive Moods Marker
ንት- /nit-/	Reciprocal and Reflexive Stems Marker
ታ- /ta-/	Causative Stem Marker
ታስተ- /tasitə-/	Causative-Reciprocal Stem Marker
ተ- /tə-/	Reflexive and Reciprocal Stems Marker
ት- /ti-/	Indicative, Subjunctive and Jussive Moods Marker
ትት- /tit-/	Reciprocal and Reflexive Stems Marker
ያ- /ya-/	Causative Stem Marker
ያስተ- /yasitə-/	Causative-Reciprocal Stem Marker
ይ- /yi-/	Indicative, Subjunctive and Jussive Moods Marker
ይት- /yit-/	Reciprocal and Reflexive Stems Marker

Source: Adopted from [19].

Suffixation

Geʿez verbs canʿt be told without having affixes which indicate at least the subject (doer of the action) done by the verb or both the subject and object of the verbs. This is done through affixation, particularly through suffixation and circumfixation. The subjects of the perfective, imperative and gerundive verbs are indicated through suffixation. On the other hand, the subjects of the indicative, subjunctive and jussive verbs are indicated through circum fixation. In Geʿez, verbal suffixes are two types. These are ,,subject markerʿʿ[19] and ,,object markerʿʿ[20] suffixes [4].

The subject marker suffix (SMS) indicates subject (doer) of the action. For example, if we take the seed **ቀተል** /*ḱətəl*/, certain suffixes can be attached to it to produce various inflected surface verbs as shown in the Table 2.5.

Table 2.5: List of Geʿez Verbal Subject Marker Suffixes along with Indicated Persons

Seed		Subject Marker Suffix	Inflected Verbs	Subject Indicated
ቀተል /*ḱətəl*/	+	-ኩ /-*ku*/	ቀተልኩ /*ḱətəlɨku*/	I
		-ነ /-*nə*/	ቀተልነ /*ḱətəlɨnə*/	We
		-ክ /-*kə*/	ቀተልክ /*ḱətəlɨkə*/	You (2psm)
		-ክሙ /-*kɨmu*/	ቀተልክሙ/*ḱətəlɨkɨmu*/	You (2ppm)
		-ኪ /-*ki*/	ቀተልኪ /*ḱətəlɨki*/	You (2psf)
		-ክን /-*kɨn*/	ቀተልክን /*ḱətəlɨkɨn*/	You (2ppf)
		-አ /-*ə*/	ቀተለ /*ḱətələ*/	He
		-ኡ /-*u*/	ቀተሉ /*ḱətəlu*/	They (3ppm)
		-አት /-*ət*/	ቀተለት /*ḱətələt*/	She
		-ኣ /-*a*/	ቀተላ /*ḱətəla*/	They (3ppf)
		-የ /-*yə*/	ቀቲልየ /*ḱətilɨyə*/	I
		-ኦ /-o/	ቀቲሎ /*ḱətilo*/	He
		-ሙ /-mu/	ቀቲሎሙ /*ḱətilomu*/	They (3ppm)
		-ን /-n/	ቀቲሎን /*ḱətilon*/	They (3ppf)
		-እ/-*ɨ*/	ቅትል /*ḱɨtɨl*/	You (2psm)
		-ኡ/-u/	ቅትሉ /*ḱɨtɨlu*/	You (2ppm)
		-ኢ/-i/	ቅትሊ /*ḱɨtɨli*/	You (2psf)

Source: Adapted from [19].

Perfective, imperative and gerundive verbs cannot be told without having one of these suffixes as subject marker. The first ten subject markers in the table are suffixes mainly for perfective verbs. The next four suffixes are used for gerundive verbs. The last most three suffixes, however,

[19] It is a morpheme that indicates a subject of a verb.
[20] It is a morpheme that indicates an object of a verb.

are used for imperative verbs. Either of these suffixes is attached to a seed to produce a verb of the abovementioned tense-mood types. The plus (+) sign in the table symbolizes the „process of concatenation" of the morphemes to produce the surface verbs indicated in the fourth column [4].

The object marker suffix (OMS), on the other hand, indicates the object (action receiver) of the action of the verb. OMS cannot be directly attached to a seed. We show the concatenation process of object markers to a verb to produce many more inflected verbs by taking the verb /*ḱətələ*/ in the table below. Table 2.6 shows the concatenation process of object markers to a verb to produce many more inflected verbs by taking the verb **ቀተለ** /*ḱətələ*/ [4].

Table 2.6: List of Ge'ez Verbal Object Marker Suffixes along with Indicated Persons

Verb	Object Marker Suffix	Object Indicated
ቀተለ /*ḱətələ*/	-ኒ /-*ki*/	Me
	-ነ /-*nə*/	Us
	-ከ /-*kə*/	You (2psm)
	-ክሙ /-*kɨmu*/	You (2ppm)
	-ኪ /-*ki*/	You (2psf)
	-ክን /-*kɨn*/	You (2ppf)
	-ክዎ/-*kɨwo*/, -ሁ/-*hu*/, -ኦ/-*o*/, -ዎ/-*wo*/, -ዮ/-*yo*/	Him
	-ዎሙ/-*womu*/, -ክሙ/-*kɨmu*/, -ሙ/-*mu*/, -ሆሙ/-*homu*/,-ዮሙ /-*yomu*/	Them (3ppm)
	-ዋ /-*wa*/,-ሃ /-*ha*/, -ኣ /-*a*/,-ያ/-*ya*/	Her
	-ዎን /-*won*/, -ሆን /*hon*/, -ን /-*n*/, -ዮን /-*yon*/	Them (3ppf)

Source: Adapted from [4].

As shown in Table 2.6 the first persons and second persons each have only one object marker suffix whereas third persons do have more than one. In this case, which object marker suffix belongs to which person depends on the subject of the verb. This results in a strict rule of suffixation that object marker suffixes can only be attached to a verb right after the subject marker. Subject marker is mandatory for any verb whereas object marker is optional. As the result, verbs choose an object marker suffix based on the subject marker already attached to them.

Two central rules of suffixation that govern the concatenation process of morphemes to produce surface verbs are [4]:

1. Seed + subject-marker = surface verb (only with SMS)
2. Seed + subject-marker + object- indicator = surface verb (with both SMS and OMS)

Therefore, we can have two forms of surface verb forms, where the one is with subject marker only and the other is with both subject and object markers. This is because object markers are optional to be attached to a verb.

Example:

Rule-1: seed + subject-marker = surface verb (only with SMS)

ቀ፡ተለ /*ḱətələ*/ + ኩ፡ /ku/ = ቀ፡ተልኩ፡ /*ḱətələku*/ (I killed)

Rule-2: seed + subject-marker + object-marker = surface verb (with both SMS and OMS)

ቀ፡ተለ /*ḱətələ*/ + ኩ፡/*-ku*/ +ሞሙ፡/*-womu*/ = ቀ፡ተልኩሞሙ፡ /*ḱətəlikɨwomu*/ (I killed them)

In this case, the subject marker /*ku*/ points that the subject „I" whereas the object marker /womu/ indicates the object 'Them'. Hence, the verb /*ḱətəlikɨwomu*/ would able to indicate both the killer (subject) and the one who is killed (object).

Circumfixes

In Ge"ez, circumfixes are the subject markers of indicative, subjunctive and jussive seeds. For such type of verbs too, the object markers are attached immediately after the circumfix of the seed. Hence, subject markers can be suffixes or circumfixes based on the type of the tense-mood of the verb. The object markers, if they appeared, however, are always suffixes to all verbs. Table 2.7 illustrates the circumfixes of troops of ቀደሰ /*ḱədəsə*/ category.

Table 2.7: List of Ge"ez Verbal Subject Marker Circum Fixes along with Their Moods

Circum fixes	Subject indicated	Used in moods
እ-እ /*ʔ-ʔ*/	I	Indicative, subjunctive and jussive
ት-እ /*t-ʔ*/	You (2psm) & She (3ps)	Indicative, subjunctive and jussive
ት-ኢ /*tɨ-ʔi*/	You (2psf)	Indicative and jussive
ት-ኡ /*tɨ-ʔu*/	You (2ppm)	Indicative and jussive
ት-ኣ /*tɨ-ʔa*/	You (2ppf)	Indicative and jussive
ን-እ /*nɨ-ʔ*/	We (1pp)	Indicative, subjunctive and jussive
ይ-እ /*yɨ-ʔ*/	He (3psm)	Indicative, subjunctive and jussive
ይ-ኡ /*yɨ-ʔu*/	You (3ppm)	Indicative, subjunctive and jussive
ይ-ኣ /*yɨ-ʔa*/	You (3ppf)	Indicative, subjunctive and jussive

Source: Adopted from [19].

The verb analysis process is done based on the way morphemes concatenate to produce surface verbs. In general, the type of morphemes and the way they concatenate to produce surface verbs can be summarized as in Table 2.8.

Table 2.8: Possible Type of Morphemes Concatenated to Form a Verb in Geʿez

No	Possible morphemes of a verb	Example
1	[NegPref][PosPre][21][Seed][SMS][OMS]	**[ኢ][አስተ][ፋቀድ][ክም][ዎመ·]**
2	[NegPref][PosPre][Seed][SMS]	**[ኢ][አስተ][ፋቀድ][ክመ·]**
3	[PosPre][Seed][SMS][OMS]	**[አስተ][ፋቀድ][ክም][ዎመ·]**
4	[NegPref][Seed][SMS][OMS]	**[ኢ][ፈቀድ][ክም][ዎመ·]**
5	[NegPref][Seed][SMS][OMS]	**[አስተ][ፋቀድ][ክመ·]**
6	[NegPref][Seed][SMS]	**[ኢ][ፈቀድ][ክመ·]**
7	[NegPref][PreCirc][22][Seed][SufCirc][23][OMS]	**[ኢ][ት][ፍቅድ][ዎመ·]**
8	[PreCirc][Seed][SufCirc][OMS]	**[ት][ፍቅድ][ዎመ·]**
9	[NegPref][PreCirc][Seed][SufCirc]	**[ኢ][ት][ፍቅዱ]**

Source: Adopted from [4].

2.5.5 The Conjugation Patterns and Stems of Verbs

The final stage of verb formation process: the tense-mood formation is discussed here. This is the third and final stage next to the person-gender-number stage of formation. The reason behind this is the fact that tense-mood formation cannot be done without personal formation [70].

Concerning to the tense-mood formation, Dillmann said that Geʿez, like the other Semitic languages, proceeds from the twofold, and not from the threefold division of time. These are the finished state (perfect) and unfinished state (imperfect) tenses [19, 70]. According to them, the following three conditions belongs to the imperfect tense: action happening in the present, something which is only to be realized in the future and something which is only thought of and willed, which may or must be realized. And therefore, the imperfect is the source of the formation of the so-called moods of the verb, through which the conditions of will and necessity are expressed [4]. In connection to this, Däse and Dillmann suggested that the name 'imperfect' is a general name for the indicative, subjunctive and imperative moods [19, 70]. This idea is used in this study while naming of the tense –mood[24] types.

[21] NegPref = Negation Prefix, PosPre = Positive Prefix.
[22] PreCirc = Prefix Circumfix (i.e., the prefix part of the circumfix).
[23] SufCirc = Suffix Circumfix (i.e., the suffix part of the circumfix).
[24] It is used to refere to all conjugation patterns (Table 3.9)

According to the Ethiopian scholars, the types of verbs described as perfective, indicative, subjunctive and imperative verbs are those which are called **ዓብይት አናቅፅ** */ʕabɨyɨt ʔənaḱɨş/* to mean verbs which can close the idea of a sentence independently without seeking a help of other verbs. In addition to these verbs, the scholars identify **ንኡሳን አናቅጽ** */nɨʔusan ʔənaḱɨş/* which cannot close a sentence unless other verbs are added to them. The second types of verbs are similar to the English auxiliary verbs. Table 2.9 summarizes the above discussion.

Table 2.9: List of Geʿez Tenses and Moods According to Both Scholars

Tense-mood Identified by Ethiopian Scholars				**Tense-mood Identified by Foreign Scholars**	
No	**Tense-mood**	**Category**		**Tense-mood**	**Category**
1	ቀዳማይ /ኃላፊ አንቀጽ (perfective)	ዓብይት አናቅጽ */ʕabɨyɨt ʔənaḱɨş/*		Perfective	Perfective
2	ካልኣይ /ትንቢት አንቀጽ (indicative)			Indicative	Imperfective
3	ሣልሣይ /ትእዛዝ (subjunctive)			Subjunctive	
4	የቅርብ ትእዛዝ አንቀጽ (Imperative)			Imperative	
5	ምክንያታዊ አንቀጽ (Jussive)	ንኡሳን አናቅጽ */nɨʔusan ʔənaḱɨş/*		-	-
6	ቦዝ አንቀጽ (Gerundive)				
7	አርእስት አንቀጽ (Infinitive)				
8	ሳቢ ዘር አንቀጽ (*sabizər ʔənɨḱəş*)				
9	ቅጽል አንቀጽ (*ḱɨşɨl ʔənɨḱəş*)				

Source: Adopted from [19].

As shown in Table 2.9, the tense-mood identified by the Ethiopian scholars are the same as that of those identified by the foreign scholars except that the Ethiopians identified extra moods which are categorized as **ንኡሳን አናቅጽ** */nɨʔusan ʔənaḱɨş/*. This study adopts the identification of tense-mood types according to the Ethiopian scholars. But, the last two moods in this table are left undone in this study. Because they have a derivational morphology as they are verbal noun and verbal adjective respectively.

The imperative verbs are command verbs as that of the subjunctive verbs except that imperatives have only second person subject markers. Likewise, the difference between subjunctive and jussive verbs lies on the fact that the jussive verbs with second person subject markers have the prefix **ተ** */tə/* whereas the subjunctive verbs with second person subject markers do not have prefixes and, hence, have got a special name: imperative verbs [4].

The perfective, imperative, gerundive and infinitive verbs do have SMS as a subject marker whereas indicative, subjunctive and jussive have subject markers being circumfixed to their seeds. However, concerning to the object marker all tense-mood types except gerundive and infinitive verbs do have OMS being suffixed after their subject markers (suffixed or circumfixed to them). Infinitive verbs can have SMS but not OMS.

Infinitive verbs can be spoken with and without having SMS. Those which don‟t have SMS are common to all pronouns. For instance: **አቅትሎት** /*ʔəḱɨtɨlot*/ is an infinitive verb which doesn't have SMS.

The infinitive verbs **አፍቅዶትየ** /*ʔəfɨḱɨdotɨyə*/ and **ኖልዊየ** /*noliwiyə*/, however, have an SMS **የ**/*yə*/ which indicates that the subject is 'I'. Generally, infinitive verbs in Ge'ez may not indicate the doer of the action of the verb.

The following sub-sections discuss the conjugation patterns and number of stems of verbs based on the aforementioned tense-mood types by taken the **ቀተለ** /*ḱətələ*/ category verbs as an example.

The Conjugation Patterns of /ḱətələ/ Category Verbs

As described in [4], According to the Ethiopian scholars, both transitive and intransitive troops of /*ḱətələ*/ have seven conjugation patterns (excluding the other two which are not verbs) based on the seven tense-mood identifications presented above. Conjugation patterns are the basic templates through which the surface verbs of /*ḱətələ*/ category are formulated. The templates are effectively used during the declaration process carried out to find the inflected surface forms of the verb. Table 2.10 depicts the basic conjugation patterns and their corresponding templates by taking the verb **ቀተለ** /*ḱətələ*/ as an example.

Table 2.10: Basic Conjugation Patterns of a Root with Their Templates and Vocalic Patterns

No	Root	Vocalic pattern (VP)	CV-Template	Conjugation patterns	Tense-mood
1	ቀ-ተ-ለ /ḱ-t-l/	111	$C_1əC_2əC_3ə$	ቀተለ /ḱətələ/	Perfective
2		166	$yC_1əC_2ɨC_3ɨ$	ይቀትል /yɨḱətɨl/	Indicative
3		666	$yC_1ɨC_2ɨC_3ɨ$	ይቅትል/yɨḱɨtɨl/	Subjunctive
4		666	$C_1ɨC_2ɨC_3ɨ$	ቅትል /ḱɨtɨl/	Imperative
5		666	yC1ɨC2ɨC$_3$ɨ	ይቅትል /yɨḱɨtɨl/	Jussive
6		137	$C_1əC_2iC_3o$	ቀቲሎ /ḱətilo/	Gerundive
7		136	$C_1əC_2iC_3ɨ$	ቀቲል /ḱətil/	Infinitive
		137	$C_1əC_2iC_3oC_4ɨ$	ቀቲሎት /ḱətilot/	

The conjugation patterns (column-IV) are produced after the intercalation of the root with vowels of varies patterns (column-V) in the templates given in column-III of the table. Almost all /*ḱətələ*/ category verbs share the template, conjugation pattern and vocalic patterns depicted in this table.

The Stems of Verbs

According to Ethiopian scholars, stems are also called **አዕማድ** /*ʔəʕimad*/ 'pillars', or shaft that support the roof of building. They are pillars or bases of verbs that support the conjugations of verbs. These scholars believe that Geʿez has five stem patterns which all are independent of each other. These are perfective, causative, causative-reciprocal, reflexive and reciprocal stems [19, 74].

However, the foreign scholars have recognized different number of stems. The stems of Ludolf are only four in number: perfective, causative, reflexive passive and causative -reflexive stems. Stems of Dillmann are four. As to him, there are no stems for reciprocity like Ludolf and unlike the local scholars. Lambdin, on the other hand, recognized six stems by adding one more stem which, according to him, is called causative of reflexive passive.

The types of stems recognized by the Ethiopian scholars are considered in this study. These are perfective, causative, causative-reciprocal, reflexive and reciprocal stems. Table 2.11 shows the stems of the aforementioned seven tense-moods of **ቀ·ተ·ለ** /*ḱətələ*/.

Table 3.11: The Five Stem Types for Each of the Seven Tense Mood Types of Geʿez Verbs

No	Tense-Mood	The Five Stem Types				
		ገቢር (Perfectve)	አንብሮ (Causative)	ተንብሮ (Reflexive)	ተጋብሮ (Reciprocal)	አስተጋብሮ(Causative –Reciprocal)
1	Perfective	ቀተለ /ḱətələ/	አቅተለ /ʔəḱɨtələ/	ተቀትለ /təḱətɨlə/	ተቃተለ /təḱatələ/	አስተቃተለ /ʔəsɨtəḱatələ/
2	Indicative	ይቀትል /yɨḱɨtɨl/	ያቀትል /yaḱətɨl/	ይትቀተል /yɨtɨḱətəl/	ይትቃተል /yɨtɨḱatəl/	ያስተቃትል /yasɨtəḱatɨl/
3	Subjunctive	ይቅትል /yɨḱɨtɨl/	ያቅትል /yaḱɨtɨl/	ይትቀተል /yɨtɨḱətəl/	ይትቃተል /yɨtɨḱatəl/	ያስተቃትል /yasɨtəḱatɨl/
4	Imperative	ቅተል /ḱɨtɨl/	አቅትል /ʔəḱɨtɨl/	ተቀተል /təḱətəl/	ተቃተል /təḱatəl/	አስተቃትል /ʔəsɨtəḱatɨl/
5	Jussive	ይቅትል /yɨḱɨtɨl/	ያቅትል /yaḱɨtɨl/	ይትቀተል /yɨtɨḱətəl/	ይትቃተል /yɨtɨḱatəl/	ያስተቃትል /yasɨtəḱatɨl/
6	Gerundive	ቀቲሎ /ḱətilo/	አቅቲሎ /ʔəḱɨtilo/	ተቀቲሎ /təḱətilo/	ተቃቲሎ /təḱatilo/	አስተቃቲሎ /ʔəsɨtəḱatilo/
7	Infinitive(a)	ቀቲል /ḱətil/	አቅትሎ /ʔəḱɨtɨlo/	አስተቃትሎ/ʔəsɨtəḱatɨlo/	ተቀትሎ /təḱətɨlo/	ተቃትሎ /təḱatɨlo/
	Infinitive(b)	ቀቲሎት /ḱətilot/	አቅትሎት /ʔəḱɨtɨlot/	አስተቃትሎት/ʔəsɨtəḱatɨlot/	ተቀትሎት /təḱətɨlot/	ተቃትሎት /təḱatɨlot/

Source: Adapted from [4].

