Addis Ababa University

College of Natural Sciences

Department of Computer Science

Development of Morphological Analyzer for Sidaamu Afoo

Desta Legese Lalego

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia

June 2020

Addis Ababa University

College of Natural Sciences

Department of Computer Science

Desta Legese Lalego

Advisor: Yaregal Assabie (PhD)

This is to certify that the thesis prepared by *Desta Legese*, titled: *Development of Morphological Analyzer for Sidaamu Afoo* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signature of the Board of Examiners for Approval

	Name	Signature	Date
Advisor:	Yaregal Assabie (PhD)	_____	_____
Examiner:	_____	_____	_____
Examiner:	_____	_____	_____

Acknowledgements

First, I would like to express my heartfelt gratitude to the almighty God. All things are done by the will of God, he gives me strength and health to finish my work. Then I would like to thanks my advisor Dr. Yaregal Assabie, who provided insights and comments with his patience and knowledge to help me to finish this work.

I would like to give special thanks to my family. Always my family encourage me to work hard and their vison is seeing my future advances. My Father: Legese Lalego, my mother: Gora Bekele, and all my brothers and sisters deserve grateful thanks because without them I would not be here. Especially I would like to thanks my little brother Birhanu Legese who helped me by several uncountable things when I do my thesis.

Also, I would like thanks Mr. Tafese G/Mariam who helped me by giving a lot knowledge about the Sidaamu Afoo Linguistics and giving some material as well as advising me by so many ways.

And I will be forever thankful to my closest friend: Lamesa Tashoma and I would like to sincerely thanks all my friends, colleagues, and classmates for their assistance, patience, encouragement, and inspiration during research.

It is difficult to mention all person's name who supported me in any respect during the completion of the thesis as well as expressing my excuse that I could not mention all.

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations and Symbols

Abbreviations:

ABS	=	Absolutive
ACC	=	Accusative
AMH	=	Amharic
C	=	Consonant
CAUS	=	Causative
CD	=	Concept Description
CONV	=	Converb
CV	=	Cross-Validation
DAT	=	Dative Case
EP	=	Epenthesis
F	=	Feminine
gen	=	Genetive
GR	=	Gain ration
HMM	=	Hidden Markov model
IB	=	Instance-Base learning algorithm
IBL	=	Instance-Based Learning
ID	=	Inverse distance
IE	=	Information extraction
IG	=	Information Gain
IGTREE	=	Information Gain tree
ILP	=	Inductive logic programming
IMP	=	Imperfect
IR	=	Information retrieval
JUSS	=	Jussive
k-NN	=	k Nearest Neighbors algorithm
LOC	=	Location
LOO	=	Leave one out
M	=	Masculine
MBL	=	Memory-based learning
MID	=	Middle
MT	=	Machine Translations
MVDM	=	Modified Value Difference Metric
NLP	=	Natural Language Processing
NML	=	Nominalizer
NOM	=	Nominal
NP	=	Noun phrase
PL	=	Plural

POL	**=**	Politeness
POS	**=**	Parts-of-speech
POS	**=**	Possession
PPRF	**=**	Present perfect
PRF	**=**	Simple perfect
SG	**=**	Singular
SVM	**=**	Support vector machine
TIBL	**=**	Typical Instance based Learning
TiMBL	**=**	Tilburg Memory-based Learner
TLM	**=**	Two-level Model
TRIBL	**=**	Tree Instance Based Learning
V	**=**	Vowel

Symbols:

‘’	=	what is inside is translated to English
//	=	what is inside is Sidaamu Afoo word
()	=	what is inside is a gloss or abbreviation or explanation
[]	=	What is inside is a reference

Abstract

Sidaamu Afoo is one of the official working languages and mostly spoken in the Southern part of Ethiopia in Sidama National Regional State. As a working language, several documents are written by this language in this State. Yet, there is no any NLP applications that are tried before in this language. Analyzing these documents of this language solves several problems for the language. Morphological analyzer is a key task in natural language processing and concerned with how words are broken down into meaningful morphological units of language, and it serves as the base for other language. In our study supervised machine leaning by tuning memory-based learning approach is used. The system has two phases: training and analysis phases. In training phases, there is the process that languages morphophonemic process is analyzed, then morphologically annotated word is prepared. The features are automatically extracted by using windowing method from annotated words as a suitable to learn by learning model. While, analysis phase takes the surface word and analysis the morphophonemic process as a training phase, then features are extracted with fixed size windowing method. During extraction in analysis phase there is no need of specifying the class label as a training phase. Morpheme identification is processed after feature extraction and finally stems are extracted.

We have used Java to develop our system and TiMBL tool as learning model with IB1 and TRIBL algorithms for implementation. The experiments performance is evaluated by using 10-fold cross-validation and Leave-one-out techniques. The default and optimized parameter setting of algorithms is used to test the experiment. Finally, the system achieved good result of accuracy for both classifiers. The result of IB1 and TRIBL with default parameter setting is 93.65% and 96.03% respectively by using 10-fold cross-validation and by using leave-one-out IB1 achieved 93.4%. Also, with optimized parameter setting the IB1 and TRIBL achieved 96.83% and 97.03% respectively by using 10-fold cross-validation and by using leave-one-out technique IB1 achieved 96.6%. Thus, TRIBL algorithm achieved better performance than IB1 algorithm for Sidaamu Afoo morphology.

Key words: - Sidaamu Afoo morphology, Morphological analyzer, memory-based learning, morphophonemic process, Feature Extraction

1. Chapter one: Introduction

1.1 Background of Research

Natural language is any language that has evolved naturally in humans through use and repetition without conscious planning or premeditation. Natural languages can take different forms, such as speech, signing or text [1] and could be transmitted from generation to generation through these forms. Linguistics is one of the scientific studies of language and natural language processing (NLP) is an approach to linguistics that employs methods and techniques of computer science. NLP is concerned with the activities of natural languages in order to provide as computers that can understand everyday human speech, translate between different human languages, and interact linguistically with people in ways that suit people rather than computers [2].

The goal of natural language processing is designing and building systems that can understand and generate natural languages. Having such types of systems will enable to communicate with machines as though one is communicating with human agent [3]. It is prior to understanding the natural language before natural language processing. Correspondingly, by the level of structural, semantical and lexical, there is ambiguity for the understanding of natural language by machine. Besides the ambiguity of words, morphological inflection and derivation of the language are other reasons that make natural language understanding very complex.

The several work in computational linguistics or NLP systems tried to develop a system for processing natural language at different levels of complexity to have a general NLU system. For example, there is a system that developed for processing NL at phoneme, word, sentences, and pragmatic levels. These system's output of a lower system can serve as the input for the other next level [4].

Morphological analysis is a key task of natural language processing (NLP); it is concerned with how words are broken down into meaningful morphological units of language that can't be further divided also known as morphemes [5]. It is a primary step for various types of text analysis of any language. Morphological analyzers are used in search engines for retrieving the documents from the keyword and increases the recall of search engines. It is also used in

the speech synthesizer, the speech recognizer, lemmatization, spell and grammar checker and machine translation.

Morphologies are motivated by four considerations as presented in [6]:

- The discovery of regularities and redundancies in the lexicon of a language (such as the pattern in a walk, walks, walking; jump, jumps, jumping);
- The need to make explicit the relationship between grammatical features (such as the nominal number or verbal tense) and the affixes whose function it is to express these features;
- The need to predict the occurrences of words not found in a training corpus; and
- 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 texts.

Sidaamu Afoo is one of the languages in Ethiopia, widely used in Sidama National Regional State (SNRS). It is also an official working language for Sidama National Regional State (SNRS). Next to Afan Oromo and Somali, Sidaamu Afoo is the biggest language under Cushitic category [7, 8] with more than 4 million speakers of the language [7]. Sidaamu Afoo is also serving as the instructional medium for primary and secondary school. Until 1990 Sidaamu Afoo used Ge'ez script for writing. However, since 1991 its official writing system is the Latin alphabet system [8].

Sidaamu Afoo is morphologically rich and complex language. Due to the morphological complexity, the knowledge of a language is needed to develop a morphological analyzer. Developing morphological analyzer for a language is essential for different purposes like the development of language technology. Thus, the aim of this study is developing a morphological analyzer for analyzing Sidaamu Afoo verbs, nouns, and adjectives.