As can be seen from Table 2.11 each of the tense-moods has five stems. Nevertheless, as most Ethiopian scholars suggest, only transitive verbs can have complete stem forms (five stems) whereas intransitive verbs have less than five [69, 74]. In connection to this, the name of the first tense-mood (perfective) seems to be confusing with the name of the first stem (perfective). To avoid this confusion, we use the name „base stem"[25] for the first column (**ገቢር** */gəbir/*) stem [4]. Verbs which belong to the base, causative and causative-reciprocal stems are transitive verbs. On the other hand, verbs which belong to the reflexive passive and reciprocal stems are intransitive verbs. Furthermore, we call each of the seven rows stems as perfective stems category, indicative stems category, subjunctive stems category, imperative stems category, jussive stems category, gerundive stems category and infinitive stems category respectively [4].

[25] We named the first stem type as base stem for it is the base of all the rest stems.

2.6 Summary

In this Chapter we discussed about the various concepts and terminologies of morphology, the approach which is used in this study for the development of Geʿez verb morphology and the verb formation process in Geʿez verbs, by dividing the formation process into three stages as seed formation, person-gender-number formation and tense-mood formation. From this one can be know how much the Geʿez language complex, i.e. the morphological complexity of Geʿez.

Chapter Three: Related Work

3.1 Introduction

Morphological analysis has been investigated by many researchers and a number of algorithms have been proposed and implemented through the years [5]. This is due to its importance for other language engineering applications ranging from spelling error correction to machine translation [2]. Memory-based learning has been successfully applied to morphological analysis and part-of-speech tagging in Western and Eastern-European languages, like Dutch and Swedish [1] and Arabic, and Swahili languages. It was also applied for Amharic morphological analysis. Except Arabic language the end results of these languages were motivating and promising to develop a morphological analyzer for other languages.

3.2 Morphological Analysis for Ge'ez Language

To the best knowledge of the researcher, Ge"ez language morphological analyzer has been investigated by Desta Berihu. For some languages it is difficult to address and analyze all the morphological features of the language. Especially a language like Ge"ez, it is difficult to address all features since it is a complex inflected language. Due to this, the author limited to Ge"ez verbs in specific **ቀተለ** /*ḱətələ*/ (he killed) category verb forms which includes **ሐተተ** /*ħətətə*/, **መሐለ** /*məħələ*/, **ወሀበ** /*wəhəbə*/, **ነቀወ** /*nəḱəwə*/, **ወረደ** /*wərədə*/, **ወደየ** /*wədəyə*/, **ጸገየ** /*ṣəgəyə*/ verbs. The author has analyzed the morphology of Ge"ez verbs using rule-based approaches specifically CV-based and Two-Level Morphology (TLM) to design the model and to implement the prototype of the analyzer. The prototype was tested with verbs which are extracted manually by domain experts from all twenty seven New Testament books of the Ethiopic Version Bible. The author found an accuracy of 92.05% at feature level and of 73.98% at verb level.

Morphological analysis using memory based learning was applied in Amharic, Swedish, Dutch, Arabic, and Swahili languages. Here we discussed the experimental results, techniques and tools researchers used for each language. We also discussed the morphological analysis of Amharic in detail using other methods. Amharic Language has many similar features with Ge"ez than others. Therefore, we discussed different researches conducted so far in Amharic language morphology using different approaches including the tools they used and their evaluation results.

3.3 Morphological Analysis for Amharic Language

This section taken from [61], Martha Yifru [15] comes up with Morpheme-based language models. The study does not have a direct relation with the current approach. However, the investigation of morphological analysis of Amharic language help to understand the morphological variations and used as a reference for this study. The model was trained on automatically and manually segmented data using the standard research institute Language modeling toolkit (SRILM)[26]. As stated there, the quality of the model has been assessed in terms of perplexity, the probability they assign to the test set, and the improvement in word recognition accuracy obtained as a result of using them in a speech recognition task. The author thoroughly assessed the morphology of Amharic language and the different ways of modeling a language. It was also compared the word base and morpheme based language modeling which the later performs well in minimizing out of vocabulary (OOV). The results show that the morpheme-based language models trained on manually segmented data always have a higher quality. The work is limited to statistical language modeling using morpheme based.

Tesfaye Bayu [39] tried to design and develop a morphological analysis system for Amharic using a Linguistica[27]. Linguistica is a freely available software package which is designed for analysis of morphology. As the author stated the package requires a large corpus ranging from 5,000 to 1,000,000 words. However, the author used a 5,236 words corpus, the smallest recommended corpus size, to train the morphology of Amharic using Linguistica. Lingustica2001 is designed for concatenative morphology of languages but it is not appropriate for non-concatenative morphology of languages like Amharic. Therefore, the author developed a stem internal morphological parser (called Amharic Stems Morphological Analyzer ASMA) based on the theory of auto segmental morphology to analyze the stems identified by Linguistica into their constituent root and pattern morphemes. However, it is also stated that the author could not integrate the stem analyzer to that of Linguistica because of time limitations. The author also suggests that to get an efficient morphological analyzer another research need to be conducted using different approach.

[26] It is a collection of C++ libraries and executable programs.
[27] Linguistica is a program designed to explore the unsupervised learning of natural language.

Wondwossen Mulugeta and Gasser [43] developed a supervised machine learning approach to morphological analysis of Amharic verbs using Inductive Logic Programming (ILP). From this approach hypothesis is drawn from background knowledge and examples, and represent them in a form of logic programming. The hypothesis and background knowledge are used to evaluate new instances. The authors tested it using CLOG[28] which is a Prolog based ILP system. The system learns first order decision lists or rules on the basis of positive example only. The rule which is applied in the CLOG has left and right clauses. The right hand side of the rule is a condition and the left hand side of it is the conclusion.

A rule to be true all conditions must possess true value. They manually labeled 206 simple verbs for training, and tried to train Amharic verb stem extraction, internal alternation and roots separately in the CLOG. Their training results 19 and 108 root and stem template extraction rules respectively and by combining those rules they tested and found an accuracy of 86.99%. They prepared a training set which contains 1,784 Amharic verbs. Their investigation was limited for simple Amharic verbs, particularly for subject markers both prefixes and suffixes. As they stated ILP is suitable for simple morphological analysis languages like English. However, if sophisticated background predicates and more examples are prepared it is also applicable for complex languages. Hence, as they said it is possible to make study for complex Amharic verbs and for other word categories of the language. On the other hand, CLOG is not capable of learning rules from incomplete examples. Therefore, ILP with CLOG needs a complete example to learn rules of morphology to analyze new instances of a given word. In reality this is not applicable because it is impossible to expect every one of the thousands of morpheme combinations to appear in the training set particularly for agglutinative languages like Amharic and Ge"ez [43].

Gasser [85] developed a morphological analyzer and generator for the three Ethiopian languages namely Amharic, Oromo and Tigriyna. The Analyzer and generator focus on verbs of the three languages and including nouns for Amharic. The analyzer segments words into their component morphemes and assign grammatical morphemes to grammatical categories and lexical morphemes to lexemes. For example, given an Amharic word, HornMorpho[29], returns the root,

[28] It is freely available ILP system at: http://www-users.cs.york.ac.uk/suresh/CLOG.html).
[29] It is freely available at http://www.cs.indiana.edu/~gasser/Research/software.html.

the lemma and a grammatical analysis in the form of a feature structure description for each possible analysis. On the other hand, the morphological generation performs the reverse process.

The author derived lexicons for the three languages from online dictionaries. For Amharic, as the author stated the lexicon is derived from the Amharic-English dictionary of Aklilu which contains 1,851 verb roots and 6,471 noun stems. For Oromo the lexicon of verb and noun roots are extracted from the dictionaries of Gragg and Bitima that contains 4,112 verb roots and 10,659 nouns stems. Likewise, for Tigrinya, the lexicon of verb roots is derived from Efrem Zacarias‟ around 602 verb roots.

As it is described in Gasser [85] the system is implemented using finite state transducer. The author developed two versions of functions; the anal_word and anal_file which analyzes single and multiple words respectively. Based on those functions the author evaluated the HornMorph with 200 Amharic and Tigriyna verbs, and 200 Amharic nous and adjectives. Each word was selected randomly. The anal_word function was run on those words and the results were evaluated by a human reader who is familiar with the languages. The program made 8 (96% accuracy) and 2 (99% accuracy) errors for Tigriyna and Amharic verbs respectively. For Amharic nouns and adjectives it made 9 errors (95.5% accuracy). Similarly the generator was evaluated only with Amharic and Tigriyna verbs. Amharic nous and adjectives were not tested in this regard. The drawback of the system is when there are multiple analyses it does not provide information about which analyses are more likely than others. Hence, it doesn‟t control ties. As the author said to handle such issues currently they are working on extending the weighted FST framework to accommodate probabilities as well as feature structures on transitions so that when there is multiple analysis results are ranked according to their weights.

Kibur Lisanu [21] studied morphological synthesis of Amharic perfective verb forms. The author developed a prototype using rule based and artificial neural network approaches. The rule-based approach generates all the roots successfully whereas the neural network predicts the type of roots in the test dataset with an accuracy of 81.48%. The author classified the verbs in to three categories as type A, B and C. Each type tested separately using the neural networks and achieved an accuracy of 80%, 25% and 100% respectively. As stated in the paper the developed a system named Amharic Morphological Synthesizer (AMS). The author only considered Amharic verbs particularly perfective forms.

Mesfin Abate [61] developed a morphological analysis of Amharic language using MBL. The corpus contains 1022 words of which 181 and 841 are nouns and verbs respectively. As the author stated the number of instances extracted from nouns and adjectives are 1356 and verbs are 6719 which accounts a total of 8075 instances. Within these instances, 26 different class labels occur. The author conducted the experiment on the handcrafted dataset on TiMBL 6.4.4 with default settings and by tuning the different parameters. The authoe used Ubuntu 12.04.3 LTS as working environment and ucto 0.3.5 for text tokenization, geany2.1, a C++ editor, for code writing, GCC 4.6.4 compiler for compiling the source and Net Beans 7.0.1 for automatic feature extraction and data preprocessing. Based on the default values of parameters of each algorithm (IB1 and IGTREE), experimental results show that the generalization performance of IB1 and IGTREE algorithms are 92.02% and 76.27% respectively. Similarly, with optimized parameters 93.59% and 82.26% results were obtained for IBl and IGTREE respectively. The author also used the default parameter settings for IB1 algorithm with leave-one-out and 10 fold cross validation and the generalization accuracy of the model obtained were 93.3% and 92.02% respectively. This shows that generalization performance of the learned model is almost the same by the two evaluation methods. The 10 fold evaluation results of IB1 and IGTREE on optimized parameter settings are 93.59% and 82.26% respectively. However, as the author described on the paper, the performance of IB1 on optimized parameter settings is raised to 96.4% when evaluated with LOOCV. This happens since LOO evaluation uses all the dataset for training except one which helps the model to learn better. From this we can conclude that IBl has a better performance than IGTREE even if it consumes more memory and time than IGTREE.

3.4 MBL Morphological Analysis for Other Languages

Bosch and Daelemans [2] have been developed a memory based learning morphological analysis for Dutch. Their model is named Memory-Based Morphological Analysis (MBMA). They also developed a memory based learner engine called TiMBL. They tested their approach with morphology of Dutch which is a productive and a bit more inflected than English. The processes of Dutch morphology include inflection, derivation, and compounding. Compounding in Dutch is concatenative. It doesn‟t have a non-concatenative nature like Amharic, Ge‟ez and Arabic languages. Inflection of verbs, adjectives, and nouns is mostly achieved by suffixation.

They drew their instances from the CELEX[30] lexical database. CELEX contains a large lexical database of Dutch word- forms, and features a full morphological analysis for 247,415 words. They took each word form and its associated analysis, and created task instances using a windowing method. Windowing transforms each word form into as many instances as it has letters. In this way they generated an instance base of 2,727,462 instances. Within these instances, 2422 different class labels occur.

The memory-based learning algorithm they used within MBMA is IBl-IG, an extension of IBl. IBl-IG constructs a database of instances in memory during learning. New instances are classified by IBl-IG by matching them to all instances in the instance base, and calculating with each match the distance between the new instance X and the memory instance Y. During their experiment they used a method that gives a good estimate of the generalization performance of an algorithm on a given instance base, which is 10-fold cross- validation. This method generates on the basis of an instance base 10 subsequent partitioning into a training set (90%) and a test set (10%), resulting in 10 experiments. Based on this, they performed 10-fold cross validation experiments in an experimental matrix in which MBMA is applied to the full instance base. The experimental analysis shows that when they discarded word-class information, type of inflection, and spelling changes, the precision and recall of basic segment types becomes over 94%. Precision is the percentage of morphemes predicted by MBMA that is actually a morpheme in the target analysis; recall is the percentage of morphemes in the target analysis that are also predicted by MBMA.

Bosch, Marsi and Soudi [1] have studied a memory based morphological analysis and part of speech tagging for Arabic. Arabic appears to be a special challenge for data-driven approaches. It is a Semitic language with a non-concatenative morphology. Their data source completely relies on the Arabic Treebank[31] 1(ATB1), version 3.0 more, specifically the "after Treebank" POS-tagged data. The analysis of Treebank is organized in rule-based, and basically consists of three steps. First, all possible segmentations of the input string in terms of prefixes (0 to 4 characters long), stems (at least one character), and suffixes (0 to 6 characters long) are generated. Next, dictionary lookup is used to determine if these segments are existing morphological units. Finally, the number of analyses is further reduced by checking for the mutual compatibility of

[30] Database for English, Dutch and German word features.
[31] Parsed corups for professional or academical researcher.

prefix+stem, stem+suffix, and prefix+stem + suffixin, three compatibility tables. The resulting analyses have to a certain extent been manually checked. These separate lexicons were created for training and testing material. The lexical entries in a lexicon were converted to instances suitable to memory based learning of the mapping from words to their analyses [2]. Instances consist of a sequence of feature values and a corresponding class, representing a potentially complex morphological operation. In their initial experiments, to test the feasibility of their approach, they first trained and tested on the full data set. TiMBL is used with its default settings (overlap distance function, gain-ratio feature weighting, k = 1). Rather than evaluating on the accuracy of predicting the complex classes, they evaluate on the complete correctness of all reconstructed analyses, in terms of precision, recall, and F-score. They got results in a near perfect recall (97.5%). The precision, however, is much lower (52.5%), indicating a substantial amount of analysis over generation. With an F-score of only 68.1, they are clearly not able to reproduce the training data perfectly. Next, to improve the precision and recall, they split the data in to different training and testing parts. The k-NN classifier is again used with its default settings. They tried to improve the recall to 99.9% for known words. However, the radical growth of percentage in recall is at the expense of the precision which in turn lowers its result to 15.6%. From this we can say that the morphological analysis of Arabic using MBL was not interesting and promising.

Atelach Alemu and Forsbom [5] developed a morphological analysis of Swedish words as a classification problem using memory-based learning (TiMBL). The aim is to find citation forms (or meaningful parts) of words rather than a detailed morphological analysis. They manually annotated 4,189 words for their main segmentation and morphology type (inflection, derivation and compounding). From this annotation, information on morphological features around the segment boundary was automatically extracted, and each unique feature concatenation was given a class label. For training, they used the top 1,000 lemmas from a frequency and dispersion ranked Swedish base lemma vocabulary pool called Stockholm-Umea Corpus, which contains one million word balanced corpora, and their corresponding word forms.

The full training set had 675 classes, of which 470 classes were used only once in the small training set. In this condition the learning problem was somewhat very hard. Because of this additionally they used a trimmed training set which contains 3,719 instances and 205 classes.

They used the default and optimized algorithm setting IBl. IBl algorithm accounts 59.2% accuracy on the full set and 67.98% accuracy on the trimmed set. A comparative analysis was done with three existing morphological analyzers (SWTWOL, Lexware and Sara) for Swedish. Even though they have used a relatively small training set, their approach gave a good performance with 85.6% accuracy for inflections than others.

Depauw and Deschryver [68] developed a morphological analyzer for Swahili language using character and Syllable-based memory based learning. They named their system Memory-Based Swahili Morphological Analyzer (MBSMA). Their experiment relies on HCS (Helsinki Corpus of Swahili) which contains 9.7 million words. They automatically extracted a morphological database of 97, 000 entries from the HCS. Since HCS has been lemmatized using an automated method, quite a few erroneous and inconsistent lemmatization can be observed in the data. Because of this they randomly extracted 10% of the data from the morphological database and had it manually annotated according to the prefix-root-suffix. Their primary evaluation metric is the standard approach of word-error rate (WER). It expresses the accuracy on the word-level, i.e., how many words have not been completely correctly segmented and lemmatized. In other words, the lower the WER, the system will be better. The experimental result shows 11.6% and 13.3% accuracy for MBSMA-s and MBSMA-c respectively. This means that it shows lower WER, i.e., a better system. The precision and recall of the experiment on quantification of retrievability of the lemmatizer also shows that 92.4% and 83.4% accuracy respectively. According to their result, MBSMA-s has a better accuracy than MBSMA-c. Moreover, they compared the accuracy of these two systems to other (Morfessor and SALAMA) morphological segmentation and lemmatization of Swahili with the same datasets. The experimental result of the comparison shows that the two (MBSMA-s and MBSMA-c) systems yield a better accuracy than others.

3.5 Summary

Amharic morphological analysis was tried by various scholars as described earlier. Wondwossen Mulugeta and Gassar‟s work was limited to Amharic simple verbs particularly subject marker. On the other hand, Tesfay Bayu was tried to cover most part of the morphological features of Amharic language. The author tested the analysis into two different approaches which needs another effort to integrate as stated in the paper [39]. The only existing morphological analyzer is HornMorpho. We have tested it with some sample data and it analyzed correctly except some

word classes and words. Moreover, it does not able to handle words that have more than one analysis. Mesfin Abate was tried to model and test machine learning perspective of morphological analysis that handles inflection and derivation of each word classes of Amharic. As the author stated in the paper [61] the morphological analyzer which developed, handles ambiguity and covers all the morphological features of the language. Desta Berihu was tried to develop a morphological analysis for Geʿez verbs. The author limited to Geʿez verbs specific to ቀተለ /*ḱətələ*/ (he killed) category verb forms. The current study tried to model and test machine learning perspective of morphological analysis which includes all category verb forms.

In this Chapter we also discussed about the performance of memory based learning in different languages which are related with our works. The Arabic morphological analysis result discourages conducting study on morphological analysis for Semitic languages using MBL. This unsatisfactory result happens because of the approach they followed. Their morphological analysis starts the construction of all possible analyses of isolated consonant word forms. However, in Amharic language morphological analysis MBL showed interesting performances. This happened because of the approach they followed. They start analyzing the vowelled inflected Amharic words with their grammatical functions of morphemes. Once stems are extracted, roots can be obtained from the stem in a separate module. The most crucial part of morphological analysis is finding the stem of the word which bears an important meaning. Therefore, we expect a good performance like Amharic for morphological analysis of Geʿez language using MBL.

Chapter Four: Design of Morphological Analyzer for Ge'ez Verbs

4.1 Introduction

As described in [2, 30, 62] the task of performing a full morphological analysis of a word form is usually considered as a segmentation of the word into morphemes, combined with an analysis of the interaction of these morphemes. Segmentation of morphemes determines the syntactic class of the word form as a whole. Morphological analysis of a language is a non-trivial task even for languages with highly inflected. The memory based learning approach of morphological analysis primarily concerns saving or learning of some patterns of the morpheme in memory and trying to classifying and analyzing the newly or unseen words by analogy [61]. In this Chapter we designed and described morphological analyzer of Ge"ez verbs.

4.2 Architecture of Ge'ez Verbs Morphological Analyzer

The morphological analyzer (Figure 4.1) has two phases. A training phase which consists of morpheme annotation to manually annotate inflected Ge"ez verbs, and feature extraction to create instances in a fixed length of windows and the memory based learning to train the dataset. On the other hand, the morphological analysis phase contains the feature extraction (instance making) to deconstruct a given text, morpheme identification to classify and extrapolate, stem and root extraction to label segmented inflected words with their morpheme functions.

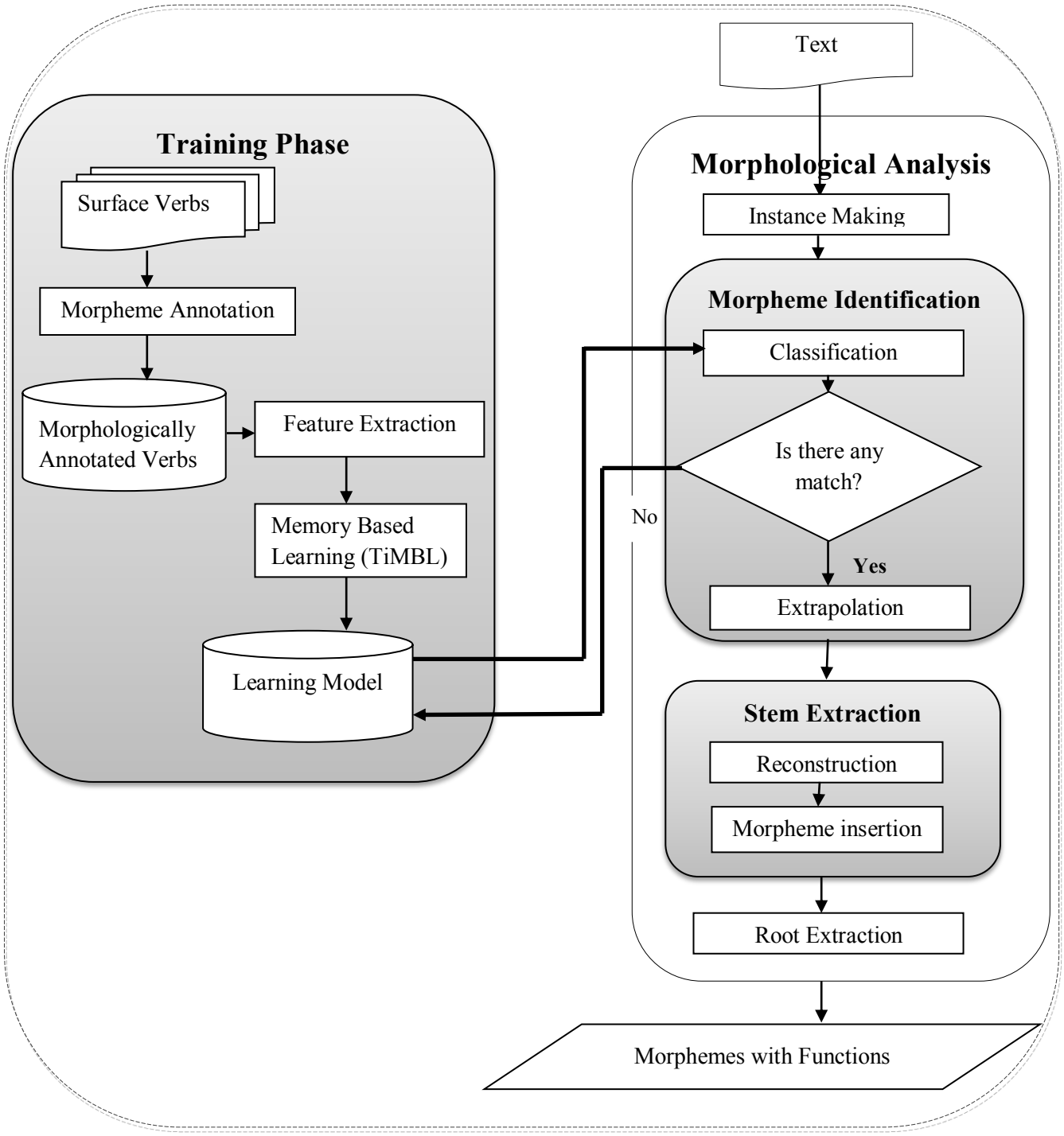

Figure 4.1: Architecture of the Proposed Geez Verbs Morphological Analyzer

4.3 Training Phase

4.3.1 Morpheme Annotation

Geʿez verb morphemes are mostly expressed by internal phonological changes in the root. Because of this the internal irregular changes of phonemes make the morphological analysis more complex. Even if it is a difficult task to find the roots of Geʿez verbs, we investigated the morphology of Geʿez language particularly for verbs. Even though Geʿez verbs have more than four prefixes and three suffixes, the morphological structure is somehow straight forward to be learned by the system. After we identify the common property of all morphological formations of Geʿez verbs and grammatical features of all the morphemes, we built morphological database. It is difficult to find a single representation or patterns of verbs as they are different in types due to a number of morphological and phonological processes. Therefore, we consider the most significant part of the word stem which bears meaning next to the root. In grammar, the stem of a word is the main part of it, which remains unchanged when the ending changes.

In this study we tried to create a memory based morphological analysis for Geʿez which handles all morphological productive word-classes particularly for verbs. To do this there is a need of preparing morphologically annotated word lists manually. We were not able to get annotated database of Geʿez morphology. This makes us to consider building morphological analyzed database. Therefore, we are forced to prepare manually annotated word. In doing so we identified the different affixations of verbs. As mentioned in Chapter one, our sources of data were different Geʿez textbooks, language experts and research papers. The annotations were cross checked by language experts and a linguist.

As shown in Figure 4.1, during preparing annotated dataset for experimentation purpose the following tasks were identified and performed in the order listed.

- Identifying inflected words
- Segmenting the word in to prefix, stem, suffix
- Putting boundary marker between each
- describing the representation of each marker

As described by Desta Berihu [4], a verb in Geʿez has three segments for prefixes before the stem and three segments for suffixes after the stem as mentioned in Chapter three Table 2.8. The positions of the affixes are also as described in Table 2.8.

NegPref/PosPref/PreCirc + **Stem** + SufCirc/SMS/OMS

Analyzing all these affixes, the root template pattern of Geʿez verbs makes morphological analysis complex [4]. It is a challenging task representing its features in to suitable memory based learning approach like Amharic [61]. Geʿez verb stems are divided into verb roots and grammatical template. The stem is the lexical part of the verb and also the source of most of its complexity. Instances are created from the manually annotated morphological data base for the purpose of learning. As described in Section 4.3.2 instances consists from a sequence of feature values and a corresponding class. The class can represent several aspects of the features like tense, aspect, mode, voice (causative, transitive, etc.) and other grammatical functions (number, gender, subject, object etc.), see more in Appendeix C.

We considered perfective, indicative, subjunctive, gerundive and jussive verb forms and all of its voice forms for testing our morphological analysis model. That is simplex, passive, transitive and causative forms of perfective and imperfective verbs. Because all the voice forms have perfective Tense Aspect Mood and have stems of the same pattern $C_1VC_2VC_3$ which makes the learning process similar for all forms. The Memory based learner is expected to store and memorize these patterns including all the other affixes [61].

To study all morphologically inflected verb types, we need a morphologically annotated word list with its possible inflection forms. Then, the tokens are manually annotated like prefix, stem and suffix pattern. The sample annotations for verbs are illustrated as shown in Table 4.1 (see more in Appendeix A):

Table 4.1: Manually Annotated Sample Verbs

ʔəsitə	3	fakə́d	S	a	F	hu	Q	-
ʔəsitə	3	fakə́d	S	a	F	homu	V	-
ʔəsitə	3	fakə́d	S	a	F	ha	C	-
ʔəsitə	3	fakə́d	S	a	F	hon	G	ɨ
tə	4	fəkə́d	S	ɨku	J	-	-	-
tə	4	gəbɨr	S	ə	A	nə	Z	-
tə	4	gəbɨr	S	ə	A	kə	M	-
-	-	noləw	S	ə	A	-	-	-

In order to easily make our data readable by the learner tool (TiMBL) the source data is translated to their equivalent Latin representation (see more in Appeix D) and tokenized. When

our source of data is paragraph (s) a toknizer is necessary to segment each word independently and pass them to the feature extractor. For this purpose we used the English Unicode toknizer called ucto[32] without modification.

4.3.2 Feature Extraction

After the annotated verbs are stored in a database, features are extracted automatically from the manually created morphological database to make instances using an Algorithm 4.1 based on the concept of windowing method in a fixed length of left and right context which is the average word length in the database. As described in [2], windowing method is dividing the windows where the instances are placed in the left and right context to hold fixed-length string of features, which describe the linguistic context of the token to be classified. Each instance is associated with a class. The class represents the morphological category in which the given word possesses.

Instances consisting of a fixed number of features are created by windowing methods-by sliding each character down as a focus letter for each character in the left and right context. Each example focuses on one letter, and includes a fixed number of left and right neighbor letters, in this case using a 10–1-10-1 window which yields twenty two features. The input character in focus, plus the ten preceding and ten following characters are placed in the windows. As stated in [1, 63] the complex morphological analysis is placed at the right most part as a class.

Like Amharic language Ge"ez is a syllabary writing system, since in the orthography consonants and vowels do not exist independently of each other. However, feature extraction can be represented in character and syllable based ways of the word. Ge"ez memory-based morphological analysis can be designed in a character or syllable representation of features. However, in Ge"ez syllabic representation of morphemes to be analyzed in memory based approach seems not feasible because during orthography vowels give their phonological behavior to consonant and eliminate themselves from that position which in turn makes the learning process difficult. Therefore, in this study we used the default one which is character based feature extractions representation.

[32] Freely available at http://ILK..uvt.nl.

Character Based Analysis

In memory based learning character based analysis is implemented in Bosch and Daelemans [2] work of Dutch morphological analysis. It gives concern for each character or letter to be considered. In this method each character is placed as instances in the left and right context. From the annotated database, instances were automatically extracted, to be suitable to memory based learning by sliding a window over the word in the lexicon.

Input: Inflected words
Output: -Instances in a fixed-length of vector size

1. Define the length of window size.
2. Fix the middle positions of arrays as a focus letter (the focus character represents where a character is started from that position on words).
3. Read from the DB and push one step forward each character until the right context Reached.
4. Put zero at the class if there is no any number and capital letters, next to the characters placed in the focus letter; if any one of those symbols exist put the value as a class(in last index)
5. Push the previous focus letter to the left and start putting each letter (as in step 3)
6. Go until it finishes that line
7. Go to the next line and repeat 3,4,5,6.

Algorithm 4.1: Feature Extraction (Adapted from [61])

For example, based on the above algorithm the character based representation of the word 'nasɨtəfaḱɨdəkɨmu' '**ናስ፡ተፋ፡ቅደክመ፡**' is as follows:

Table 4.2: Character Based Representation of the Word „ናስተፋቅደክሙ"

No	Left context	Focus	Right context	Class
1	= = = = = = = = = = = =	**n**	a s ɨ t ə f a ɡ́ ɨ d	**0**
2	= = = = = = = = = = = n	**a**	s ɨ t ə f a ɡ́ ɨ d ə	**0**
3	= = = = = = = = = = n a	**s**	ɨ t ə f a ɡ́ ɨ d ə k	**0**
4	= = = = = = = = = n a s	**ɨ**	t ə f a ɡ́ ɨ d ə k ɨ	**0**
5	= = = = = = = = n a s ɨ	**t**	ə f a ɡ́ ɨ d ə k ɨ m	**0**
6	= = = = = = n a s ɨ t	**ə**	f a ɡ́ ɨ d ə k ɨ m u	**7**
7	= = = = = n a s ɨ t ə	**f**	a ɡ́ ɨ d ə k ɨ m u =	**0**
8	= = = = n a s ɨ t ə f	**a**	ɡ́ ɨ d ə k ɨ m u = =	**0**
9	= = = n a s ɨ t ə f a	**ɡ́**	ɨ d ə k ɨ m u = = =	**0**
10	= = n a s ɨ t ə f a ɡ́	**ɨ**	d ə k ɨ m u = = = =	**0**
11	= n a s ɨ t ə f a ɡ́ ɨ	**d**	ə k ɨ m u = = = = =	**S**
12	n a s ɨ t ə f a ɡ́ ɨ d	**ə**	k ɨ m u = = = = = =	**W**
13	a s ɨ t ə f a ɡ́ ɨ d ə	**k**	ɨ m u = = = = = = =	**0**
14	s ɨ t ə f a ɡ́ ɨ d ə k	**ɨ**	m u = = = = = = = =	**0**
15	ɨ t ə f a ɡ́ ɨ d ə k ɨ	**m**	u = = = = = = = = =	**0**
16	t ə f a ɡ́ ɨ d ə k ɨ m	**u**	= = = = = = = = = =	**K**

The equality mark (=) is used as a filler symbol for positions beyond the beginning or end of the word. Table 4.2 shows the structure of features to make sixteen (16) instances associated with their classes, which are derived from the word „nasɨtəfaɡ́ɨdəkɨmu" (**ናስተፋቅደክሙ**). For example, The class of the sixth instance is „**7**", which indicates the morpheme ending in „**ə**" is a prefix which represents the Indicative, Subjunctive and Jussive causative reciprocal stem marker „nasɨtə" (**ናስተ**). This shows the character based representation of words transcribes their deep structure of phonological process and segments each letter one at a time.

In this study we tried to develop a corpus based morphological analysis for Ge"ez verbs which handles all morphologically productive verb classes. But there are irregular verbs that need to mange spelling changes by insertion and deletion methods. However, due to time and lack of detail knowledge of the language, we did not consider such type of operations in this study.

The dataset for the learner consists of description of instances in terms of fixed number of feature values. Each character is used once as a focus character (**F**) and associated with the ten characters to its left (**L1→L10**) and the ten characters to its right (**R1→R10**). The morphological representation is placed as a class at the right most part as shown in Table 4.2. Each feature is separated by comma to be used as input for the learner. Different formats for the training and test files are supported by TiMBL, including Compact, C4.5, ARFF, Columns, Sparse and Binary. In this study, the C4.5 format is the default method by which the training data is presented to TiMBL. C4.5 format implies that feature values are separated by commas and that the last feature value denotes the class of the instance (see more in Appendixes B). Therefore, it requires feature values and classes [63].

Table 4.3: Feature Extraction of the Word „ḱədəsomu‟ “ቀደሰመ”

L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	F	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	Class
=	=	=	=	=	=	=	=	=	=	**ḱ**	ə	d	ə	s	o	m	u	=	=	=	0
=	=	=	=	=	=	=	=	=	ḱ	**ə**	d	ə	s	o	m	u	=	=	=	=	0
=	=	=	=	=	=	=	=	ḱ	ə	**d**	ə	s	o	m	u	=	=	=	=	=	0
=	=	=	=	=	=	=	ḱ	ə	d	**ə**	s	o	m	u	=	=	=	=	=	=	0
=	=	=	=	=	=	ḱ	ə	d	ə	**s**	o	m	u	=	=	=	=	=	=	=	S
=	=	=	=	=	ḱ	ə	d	ə	s	**o**	m	u	=	=	=	=	=	=	=	=	Q
=	=	=	=	ḱ	ə	d	ə	s	o	**m**	u	=	=	=	=	=	=	=	=	=	0
=	=	=	ḱ	ə	d	ə	s	o	m	**u**	=	=	=	=	=	=	=	=	=	=	V

The class of the fifth instance „S‟ in Table 4.3 indicates that the morpheme ending in‟s‟ is a stem. In identifying the class knows the corresponding focus letter is end of a certain affixation of morpheme. Hence, in this example „S‟ indicates the character corresponding to the focus letter is the end of stem and „Q‟ and „V‟ indicates the character corresponding to the focus letter is the end of accusative (object marker) suffix. Characters that do not mark the end of a morpheme are classed with the default category 0. In order to prepare Table 4.3 for training and testing data, the morphological features must be separated by a comma (C4.5 format) and the others (L1…L10 and R1 … R10) must be removed and the following instances are left.

=, =, =, =, =, =, =, =, =, =, **ḱ,** ə, d, ə, s, o, m, u, =, =, =, 0
=, =, =, =, =, =, =, =, =, ḱ, **ə,** d, ə, s, o, m, u, =, =, =, =, 0
=, =, =, =, =, =, =, =, ḱ, ə, **d,** ə, s, o, m, u, =, =, =, =, =, 0
=, =, =, =, =, =, =, ḱ, ə, d, **ə,** s, o, m, u, =, =, =, =, =, =, 0
=, =, =, =, =, =, ḱ, ə, d, ə, **s,** o, m, u, =, =, =, =, =, =, =, S
=, =, =, =, =, ḱ, ə, d, ə, s, **o**, m, u, =, =, =, =, =, =, =, =, Q
=, =, =, =, ḱ, ə, d, ə, s, o, **m**, u, =, =, =, =, =, =, =, =, =, 0
=, =, =, ḱ, ə, d, ə, s, o, m, **u**, =, =, =, =, =, =, =, =, =, =, V

The number of classes which represents the complex morphological analysis in our dataset are 31 (see Appendix C with their description). These classes are extracted from the morphologically annotated word lists. But if our dataset increases with different affixations, the number of classes will increase. Each class represents the complex morphological analysis which represents the grammatical functions of the morphemes.