1.2 Motivation

Nowadays, there are no NLP applications for Sidaamu Afoo language like morphological analyzer, machine translator, spell checker, grammar checker, lexical dictionary, etc. Lack of those applications may consequent for different problems. The problems may occur while retrieving the document, writing on the computer, etc. Morphological analyzer plays a vital role in NLP systems. It is used to generate surface word forms found in everyday communication, from lexical components that could be stored separately in different databases

(lexicons). Such systems are used as a subcomponent of NLP in applications like machine translation, dictionary (lexicon) development, and spelling and grammar checking, and etc. [3]. Developing morphological analyzer is the groundwork for future development of other NLP applications. Some NLP applications use morphological analyzer's output as an input; For instance, spell checker use for a suggestion list and information retrieve use for a searching keyword. Also, developing the morphological analyzer is one of the contributions for development of the language. Hence, the purpose of this study is to explore the possibility of developing the morphological analyzer that is useful for generating Sidaamu Afoo words. Therefore, the main motivation of this work is to design morphological analyzer of Sidaamu Afoo.

1.3 Statement of the Problem

Sidaamu Afoo is one of the widely spoken and official working languages in Sidama National Regional State. Sidaamu Afoo is under-resourced language, where there is no well-studied linguistic resource that helps to develop NLP research and also no NLP applications tried for this language like morphological analyzer, machine translator, spell checker, grammar checker, lexical dictionary, etc. The lack of these applications for one language may impedes the development of language technology and NLP applications. The problems may occur when retrieving the document; the information retrievals uses the keyword for retrieving the necessary documents. But rather than giving whole words as a keyword, it is better to breakdown words into morphemes before giving words to search engine. Obviously, these extensive inflectional and derivational features of the language are presenting various challenges for text processing and information retrieval tasks in Sidaamu Afoo. In addition, when we write a sentence on our computer by using Sidaamu Afoo, automatic spell checker (i.e., English spell checker) marks red because there is no spell checker for Sidaamu Afoo. It may also be a struggle to develop other NLP applications like spell checker before developing morphological analyzer. For this reason, developing a morphological analyzer is the ground work for future development of other NLP applications. Also, developing the morphological analyzer is one of the contributions for development of the language.

Morphological analyzer plays a vital role in NLP systems. It is used to generate lexical word forms found in every day communication from surface word components that could be stored separately in different databases (lexicons). Such systems are used as a subcomponent of NLP

in applications like machine translation, dictionary (lexicon) development, and spell and grammar checking, etc. [3]. This means, the output of morphological analyzer is served as an input for other NLP applications.

There are different research attempts of the morphological analyzer in different languages like English [9], Amharic [10, 11, 12], Ge'ez [6, 4], Afan Oromo [3, 5]. There are some techniques which have been used in these researches and there are no common techniques for all languages. Also, techniques used in other languages may not be directly applied to Sidaamu Afoo morphology because of the morphological structure of the language.

Hence, the purpose of this study is to explore the possibility of generating Sidaamu Afoo words that are helpful for developing other NLP applications by developing the morphological analyzer.

1.4 Objectives

General objective

The general objective of this work is to develop a morphological analyzer for Sidaamu Afoo.

Specific objective

To achieve the general objective, the following specific objectives are performed: -

- Review related papers on morphological analyzer
- Study the linguistic features of Sidaamu Afoo.
- Study appropriate techniques of morphological analyzer that match with the morphological structure of Sidaamu Afoo.
- Collect the corpus to be learned and tested by the model in morphological analyzer.
- Study possible morphological rules to prepare morphologically annotated data.
- Adopt a machine learning algorithm that is appropriate to the morphological property of Sidaamu Afoo.
- Develop an algorithm of morphophonemic process for morphological analyzer of Sidaamu Afoo language.
- Develop a prototype of the system and test performance of system for morphological analyzer of Sidaamu Afoo.

1.5 Methods

To achieve our specific objectives, we would use the following methods:

Literature Review

For further study of a morphological analyzer, we will review different related works in different languages that are done by using different approaches. But for detail understanding of the language's morphological structure, we will review the papers that are done in the area of linguistics for Sidaamu Afoo and other related languages.

Data Collections

For the training and testing our study, we will collect the data by reading different primary and secondary source of Sidaamu Afoo documents as well as interviewing the people that are familiar with the language. It is also possible to collect corpus from Sidaamu Afoo department at Hawassa university (if present) and we will communicate with the experts for a further morphological rule of the language.

Prototype Development

Based on the experiment result which is obtained from the training and testing data, the prototype will be developed. Firstly, we should develop a prototype for developing a morphological analyzer for Sidaamu Afoo and examine the study with different methodology and techniques. Finally, the evaluation is drawn from the training and testing data by using different evaluation technique and the evaluation result is reported with general performance.

1.6 Scope and Limitations

The Scope of the study

The scope of this study is determining the inflections of the word categories of verbs, nouns, and adjectives. The study should not discuss the compound word formation. Our work proposes that the only study of a morphological analyzer and would not address the study of morphological synthesizer. Also, it doesn't include the other NLP applications except morphological analyzer.

Limitation of study

The main limitation of this study is the absence of enough resources. This means there are no well written linguistic materials in Sidaamu Afoo and there is no suitable lexicon for Sidaamu Afoo which is prepared and stored previously. The preparation of lexicon is very difficult

because the affixation is not the same for all verbs and nouns as well as adjectives. Also, it is difficult and time-consuming to build lexicon manually.

1.7 Application of Results

Incorporating morphological analyzer in NLP improves the performance of the NLP applications. There are so many applications that bases on the morphological analyzer contributions; the output of a morphological analyzer is the input for other NLP applications. These applications that takes morphological analysis's output as an input are spell checker, grammar checker, machine translator, information retrieval, lexical dictionary, etc. Therefore, the application result of a morphological analyzer is to develop the spell checker, grammar checker, dictionary (lexical), machine translator, etc. In addition, it may contribute as teaching and learning process of Sidaamu Afoo.

1.8 Organization of the thesis

The rest of thesis is organized as follows: Chapter 2 discusses the concept of morphology, different approaches to morphological analyzer, memory-based learning and morphology of Sidaamu Afoo. In Chapter 3 related work are discussed. The architecture of Morphological analysis for Sidaamu Afoo with different algorithms used in implementation is discussed in Chapter 4. Chapter 5 discusses the experimental results of the system. Conclusion and future works are discussed in Chapter 6.

2. Chapter Two: Literature Review

2.1 Introduction

Morphological analysis is the base for one language in natural language processing, it is a prior task in NLP. Morphological analysis studies the internal structure of word in a certain language. It is the segmentation of words into their component morphemes and the assignment of grammatical morphemes to grammatical categories and lexical morphemes to lexemes [13]. Morphological analyzer is a computer program which takes a word (i.e. surface level) as input and produces its grammatical structure (lexical level) as output. It will return its stem word along with its grammatical information depending upon its word category [6]. The main objective of this study is developing morphological analyzer for Sidaamu Afoo.

In this chapter, basic concepts of morphology and different approaches to morphological synthesis are discussed. Section 2.2 discusses concepts and terminologies of morphology and Section 2.3 discusses the different approaches of morphological analysis that have close relevancy to this study.

2.2 Concepts and Terminologies

Linguistics is concerned with languages and their structures. It studies the languages at different levels such as at phone, word, sentence and the like. There are also different branches in Linguistics to deal with specific features of a language. The four major branches of linguistics commonly available in the literature are phonology, morphology, syntax and semantics [14].

Phonology concerns sounds (or phones) and their features. The general understanding in linguistics is that human utterance is produced from distinct sounds (or phones) that in combination are used to form words or any other meaningful linguistic entity [14]. Morphology is the study of word formation and structure. In synthetic languages, a single word stem (for example, walk) may have a number of different forms (for example, walks, walking, and walked). However, for some purposes these are not usually considered to be different words, but rather different forms of the same word. In these languages, words may be considered to be constructed from a number of morphemes. Morphology is the study of the way words are built up from smaller meaningful units, morpheme [2]. There are a lot of questions that takes in to account when we design morphological analysis. These are: what

pieces does a word has? what does each of them mean? how they are combined and form a word? [6].