4.3.3 Memory Based Learning

Memory-Based Learning (MBL) is founded in the hypothesis that the extrapolation of behavior from stored representations of earlier experience to new situations, based on the similarity of the old and the new situation. MBL algorithms take a set of examples (fixed-length patterns of feature-values and their associated class) as input, and produce a classifier that can classify new, previously unseen, input patterns [76]. Our MBL system is based on the use of TiMBL. It is implemented in C and C++. It has also a Python version.We used both versions interchangeably because the python version has the capability to classify individual instances with simple scripts [61]. It implements several memory-based learning algorithms (IB1, **IB2**, IGTREE, TRIBL and **TRIBL2**). All implemented algorithms have in common that they store some representation of the training set explicitly in memory. During test, new cases are classified by extrapolation from the most similar stored cases. The main differences among the algorithms incorporated in TiMBL lie in the definition of similarity, the way the instances are stored in memory, and the way the search through memory is conducted [76].

There are different optimized parameters to be tuned in TiMBL. These are the MVDM (modified value difference metric) from distance metrics, IG (information gain) from weighting metrics, ID (inverse distance) from class voting weights, and k= 3 and 5 from the nearest neighbor. They are used together with the different classifiers (algorithms). The classifier engines used here are TRIBL2 and IB2 which construct a database of instances in memory during learning. In those

classifier a training instances are represented by tree structures which is similar with IBl and IGTREE algorithms.

TRIBL2 is designed as a combination between IB1 and IGTREE with the aim to exploit the trade-off between the search speed of IGTREE and the generalization accuracy of IBl. In other words, it is built to capture the best of both algorithms. It splits the classification of new instances into a quick decision-tree (IGTREE) traversal based on the first (most important and most class-disambiguating) features, followed by a slow but accurate K-NN (IB1) classification based on the remaining less important features. During the classification of an instance it continues to use IGTree as long as it finds matching feature values in the weighting-governed feature ordering. It only reverts to IB1 classification when a mismatch is found. The reasoning behind this mismatch-based switching is that the switch to 1B1 is only invoked when mismatching occurs, being the typical point in which IB1 can improve on decision-tree-style classification. It has no parameter which determines a switching point from IGTREE to IBl like TRIBL [77].

The operation of IB2 is identical to IBl, except that IB2 only saves misclassified instances in the training data [77]. It starts with an instance base containing only a small portion of the available training instances. This initial number of training instances in memory can be set by the user. IB2 then adds instances into memory only when they are misclassified by the k-NN algorithm, on the basis of the instances already in memory at that point. Additional instances are added because they do not form part of the concept description of the instances in memory at that particular time. They are included to expand the concept description because we can assume that they represent a part of the instance space where there is not enough instances contained in memory for accurate classification [17]. Generally, the procedure of building a tree and how IB2 and TRIBL2 work is shown in Algorithms 4.2 and 4.3.

Input: Training Set (TS)
Output: Concept Description (CD)

CD = 0;
For each x ϵ Training Set **do**
1. **for each** y ϵ CD **do**
   Sim[y] = Similarity(x, y)
2. Y_{max} = some y ϵ CD with maximal Sim[y]
3. **if** class(x) = class(Y_{max})
   then classification = **correct**
   else
   3.1.classification = **incorrect**
   3.2.CD = CD U { x }
   End if
   End for

End for
return CD

Algorithm 4.2: IB2 Algorithm (Adopted from [77])

As described in [77], concept descriptions are determined by how the IB2 algorithms selected similarity and classification functions use the current set of saved instances.

Similarity Function: This computes the similarity between a training instance i and the instances in the concept description. Similarities are numeric-valued.

Classification Function: This receives the similarity function's results and the classification performance records of the instances in the concept description. It yields a classification for i.

Concept Description Updater: This maintains records on classification performance and decides which instances to include in the concept description. Inputs include i, the similarity results, the classification results, and a current concept description. It yields the modified concept description.

Input:

- A training set T of instances with their classes (start value: a full instance base),
- An information gain ordered list of features (tests) Fi…Fn (start value: F1…Fn)

Output:

1. If T is unambiguous (all instances in T have the same class c), or i = (n+1), create a leaf node with class label c.
2. otherwise, until i = n(the number of features)
    - Select the first feature (test) Fi in Fi…Fn, and construct a new node N for feature Fi, and as default class c (the class occurring most frequently in T).
    - Partition T into subsets T1…Tm according to the values V1…Vm which occur for Fi in T (instances with the same value for this feature in the same subset).
    - For each j an element of {1,…m}:
        - If not all instances in Tj map to class c, BUILD-IG-TREE (Tj, Fi+1…Fn),
        - Connect the root of this sub tree to N and label the arc with Vj.

Algorithm 4.3: Algorithm for Building IGTREE (Adopted from [16])

Basically TRIBL2 starts out as IGTREE and switches to IB1 when a mismatch occurs, committing however to the matches found before the mismatch occurred. The advantage of being that much speed can be gained for instances that have a consecutive match on many features, while test instances that mismatch early on in the tree, will not be classified on the basis of these few matching features alone, instead the remaining tree will be searched according to the IB1 search method. It built a complete tree. The tree is ordered on the basis of feature weights (highest weighting features highest in the tree). The tree is searched top-down (Algorithm 4.4). Matches are committed to. When a mismatch occurs, search continues exhaustively for the remainder of the tree, identical to the search method applied by IB1.

Input:

- The root node N of a sub tree (start value: top node of a complete IGTree),
- An unlabeled instance I with information-gain-ordered feature values fi…fn (start value: f1…fn).

Output: A class label.

1. If N is a leaf node, output default class c associated with this node.
2. Otherwise, if test Fi of the current node does not originate an arc labeled with fi, output default class c associated with N.
3. Otherwise,
    - New node M is the end node of the arc originating from N with as label fi.
    - SEARCH-IG-TREE (M, fi+1…fn)

Algorithm 4.4: Algorithm for Searching IGTREE (Adopted from [16])

From the training phase, we get a memory based learning model which will be used during the morphological analysis phase. Therefore, the implementation of the memory based morphological analysis relies on this learning model including training our dataset. The output of the trained dataset is taken as a tree files to be used during implementation [61].

4.4 Morphological Analysis

The training phase is the base to implement the morphological analysis phase. In this phase the instance making to make the input words to be suitable for memory based learning classification, the morpheme identification to classify and extrapolate the class of new instances, the stem extraction to reconstruct and insert identified morphemes, and finally the root extraction to get root forms of verb stem with their grammatical functions are described in detail.

4.4.1 Instance Making

As described in Section 4.3.2, instance is a sequence of features (characters) separated with comma. When unknown word is given to be analyzed by the system, it accepts and deconstructs[33] as instances to make similar representation with the stored instance in memory. Feature extraction in this section is different from that described in the training phase. It states how to make instance to be analyzed here. The word is deconstructed in a fixed length of instances without identifying the class labels at the last index. For example, when a new previously unseen word needs to be segmented, the words are similarly deconstructed and represented as instances using the same information. For instance if we want to segment the word ‚ɬəkə́dəsomu‟ "**ተ·ቀደሶመ·**" the system accepts the word and extract its feature as shown in Table 4.4.

Table 4.4: Instance to be Classified

10	9	8	7	6	5	4	3	2	1	**F**	1	2	3	4	5	6	7	8	9	10	class
=,	=,	=,	t,	ə,	́k,	ə,	d,	ə,	s,	o,	m,	u,	=,	=,	=,	=,	=,	=,	=,	=,	?

4.4.2 Morpheme Identification

Morpheme identification is a process of identifying each morpheme of a given verbs. A new inflected word is deconstructed as instances and given to the system to be classified by the classifier. In **classification** process, if there is an exact match in the memory, the classifier

[33] The verbs are extracted as instances by making class question mark (?) to show the morphological representation of the instances (class).

returns the class of that instance to the new instance. If it is misclassified by the algorithm, on the basis of the instances already in memory at that point, it will be added into memory. **Extrapolation** is performed to assign the most likely neighborhood class with its morphemes based on their boundaries. This will be done based on the similarity metric applied on the training data.

For example, to find the class of the new instance in Table 4.4, the instance is compared to each and every instance in the memory-based learner. The classifier tries to find that training instance in memory that most closely resembles it. So this might be instance six in Table 4.3, as they share almost all features **(L5, L4, L3, L2, L1, F, R1, R2)** except **L6** and **L7**. The memory-based learner then extrapolates the Q class of this training instance and predicts it to be the class of the new instance. Instances which are only misclassified by the classifier will be stored in the memory based model for farther analysis [68].

4.4.3 Stem Extraction

After morphemes are identified, the next step is the stem extraction process. In stem extraction process **reconstruction** of individual instances into a meaningful (their original word forms) and **morpheme insertions** in their segmentation point are performed. The steps of this post-processing phase (the recompilation of words and insertion of the predicted classes in their proper morpheme boundaries) are performed based on an Algorithm 4.5.

```
Input: - Ge"ez word
Output: - Stem with morpheme functions
        De-construct the given word
        For each instances given
                Read TRIBL2/IB2tree
                If similar /related instances exist in memory
                        Assign the right most part to the new instances as class
                        If more than one related instances exist
                                Select the nearest-use distance metrics
                                End if
                Reconstruct and insert the morpheme boundary
                Return segmented word
                Else
                        Add to training data/learning model
                End if
        End for
```

Algorithm 4.5: Extraction of Stems from a Given Word (Adapted from [61])

The system tries to search similar, that resembles from previously stored instances in memory and tries to find a match after extracting it. If there are no similar instances in memory, it uses a distances similarity matrix to find more nearest neighbor. Modified difference metric (MVDM) a method which determines the similarity of the values of a feature by looking at co-occurrence of values with target classes. It can be computed by Equation 4.1 [10, 62].

$$\delta(v_1, v_2) = \sum_{i=1}^{n} |P(C_i|v_1) - P(C_i|v_2)| \tag{4.1}$$

Where, v_1 and v_2 are feature values, $C_1 \ldots n$ (C_i) are class labels of the feature-values and P (C_i/v_l) is an example of a conditional probability which, expressed in words, means the probability of class C, occurring, given feature value v.

For instance, the reconstruction and morphem insertion of the whole instances of the word ተቀደሰመ ' təḱədəsomu' could be as shown in Figure 4.2.

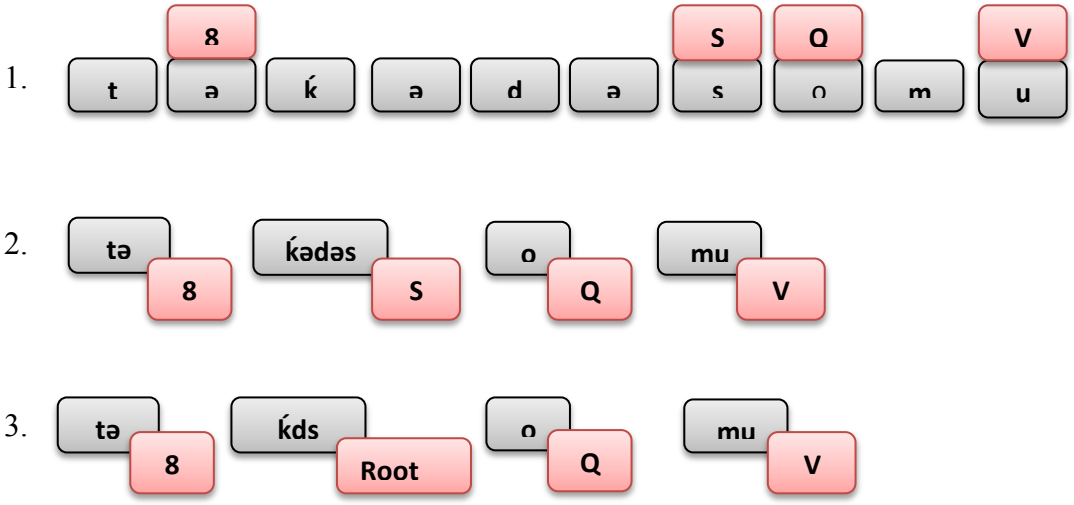

Figure 4.2: Reconstruction of the Morphological Analysis of the Word ተቀደሰመ ' təḱədəsomu'

As shown in Figure 4.2, reconstruction of the morphological analysis of the word ' təḱədəsomu', four non-null classes are predicted in the classification (1), the letters of the four identified morphemic segments are concatenated (2) and in the third step root extraction can be performed.

4.4.4 Root Extraction

To extract the root of the verb reprocess will be taken. For example, consider the verb „ḳ́ədəsə‟ in Figure 4.8: reprosse was taken in order to remove the vowels and to get the root „ḳ́ds‟. In our system we assumed that the root of Ge‟ez verbs can be deduced from the stems by removing the vowels. Because like Amharic language root in Ge‟ez is represented by only with the sixth characters of consonants which we called in Ge‟ez 'sads' except some words that begin with vowel. Therefore, any characters which are vowels in the stem (which were given to the system to be analyzed) are removed. The root of the word is returned with its segmentation and grammatical features.

There is an exception that we must consider. Whenever the system finds some verbs that start with vowel, it doesn't remove the vowel immediately rather it ignores and passes it as analyzed root. Moreover, mono and bi-radical verb types should also be considered not to remove their vowels since those verb types have valid meaning when they exit with vowels. For example we can consider the three inflected words as shown in Table 4.5.

Table 4.5: Stem and root extraction of verbs „fəḳ́ədɨku‟, „ś́əməni‟ and „gəś́ə‟

	Word	Stem		Root	
1	fəḳ́ədɨku	[fəḳ́əd]$_{perfective}$	[ɨku]$_{1ps}$	[fḳ́d]$_{perfective}$	[ɨku]$_{1ps}$
2	śəməni	[śəmə]$_{perfective}$	[ni]$_{object}$		
3	gəś́ə	[gəś́ə]$_{perfective}$			

Source: Adopted from [19].

The morphological analysis process of some bi-radical verbs will end in stem extraction. For example, as shown in Table 4.5 the last two verbs end in stem extraction. Because when we remove vowels „ə-ə‟ from ሳሟመ /śəmə/ and ገሠ /gəśə/ gives another meaning ሥም /śm/ ግሥ /gś/ which have different meanings respectively. Therefore, to get the roots of such type of verbs first we have to change into three radicals like ሳሟመ /śəmə/ and ገሠ /gəśə/ will be changed to ሠየመ and ገየሠ respectively. This will be done by spelling insertion with the other regular formations of morphological analysis. It is possible to make the memory based learner to learn spelling insertion/changes if proper re-framed representation of the annotation is performed. Such exceptions should be considered during the implementation process [10, 60].

4.5 Summary

In this Chapter we discussed about the designing of morphological analysis of Ge‟ez verbs by dividing in training and analysis phases. In the training phase we described about the annotation process for our dataset in a character based representation of features; then how those annotated dataset are extracted in a fixed length of feature vectors using windowing method. Next instances are passed to the memory based learner tool (TiMBL) to be learned by the machine. In the analysis phase we described the general processes of morpheme identification, stem and root extractions.

Chapter Five: Experiment

5.1 Introduction

Memory-based learning (MBL) has been successfully applied to a wide range of NLP tasks as described in chapter two and for most of the tasks it shows an interesting result. In this Chapter, the experimentation environment (tool used), the corpus (datasets), some of different algorithm optimization parameters which are used here, the evaluation using the selected algorithms and techniques, and the different experimentations and results obtained from the experiments are presented in detail.

5.2 Experimentation Environment

Experiments with memory-based learning described here were carried out using the Tilburg Memory-based Learner, or TiMBL. This software package is developed and maintained by the Induction of Linguistic Knowledge group at Tilburg University, and is a very efficient and feature rich implementation. It can be run either as a "one-shot" command line classier or as a server, and there is also a C++ API that makes it possible to integrate the memory-based learning functionality into other programs [48]. We trained and tested our dataset on TiMBL that runs on Unix machine. It provides options to tune some of the parameters depending on the nature of tasks. The experiment conducted here is on the handcrafted dataset on TiMBL 6.4.4 with selected settings and by tuning the different parameters. We used Mint Linux as working environment and ucto 0.3.5 for text tokenization, geany2.1 a C++ editor for code writing, GCC 4.9 compiler for compiling the source. NetBeans 7.0.1 was for automatic feature extraction and data preprocessing. Moreover, the experiment has done with IB2 and TRIB2 algorithms only.

5.3 The Corpus

In order to train a classifier that can predict the class of new, previously unseen words correctly, a set of training examples that are manually prepared with the correct input format are needed. Because the basis of classification in TiMBL is the storage of all training examples in memory, a test of the classifier"s accuracy must be done on a separate test set [10]. In our case we call these datasets *Geez.train* and *Geez.test* files respectively. Each of these file contains a number of instances in the same format. The only difference between the files is that the training file usually contains more instances than the test file. The training set *Geez.train* contains 90% of the

words from the total numbers and the test set contains 10% of the words, none of which are present in the training set. The total of our corpus contains 1105 words which counts 12135 instances. Within these instances, 31 different class labels occur.

We started the experiment executing TiMBL by specifying the two files, *Geez.train* and *Geez.test* together with selected algorithm and default metrics. Upon completion, a new is created which is identical to the input test file. The selected TiMBL algorithm and the default metrics used during the experiment are appended to the new output file name. The process as a whole can be divided into three phases: *Phase 1* - Examining the training file; *Phase 2* - Learning from the training file; and *Phase 3* - Applying the classifier to the test set. Phases 1 and 2 can be viewed as the construction of the classifier, with Phase 3 as the evaluation phase. The first two phases of the process will proceed if a training file in the correct format is present. Phase 1 entails the examination of the contents of the training file, followed by the computation of a number of statistics based such on this examination. These statistics include entropy, the information Gain and Gain Ratio of the attributes. Entropy in this case is a probability-based measure used to calculate the amount of uncertainty. As the data become purer and purer, the entropy value becomes smaller and smaller. The goal in machine learning is to get very low entropy in order to make the most accurate decisions and classifications [61, 62]. Phase 2 is the learning phase and this merely consists of efficiently storing instances to memory according to the algorithm specified earlier. The system will exit at this point if a test file was not specified at the start, writing the results of the first two phases to an output file. Unless otherwise stated, the name of this output file is a combination of the training file, the chosen algorithm and parameter settings.

If test file was indeed specified, the process will continue to Phase 3, which consists of applying the classifier created in the two previous phases to the evaluation data, and writing the results to the output file. The results to be reported can be customized by the user, but TiMBL will by default report the accuracy obtained (percentage of correctly classified test instances). We can illustrate these processes in the following ways (Figure 5.1).

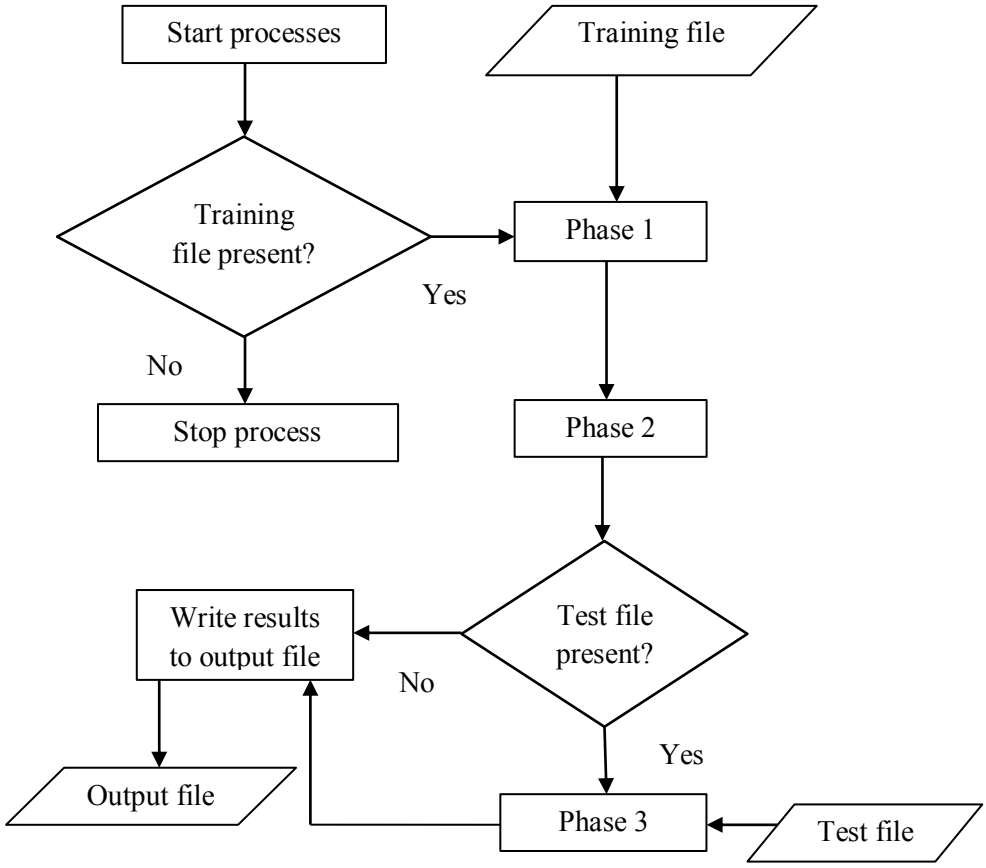

Figure 5.1: The Working of TiMBL

The Three Process Phases of the TiMBL Output:

Examine datafile 'Geez.train' gave the following results:
Number of Features: 21
InputFormat : C4.5
Phase 1: Reading Datafile: Geez.train
Start: 0 @ Tue Sep 16 07:35:27 2014
Finished: 11029 @ Tue Sep 16 07:35:28 2014
Calculating Entropy Tue Sep 16 07:35:28 2014
Lines of data: 11029
DB Entropy: 2.2107963
Number of Classes: 31

Feats	Vals	InfoGain	GainRatio
1	11	0.19631971	0.19331920

Phase 2: Building multi index on Datafile: Geez.train
Start: 0 @ Tue Sep 16 07:35:28 2014
Finished: 11029 @ Tue Sep 16 07:35:28 2014

Phase 3: Learning from Datafile: Geez.train
Start: 0 @ Tue Sep 16 07:35:28 2014
Finished: 11029 @ Tue Sep 16 07:35:28 2014
Size of InstanceBase = 93582 Nodes, (3743280 bytes), 56.07 % compression
Learning took 0 seconds, 345 milliseconds and 725 microseconds
Examine datafile 'Geez.test' gave the following results:
Number of Features: 21
InputFormat : C4.5
Starting to test, Testfile: Geez.test
Writing output in: Geez.test.TRIBL2.O.gr.k1.out
Algorithm : TRIBL2
Global metric : Overlap
Deviant Feature Metrics:(none)
Weighting : GainRatio

5.4 Algorithm Parameter Optimization

A good choice of parameter settings can have a large effect on accuracy in TiMBL. In our experiment, the following algorithm parameters could be optimized:

5.4.1 The Distance Metrics

There are different types of distance metrics such as including overlap and MVDM. As described in [10, 63], the most basic metric that works for patterns with symbolic features is the Overlap metric which is the default metric in TiMBL. But it is limited to exact matches between feature values. All values of a feature are seen as equally dissimilar. To optimize this we used modified value difference. Logically, we know that some feature values are more similar than others. Modified Value Difference Metric (MVDM) was defined to include this information about similar feature values. It determines the similarity of a feature by looking at the co-occurrence of values with target classes. The distance between two values v_1 and v_2 of a feature is computed through the difference of the conditional difference of the classes C_i for those values.

The way in which the nearest neighbor sets are composed in MVDM differs from the way in which it is done in Overlap-based metrics. It causes an abundance of ties in the nearest neighbor position; this problem is reduced or completely resolved by using MVDM. For example, if a test instance differ one mismatch from the nearest neighbor, Overlap will yield all the instances which have different feature values in the mismatch position as nearest neighbors. Contrary to this, MVDM will only propose the nearest neighbor with the lowest delta in the mismatch position. This means that MVDM yields a much smaller nearest neighbor set than the Overlap-based metrics [10].

5.4.2 Feature Weighting Metrics

Feature weighting in supervised learning concerns the development of methods for quantifying the capability of features to discriminate instances from different classes [78]. The K-NN performance can be improved by identifying the significant features and finding the feature weights. It includes Gain ratio, Information gain, shared variance and chi-square and the last three parameters can be optimized [80]. As described in [79], a K-NN classifier in its most basic form operates under the implicit assumption that all features are of equal value as far as the classification problem at hand is concerned. When irrelevant and noisy features influence the neighborhood search to the same degree as highly relevant features, the accuracy of the model is

likely to deteriorate. It is a technique used to approximate the optimal degree of influence of individual features using a training set. When successfully applied, relevant features are attributed a high weight value, whereas irrelevant features are given a weight value close to zero. It can be used not only to improve classification accuracy but also to discard features with weights below a certain threshold value and thereby increase the resource efficiency of the classifier. Therefore, feature weighting is a measure, which assigns higher weights to more important features.

5.4.3 Number of Nearest-Neighbor

Nearest-neighbor sets are useful in explaining the behavior of the classifier; the distance of the nearest neighbor can be used as a certainty factor [10]. Optimizing of nearest neighbor representing the number of nearest distances in which memory items are searched. In the experiments, it varied between 1(default), 3, 5, 7, 9, 11, 15, 25, 35 and 45 [80]. Using a larger value of k can often lead to higher accuracy for discrete classes and larger training set. On the other hand, small value of k captures fine structure of problem space better and may be necessary with small training sets [10, 63]. Our training dataset is relatively small among the large set of inflected Ge"ez words. We examined our dataset by varying the value of k to 1, 3 and 5 to see the effect of the k value in extrapolating the class of new instances and to get better generalization by setting all the parameters the same. When k=1 gives this result:

```
=,=,=,=,=,b,a,r,ə,k,a,=,=,=,=,=,=,=,=,=,=,C,F { R 0.775647, F 0.775647 }      1.2892466208571
# k=1, 2 Neighbor(s) at distance:          1.2892466208571
#         =,=,=,t,ə,f,a,ḱ,ə,d,a,=,=,=,=,=,=,=,=,=,=,{ R 1.00000, F 1.00000 }
```

As the value of k is 1 the k-NN classifier is looking for its nearest neighbors and finds only the instance =,=,=,t,ə,f,a,ḱ,ə,d,a,=,=,=,=,=,=,=,=,=,=,{ R 1.00000, F 1.00000 } with class **F** and **R** (3ppf subject marker and 2psf object marker respectively). However, the eleventh character in the instance =,=,=,=,=,b,a,r,ə,k,a,=,=,=,=,=,=,=,=,=,=, as shown above is „a" which is a morpheme boundary marker. It is coded as class „**C**" (3psf object marker). But the class predicted by the classifier is „**F**" because there are no other options and „**R**" has same weight. When the value of k equals to 3 the classifier finds more than one nearest neighbor instances.

```
=,=,=,=,=,b,a,r,ə,k,a,=,=,=,=,=,=,=,=,=,=,=,C,R { R 1.36072, G 0.585073, D 0.642549, F 0.775647 }
                                                                                    1.2892466208571
# k=1, 2 Neighbor(s) at distance:          1.2892466208571
#         =,=,=,t,ə,f,a,ḱ,ə,d,a,=,=,=,=,=,=,=,=,=,=,=,{ R 1.00000, F 1.00000 }
# k=2, 1 Neighbor(s) at distance:          1.5563009330755
#         =,=,=,t,ə,f,a,ḱ,i,d,a,=,=,=,=,=,=,=,=,=,=,=,{ D 1.00000 }
# k=3, 2 Neighbor(s) at distance:          1.7091889088213
#         =,y,i,t,i,f,a,ḱ,ə,d,a,=,=,=,=,=,=,=,=,=,=,=,{ G 1.00000 }
#         =,t,i,t,i,f,a,ḱ,ə,d,a,=,=,=,=,=,=,=,=,=,=,=,{ R 1.00000 }
```

There are five neighbors at distance 1.2892466208571 which have different verb markers (**R**-2ppf marker, **G**-3ppf object marker, **D**-3psf marker and **F**-3ppf). The k-NN classifier selects the class with the highest feature weight and assign for the test instance. As shown in the result "**R**" gets higher weight (1.36072) than other classes and it is assigned to the tested instance.

However, when the value of k becomes five the k-NN predicted the exact class of the test instance. There are thirteen neighbors at distance 1.2892466208571, six of which have 3psf object marker "**C**", while others (two: **R**-2ppf marker, one: **G**-3ppf object marker, two: **D-**3psf marker and two: **F**-3ppf) which have low feature weight"s than "**C**". Hence, the K –NN classifier selects the class with the higher feature weight (2.83471) and assigns for the tested instance.

```
=,=,=,=,=,b,a,r,ə,k,a,=,=,=,=,=,=,=,=,=,=,=,C,C { R 1.36072, C 2.83471,
                                                    G 0.585073, D 1.10418, F 1.30219}  1.2892466208571
# k=1, 2 Neighbor(s) at distance:          1.2892466208571
#         =,=,=,t,ə,f,a,ḱ,ə,d,a,=,=,=,=,=,=,=,=,=,=,=,{ R 1.00000, F 1.00000 }
# k=2, 1 Neighbor(s) at distance:          1.5563009330755
#         =,=,=,t,ə,f,a,ḱ,i,d,a,=,=,=,=,=,=,=,=,=,=,=,{ D 1.00000 }
# k=3, 2 Neighbor(s) at distance:          1.7091889088213
#         =,y,i,t,i,f,a,ḱ,ə,d,a,=,=,=,=,=,=,=,=,=,=,=,{ G 1.00000 }
#         =,t,i,t,i,f,a,ḱ,ə,d,a,=,=,=,=,=,=,=,=,=,=,=,{ R 1.00000 }
# k=4, 2 Neighbor(s) at distance:          1.8991721698954
#         ə,s,i,t,ə,f,a,ḱ,ə,d,a,=,=,=,=,=,=,=,=,=,=,=,{ C 1.00000, F 1.00000 }
# k=5, 6 Neighbor(s) at distance:          2.1662264821139
#         ə,s,i,t,ə,f,a,ḱ,i,d,a,=,=,=,=,=,=,=,=,=,=,=,{ C 2.00000 }
#         a,s,i,t,ə,f,a,ḱ,i,d,a,=,=,=,=,=,=,=,=,=,=,=,{ C 3.00000 }
#         ə,s,i,t,ə,f,a,ḱ,i,d,a,=,=,=,=,=,=,=,=,=,=,=,{ D 1.00000 }
```

5.4.4 Class Voting Weights

They are used for extrapolation from the nearest neighbor set, which includes normal majority voting (default), inverse distance weighting, inverse linear weighting, and exponential decay weighting. The most straightforward method for letting the k-nearest neighbors vote on the class of a new case is the majority voting method, in which the vote of each neighbor receives equal weight, and the class with the highest number of votes is chosen [10, 63]. It is the basic method used to determine the class to be assigned to a query. It consists of doing a frequency count of the classes occurring in the nearest neighbor set, and then assigning the most frequently occurring class to the query. This means that all instances in the set of nearest neighbors contribute equally to the decision about which class to assign to the query, irrespective of their distances from the query point. It is the default setting. However, using majority voting for our learning approach misleads identification of some of the morphemes.

```
=,=,ʔ,ə,f,ɨ,ḱ,ə,d,ə,t,ɨ,=,=,=,=,=,=,=,=,=,D,D { C 1.00000, D 3.00000 }      0.45356618230956
# k=1, 2 Neighbor(s) at distance:          0.45356618230956
#         =,=,t,ə,f,ə,ḱ,ə,d,ə,t,ɨ,=,=,=,=,=,=,=,=,=,{ C 1.00000 }
#         =,=,t,ə,f,a,ḱ,ə,d,ə,t,ɨ,=,=,=,=,=,=,=,=,=,{ D 1.00000 }
# k=2, 1 Neighbor(s) at distance:          0.69095981454974
#         =,=,=,=,f,ə,ḱ,ə,d,ə,t,ɨ,=,=,=,=,=,=,=,=,=,{ D 1.00000 }
# k=3, 1 Neighbor(s) at distance:          0.86583095625387
#         s,ɨ,t,ə,f,a,ḱ,ə,d,ə,t,ɨ,=,=,=,=,=,=,=,=,=,{ D 1.00000 }
```

Four possible solutions are presents by the classifier. In order to extrapolate the class of the test instance (=,=,ʔ,ə,f,ɨ,ḱ,ə,d,ə,t,ɨ,=,=,=,=,=,=,=,=,=,) the k-NN classifier is looking for its nearest neighbors and finds two neighbors; three with „D" (3psf subject marker) and one with „C" (3psf object marker). Therefore, the class predicted by the classifier is „D" (3psf subject marker), because this is the majority class in the NN-set. If the number is reversed (three C class and one D class), the classifier will assign class C for the tested instance instead of D because it is the majority class. So, even if a part of the correct solution is present in the NN-set, simple majority voting prevents it from surfacing in the output. In Figure 5.6 the classifier assigned the correct class for the tested instance.

Sometimes when the value of k is small, and the data is very sparse, or the class labels are noisy, the probability estimation is very unreliable to use this method. When the radius of the k-NN region can get extended far beyond the local neighborhood of the query point, but the far neighbors will have the same influence in voting as the close neighbors. This can result in

classification errors. To remedy this, distance weighted voting can be used [10, 63]. In our experiment both inverse distance weighting and inverse linear weighting are used. Inverse distance weighting implies that weights, inversely proportional to the distance from the query instance, are awarded to the neighboring instances. On the other hand, inverse linear weighting assigns a heavier weight to a neighbor that is a smaller distance away from the query than a neighbor that is a greater distance away [63]. A neighbor with smaller distance is weighted more heavily than one with a greater distance: the nearest neighbor gets weight of 1, the furthest neighbor a weight of 0 and the other weights are scaled linearly to the interval in between [10].

5.5 Evaluation and Results

In this section we discussed the evaluation technique, testing results and comparison with related work which are done using MBL approaches.

5.5.1 Evaluation Technique

The memory-based classier has been trained and tested by the more common method called 10-fold cross-validation technique. In it each data set was split into 10 parts of equal size. Each of these parts was used as test set once and the remaining parts were concatenated to be the training set. This testing have been applied in two batches, the first one using TiMBL‟s default parameter settings (Table 5.1) and the second one with a limited form of parameter optimization. This optimization procedure consists of manually selecting a few values for each input parameter of the machine learning algorithm and testing all combinations of these values.

Cross-validation is a computer intensive technique, using all available examples as training and test examples. It mimics the use of training and test sets by repeatedly training the algorithm K times with a fraction 1/K of training examples left out for testing purposes [81]. It has two possible goals: to estimate performance of the learned model from available data using one algorithm and to compare the performance of two or more different algorithms and find out the best algorithm for the available data. The basic form of cross-validation is k-fold cross-validation. In this form the data is first partitioned in to k equally (or nearly equally) sized segments or folds. Subsequently k iterations of training and validation are performed such that within each iteration a different fold of the data is held-out for validation while the remaining k-1 folds are used for learning. In data mining and machine learning 10-fold cross-validation (k= 10) is the most common [82].

In the 10-fold cross-validation experiments, the system is trained on approximately 90% of the corpus and then tested on the remaining 10%. This is repeated 10 times (i.e., for 10 folds), with a different part of the corpus being used as a test corpus each time. All instances in the corpus are entirely included in either the test corpus or the training corpus. In other words, the split between training and test corpora is always located at instance boundary. The overall accuracy is the average of the sum of each training and test set as illustrated in Tables 5.4 and 5.7. There is a certain variation in the sizes of the training and test corpora used in the different folds. Figure 5.2 demonstrates our 10-fold cross validation evaluation. The darker sections of the data are used for training while the whiter sections are used for validation (testing data).

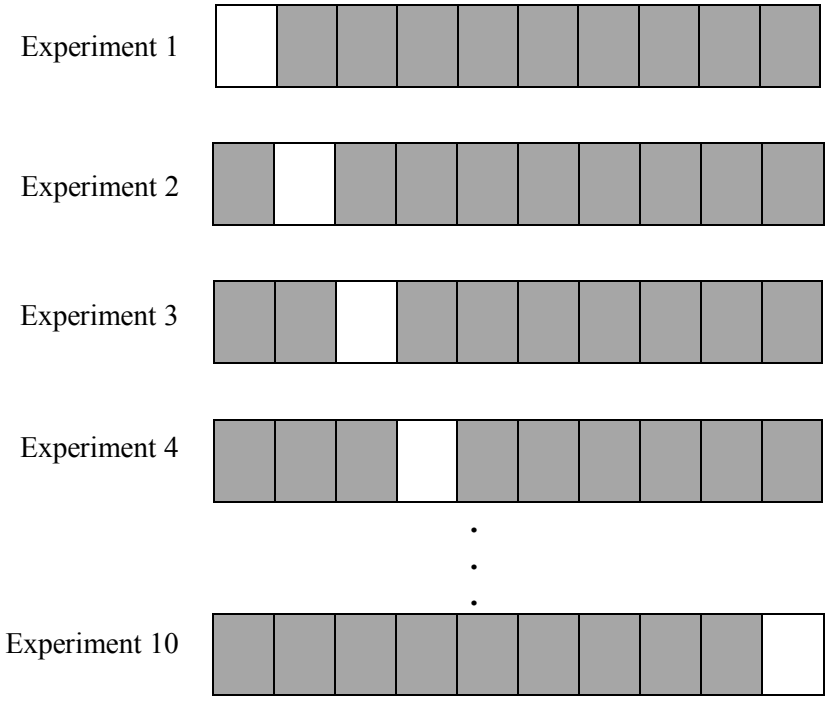

Figure 5.2: Procedure of 10-fold Cross-Validation

Table 5.1: Default Parameters with Their Values and Descriptions

Parameter	Default values	Description
Nearest neighbor	K=1	K=1 number of nearest neighbors used for extrapolation
Feature-weighting	-m0	The distance between two patterns is simply the sum of the differences between the features (overlap)
Distance metrics	-w0	No weight. All features have the same importance
Class voting	-dz	Normal majority voting-all neighbors have equal weight

5.5.2 Testing Results

Testing was done using TiMBL‟s default and optimized parameter settings. Based on the default values of parameters of each algorithm (IB2 and TRIB2), we obtained results[34] shown in Tables 5.2 and 5.3 in each training and testing experiments (10-fold cross validation).