Syntax concerns the combinations of words as phrases and phrases as sentences and the rules that govern these combinations. Semantics is concerned with word and sentence meanings and their interpretations.

2.2.1 Morphemes

The basic building blocks in morphology are Morphemes. Realizing morphemes as part of a word is called morph. Often, there is a one-to-one relation between morpheme and morph. For instance, the morpheme /door/ is realized as the morph /door/ [15]. Morpheme is the smaller units which are used to form words. For example, when we see the words /*iteemmo/,* it is produced from /*it/,* and /*–eemmo*/ and */helps*/ is produced from /help/ and /-s/. These are able to act as words in isolation and bound morphemes can operate only as parts of other words [6]. Word and morpheme are not meaningfully the same, a morpheme may or may not stand alone but word can give meaning anytime. One or several morphemes compose a word. The process involved in forming words from one or more morphemes is called **word formation** [14]**.** In word formation, a morpheme can make some changes or can have a number of ways to form the word. For instance, /*door/* and /*baatto/* forms a word /*door*/ and /*baatto/* respectively without any change. But, if we take a plural in English, it is represented in a number of ways such as /-s/ as in /dogs/, through a morph /-en/ as in /oxen/, through stem vowel alteration as in /men/ and /women/ or through zero morph as in /sheep/ likewise in Sidaamu Afoo, in a number /*-uwa/ubba/* as in /*su'muwa/ 'names'* and /*dargubba/ 'places'*, through word alteration as in /labballo/ 'men' and /*meento/ 'women'*. Such process of making a morpheme a word (or word components) is called morpheme realization. Moreover, all different forms used to represent a morph are called **allomorphs.** In English, Plural representing forms can be taken as allomorphs. [14].

As Yitayal Abate and Yaregal Assabie [6] present, Words can be formed from morphemes in two ways. Based on these way morphemes can be classified into two:

- Derivational Morpheme
- Inflectional Morpheme

Derivational morphemes can be added to a word to create (derive) another word (for example, generating a verb from a noun: the addition of /*-ness*/ to /*happy*/ to give /*happiness*/). They

carry semantic information. On the hand, 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 (e.g., The */walk/* morpheme with the plural marker morpheme */-s/* becomes /*walks*/). They carry grammatical information [16].

As Tesfaye Bayu [14] presents, the two morpheme processes differ in the type of words they produce. Here examples that show both processes: -

1) *en* + *large* = *enlarge*

bound morph Adj. verb

duumo (*Adj.*) +*i* (*bound morph*) =*duu'mi*(*verb*)

} Derivational process

2) *help* + *ed* = *helped*

verb morph bound verb

sirb(*verb*)++*ino*(*bound*)=*sirbino* (*Verb*) *'he sang'*

} Inflectional process

There are three types of morphological structure. These are: - Isolating, agglutinative, and inflectional [17]. Languages with isolating morphological structures have morphemes representing words in the language in most cases. There is little or no morphological change in words, and such languages do not require extensive study on morphological analysis. An isolating language is a type of language with a very low morpheme per word ratio and no inflectional morphology whatsoever [18]. For example, Yoruba is a language. Languages with agglutinative morphological structures have words formed from lots of morphemes that are glued together. Words in these language groups have lots of easily separable morphemes. For example, languages like Sidaamu Afoo is the language that have agglutinative morphological structure as it has only suffixes inflections [8].

In languages with inflectional morphological structures, morphemes are fused together and require complex morphological analyzer to separate morphemes. Morphemes may be fused together in several ways such as affixation and doubling all or part of a word. For example, the languages like Amharic and Afan Oromo is the languages that have inflectional morphological structure [3, 11].

Morphemes can be classified in to two ways.

- Free vs bound morpheme
- Root, Affixes and combining forms

A free morph may form a word by its own while bound morphs occur only in combination with other forms. For instance, the morph /*door*/ stands as a word /*door*/ by itself; so, it is a free morph. But /-*s*/ cannot stand alone as an independent word; hence it is a bound morph. In word forms such as /doors/, for example, there are two morphemes the free morph /*door/* and the bound morph /-*s*/, of which the morph /-*s*/ cannot stand by itself [3, 11]. Hence, the majority of affixes are bound morphemes.

Morpheme can also be further categorized into affixes, roots/stem and combining [17].

Roots/Stems morphemes that makes the most precise and concrete contribution to the word's meaning, and is either the sole morpheme or else the only one that is not an affix. For example, /*walk/* in /*walks/* and /*hiiqq/* in /*hiiqqitu/ and /hiiqqi/.* In Sidaamu Afoo there is no difference in stem with root unlike Amharic or Ge'ez which has root with the stems.

Affixes - In languages with a reasonably rich morphology, affix ordering is an important issue for morphological analysis. Affixation is a common process used in word formation [6]. Bound morphemes that either precede, follow or are inserted inside the root or stem are affixes. Affixes also classified into four parts (i.e. prefix, infix, suffix, circumfix).

- Prefix-***en***- in ***en****able* and ***di***- in ***di****hiiqq* is the affix that precedes the stem /*able/* and /*hiiqq/.*
- Suffix- ***-s*** in *walk****s*** and ***-tu*** in ***ittu*** is the suffix that follows the root/stem /*walk/* and /*it/.*
- Infix- ***-qqa-*** in *hii****qqa****qq* is the infix that inserted inside the root/stem /*hiiqq/.*
- Circumfix- ***di- -ino*** in ***di****hiiqq****ino*** is an affix that precedes and follows the stem /*hiiqq/.*

Combining Forms - morphemes that are formed from two bound or free-like roots.

e.g. two free roots: /*photo/* and /*graph/* in /*photograph/*

two bound roots: *electro-*/ and /*-lysis*/ in /*electrolysis/*

bound and free roots: /*Ethio-*/ and /*America/* in /*Ethio-American*/

In addition, bound morphemes can also be further categorized into two groups as affixes and contracted (clitics) forms [6]. Affixes have been discussed earlier shortly. As Yitayal Abate and Yaregal Assabie [6] present, a clitic is an element that behaves like an affix and a word. For example, English has an obvious clitic, the /**'s/** used to denote the possessive. Sidaamu Afoo has different clitic and affixes morphemes (see Chapter 2.5.4). Reduplication is a border case of affixation. The form of the affix is a function of the stem to which it is attached, i.e., it copies (some portion of) the stem. Reduplication is used in Amharic to show an action done repeatedly [8]. For example, in Amharic, /ሰበረ - ሰባበረ/, in Sidaamu Afoo, */bada/ 'to separate' -/babbada/ 'to separate repeatedly'.*

2.2.2 Morpho-Phonemics

Morpho-phonology or Morpho-phonemics studies the phonemic changes when a morpheme is inflected with another [16]. There are different types of morphophonemic rules in Sidaamu Afoo; epenthesis, metathesis, a set of assimilation rules, and others. All of the rules operate on verbs, and one of them concerns nouns and adjectives as well. Most of them are motivated by the phono-tactics of Sidaamu Afoo language [8]. For more detail (see Chapter 2.5.2).

2.2.3 Morphotactic

Languages have patterns or structures in which morphs are combined to form words. These word patterns (or word structures) are called morphotactic [14]. Morphs must somehow be put together to form words. A word grammar determines the way this has to be done is also morphotactic. The model of morpheme ordering that explains which classes of morphemes can follow other classes of morphemes inside a word. For instance, the fact that the English plural morpheme follows the noun rather than preceding it, morphotactic fact [2]. For example, the English word pseudohospitalization is formed from /pseudo-/, /hospital/ /-ize/ and /-ation/. But these morphemes can be randomly arranged as /**hospitalationizepseudo/*, /**pseudoizehospitalation*/, /**pseudohospitalationize/* if such a morphotactics exits in the language. Also, in Sidaamu Afoo, from morpheme, /*di*-/, /*it*/ (*eat*), and /-*i*-*s*-*ino*-*se*/ different word forms can be formed as /*itdisinose/, /isinoseitdi/, and /disinoseit*/ if there is no morphotactic, but the grammatically correct one is /*ditisinose*/ (he not make her to eat). Therefore, the morphemes of a word cannot occur in random order [16]. Morphotactics is responsible for governing the rules for the combination of morphs into larger entities. But phonological rules may apply and change the shape of morphs. Morphophonology, a discipline that merges morphology and phonology, deal with these changes and their underlying reasons [15]. Therefore, a system for morphological analysis should consider the morphotactic, morphophonological and phonological features in order to generate linguistically acceptable word forms. This process is common for Sidaamu Afoo also, it described in Chapter 2.5.2.