Table 5.2: 10 Fold CV Experiment on IB2 with Default Setting and b=5330

No of experiment	1	2	3	4	5	6	7	8	9	10
No instances	11029	10739	10629	11017	10727	11042	10935	10979	10963	11155
Seconds taken	0.211	0.193	0.175	0.116	0.157	0.159	0.118	0.114	0.244	0.279
Size of instance base (byte)	1869800	1889840	1790880	1862720	1973680	1936880	1951320	1933120	1922840	1850960
compression	56.65	55.98	57.30	56.73	55.28	56.29	56.07	56.07	56.20	58.18
Accuracy	94	94.9	95	93.5	94.4	94.9	96.5	92.1	77.3	84.5

In Table 5.2 b=5330 is the number of instances, to be taken from the top of the training file, to act as the bootstrap (sample with replacement) set of memorized instances before IB2 starts adding new instances. It is only applicable in conjunction with IB2 algorithm. The value (5330) here is the average value of b from the 10 fold experiments which scores high performance in each fold.

[34] The syntax how to use those parameters is described in [10].

Table 5.3: 10 Fold CV Experiment on TRIB2 with Default Setting

No of experiment	1	2	3	4	5	6	7	8	9	10
No instances	11029	10739	10629	11017	10727	11042	10935	10979	10963	11155
Seconds taken	0.0738	0.0565	0.060	0.046	0.089	0.0707	0.0765	0.0473	0.0814	0.117
Size of instance base (byte)	3743280	3705960	3757200	3756240	3836560	3769840	3804200	3833800	3709760	3693920
compression	56.07	55.82	55.50	55.96	55.20	56.21	55.83	55.65	56.42	57.09
Accuracy	96.1	97.9	96.8	96	93.1	93.9	97.9	93.9	75.9	70.5

The overall performance of the experiment is the average value of each fold experiment results. The average experiment result of the two algorithms with default parameter and optimized parameter setting described in Table 5.4 and Table 5.7 respectively.

Table 5.4: Average Performance of 10 fold CV Experiment With Default Parameter Setting

Evaluation method	*Algorithm*	*Compression (%)*	*Time taken (seconds)*	*Size of instances base (byte)*	*Accuracy (%)*
10 fold CV	IB2	56.48	0.177	1898204	91.72
	TRIBL2	55.98	0.0725	3761076	91.19

As we observe from Table 5.4 IB2 algorithm perform a little bit better generalization of the accuracy of the model than TRIBL2 algorithm. Moreover, IB2 algorithm used much less memory storage than TRIBL2. But IB2 consumes much more time than TRIBL2 to process.

Many classifiers are parameterized and their parameters can be tuned to achieve the best result with a particular dataset. In most cases it is easy to learn the proper value for a parameter from the available data [82]. After tuning a number of parameters along with the algorithms, we identified the parameters with high performance as optimized parameters. These are nearest neighbor: k=5, distance metric: modified value difference metric (-mM), feature weighting: information gain (-w2) and distance-weighted class voting: inverse distance (-d ID). We did the experiments with those parameter settings and we have got the results shown in Tables 5.5 and 5.6.

Table 5.5: 10 Fold CV Experiment on IB2 with Optimized Parameter Settings and b=5330

No of experiment	1	2	3	4	5	6	7	8	9	10
No instances	11029	10739	10629	11017	10727	11042	10935	10979	10963	11155
Seconds taken	0.288	0.229	0.254	0.188	0.232	0.213	0.185	0.188	0.373	0.484
Size of instance base (byte)	1817120	1873200	1926200	1833400	1944720	1905360	1922640	1900320	1885440	1813480
compression	57.22	56.16	56.23	57.03	55.56	56.57	56.37	56.42	56.50	58.50
Accuracy	98	98.7	97.1	96.9	97.9	95.8	98.5	92.6	78.7	78.2

Table 5.6: 10 Fold CV Experiment on TRIBL2 with Optimized Parameter Settings

No of experiment	1	2	3	4	5	6	7	8	9	10
No instances	11029	10739	10629	11017	10727	11042	10935	10979	10963	11155
Seconds taken	0.055	0.069	0.068	0.053	0.098	0.081	0.087	0.079	0.089	0.143
Size of instance base (byte)	4033800	4003320	4060320	4084440	4158480	4079720	4128040	4115320	4010960	3988600
compression	52.66	52.28	51.91	52.12	51.44	52.6	52.07	52.39	52.88	53.66
Accuracy	97.7	99.1	98.3	96.6	94.2	94.6	98.1	95.8	75.4	73.3

Table 5.7: Average Performance of 10 fold CV Experiment with Optimized Parameter Setting

Evaluation method	*Algorithm*	*Compression (%)*	*Time taken (seconds)*	*Size of instances base (byte)*	*Accuracy (%)*
10 fold CV	IB2	56.66	0.263	1882188	93.24
	TRIBL2	52.4	0.082	4066300	92.31

As shown in Tables 5.4 and 5.7 the generalization performance of IB2 algorithm with default and optimized parameter setting is 91.72% and 93.24% respectively. Similarly, the generalization performance of TRIBL2 algorithm with default setting is 91.19% and with optimized parameters is 92.31%. Therefore, IB2 shows better general performance than TRIBL2 in both default and optimized parameter settings. It also performs better in compression the instance trees and uses less memory than TRIBL2. On the other hand, TRIBL2 processes within short seconds than IB2 on the same number of instances.

Aside from the percentage of correctly classified test instances, we used some more evaluation metrics that have become common in information retrieval and machine learning namely precision, recall, and F-score (f-measure). Precision can be defined as the percentage of correct morph boundaries among all morph boundaries suggested by the system and recall is the percentage of correct boundaries discovered by the classifier. F-score or f-measure is a harmonic

mean of precision and recall. It is a commonly used metric to summarize precision and recall in one measure [10]. The precision, recall and F-score for unknown- words with default and optimized parameters described in Table 5.8.

Table 5.8: 10 folds CV Results of Average Precision, Recall and F-score with Default and Optimized Parameter Settings

Algorithm	*With default parameters*			**With optimized parameters**		
	Precision	***Recall***	***F-score***	**Precision**	**Recall**	**F-score**
IB2	*52.9*	*52.1*	*52.49*	55.6	56.3	59.95
TRIBL2	*55.4*	*56.6*	*55.99*	58.8	60.3	59.54

As shown in Table 5.8 optimizing some of the parameters achieved better result than default parameter settings on both algorithms. This result is achieved with small dataset (1105- which doesn‟t include all verb categories complete inflection). We observed that the ill-balanced distribution of dataset matters the extrapolation of new instances. On the other hand, it has negative impact on the general performance of the system. Our data set contains complete derivation of one verb and some sample verbs from the remaining categories. Thus we believe that if our dataset contains more inflected verbs which include complete inflected verbs of each category, the system will achieve better result than this. So it is possible to say that, the result obtained here is acceptable to implement the system in large scale examples and words with more complexity.

In memory based learning the minimum size of the training set to begin with is not yet specified. However, as stated in many literatures the size of the training data matters the learning performance of the algorithm. Hence, it is crucial to draw learning curves in addition to reporting the experimental results [62]. We perform series of experiments on systematically increased amounts of training material up to the currently available total dataset which is 1105. In most cases to draw a learning curve, the learning can be measured by fixing the test set against which the increased model is systematically tested. We took the test set by calculating the average of 10% of each number of training set and set here to be 70.

Table 5.9: Generalization Accuracy with Increasing Number of Words

No of instances	49	885	1949	3325	4313	5225	6409	7310	8220	9230	10146	11116	11924
Training words	5	90	180	270	360	450	540	630	720	810	900	990	1080
Performance(%)	69.9	78.5	81.2	82.9	86.8	86.7	86.1	86.4	86.4	86.1	89.3	94.5	99.8

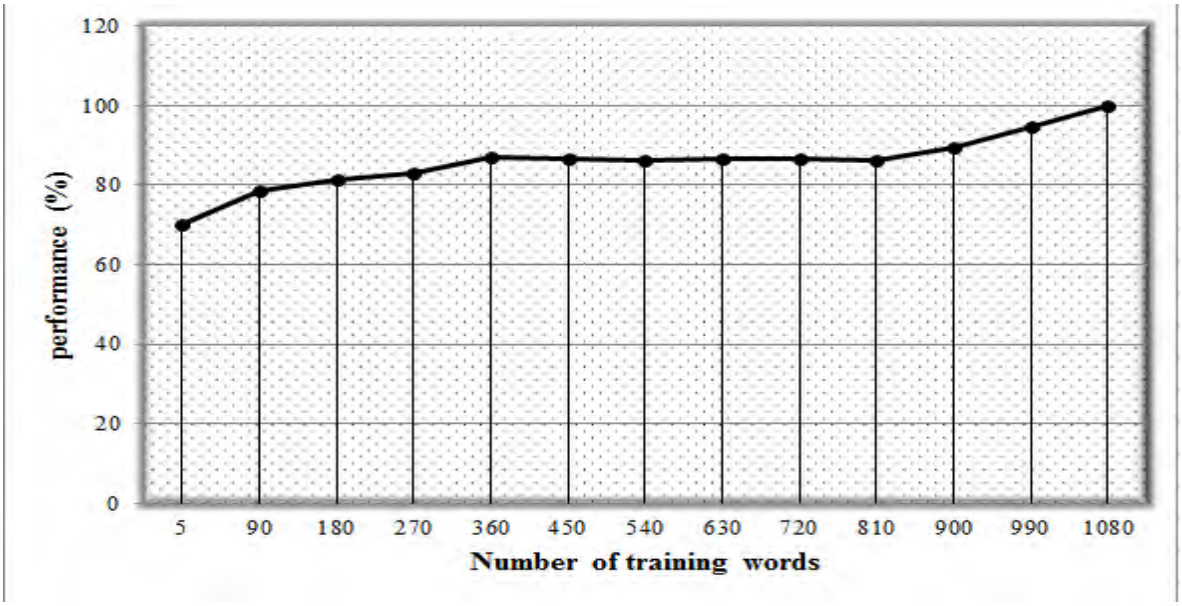

Figure 5.3: Learning Curve with Increasing Number of Words

The learning curve displays on the Ge‟ez verbs morphology test set, with increasing amounts of instances in the training set (subsets which was labled in the X-axis were simply created by taking the n first instances of the full training set). To include all the dataset we start simply taking 5 and 90 as initial training set and the rest taking multiple of 90. The x axis has a logarithmic scale. The curve shows that at the maximal amount of training material currently available (1080 words in the 10% split training set) the curve has not flattened; with larger amounts of training material better generalization accuracies on unseen data can be expected. The curve suggests that if the training set is doubled in size about three or four times more, a 100% score on the test set appears to be in reach. In general, we can infer from the learning curve that more training data gives better generalization.

5.5.3 Comparison with Related Works

As described in chapter three, to the best knowledge of the researcher, Geʿez language morphological analyzer has been investigated by Desta Berihu. The study limited to ቀተለ /*ḱətələ*/ (he killed) category verb forms. The author has analyzed the morphology of Geʿez verbs using rule-based approaches specifically CV-based and Two-Level Morphology (TLM) to design the model and to implement the prototype of the analyzer. The experimental result the author found an accuracy of 92.05% at feature level and of 73.98% at verb level.

The current study tried to model and test machine learning (MBL) perspective of Geʿez morphological analysis which includes all category verb forms. During the experiment we used 10-fold cross-validation technique. The experimental result shows that the generalization performance of IB2 and TRIBL2 algorithms are 93.24% and 92.31% respectively.

Comparison of morphological analysis of similar natural languages which developed by using memory based learning approach shown in Table 5.10.

Table 5.10: Comparison of Ge'ez Morphological Analysis with Others

Morphological Analysis of	Evaluation Techniques	Size of Dataset (in words)	Word Class	Algorithm	Accuracy (%)
Dutch language	10 fold CV	247,415	Verbs, Nouns, Adjectives	IBl	94
Arabic language	10 fold CV	16,626	Verbs, Nouns, Adjectives	IBl	15
Swedish language	10 fold CV	4,189	Verbs, Nouns, Adjectives	IBl	85.6
Swahili language	WER	9,700	Verbs, Nouns, Adjectives	IBl	13.3 (86.7)
Amharic language	10 fold CV	1022	Verbs, Nouns, Adjectives	IBl	93.59
				IGTREE	82.26
Geʿez language (our work)	10 fold CV	1105	Verbs	IB2	93.24
				TRIBL2	92.31

As shown in Table 5.10, except Swahili morphological analysis, the evaluation techniques are similar. Swahili morphological analysis was evaluated using word error-rate (WER) technique. The lower the WER means the system will be better.

It is difficult to address and analyze all the morphological features of the languages like Geʿez. Geʿez is a complex inflected language. Due to this, in this study, we addressed the morphological analysis of Geʿez verbs only. Verbs are morphologically the most complex POS in Geʿez. The experimental result shows that the generalization performance of IB2 and TRIBL2 using

optimized parameters 93.24% and 92.31%, respectively. This result is achieved with small dataset which doesn‟t include all verb categories complete inflection. We observed that the ill-balanced distribution of dataset has negative impact on the general performance of the system. Our data set contains complete derivation of one verb and some sample verbs from the remaining categories. This due to time limitation to collecte all complete inflected verbs. Thus, we believe that if our dataset contains more inflected verbs which include complete inflected verbs of each category, the system will achieve better result than this. So it is possible to say that, like other languages, the result obtained here is acceptable to implement the system in large scale examples and words with more complexity.

5.6 Summary

We experimented the proposed morphological analysis with 10 fold cross validation. The system is trained on approximately 90% of the corpus and then tested on the remaining 10%. The performance of the system in terms of accuracy, classification time and memory usage was determined by evaluating the morpheme identification by training the system with the default and optimized algorithmic parameter settings. The evaluation criteria used for morpheme identification were accuracy, recall, precision, and F- score.

The accuracy of the model with default settings are 91.72 % and 91.19% for IB2 and TRIBL2 classifiers respectively. After a number of experimentations of one parameter against the other; the nearest neighbor is 5, the distance metrics is MVDM, the feature weighting metric is IG and the class voting weight is inverse-distance, taken as optimized parameters for both IB2 and TRIBL2 results better performance. Therefore, using those optimized parameters the generalization accuracy of IB2 and TRIBL2 becomes 93.24% and 92.31% respectively.

IB2 algorithm showed significant performance even though it takes more time than TRIBL2. Like Amharic morphology, we believe that the problem of speed may not be a serious problem for Ge‟ez morphological analysis because the fundamental concern is obtaining a morphological analyzer which performs better in terms of accuracy. Therefore, IB2 classifier is better than TRIBL2 regardless of more processing time.

The precision, recall and F-score were also calculated by taking the average of the 10 fold cross validation. The results using IB2 algorithm with default parameter settings are 52.9%, 52.1% and 52.49 %, respectively. Similarly, TRIBL2 classifier was also evaluated in the same manner with

IB2 and obtained 55.4%, 56.6%, 55.99% precision, recall and F-score, respectively. These algorithms are also evaluated using optimized parameters to obtain good result than the default ones. Therefore, we obtained 55.6%, 56.3% and 59.95% precision, recall and F-score respectively using IB2 algorithm. In the same manner we obtained 58.8%, 60.3% and 59.54% precision, recall and F-score respectively using TRIBL2 algorithm. In general, both algorithms have insignificant difference. Therefore, they are suitable for Ge‟ez morphological analysis.

Chapter Six: Conclusion and Future work

6.1 Conclusion

In this study the memory-based learning for Ge"ez verbs is achievable. Based on the experiment results obtained in the previous chapter, memory based learning showed a good result for morphological analysis of Ge"ez verbs relative to small number of datasets. The unavailability of complete inflection verbs of all Ge"ez categories verbs and annotated morphological database forced us to spend much more time in preparing dataset. This is also the main reseaon for the small number of our datasets. We annotated manually 1105 verbs to be suitable to TiMBL algorithms. From these annotated verbs, we extracted 12135 instances automatically. This data set was divided into training and testing data from which 90% for training and 10% for testing. The training data is used to assess how much the model is able to learn and the test (unseen) data used for evaluating the performance of the algorithms.

By adjusting the default and optimized parameter settings of TiMBL tools, we trained and tested our dataset. To do this we used both IB2 and TRIBL2 algorithms. We found that IB2 is good at memory usage on both default and optimized settings (with 91.72% and 93.24% accuracy) but it has low processing speed which in turn takes more time. On the other hand, TRIBL2 algorithm performs a little bit different from IB2. It performs 91.19% and 92.31% with default and optimized parameter settings respectively. TRIBL2 classifier needs more memory usage and high speed during training and test of dataset. Therefore, there is tradoff between both algorithms is respective of their advantages. In summary, memory storage and speed may have a matter in choosing from both algorithms for Ge"ez morphological analysis.

6.2 Future Work

Future works based on shortcomings of the current study are outlined here. In other words, the tasks which were not included in our work are described as a recommendation for future works to make a complete full flagged morphological analysis.

- This study addressed the morphological analysis of Ge"ez verbs and with relatively very small datasets. This can be extended to nouns, adjectives, compound- words and complex verbs. The use of a large training data has huge importance in enhancing the performance of the system. In other words, initiation of a big project to develop an efficient full-

fledged automatic morphological analyzer for Ge‟ez language which works for all POS categories is required.

- The complex nature of Ge‟ez like Amharic morphology the system can‟t segment some words. So managing spelling changes (inserting, deleting) and extracting roots are possible project areas.
- A comparative analysis for Ge‟ez and other local languages, using other approaches (e.g., HMM, SVM, MBL, ILP and finite state morphological analysis) can be done.

References

[1] Antal van den Bosch, Erwin Marsi, and Abdelhadi Soudi (2005). Memory-based morphological analysis and part-of-speech tagging of Arabic: ILK / Dept. of Language and Information Science, Faculty of Arts, Tilburg University.

[2] Antal van den Bosch and Walter Daelemans (1999). Memory based morphological analysis for Dutch language: computational linguistics, Tilburg University.

[3] Bender Myron Lienel (1976). Language in Ethiopia: pages 23-27 and 99-106. Oxford University Press, London.

[4] Desta Berihu (2010). Design And Implementation of Automatic Morphological Analyzer for Geez Verbs: a thesis submitted to the school of graduate studies of the Addis Ababa University in partial fulfillment for the degree of masters of Science in computer science

[5] Atelach Alemu and Eva Forsbom (2005). Morphological Classification of Swedish Words using Memory-Based Learning: Stockholm University/KTH/GSLT Uppsala University/GSLT

[6] Igor Bolshakov and Alexander Gelbukh (2004). Computational Linguistics Models: Resources, Applications, 1st ed.

[7] John Goldsmith (2009). Segmentation and morphology: university of Chicago

[8] Jyh-Han Lin (1994). A theory for memory based learning: Machine Learning, 17, 1-26 (1994), Brown University

[9] Koray Ak, Olcay Taner Yıldız (2011). Unsupervised Morphological Analysis Using Tries: Dept. of Computer Science and Engineering. Isık University

[10] Walter Daelemans, Jakub Zavrel, Ko van der Sloot and Antal van den Bosch (2010). TiMBL: Tilburg Memory-Based Learner version 6.3 Reference Guide: ILK Technical Report – ILK10-01http://ilk.uvt.nl/timbl/

[11] Edwared Liddy (2001). Natural Language Processing: In Encyclopedia of Library and Information Science, 2nd Ed. NY. Marcel Decker, Inc

[12] Muluken Andualem (2007). Geez Verb Classification in the Three Tradition Schools of Qene: A Thesis Submitted to Graduate Studies of Addis Ababa University in Partial Fulfillment of the Degree of Masters of Science in Information Science.