2.2.4 Types of Morphology

There are two productive ways to form words from morphemes: inflection and derivation [15]. So, we have correspondingly inflectional and derivational morphology.

Inflectional Morphology: deals with the combination of a word with a grammatical morpheme, usually resulting in a word of the same class as the original stem, and serving some syntactic function, for example, plurals of nouns. They do not change the part-of-speech category but the grammatical function (also called morphosyntactic information) is changed [3, 13]. Many inflectional features appear on words to express agreement (person, number, and gender) as well as to express case, aspect, mood, and tense [6]. For example, in English, the word /*work/* is a verb, and inflectional forms like /*works*/, /*working*/, and /*worked*/ and in Sidaamu Afoo the word */hiiqq/* is a verb, and inflectional forms like */hiiqqitu*/, */hiiqqanni/*, */hiiqqino/* are produced by adding the 3rd person singular maker /-s//-*itu(F.)*/, the present continuous marker /-ing//-*anni*/ and the perfective /-ed//-*ino*/ respectively. These four-word forms of /*work/* and /*hiiqq*/ (i.e., /*work, works, working*/, and */worked*/ and /*hiiqqi/*, */hiiqqitu/*, */hiiqqanni/*, and /*hiiqqino*/) are all verbs and there is no change in the part-of-speech category due to the affixation.

On the other hand, **derivational morphology** creates new words (i.e., words with a different part-of-speech category) by adding a bound morpheme to a stem. Derivation can be applied recursively, i.e., words that are already the product of one derivation process can undergo the process again.

For example from English: /*large*/ (adj.) →/*enlarge/* /en- + large/ (v) →/*enlargement/* /enlarge+-*ment*/ (noun) and from Sidaamu Afoo: /*waajjo/*(adj.) 'white, pale'→/*waajjidhu/* */waajjo+-i-dh-u/(v) ' to become white, pale'.*

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; in English, /*dogcatcher/* and in Sidaamu Afoo, /*haqqiboce/* is therefore a compound, because both /*dog/* and /*catcher/*, and /*haqqa/* and */boce/*, 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 respectively [6]. The rules of compound word formation in Sidaamu Afoo is unpredictable, and thus needs further linguistic study in the language [8]. Hence, it is out of the scope of this research.

2.2.5 Computational Morphology

Computational morphology is the branch of computational linguistics concerned with word structure. It deals with developing theories and techniques for computational analysis and synthesis of word forms [6]. On the other hand, Computational morphology is intended to handle the task of morphology automatically with the use of computers and computational methods. Generally, the tasks involved in computational morphology can be grouped into two parts [15]:

a) Word-form synthesis and analysis;
b) Parts-of-speech (POS)-or inflectional-category determination.

Word-Form Synthesis and Analysis

Synthesis or generation is the processes of producing word forms from their constituent morphemes, by tokenizing word forms into their ingredient morphemes, but analysis or recognition does the reverse process. Morphological analysis, by which a surface word form such as /cries/ and /*hiiqqitu*/ is analyzed into a lexical representation (such as /cry/ + /s/ and /*hiiqq*/+/*i*/+/*tu*/), consisting of the word's component morphemes or grammatical features. Whereas, morphological generation, by which a lexical representation (such as /cry/ + /s/ and /*hiiqq*/+/*i*/+/*tu*/), is converted to a surface word (such as /cries/ and /*hiiqqitu*/) form. These processes demand identification of word form components (for example stems and suffixes) and taking account of the regular phonological or orthographical alternations due to morphological, and morphophonological processes involved [6].

PoS or Inflectional Category Determination

Computational morphological systems operate as a morphological front end of syntactic parsers. Such syntactic parsers need not only tokenized word forms but also determine the PoS or inflectional category of the entire word. The PoS tag or inflectional category of words is often taken from a morpheme that serves as a "head of a word" for that particular word/word form [19]. For example, the English word /*sadness*/ comes from the following morphemes: the adjective stem /*sad*/ and the noun marker suffix /*-ness*/. Of these morphemes, the noun-marker suffix /*-ness*/ is a head of the word /*sadness*/, and thus, the entire word is a noun [15].

Likewise, in Sidaamu Afoo a word /*Feyaancho*/ is comes from the morphemes: /*fey*-/ the verb stem and /–*aancho*/ nominalizer marker suffix, here the nominalizer marker /–*aancho*/ is the head of the word /*Feyaancho*/.

2.2.6 Morphological Analyzer

Morphological analysis is an essential component in language engineering applications ranging from spelling error correction to machine translation. It is an analysis of internal structure of words. For languages with rich morphology like Sidaamu Afoo, using the morphological analysis is possible to figure out if a word can be a noun or a verb, or if it can be singular or plural [20]. Performing a full morphological analysis of a word form is usually regarded as a segmentation of the word into morphemes, combined with an analysis of the interaction of these morphemes that determine the syntactic class of the word form as a whole [21]. As Yitayal Abate and Yaregal Assabie [6] described, Morphological analyzer is a computer program for analyzing the morphology of an input word; the analyzer reads the inflected surface form of each word in a text and produce its lexical form after analyzing it [22]. Therefore, the role of morphological analyzer is very important in the field of natural language processing (NLP) applications like machine translation (MT), information extraction (IE), information retrieval (IR), spell checker, lexicography, etc. Morphological analysis deals with the process of splitting given words into their constituents, called **morphemes**, and describing how words are built by combining these morphemes. The morphological analyzer maps an inflected word into its stem, parts of speech and feature equations corresponding to inflectional information.

Traditionally, morphological analysis has been done in a manual fashion. 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 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 [23]. The morphological structure of an agglutinative language is unique and capturing its complexity in a machine analyzable and generatable format is a challenging job [6].

To analyze a given word, morphological analyzers should have basic knowledge about the language which is common to most analyzers. As it is described in [24], to analyze certain

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;
- 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

2.3 Approaches to morphological analyzer

According to [25], the different approaches to morphology are categorized as corpus based and rule-based.

2.3.1 Rule based

Rule-based approaches are based on a theory or rule of morphology laid down by experts for a certain language. It enables one to incorporate sophisticated linguistic theory, such as generative phonology, into computational morphology processes. Because of their reliance on linguistic theories, systems developed using such approaches are often efficient and produce better quality outputs [15]. According to [3], The rules are either created by linguists or automatically by computer programs, and may contain a large number of morphological, lexical and/or syntactical information. With human rule creation, there is a large set of manually constructed rules based on a specific grammar, written in a formal notation so that they can be used by the computer for further parsing. Adding a rule to the system may involve over-generation (i.e., one extra rule can result in more harm to the accuracy of rule-based approach). The rule-based tagging system has also some limitations: requires hand-written rules, costly and time consuming.

2.3.2 Machine Learning

Corpus-based approaches do not strictly follow explicit theory of linguistics. Corpus-based approaches, also called machine learning approaches. The approaches are completely based on training and testing corpora, which constitute the input data. Machine learning deals with techniques that allow computers to automatically learn and make accurate predictions based on past observations [16]. 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 (i.e. speech recognition,

morphological analysis, document categorization, document segmentation, part-of-speech tagging, and word-sense disambiguation, named entity recognition, parsing, machine translation and transliteration.).

As Yitayal Abate and Yaregal Assabie [6] present, 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 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 [16]. 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 unlike rule-based. Based on information type (i.e. based on the type of corpora they used) employed in the machine learning task, machine learning can be classified into two: unsupervised learning and supervised learning. Recently, machine learning approaches are found to be dominating the Natural Language Processing field [6]. 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 or annotated corpora [6]. Henceforth, we use this approach for our study.

2.3.2.1 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. Unsupervised approaches use heuristics or probability information

generated from the test corpora to generate the morphological analysis system [15]. In this approach, no sample outputs are given. Hence, this approach reduces the cost of browsing annotated corpora.

As Yitayal Abate and Yitayal Abate [6] discussed, Unsupervised learning seems much harder: the goal is the computer must learn how to do something without 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. We don’t discuss this approach in detail because we don’t use this approach.

2.3.2.2 Supervised Learning

Supervised approaches use annotated text corpora while unsupervised approaches use natural corpus as those found in newspaper and books. Annotated text can be word-forms tokenized into constituent morpheme by human expert or words with their grammatical properties assigned pre-hand [27]. 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 [28].

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 [6]. 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 [28]. 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. In supervised learning the target function is completely specified by the training data. There is a label associated with each example. If the label is discrete, the task is called classification. Otherwise, for continuous-valued (real) valued labels, the task becomes a regression problem. Based on the examples in the training data, the label for new case is predicted. Hence, learning is not only a question of remembering but also of generalization to unseen cases.

As Anand [16] described, any change in the learning system can be seen as acquiring some kind of knowledge. So, depending on what the system learns, the learning is categorized as

- Model Learning: The system predicts values of unknown function. This is called as prediction and is a task well known in statistics.
- Concept learning: The systems acquire descriptions of concepts or classes of objects.
- Explanation-based learning: Using traces (explanations) of correct (or incorrect) performances, the system learns rules for more efficient performance of unseen tasks.
- Case-based (exemplar-based) learning: The system memorizes cases (exemplars) of correctly classified data or correct performances and learns how to use them (e.g. by making analogies) to process unseen data.

As Yitayal Abate and Yaregal Assabie [6] descried, A supervised learning algorithm takes a set of training examples or instances as input and produces a classifier or predictor as output after learned from examples. The set of training examples provides two kinds of information. First, it tells the learning algorithm the observed output values *yi* for various input values *xi*. 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 [29]. 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 predetermined classifications [30]. 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 [16]. It is an algorithm that analyzes data and recognizes patterns from those data. In SVM the basic

idea is to represent the examples as points in space, making sure that separate classes are clearly divided [31]. SVM 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 [32].

Inductive Logic Programming (ILP): is a supervised machine learning framework based on logic programming. It is an algorithm which learn first-order decision lists represented as ordered Prolog clause ending with cut. In ILP a hypothesis is drawn from background knowledge and examples. The examples, background knowledge and hypothesis all take the form of logic programs. The background knowledge and the final hypothesis induced from the examples are used to evaluate new instances [10].

Hidden Markov Model (HMM): Hidden Markov Model is one of the most important machine learning models in speech and language processing. HMM is a probabilistic sequence classifier, given a sequence of units and its job is to compute the probability distribution over possible labels and choose the best label sequence [33].

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 as it proposed to learn the NLP classification Problems. It is a classification based, supervised learning approach: a memory-based learning algorithm constructs a classifier for a task by storing a set of examples [34]. Each example associates a feature vector (the problem description) with one of a finite number of classes (the solution). Given a new feature vector, the classifier extrapolates its class from those of the most similar feature vectors in memory [35]. 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 [36]. 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. It is a supervised inductive learning algorithm for learning classification tasks [37]. 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 [38].

Memory-based learning has a promising feature in analyzing NLP tasks like part-of-speech tagging, text translation, chunking and morphophonology due to its capabilities of incremental learning from examples. part-Of-Speech Tagging e.g., [37, 39,40], Grammatical Relation Finding, e.g., [41], Shallow Parsing (combining memory based tagging, chunking, PP finding, and grammatical relation finding), e.g., [35,42,43], Morphological Analysis, e.g., [6,11,36,37,44], Phoneme-To-Grapheme Conversion, e.g., [45], Pitch Accent Placement, e.g., [46], Semantic Role Labeling, e.g., [47], Word Sense Disambiguation, e.g., [48,49], Named Entity Recognition, e.g., [50], PP Attachment Disambiguation, e.g., [51], Co-Reference Resolution, e.g., [52], and Information Extraction, e.g., [53].

The major techniques that employed by memory-based learning system are the nearest neighbor search, space decomposition techniques, and clustering. An example memory-based learning system is shown in Figure 2.1. 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 memory-based learning systems can be organized as hash tables, trees, or full-search tables. Source [54].

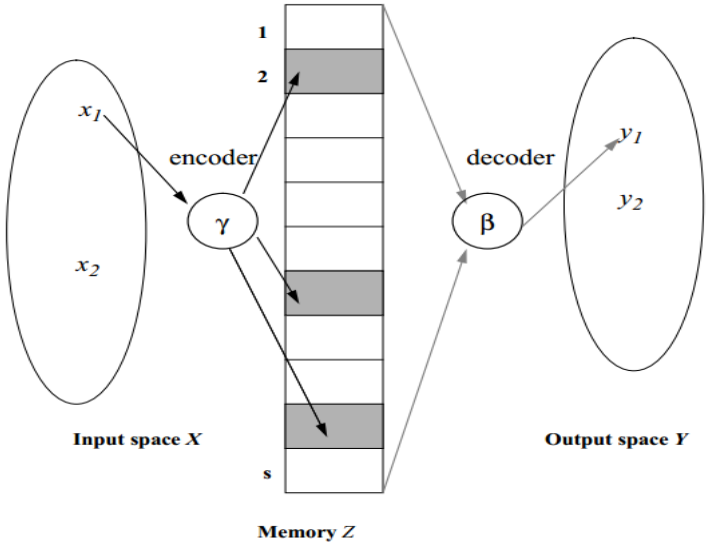

Figure 2.1 An example memory-based learning system.

As Yitayal Abate and Yaregal Assabie [6] discussed, 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 for a training purpose, 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 according to a distance metrics and weight.

The General architecture of an MBL system which is shown in Figure 2.2, contains two components: a learning component which is memory-based (from which MBL gets its name), and a performance component which is similarity-based. Source Adopted from [55].

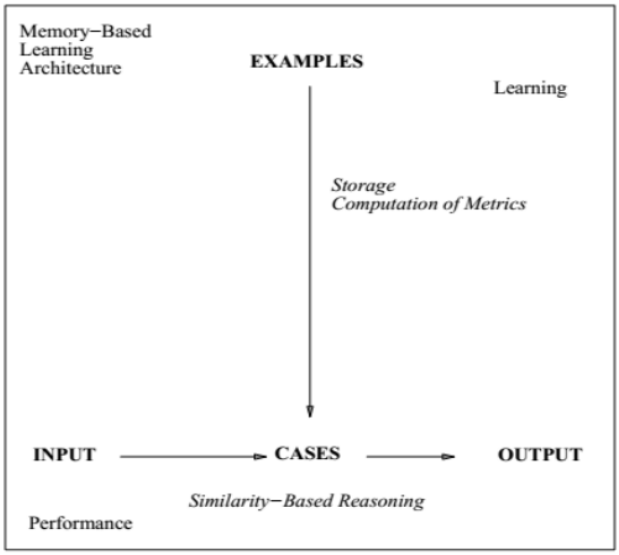

Figure 2.2 General Architecture of an MBL System

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. But, 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 [55]. Memory-based learning has the following advantages in contrast to most other machine learning algorithms: It presupposes no more linguistic knowledge than explicitly present in the corpus used for training, i.e., it avoids a knowledge-acquisition bottleneck(but to prepare a morphologically annotated corpus it is mandatory to have knowledge about a language); 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 forms being processed [6]. Memory-based techniques, based on principles of non-parametric density estimation, are a powerful form of machine learning well-suited to natural language tasks [56]. Because of these advantages, it is selected to test the morphological analysis of Sidaamu Afoo.

As explained in [6], the major disadvantage of memory-based learning, compared to most opponent 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.

2.4 Memory-Based Language Processing

A memory-based learning system is an extended memory management system that decomposes the input space either statically or dynamically into sub regions for the purpose of storing and retrieving functional information. The main generalization techniques employed by memory-based learning systems are the nearest-neighbor search, space decomposition techniques, and clustering [54]. Memory-based learning algorithms can learn mappings (classifications), if a sufficient number of instances of these mappings is presented to them [38]. As Walter and Antal [57] present, 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 [57].