[13] Sebastian Reiner Spiegler (2011). Machine learning for the Analysis of Morphologically Complex languages: a thesis submitted to the University of Bristol in accordance with the

requirements for the degree of Doctor of Philosophy, Merchant Venturers School of Engineering University of Bristol

[14] Steven Bird, Ewan Klein and Edward Loper (2009). Natural Language Processing with Python, 1st ed. USA

[15] Marta Yifru (2010). Morphology-Based Language Modeling for Amharic Dissertations schrift zum Erlangung des Grades eines Doktors der Naturwissenschaften Department Informatik an der Fakultät für Mathematik, Informatik und Naturwissenschaften der Universität Hamburg

[16] Walter Daelemans, Antal Van den Bosch and A. Weijters (1997b). IGTree: using trees for compression and classification in lazy learning algorithms: Artificial Intelligence Review11:407–423.

[17] Walter Daelemans, Jakub Zavrel, Ko van der Sloot and Antal van den Bosch (2003). TiMBL: Tilburg Memory-Based Learner version 5.0 Reference Guide: ILK Technical Report – ILK 03-10 http://ilk.uvt.nl/timbl/

[18] http://en.wikipedia.org/wiki/Morphology_%28linguistics% last accessed on 8 May 2014 at 09:11.

[19] ደሴ ቀለብ (መምህር) (2002). ትንሣኤግእዝ, በኢትዮጵያ ኦርቶዶክስ ተዋሕዶ ቤተክርስተያን በሰንበት ትምህርትቤቶች ማደራጀ መምሪያ ማኅበረቅዱሳን: አዲስአበባ

[20] Richard Wicentowski (2002). Modeling and Learning Multilingual Inflectional Morphology in a Minimally Supervised Framework*:* A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy. Baltimore, Maryland

[21] Kibur Lisanu (2002). design and development of automatic morphological Synthesizer for Amharic perfective verb forms a thesis submitted in partial fulfillment of the requirement for the degree of masters of science in information science, Addis Ababa University.

[22] Geert Booij (2009). The Oxford Handbook of Grammatical Analysis Oxford: Oxford University Press pp 563-589

[23] Elvis Comrie (1981). Language universals and linguistic typology: Oxford UK: Blackwell.

[24] Professor Oiry Fall (2009). Morphology

[25] Wolf Leslau (1987, 1991). Comparative Dictionary of Ge‟ez (Classical Ethiopic): Ge‟ez-English / English-Ge‟ez with an index of the Semitic roots, Wiesbaden Harrassowitz, ISBN- 1-447-02592-1

[26] Pachalam Antony, Anand Kumar and Krishnan Soman (2010). Paradigm based Morphological Analyzer for Kannada Language Using Machine Learning Approach: Vol. 3 Issue 4, p457

[27] Anand Kumar, Dhanalak shmi, and Krishnan Soman (2010). A Sequence Labeling Approach to Morphological Analyzer for Tamil Language, Department of Linguistics, Tamil University, Thanjavur, India.

[28] Michael Gasser (2011). Computational Morphology and the Teaching of Indigenous Languages, School of Informatics, Indiana University

[29] Jisha Jayan and Rajeev Rajendran (2011). Morphological Analyser and Morphological Generator for Malayalam - Tamil Machine Translation, Tamil University, Thanjavur

[30] Walter Daelemans and Antal van den Bosch (2005). Memory-Based Language Processing, University of Antwerp and Tilburg University

[31] Mark Aronoff and Kirsten Fudeman (2011). What is Morphology?,2nd Ed.Blackwell Publishing.

[32] Graeme Ritchie (1992). Computational Morphology: Practical Mechanisms for the English Lexicon, Massachsetts Institute of Technology, USA

[33] Ayugo Ignatius, Adetunmbi Adebayo and Kammelu Nkiru (2013). Finite state Concatenative Morphotactics: The Treatment of Igbo Verbs. International Journal of Computing and ICT Research, Vol. 7 Issue 1, pp70-80

[34] Sander Canisius and Maarten van Gompel (2006). Python timbl, Tilburg University http://github.com/proycon/python-timbl/

[35] http://www.cs.bham.ac.uk/~pjh/sem1a5/pt2/pt2_intro_morphology.html

[36] Anand Kumar (2013). Morphology Based Prototype Statistical Machine Translation System for English to Tamil Language, A Thesis Submitted for the Degree of Doctor of Philosophy in the School of Engineering, Amrita School Of Engineering Amrita Vishwa Vidyapeetham Coimbatore-641 112, Tamilnadu, India

[37] Saranya Kittanakom (2008). Morphological Analyzer for Malayalam Verbs, a project report submittedin partial fulfillment for the award of the degree of Master of Technology in

Computational Engineering and Networking, Amrita Vishwa Vidyapeetham Amrita School of Engineering, Coimbatore, 641105

[38] Stave Pulman, Graham Russell, Graeme Ritchie, and Alan Black (1989), Computational Morphology of English, Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet: http://www.cl.cam.ac.uk/TechReports/

[39] Tesfaye Bayu (2002). Automatic morphological analyzer for Amharic: An experiment employing unsupervised learning and auto segmental analysis approaches. Master"s thesis, Addis Ababa University

[40]http://www.aihorizon.com/essays/generalai/supervised_unsupervised_machine_learning.htm

[41] Xuri Tang (2001). English Morphological Analysis with Machine-learned Rules, Dept. Foreign Languages, Wuhan University of Science and Engineering, 430073, Wuhan, P. R. China

[42] Ethem Alpaydın (2010). Introduction to Machine Learning 2nd ed. the Massachusetts Institute of Technology Press Cambridge, Massachusetts London, England

[43] Wondwossen Mulugeta and Michael Gasser (2012). Learning Morphological Rules for Amharic Verbs Using Inductive Logic Programming. Workshop on Language Technology for Normalization of Less-Resourced Languages SALTMIL8/AfLaT2012).

[44] Andrew Barto and Thomas Dietterich (2003). Reinforcement Learning and its Relationship to Supervised Learning. University of Massachusetts Amherst, MA and Oregon State University

[45] Sotiris Kotsiantis (2007). Supervised Machine Learning: A Review of Classification Techniques, Department of Computer Science and Technology University of Peloponnese, Greece End of Karaiskaki, 22100, Tripolis GR.

[46] Samer Al Moubayed, Anantha krishnan and Laura Enflo (2011). Automatic Prominence Classification in Swedish Centre for Speech Technology, Royal Institute of Technology (KTH), Lindstedtsvägen 24, SE-10044, Stockholm

[47] Phil Blunso (2004). Hidden Markov Models,

[48] Anders Nklestad (2009). A Machine Learning Approach to Anaphora Resolution Including Named Entity Recognition, PP Attachment Disambiguation, and Animacy Detection

[49] Stale Buchholz (2002). Memory-based grammatical relation finding. Ph. D. thesis, Tilburg University

[50] Sander Canisius and Antal van den Bosch (2004). A memory-based shallow parser for spoken Dutch. Selected papers from the Thirteenth Computational Linguistics in the Netherlands Meeting, Antwerp, Belgium, pp.31- 45

[51] Bart Decadt, Jacques Duchateau, Walter Daelemans, and Piet Wambacq (2002). Memory-based phoneme-to-grapheme conversion. Computational Linguistics in the Netherlands 2001. Selected Papers fromthe Twelfth CLIN Meeting, Amsterdam, pp. 47- 61.Rodopi.

[52] Paul Lendvai, Antal van den Bosch, Emiel Krahmer, and Sander Canisius (2004). Memory-based robust interpretation of recognised speech. In Proceedings of the 9th International Conference on "Speech and Computer", SPECOM'04, St. Petersburg, Russia, pp.415 - 422.

[53] Paul Lendvai, Antal van den Bosch, and Emiel Krahmer (2003). Memory-based disfluency chunking. In Proceedings of DISS'03, Disfluency in Spontaneous Speech Workshop, G□oteborg University, Sweden, pp. 63 - 66.

[54] Erwin Marsi, Brian Busser, Walter Daelemans, Vectory Hoste, Martin Reynaert, and Antal van den Bosch (2002). Combining information sources for memory-based pitch accent placement. In Proceedings of the International Conference on Spoken Language Processing, pp. 1273 - 1276.

[55] Antal van den Bosch, Sander Canisius, Walter Daelemans, Jens Hendrickx, and Karl Sang (2004). Memory-based semantic role labeling: Optimizing features, algorithm, and output. Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004), Boston, MA.

[56] Jeremy Veenstra, Antal van den Bosch, Stale Buchholz, Walter Daelemans, and Jakub Zavrel (2000). Memory-based word sense disambiguation. Computers and the Humanities.

[57] Daniele Meulder, and Walter Daelemans (2003). Memory-based named entity recognition using unannotated data. Proceedings of CoNLL-2003, the Seventh Conference on Natural Language Learning, Edmonton, Canada, pp. 208-211.

[58] Jakub Zavrel, and Walter Daelemans (1997). Memory-based learning: Using similarity for smoothing. In Proceedings of the 35th annual meeting of the ACL, Madrid.

[59] Antal van den Bosch and Vectory Hoste (2007). A modular approach to learning dutchco-reference.In C. Johansson (Ed.), Proceedings of the Workshop on Anaphora Resolution, Mjlfjell, Norway, September 28-30, 2005.

[60] Walter Daelemans and Jakub Zavrel (2003). Text Mining, Theoretical Aspects and Applications, Chapter Feature-rich memory-based classification for shallow NLP and information extraction, pp. 33-54. Springer Physica-Verlag

[61] Mesfin Abate (2014). Amharic Morphological Analysis Using Memory Based Learning, A Thesis Submitted in Partial Fulfillments of The Requirement for the Degree of Master of Science in Computer Science, Addis Ababa University, Ethiopia.

[62] Walter Daelemans and Antal vanden Bosch (2005). Memory based language processing, Published in the United States of America by Cambridge University Press, New York

[63] Walter Daelemans and Antal van den Bosch (2009). Memory based language processing, CLiPS Research Group, University of Antwerp, ILK / Tilburg centre for Creative Computing, Tilburg University, the Netherlands.

[64] Walter Daelemans (1999). Introduction to the special issue on memory-based language processing, ILK , Computational Linguistics , Tilburg University, the Netherlands, pp 287-296

[65] Demon Hand, Metro Mannila, Padhraic Smyth (2001). Principles of Data Mining. The MIT Press.

[66] GongdeGuo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer (2003). KNN Model-Based Approach in Classification, School of Computing and Mathematics, University of Ulster.

[67] Walter Daelemans, Antal van den Bosch, and Jakub Zavre (1997). Feature-Relevance Heuristic for Indexing and CompressingLargeCaseBases, Computational Linguistics, Tilburg University, the Netherlands.

[68] Guy DePauw and Gilles-Maurice deSchryver (2008). Improving the Computational Morphological Analysis of a Swahili Corpus for Lexicographic Purposes, Language Technology Group, University of Antwerp and Department of African Languages and Cultures, Ghent University, Belgium

[69] **ኪዳነ ወልድ ክፍሌ (አለቃ) (1948) ፣ መጽሐፈ ሰዋስው ወግስ መዝገበ ቃላት ሐዲስ ፣ አዲስ አበባ ፣አርቲስቲክ ማተምያ ቤት**

[70] Dillmann Carl Bezold (1899). Ethiopic Grammar. London: Amsterdam Hilo Press

[71] Lambdin Thomas (1978). Introduction to Classical Ethiopic (Ge‟ez). Michigan: Edwards Brothers, Inc

[72] Ludolf Hiob (1699). Lexicon Aethiopico-Latinum. (ed.).

[73] በላይ መኮንን (ሊቀ ጎሩያን) (2002) ፣ ሕያዉ ልሣነ ግእዝ-አማርኛ መዝገበ ቃላት ኣዲስ አበባ ፣ንግድ ማተምያ ድርጅት ::

[74] ዘርአዳዊት አድሐና (መምህር) (1996) ፣ መርኖ ሰዋስዉ ዘልሣነ ግእዝ ፣ አዲስ አበባ ፣ብርሃንና ሰላም ማተምያ ድርጅት

[75] Gabriella Scelt (2001). The Comparative Origin and Usage of the Ge'ez writing system of Ethiopia, A paper submitted to Professor Pilar Quezzaire-Belle in partial fulfilment of the requirements for Arts of Africa, AH 2 15

[76] Jaime Teixeira, Carlos Oliveira and Coentrao Moutinho (2006). Machine Learning of European Portuguese Grapheme-To-Phone Conversion using a Richer Feature Set, vol. 4, no 6

[77] Daha Aha, John Kibler and Jolm Albcrt (1991). Instance based Algorithms in Machine learning vol. 6, pp. 37-66.

[78] Elena Marchiori (2013). Class dependent feature weighting and K-nearest neighbour classification, Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands.

[79] http://www.fon.hum.uva.nl/praat/manual/kNN_classifiers_1_1_1__Feature_weighting.html last accessed on 17 Sep 2014 at 14:37

[80] Walter Daelemans, Veronique Hoste (2002). Evaluation of Machine Learning Methods for Natural Language Processing Tasks, CNTS Language T echnology Group, University of Antwerp UIA, Universiteitsplein 1 (bldng A), B-2610 Antwerpen, Belgium

[81] Ava Stone (1974). Cross-validatory choice and assessment of statistical predictions, Journal of the Royal Statistical Society, B, 36(1):111–147.

[82] Payam Refaeilzadeh, Leitang Huanliu (2008). Cross validation Arizona State University.

[83] John McCarthy (1981). A prosodic theory of non-concatenative morphology, Linguistics Department Faculty Publication Series, University of Massachusetts – Amherst

[84] Jacqueline Dake (2005). Explorations of the speed-accuracy trade-off in Memory Based Learning algorithms

[85] Michael Gasser (2011). HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya. Conference on Human Language Technology for Development, Alexandria, Egypt.

Appendix A: Sample Manually Prepared Verbs

ḱədəsSəA
ḱədəsSəAniJ
ḱədəsSəAnəZ
ḱədəsSəAkəM
ḱədəsSəAkɨmuK
ḱədəsSəAkiL
ḱədəsSəAkɨnRɨ
ḱədəsSoQ
ḱədəsSoQmuV
ḱədəsSaC
ḱədəsSoQnGɨ
tə4ḱədəsSəA
tə4ḱədəsSəAniJ
tə4ḱədəsSəAnəZ
tə4ḱədəsSəAkəM
tə4ḱədəsSəAkɨmuK
tə4ḱədəsSəAkiL
tə4ḱədəsSəAkɨnR
tə4ḱədəsSoQ
tə4ḱədəsSoQmuV
tə4ḱədəsSaC
tə4ḱədəsSoQnGɨ
gəbɨrSəA
gəbɨrSəAniJ
gəbɨrSəAnəZ
gəbɨrSəAkəM
gəbɨrSəAkɨmuK
gəbɨrSəAkiL
gəbɨrSəAkɨnRɨ
gəbɨrSoQ
gəbɨrSoQmuV
gəbɨrSaC
gəbɨrSoQnGɨ
tə4gəbɨrSəA
tə4gəbɨrSəAniJ
tə4gəbɨrSəAnəZ
tə4gəbɨrSəAkəM
tə4gəbɨrSəAkɨmK
tə4gəbɨrSəAkiL
tə4gəbɨrSəAkɨnRɨ
tə4gəbɨrSoQ
tə4gəbɨrSoQmuV
tə4gəbɨrSaC
tə4gəbɨrSoQnGɨ
barəkSəA
barəkSəAniJ
barəkSəAnəZ
barəkSəAkəM
barəkSəAkɨmuK
barəkSəAkiL
barəkSəAkɨnRɨ
barəkSoQ
barəkSoQmuV
barəkSaC
barəkSoQnGɨ
tə4barəkSəA
tə4barəkSəAniJ
tə4barəkSəAnəZ
tə4barəkSəAkəM
tə4barəkSəAkɨmuK
tə4barəkSəAkiL
tə4barəkSəAkɨnRɨ
tə4barəkSoQ
tə4barəkSoQmuV
tə4barəkSaC
tə4barəkSoQnGɨ
noləwSəA
noləwSəAniJ
noləwSəAnəZ
noləwSəAkəM
noləwSəAkɨmuK
noləwSəAkiL
noləwSəAkɨnRɨ
noləwSoQ
noləwSoQmuV
noləwSaC
noləwSoQnGɨ
tə4noləwSəA
tə4noləwSəAniJ
tə4noləwSəAnəZ
tə4noləwSəAkəM
tə4noləwSəAkɨmuK
tə4noləwSəAkiL
tə4noləwSəAkɨnRɨ
tə4noləwSoQ
tə4noləwSoQmuV
tə4noləwSaC
tə4noləwSoQnGɨ
ħokSəA
ħokSəAniJ
ħokSəAnəZ
ħokSəAkəM
ħokSəAkɨmuK

ħokSəAkiL
ħokSəAkɨnRɨ
ħokSoQ
ħokSəAmuV
ħokSaC
ħokSoQnGɨ
tə4ħəwɨkSəA
tə4ħəwɨkSəAniJ
tə4ħəwɨkSəAnəZ
tə4ħəwɨkSəAkəM
tə4ħəwɨkSəAkɨmuK
tə4ħəwɨkSəAkiL
tə4ħəwɨkSəAkɨnRɨ
tə4ħəwɨkSoQ
tə4ħəwɨkSoQmuV
tə4ħəwɨkSaC
tə4ħəwɨkSoQnGɨ
tənɨbəlSəA
tənɨbəlSəAni
tənɨbəlSəAnəZ
tənɨbəlSəAkəM
tənɨbəlSəAkɨmuK
tənɨbəlSəAkiL
tənɨbəlSəAkɨnRɨ
tənɨbəlSoQ
tənɨbəlSoQmuV
tənɨbəlSaC
tənɨbəlSoQnGɨ
tə4tənɨbəlSəA
tə4tənɨbəlSəAniJ
tə4tənɨbəlSəAnəZ
tə4tənɨbəlSəAkəM
tə4tənɨbəlSəAkɨmuK
tə4tənɨbəlSəAkiJ
tə4tənɨbəlSəAkɨnRɨ
tə4tənɨbəlSoQ
tə4tənɨbəlSoQmuV
tə4tənɨbəlSaC
tə4tənɨbəlSoQnGɨ
śemSəA
śemSəAniJ
śemSəAnəZ
śemSəAkəM
śemSəAkɨmuK
śemSəAkiL
śemSəAkɨnRɨ
śemSoQ
śemSoQmuV
śemSaC
śemSoQnGɨ
tə4śəyɨmSəA
tə4śəyɨmSəAniJ
tə4śəyɨmSəAnəZ
tə4śəyɨmSəAkəM
tə4śəyɨmSəAkɨmuK
tə4śəyɨmSəAkiL
tə4śəyɨmSəAkɨnRɨ
tə4śəyɨmSoQ
tə4śəyɨmSoQmuV
tə4śəyɨmSaC
tə4śəyɨmSoQnGɨ