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.

As Bosch and Walter [44] present, Memory-based learning is a class of inductive, supervised machine learning algorithms that learn by storing examples of a task in memory. Computational effort is invested on a “call-by-need" basis for solving new examples (henceforth called instances) of the same task. When new instances are presented to a memory-based learner, it searches for the best matching instances in memory, according to a task-dependent similarity metric. When it has found the best matches (the nearest neighbors), it transfers their solution (classification label) to the new instance. Memory-based learning has been shown to be quite adequate for various natural-language processing tasks such as stress assignment [58], grapheme-phoneme conversion [59,60], and part-of-speech tagging

[61]. Memory based learning and problem solving incorporates two principles [6]: 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. 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. To analyze natural language processing (NLP) problems from spell checker to machine translation and automatic extraction of knowledge from text, memory-based learning has a great future potential as many researches implies [36,42,62]. Memory-Based Learning (MBL) is an elegantly simple and robust machine-learning method applicable to a wide range of tasks in NLP [55].

An experiment that has tried to explain so far with memory-based learning described here were carried out using the Tilburg Memory-based Learner tool, TiMBL. This software package is developed and maintained by the Induction of linguistic knowledge group at Tilburg University center for Language and Speech Technology, and is a very efficient feature rich implementation. Firstly, it is developed for Dutch language and later it be a contribution for different language. TiMBL is free software and any one can redistribute it and/or modify it under the term of GNU General Public License as published by the free Software foundation [63]. 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 [57]. 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 [64]. Several wrappers, bindings and other extensions to TiMBL have been developed: for instance, TimblServer, python-timbl, Dimbl, paramsearch, rtimbl, Timpute, knngraph. Also, TiMBL is a core component of various NLP software systems such as MBT(Memory-based tagger generator), Frog(Dutch morpho-syntactic analyzer), Gecco (Context-sensitive spelling corrector, used by Valkuil.net for Dutch, and Fowlt,net for English) [63].

TiMBL is an open source software package implementing several memory-based learning algorithms such as **IB1**, IGTREE, **TRIBL**, TRIBL2, IB2, etc., among these IB1 and TRIBL

are used in this study. IB1 is an implementation of k-nearest neighbor classification with feature weighting suitable for symbolic feature spaces, and TRIBL 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 [36,65].

For last fifteen years, TiMBL has mostly used as a machine learning classifier component in natural language processing. But it its use is encompasses to virtually any supervised machine learning domain. TiMBL is in many cases for more efficient in classification than a standard k-nearest neighbor algorithm would be, due to its particular decision-tree-based implementation [63].

KNN:

The k Nearest Neighbors algorithm (kNN for short) is one of the nearest neighbor Rule from different types of nearest neighbor rules (i.e. Condensed Nearest Neighbor Rule CNN, Selective Nearest Neighbor Rule SNN, Reduced Nearest Neighbor Rule RNN, Edited Nearest Neighbor Rule ENN, All k-NN) [66]. It is an instance-based, or a lazy learning method. As Debo et.al [67] propose, it has been regarded as one of the simplest of all machine learning algorithms. The rational of kNN is that similar samples belonging to the same class have high probability, while the key idea of kNN algorithm is to first select k nearest neighbors for each test sample, followed by using the learnt k nearest neighbors to predict this test sample. Therefore, kNN algorithm was often thought as an algorithm, in which no explicit training step is required. It is a non-parametric classification method, which is simple but effective use in many cases in machine learning [68]. The K Nearest Neighbor (kNN) method has widely been used in the applications of data mining and machine learning due to its simple implementation and distinguished performance. However, to apply k-NN it is better to choose an appropriate value for k. We can choose the value of k by running an algorithm recursively with different k values and then it is simple to choose the k value with the best performance [6].

An MBL system, as describe in Figure 2.2 above, contains learning and performance components. A learning component which is memory-based (from which MBL its name got) and a performance component which is similarity-based. In MBL input taken and generate

output after learning and trained after taking input. For the basis of mapping input to output is the result of learning component and taken as the form of performing classification [69]. During classification, a previously unseen test instance is presented to the system. The similarity between the new instance X and all examples Y in memory is computed using a *distance metric* Δ(X; Y). The extrapolation is done by assigning the most frequent category within the *k* most similar example(s) as the category of the new test example [70]. To calculate the distance between two example X and Y, represented by n features, and δ is the distance per feature, the distance metric should be used through equation 1 given below. The distance between two patterns is simply the sum of the differences between the features. The value of ***k*** refers to k-nearest distance rather than k-nearest cases [70]. With k=1, for example, TiMBL's nearest neighbor set can contain several instances that are equally distant to the test instance. As Walter et. Al [65] present, their K-NN kernel could therefore be called k-nearest distances classification.

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

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

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

The k-NN has its own disadvantage: it is computationally expensive since it saves all training instances. It also not good with noisy attribute values and irrelevant attributes. Its performance depends on the choice of the similarity function to compute distance between two instances. There is no simple or natural way to work with nominal valued attributes or missing attributes. No one can know how much data is structured [71]. Even if k-NN has these disadvantages, k-NN has extent advantage. As Aha et al., [71] stated, k-NN is supervised learning algorithms that are also used in machine learning where instances are used incrementally to classify object previously unseen. Its cost for updating object instances is low and learning rate or speed of learning or training is fast. It also possible to extend the algorithm to obtain concept description. In addition to this, there is a number of advantages of k-NN classifier has chosen for memory-based learning during machine learning while compared to other discriminative supervised machine-learning algorithms as stated in [57]:

- 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;
- 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;
- 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 decrement allowing for fast leave-one-out testing.
- 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.

To overcome those disadvantage of k-NN listed before, nowadays 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 IB1, IB2, IB3, IGTREE, TRIBL, TRIBL2 algorithms implemented in TiMBL [6].

IB1: (Instance Based learning algorithm 1)

Instance-based Learning (IBL) is classified into lazy classifier for storing all of training instances and does not really work until the classification process runs. IB1 relatively accurate in classifying but require a large storage space for storing all the training instances. IBL algorithms classify instance by comparing it with training data (predefined class) so that a similar instance will have a similar classification using the similarity function [72]. 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 IB1 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 [6]. The IB1 algorithm, is the simplest instance-based learning algorithm. The similarity functions that IB1 used is:

$$\text{Similarity}(x,y) = -\sqrt{\sum_{i=0}^{n} f(x_i, y_i)} \qquad (2)$$

Where the instances are described by n attributes. We define $f(xi,yi) = (xi-yi)^2$ for numeric-valued attribute and define $f(xi,yi) = (xi \neq yi)$ for Boolean and symbolic-valued attributes.

IB1 is identical to the nearest neighbor algorithm except that it normalizes its attributes' ranges, processes instance incrementally, and has a simple policy for tolerating missing values [73]. In lazy learning (e.g., the IB1 algorithm in Aha et.al [71]), similarity of a new instance to stored instances that is trained one, is used to find the nearest neighbors of the new instance that should be tested one. The classes associated with the nearest neighbor instances are then used to predict or classify the class of the new instance previously unseen one. In IB1, all features are assigned the same relevance, which is undesirable for some linguistic problems [74].

IB2

The IB2 algorithm is incremental [66]. It is identical to IB1 except that it saves only misclassified instances. IB2's storage requirements can be significantly smaller than IB1's in those applications where the instances vary greatly in their distance from the concept boundary. It can significantly reduce IBl's storage requirements [73]. The IB1 algorithm stores all the instances of the training data in the memory. This implies that all instances, even those that fall well within the borders of the target description, are stored 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 [55]. This is the basis of the operation of the IB2 algorithm. 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. 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, it has a strategy of only storing misclassified instances [71].

IB3

The IB3 algorithm (Aha et al. [71]) is another incremental algorithm that addresses IB2's problem of keeping noisy instances by retaining only acceptable misclassified instances [66]. IB3 had higher accuracy and lower storage requirements in the presence of noise than IB2, though it still suffered a dramatic decrease in accuracy (and a slight increase in storage) when compared to its performance in the noise-free case.

As David et al., [73] present, IB3 is an extension of IB2 that employs a "wait and see" evidence gathering method to determine which of the saved instances are expected to perform well during classification. Its similarity function is identical to IB2's. Its classification function and updating algorithm differ as follows:

1. IB3 maintains a classification record (i.e., the number of correct and incorrect classification attempts) with each saved instance. A classification record summarizes an instance's classification performance on subsequently presented training instances and suggests how it will perform in the future.
2. IB3 employs a significance test to determine which instances are good classifiers and which ones are believed to be noisy. The former is used to classify subsequently presented instances. The latter are discarded from the concept description

IGTREE

As stated in [74], 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 reorganization 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. [6]. Instead of converting the instance base to a tree in which all instances are fully represented as paths, storing all feature values, Walter et.al [74] compress the tree even more by restricting the paths to those input feature values that disambiguate the classification from all other instances in the training material.

TRIBL

It is an approach called the *Typical Instance Based Learning* (TIBL) algorithm, which attempted to save instances near the center of clusters rather than on the border. This can result in much more drastic reduction in storage and smoother decision boundaries, and is robust in the presence of noise. It is designed as a hybrid generation of IB1 and IGTREE with the aim to exploit the tradeoff between the search speed of IGTREE and the generalization accuracy of' IB1. During search, the normal IGTREE search algorithm is used, until the case-base nodes are reached, in which case regular IB1 nearest neighbor search is used on this sub-case-base [70]. The *typicality* of an instance is defined as the ratio of its average similarity to instances of the same class to its average similarity to instances of other classes [66]. The *similarity* of two instances *x* and *y* is defined as 1 - distance(*x*,*y*), where

$$\text{distance}(x,y)=\sqrt{\frac{1}{m}\sum_{i=0}^{m}\left(\frac{xi-yi}{maxi-mini}\right)^2} \qquad (3)$$

TRIBL2

It also designed as a combination between IB1 and IGTREE with the aim to exploit the tradeoff between the search speed of IGTREE and the generalization accuracy of' IB1 like TRIBL. it works 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 [6]. 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. TRIBL2 does not employ a fixed switching point like TRIBL algorithm even has a close similarity. 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 [70].

Thus, from the described algorithms, we have chosen IB1 and TRIBL. The reason is that what they give the result of generalization accuracy, learning and testing speed, compression. IB1 performs a better result, rather it contains larger storage. To handle this, we have used TRIBL as an additional algorithm and tuned optimized parameters to both algorithms. The other option to reduce the storage size is using IB2. But, rather only focusing on the reducing of the storage, we focused on is instances are correctly classified or not? Although, it has no comparable result with IB1 by general accuracy. IB1 achieves better performance than other classifiers. For this reason, we have chosen IB1 and TRIBL (to reduce storage) with default and optimized parameters. To achieve the better performance of generalization accuracy and learning speed we chose IB1 and TRIBL algorithms for our study.

2.5 Morphology of Sidaamu Afoo

Sidaamu Afoo belongs to High Land East Cushitic language and closely related to Kambaatisa (62%), Halaaba (Alaba) (64%), Hadiyyisa (53%), Kabeena (64%), Gedeuffa (40%), and Burji (41%) [75]. As Tafesse G/Mariam [75] stated, Sidaamu Afoo is witnessing a rapid standardization and development in the last two decades starts from late 1990's. it has been used for different purpose as we have discussed in Chapter 1. Sidaamu Afoo is an agglutinative language, where almost all derivational morphology and all inflectional morphology involves affixation. Most affixes are suffixes, though there are case suprafixes [8] (discussed in Chapter 2.5.4 briefly). In order to give more emphasis about the morphology of a language that has involved in word formation, it is better to precede the discussion of phonology and word class of a language.

2.5.1 Phonology of Sidaamu Afoo

During inflection, derivation and compounding the segmental phonemes of Sidaamu Afoo and some of the major morphophonemic process is involved [76]. As Anbessa Teferra [76] states, Sidaamu Afoo has Consonantal Phonemes and Vowel phonemes. These Consonant morphemes as shown in Table 2.1. **Source:** Kawachi [8].

Table 2.1 Sidaamu Afoo Consonant Phonemes

		Bilabial	Labio-dental	Dental	Alveolar	Palatal	Velar	Glottal
Stop	Plosive	b		t d			k g	’
	Ejective	P’		t’			k’	
	Implosive			dh				
Affricate					c j			
	Ejective				sh			
Fricative					s z			h
Nasal		m		n		ny		
Tap/Flap				r				
Lateral Approximant				l				
Approximant		w				y		

According to Kawachi [8], Sidaamu Afoo has a the glottalized sonorants **/‘l, ‘m, ‘n, ‘r, ‘y/** are clusters made up of two phoneme segments consisting of a glottal stop and sonorant. From these glottalized sonorants, /**‘m, ‘n, ‘l**/ can occur in open-class words [77]. The glottal stop is a phoneme in Sidaamu Afoo.

Sidaamu Afoo has five short and long vowel phonemes /**a, e, i, o, u**/ and **/aa, ee, ii, oo, uu**/ respectively [78]. Accordingly, Vowel lengthening is phonemic in Sidaamu Afoo as it brings meaning difference. E.g., /***dora****/ ‘red clay soil’,* and */**doora**/- ‘to choose, to elect, prefer’* is different by their meaning. Anbessa Teferra [76] presents, Sidaamu Afoo has both open and closed syllables. Closed syllables occur word internally in words such as /**wor-ba**/ ‘brave’ and /**kub-ba**/ ‘to jump’. Though, open syllables tend to dominate in Sidaamu Afoo since every inflected word ends in a vowel.

Consonants and Vowels Cluster:

Sidaamu Afoo allow the maximum of two consonants successively in consonant sequence. Consonant cluster refers to tauto-syllabic sequences i.e., sequences of consonants within a syllable. All Sidaamu Afoo consonant sequences are hetero-syllabic i.e., sequences across syllable boundaries as consonants cluster [76]. There are three types of cluster sequences: Sonorant-obstruent, sonorant-sonorant, glottal-sonorant [8,76]: these types are:

Sonorant-obstruent: The nasals and the liquids can each be followed immediately by obstruent. Consonants that said to be sonorant are /m, n, r, l/. According to Girum Tesfaye [79], most of Sidaamu Afoo consonant cluster in ideophones are sonorant-obstruent pattern.

e.g., /si**r.b**a/- ‘to sing’, /a**n.g**a/- ‘hand’, /a**l.b**a/- ‘face’, /a**m.b**ooma/- ‘hyena’.

Sonorant-sonorant: There are limited words that contain a sonorant-sonorant sequence. Some of these words are loan words from Amharic or other languages.

e.g., /baa**y.r**a/- ‘elder’, /xa**r.m**use/- ‘(AMH) bottle’.

Glottal-sonorant: e.g., /su**’m**a/- ‘name’, /ki**’n**e/- ‘you-2PL’, /hala**’l**ado/- ‘wide’, /so**’r**o/- ‘mistake’

Although, In Sidaamu Afoo, the cluster of vowels have their own phonological rules. The maximum similar vowel can be cluster in Sidaamu Afoo is two and it never usual be three in a cluster. Sometimes it seems to take more than two similar vowels with different utterance, at that time it must take double glottal stop or single glottal stop to segment these vowels. For instance**, /**kuu**”**u/ ‘those’ **,** /e**’i**no/- ‘she entered’, /e**”i**no/ ‘he entered’, /ko**”**ee/ ‘this one (M)’, /ti**’**i/ ‘that one (F).’, etc. An acceptance of double or single glottal stops are according to the utterance of that word gives. Mostly in Sidaamu Afoo, word final vowel is limited as a single in the words, but sometimes as an exception, those the final vowel to be doubled or lengthen and it is limited in a number. E.g., **lemoo** ‘twenty’, **saa** ‘cow’, etc., and some command giving words like **/agii/** ‘please drink!’, /**itii/** ‘please eat!’ can lengthen their final vowel while they derived to be commanded word. Note that, it is forbidden that only similar vowels can’t come successively being more than two with in one word. But it is possible to come more than two vowels successively, if those vowels are different. For example, /**kooee/** ‘that (far)’ it doesn’t take any glottal stop.