Appendix B: Sample Extracted Features

=,=,=,=,b,a,r,ə,k,ə,n,ə,=,=,=,=,=,=,=,=,=,0

=,=,=,b,a,r,ə,k,ə,n,ə,=,=,=,=,=,=,=,=,=,=,Z

=,=,=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,ə,=,=,=,0

=,=,=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,ə,=,=,=,=,0

=,=,=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,ə,=,=,=,=,=,0

=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,ə,=,=,=,=,=,=,=,0

=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,ə,=,=,=,=,=,=,=,=,S

=,=,=,=,=,b,a,r,ə,k,ə,k,ə,=,=,=,=,=,=,=,=,=,A

=,=,=,=,b,a,r,ə,k,ə,k,ə,=,=,=,=,=,=,=,=,=,=,0

=,=,=,b,a,r,ə,k,ə,k,ə,=,=,=,=,=,=,=,=,=,=,=,M

=,=,=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,ɨ,m,u,=,0

=,=,=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,ɨ,m,u,=,=,0

=,=,=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,ɨ,m,u,=,=,=,0

=,=,=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,ɨ,m,u,=,=,=,=,0

=,=,=,=,=,=,b,a,r,ə,k,ə,k,ɨ,m,u,=,=,=,=,=,=,S

=,=,=,=,=,b,a,r,ə,k,ə,k,ɨ,m,u,=,=,=,=,=,=,A

=,=,=,=,b,a,r,ə,k,ə,k,ɨ,m,u,=,=,=,=,=,=,=,0

=,=,=,b,a,r,ə,k,ə,k,ɨ,m,u,=,=,=,=,=,=,=,=,0

=,=,b,a,r,ə,k,ə,k,ɨ,m,u,=,=,=,=,=,=,=,=,=,0

=,b,a,r,ə,k,ə,k,ɨ,m,u,=,=,=,=,=,=,=,=,=,K

=,=,=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,i,=,=,=,0

=,=,=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,i,=,=,=,=,0

=,=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,i,=,=,=,=,=,0

=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,i,=,=,=,=,=,=,0

=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,i,=,=,=,=,=,=,S

=,=,=,=,=,b,a,r,ə,k,ə,k,i,=,=,=,=,=,=,=,=,A

=,=,=,=,b,a,r,ə,k,ə,k,i,=,=,=,=,=,=,=,=,=,0

=,=,=,b,a,r,ə,k,ə,k,i,=,=,=,=,=,=,=,=,=,=,L

=,=,=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,ɨ,n,ɨ,=,0

=,=,=,=,=,=,=,=,=,b,a,r,ə,k,ə,k,ɨ,n,ɨ,=,=,0

=,=,=,t,ə,b,a,r,ə,k,o,m,u,=,=,=,=,=,=,=,=,Q

=,=,=,=,=,=,=,=,=,n,o,l,ə,w,a,=,=,=,=,=,=,0

=,=,=,=,=,=,=,=,=,n,o,l,ə,w,a,=,=,=,=,=,=,=,0

=,=,=,=,=,=,=,=,n,o,l,ə,w,a,=,=,=,=,=,=,=,=,0

=,=,=,=,=,=,=,=,n,o,l,ə,w,a,=,=,=,=,=,=,=,=,S

=,=,=,=,=,=,=,=,n,o,l,ə,w,a,=,=,=,=,=,=,=,=,=,C

=,=,=,=,=,=,=,=,=,=,n,o,l,ə,w,o,n,ɨ,=,=,=,0

=,=,=,=,=,=,=,=,=,=,n,o,l,ə,w,o,n,ɨ,=,=,=,=,0

=,=,=,=,=,=,=,=,=,=,n,o,l,ə,w,o,n,ɨ,=,=,=,=,=,0

=,=,=,=,=,=,=,=,=,=,n,o,l,ə,w,o,n,ɨ,=,=,=,=,=,=,0

=,=,=,=,=,=,=,=,=,=,n,o,l,ə,w,o,n,ɨ,=,=,=,=,=,=,S

=,=,=,=,=,=,=,=,=,=,n,o,l,ə,w,o,n,ɨ,=,=,=,=,=,=,=,Q

=,=,=,=,n,o,l,ə,w,o,n,ɨ,=,=,=,=,=,=,=,=,=,G

=,=,=,n,o,l,ə,w,o,n,ɨ,=,=,=,=,=,=,=,=,=,=,0

=,=,=,=,=,=,=,=,=,=,t,ə,n,o,l,ə,w,ə,=,=,=,0

=,=,=,=,=,=,=,=,=,=,t,ə,n,o,l,ə,w,ə,=,=,=,=,4

=,=,=,=,=,=,=,=,=,=,t,ə,n,o,l,ə,w,ə,=,=,=,=,=,0

=,=,=,=,=,=,=,=,=,=,t,ə,n,o,l,ə,w,ə,=,=,=,=,=,=,0

=,=,=,=,=,=,t,ə,n,o,l,ə,w,ə,=,=,=,=,=,=,=,0

=,=,=,=,=,t,ə,n,o,l,ə,w,ə,=,=,=,=,=,=,=,=,=,0

=,=,=,=,t,ə,n,o,l,ə,w,ə,=,=,=,=,=,=,=,=,=,S

=,=,=,t,ə,n,o,l,ə,w,ə,=,=,=,=,=,=,=,=,=,=,A

=,=,=,=,=,=,=,=,=,=,t,ə,n,o,l,ə,w,ə,n,i,=,0

=,=,=,=,=,=,=,=,=,t,ə,n,o,l,ə,w,ə,n,i,=,=,4

=,=,=,=,=,=,=,=,=,t,ə,n,o,l,ə,w,ə,n,i,=,=,=,0

=,=,=,=,=,=,=,=,t,ə,n,o,l,ə,w,ə,n,i,=,=,=,=,0

=,=,=,=,=,=,t,ə,n,o,l,ə,w,ə,n,i,=,=,=,=,=,0

=,=,=,=,=,t,ə,n,o,l,ə,w,ə,n,i,=,=,=,=,=,=,0

=,=,=,=,t,ə,n,o,l,ə,w,ə,n,i,=,=,=,=,=,=,=,S

=,=,=,t,ə,n,o,l,ə,w,ə,n,i,=,=,=,=,=,=,=,=,A

=,=,=,=,=,=,=,=,ɦ,o,k,ə,k,i,=,=,=,=,=,=,=,S

=,=,=,=,=,=,=,=,ɦ,o,k,ə,k,i,=,=,=,=,=,=,=,=,A

=,=,=,=,=,=,=,t,ə,ħ,ə,w,ɨ,k,ə,k,ɨ,n,ɨ,=,=,0

=,=,=,=,=,=,=,t,ə,ħ,ə,w,ɨ,k,ə,k,ɨ,n,ɨ,=,=,=,0

=,=,=,=,=,=,t,ə,ħ,ə,w,ɨ,k,ə,k,ɨ,n,ɨ,=,=,=,=,0

=,=,=,=,t,ə,ħ,ə,w,ɨ,k,ə,k,ɨ,n,ɨ,=,=,=,=,=,S

=,=,=,t,ə,ħ,ə,w,ɨ,k,ə,k,ɨ,n,ɨ,=,=,=,=,=,=,A

=,=,t,ə,ħ,ə,w,ɨ,k,ə,k,ɨ,n,ɨ,=,=,=,=,=,=,=,0

=,t,ə,ħ,ə,w,ɨ,k,ə,k,ɨ,n,ɨ,=,=,=,=,=,=,=,=,0

t,ə,ħ,ə,w,ɨ,k,ə,k,ɨ,n,ɨ,=,=,=,=,=,=,=,=,=,R

ə,ħ,ə,w,ɨ,k,ə,k,ɨ,n,ɨ,=,=,=,=,=,=,=,=,=,=,0

=,=,=,=,=,=,=,=,=,=,t,ə,ħ,ə,w,ɨ,k,o,=,=,=,0

=,=,=,=,=,=,=,=,=,t,ə,ħ,ə,w,ɨ,k,o,=,=,=,=,4

=,=,=,=,=,=,=,=,t,ə,ħ,ə,w,ɨ,k,o,=,=,=,=,=,0

=,=,=,=,=,=,=,t,ə,ħ,ə,w,ɨ,k,o,=,=,=,=,=,=,0

=,=,=,=,=,=,=,t,ə,ħ,ə,w,ɨ,k,o,=,=,=,=,=,=,0

=,=,=,=,=,t,ə,ħ,ə,w,ɨ,k,o,=,=,=,=,=,=,=,=,0

=,=,=,=,t,ə,ħ,ə,w,ɨ,k,o,=,=,=,=,=,=,=,=,=,S

=,=,=,t,ə,ħ,ə,w,ɨ,k,o,=,=,=,=,=,=,=,=,=,=,Q

=,=,=,=,=,=,=,=,t,ə,ħ,ə,w,ɨ,k,a,=,=,=,=,=,0

=,=,=,=,=,=,=,=,t,ə,ħ,ə,w,ɨ,k,a,=,=,=,=,=,=,0

=,=,=,=,=,=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,=,=,=,0

=,=,=,=,=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,=,=,=,=,0

=,=,=,=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,=,=,=,=,=,0

=,=,=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,=,=,=,=,=,=,0

=,=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,=,=,=,=,=,=,=,0

=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,=,=,=,=,=,=,=,=,0

=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,=,=,=,=,=,=,=,=,=,S

=,t,ə,t,ə,n,ɨ,b,ə,l,ə,=,=,=,=,=,=,=,=,=,=,A

=,=,=,=,=,=,=,=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,n,0

=,=,=,=,=,=,=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,n,i,4

=,=,=,=,=,=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,n,i,=,0

=,=,=,=,=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,n,i,=,=,0

=,=,=,=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,n,i,=,=,=,0

=,=,=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,n,i,=,=,=,=,0

=,=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,n,i,=,=,=,=,=,0

=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,n,i,=,=,=,=,=,=,0

=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,n,i,=,=,=,=,=,=,=,S

=,t,ə,t,ə,n,ɨ,b,ə,l,ə,n,i,=,=,=,=,=,=,=,=,A

t,ə,t,ə,n,ɨ,b,ə,l,ə,n,i,=,=,=,=,=,=,=,=,=,0

ə,t,ə,n,ɨ,b,ə,l,ə,n,i,=,=,=,=,=,=,=,=,=,=,J

=,=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,n,ə,=,=,=,=,=,0

=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,n,ə,=,=,=,=,=,=,0

=,=,t,ə,t,ə,n,ɨ,b,ə,l,ə,n,ə,=,=,=,=,=,=,=,S

=,=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,o,n,ɨ,=,=,=,=,=,0

=,=,=,t,ə,t,ə,n,ɨ,b,ə,l,o,n,ɨ,=,=,=,=,=,=,0

=,=,=,=,t,ə,ś,ə,y,ɨ,m,ə,k,i,=,=,=,=,=,=,=,S

=,=,=,t,ə,ś,ə,y,ɨ,m,ə,k,i,=,=,=,=,=,=,=,=,A

=,=,t,ə,ś,ə,y,ɨ,m,ə,k,i,=,=,=,=,=,=,=,=,=,0

=,t,ə,ś,ə,y,ɨ,m,ə,k,i,=,=,=,=,=,=,=,=,=,=,L

=,=,=,=,=,=,=,=,=,=,t,ə,ś,ə,y,ɨ,m,ə,k,ɨ,n,0

=,=,=,=,=,=,=,=,=,t,ə,ś,ə,y,ɨ,m,ə,k,ɨ,n,ɨ,4

=,=,=,=,=,=,=,=,t,ə,ś,ə,y,ɨ,m,ə,k,ɨ,n,ɨ,=,0

=,=,=,=,=,=,=,t,ə,ś,ə,y,ɨ,m,ə,k,ɨ,n,ɨ,=,=,0

=,=,=,=,=,t,ə,ś,ə,y,ɨ,m,ə,k,ɨ,n,ɨ,=,=,=,0

=,=,=,=,t,ə,ś,ə,y,ɨ,m,ə,k,ɨ,n,ɨ,=,=,=,=,0

=,=,=,t,ə,ś,ə,y,ɨ,m,ə,k,ɨ,n,ɨ,=,=,=,=,=,S

=,=,t,ə,ś,ə,y,ɨ,m,ə,k,ɨ,n,ɨ,=,=,=,=,=,=,A

=,t,ə,ś,ə,y,ɨ,m,ə,k,ɨ,n,ɨ,=,=,=,=,=,=,=,0

t,ə,ś,ə,y,ɨ,m,ə,k,ɨ,n,ɨ,=,=,=,=,=,=,=,=,=,R

ə,ś,ə,y,ɨ,m,ə,k,ɨ,n,ɨ,=,=,=,=,=,=,=,=,=,=,0

Appendix C: **Boundary Marker Symbols**

S – Stem marker

I – First person singular subject marker/ nominative

W – First person plural subject marker/ nominative

Y – Second person singular masculine subject marker/ nominative

U – Second person plural masculine subject marker/ nominative

H – Second person singular feminine subject marker/ nominative

E – Second person plural feminine subject marker/ nominative

A – Third person singular masculine subject marker/ nominative

B – Third person plural masculine subject marker/ nominative

D – Third person singular feminine subject marker/ nominative

F – Third person plural feminine subject marker/nominative

J - First person singular object marker/accusative

Z - First person plural object marker/ accusative

M - Second person singular masculine object marker/ accusative

K - Second person plural masculine object marker/ accusative

L - Second person singular feminine object marker/ accusative

R - Second person plural feminine object marker/ accusative

Q - Third person singular masculine object marker/ accusative

V - Third person plural masculine object marker/ accusative

C - Third person singular feminine object marker/ accusative

G - Third person plural feminine object marker/ accusative

N – Negative marker

X – Infinitive indicator

0 – nothing indicates

2 – Perfective, indicative, jussive and gerundive causative stem marker

3- Perfective, indicative, jussive and gerundive causative reciprocal stem marker

4 – Perfective, subjunctive and gerundive reflexive and reciprocal stem marker

5- Indicative, subjunctive and jussive base stem marker

6 – Indicative, subjunctive and jussive causative stem marker

7 – Indicative, subjunctive and jussive causative reciprocal stem marker

8 – Indicative, subjunctive and jussive reflexive and reciprocal stem marker

Appendix D: Transliteraions used in the study (adapted from [19])

No	1st	2nd	3rd	4th	5th	6th	7th
1	ሀ=hə	ሁ=hu	ሂ=hi	ሃ=ha	ሄ=he	ህ=hɨ	ሆ=ho
2	ለ=lə	ሉ=lu	ሊ=li	ላ=la	ሌ=le	ል=lɨ	ሎ=lo
3	ሐ=ħə	ሑ=ħu	ሒ=ħi	ሓ=ħa	ሔ=ħe	ሕ=ħɨ	ሖ=ħo
4	መ=mə	ሙ=mu	ሚ=mi	ማ=ma	ሜ=me	ም=mɨ	ሞ=mo
5	ሠ=śə	ሡ=śu	ሢ=śi	ሣ=śa	ሤ=śe	ሥ=śɨ	ሦ=śo
6	ረ=rə	ሩ=ru	ሪ=ri	ራ=ra	ሬ=re	ር=rɨ	ሮ=ro
7	ሰ=sə	ሱ=su	ሲ=si	ሳ=sa	ሴ=se	ስ=sɨ	ሶ=so
8	ቀ=ḱə	ቁ=ḱu	ቂ=ḱi	ቃ=ḱa	ቄ=ḱe	ቅ=ḱɨ	ቆ=ḱo
9	በ=bə	ቡ=bu	ቢ=bi	ባ=ba	ቤ=be	ብ=bɨ	ቦ=bo
10	ተ=tə	ቱ=tu	ቲ=ti	ታ=ta	ቴ=te	ት=tɨ	ቶ=to
11	ኀ=xə	ኁ=xu	ኂ=xi	ኃ=xa	ኄ=xe	ኅ=xɨ	ኆ=xo
12	ነ=nə	ኑ=nu	ኒ=ni	ና=na	ኔ=ne	ን=nɨ	ኖ=no
13	አ=ʔə	ኡ=ʔu	ኢ=ʔi	ኣ=ʔa	ኤ=ʔe	እ=ʔɨ	ኦ=ʔo
14	ከ=kə	ኩ=ku	ኪ=ki	ካ=ka	ኬ=ke	ክ=kɨ	ኮ=ko
15	ወ=wə	ዉ=wu	ዊ=wi	ዋ=wa	ዌ=we	ው=wɨ	ዎ=wo
16	ዐ=ʕə	ዑ=ʕu	ዒ=ʕi	ዓ=ʕa	ዔ=ʕe	ዕ=ʕɨ	ዖ=ʕo
17	ዘ=zə	ዙ=zu	ዚ=zi	ዛ=za	ዜ=ze	ዝ=zɨ	ዞ=zo
18	የ=yə	ዩ=yu	ዪ=yi	ያ=ya	ዬ=ye	ይ=yɨ	ዮ=yo
19	ደ=də	ዱ=du	ዲ=di	ዳ=da	ዴ=de	ድ=dɨ	ዶ=do
20	ገ=gə	ጉ=gu	ጊ=gi	ጋ=ga	ጌ=ge	ግ=gɨ	ጎ=go
21	ጠ=t’ə	ጡ=t’u	ጢ=t’i	ጣ=t’a	ጤ=t’e	ጥ=t’ɨ	ጦ=t’o
22	ጰ=Pə	ጱ=Pu	ጲ=Pi	ጳ=Pa	ጴ=Pe	ጵ=Pɨ	ጶ=Po
23	ጸ=ṣə	ጹ=ṣu	ጺ=ṣi	ጻ=ṣa	ጼ=ṣe	ጽ=ṣɨ	ጾ=ṣo
24	ፀ=ḍ’ə	ፁ=ḍ’u	ፂ=ḍ’i	ፃ=ḍ’a	ፄ=ḍ’e	ፅ=ḍ’ɨ	ፆ=ḍ’o
25	ፈ=fə	ፉ=fu	ፊ=fi	ፋ=fa	ፌ=fe	ፍ=fɨ	ፎ=fo
26	ፐ=pə	ፑ=pu	ፒ=pi	ፓ=pa	ፔ=pe	ፕ=pɨ	ፖ=po

ኰ=k^{w}ə	ኲ=k^{w}ɨ	ኩ=k^{w}i	ኳ=k^{w}a	ኴ=k^{w}e
ቈ=ḱ^{w}ə	ቊ=ḱ^{w}ɨ	ቁ=ḱ^{w}i	ቋ=ḱ^{w}a	ቌ=ḱ^{w}e
ጐ=g^{w}ə	ጒ=g^{w}ɨ	ጉ=g^{w}i	ጓ=g^{w}a	ጔ=g^{w}e
ኈ=x^{w}ə	ኊ=x^{w}ɨ	ኍ=x^{w}i	ኋ=x^{w}a	ኌ=x^{w}e

Declaration

This thesis is my original work and has not been presented or submitted as a partial requirement for a degree in any other university.

Declared by:

Name: Yitayal Abate Aleka

Signature: _____

Date: _____

Confirmed by advisor:

Name: Yaregal Assabie (PhD)

Signature: _____

Date: _____

Place and date of Submission: Addis Ababa, October 07, 2014