Geminates of Consonants

The consonants */ch, dh, ny, ph, sh, ts, and zh/* are seven two-digit letter that has been used as a single consonant in Sidaamu Afoo, said to be “**siwiilu fidalla**”, their phonetic representation is /č, ɓ, ñ, p’, š, š’, ž/ respectively [78]**.** Hence, except */sh-š/* these consonants cannot be geminate. Because of one or more reasons; it misses the rule that disallowed in the language like the maximum two consonant sequence (i.e., it cannot bring any meaning difference except */sh/* when they lengthen or shorten. E.g., */bu**sh**a/- ‘bad’* and */bu**shsh**a/- ‘soil’*). As Kawachi [8] states, all consonants can be geminate in Sidaamu Afoo. Although, Anbessa Teferra [76] concluded that /h/ cannot be geminated in Sidaamu Afoo. But Kawachi [8] presents, sometimes it can be geminated in example

/aha***hh****e/- 'grandparents',* what we have concluded for this example that /h/ never geminated anywhere in Sidaamu Afoo, the meaning of '*grandparents'* is /*ahaah****uwa****/ahaa****kka****/* not /*ahaahhe*/, it also not give a sense in the language.

Some examples for geminated consonants are, /go**bb**a/ 'country', /bee**tt**o/ 'child', /ha**shsh**a/ (*hašša* by phonetic) 'evening', /duree***ss***a/ 'rich'.

2.5.2 Morphophonemic Rules

There are several morphophonemic rules in Sidaamu Afoo. These are: epenthesis, metathesis, a set of assimilation rules and others. All these rules operate on verbs, and some of them are apply on nouns and adjectives while affixes append with the stem for word formation [8].

i. Epenthesis: inserting vowel /**i**/ between a stem-final consonant cluster/geminate and suffix-initial consonant. Epenthesis is occurred for the following three conditions as Anbessa Teferra [76] presents.

   a. When suffixes such as /-**tV**/ are added to mono-consonantal verb stems and those epenthetic vowel /i/ inserted between stem and suffix. Ø→i/ δ[C1-C2]------rule
      e.g., /**sh-tu**/→/**shitu**/ 'she/they killed'

   b. Epenthetic vowel /i/ inserted between stem and suffix to break the three consonant sequence that misses a rule of maximum of two consonant sequence. This could apply when consonant-initial suffixes are added to a verb stem that ends in consonant sequence or geminates. Ø→i /CC-C------rule e.g., /*sirb-tu*/→/*sirb****i****tu*/ 'she/they sang'

   c. When a single causative /-***s***/ or double causative /-***sii****s*/ is added to stem-final obstruent. **Ø→i /C-s/**----rule. e.g., /it-s/→/itis-/ 'feed'**, /it-siis/→ /it*i***siis-/ 'cause to be eaten'

ii. Metathesis: is a phonological process whereby two adjacent consonant phonemes are transposed. There is nasal and glottal metathesis in Sidaamu Afoo.

   a. Metathesis of a stem-final sonorant and glottal stop (as an allomorph of the MID suffix) [8]. The allomorph of the middle suffix for a verb ending in a sonorant is the glottal stop. The root-final sonorant and the glottal stop usually metathesize when stem takes auto benefactive affix /-*dh*/, with the result that the stem ends in /'m, 'n, 'l/
      e.g., /fan-/ 'open'+ /-ɓ(dh)/→/fa**'**n-/ 'to open for oneself'

   b. Metathesis of a stem-final obstruent is permuted with /n/ of 1PL, 3SGMPPRF suffix -/n – mmo/. When a stem ending in an obstruent (neither part of consonant cluster or geminate)

occurs with the suffix, the obstruent and the /n/ are exchanged to form the cluster, -n-obstruent [8].

e.g., /it-/ ‘eat + /-n –**ummo**/→ /i**n**tummo/ ‘we ate’

iii. Assimilation can be either total or partial and it takes place in order to assure the correct syllable-structure which are permitted in the language. There are also so many types of assimilation rule [8].

a. Lenition of stops (/b/, /k/, /ɓ/) in an intervocalic position. Lenition also said to be weakening. It is a sound change from left to right on the stop-fricative –approximant dimension. These stops (/b/, /k/, /ɓ/) change to (/w/, /h/, /’r/), respectively in intervocalic position [75] [76]. Lenition can be applied to both in verb and nouns [8].

e.g., */duu**b-/** ‘be satisfied’ + /-**i/ →/**duu**wi/** ‘he was satisfied’

*/haɓ**(**dh**)**-/* ‘go’ + /***i/→ /**ha**’ri/** ‘he went’. **/**duk**-/** ‘carry’ +/ ***i/ →/**du**hi/** ‘he carried’.

b. Assimilation of the stem-final /m/ to the dental place of articulation of /tu/ or /tin/ (the 3SG.F/3PL suffix /–tu/ or /tin/ of the 2PL/ 2SG.POL suffix -tin). when a verb stem ending in /m/ (including a stem consisting of a verb root and the passive or reciprocal suffix *–am*) is followed by one of the /t/-initial suffixes, the /m/ changed to /n/ during assimilation to following /t/.

e.g., /*morom*-/ ‘deny’ +/-***tu/*** →/moro**n**tu/ ‘she/they denied’

/*morom*-/ ‘deny’ +/-***tin/*** →/moro**n**tin/ ‘you/they denied’

c. Assimilation of /n/ of the 1PL suffix /-n –mm-/ to the bilabial place of articulation of the stem-final /b/. e.g., /gib-/ ‘refuse’ +/-nummo /→ /gi**m**bummo/ ‘we refused’

d. Total assimilation of /tu/ (the 3SG.F/3PL suffix -tu or /tin/ of the 2PL suffix -tin) to a stem-final obstruent. Here total assimilation is applied, stem-final obstruent forms a geminate and added suffixes to it. In order to avoid more than two Consonant sequence by breaking consonant cluster, suffix initial consonant /t/ should be deleted then added to stem.

e.g., /gib-/ ‘refuse’ +/-**tu/ →**/gib**bu** ‘she/they refused’

/gib-/ ‘refuse’ +/-**tin/ →**/gib**bin/** ‘you (PL) refused’

e. Total assimilation of /n/ of the 1PL, 3POL, 3SGMPPRF suffix -n -mm to a stem-final sonorant. It also named that total sonorant assimilation.

e.g., /kul-/ ‘tell’ + /nummo/→ /kullummo/ ‘we told’

/kul-/ ‘tell’ +/-no/→/kullo/ ‘he has told/ let us tell’

/kul-/ ‘tell’ +/- ni/→/kulli/ ‘he (pol) told’

f. Elision of a vowel or /h/ after the negative proclitic /di-/. According to Anbessa Teferra [76], during fast speech, stem-initial /h/ is elided when preceded by a vowel. A negative prefix (only prefix in Sidaamu Afoo) /*di*-/ is added to verb stem-initial /h/, elision of /h/ is takes place. Also, in vowel elision, a negative prefix /di-/ is prefixed to a vowel-initial verb stem. A vowel /i/ is deleted from the prefix /*di*-/ and vowel-initial stem added to remaining prefix clitic /d-/.

e.g., /di-/+/has-/+/-anno/→/d**a**sanno/ ‘he will not look for’

/di-/+/ag-oommo/ →/d**a**goommo/ ‘I have not drunk’

iv. Others: according to Kawachi [8], there are few morphophonemic processes that cannot be classified as any of the above type.

a. Ejectivization: it takes place when the auto benefactive formative, i.e., /-ɓ(*dh)/* is added to stem-final consonants [75].

e.g., /**dib**-/ ‘cover’→/**dip’** (**diph**)/ ‘cover oneself’

b. Replacement of /ɓ(dh)/ by /’/before /n/ of 1PL suffix /–**n-mm-/**

The middle suffix - ɓ(dh) becomes the glottal stop when followed by the 1PL suffix –n-mm- e.g., /**loos**-/ ‘to work’ + / ɓ(dh)/ + /-**nummo**/→/loos-i-’-nummo/ (V-EP-MID-PRF.1PL) ‘we have work for ourselves’.

2.5.3 Morphological categories of Sidaamu Afoo

According to Anbessa Teferra [76], A lexical category is a set of words which share a common set of grammatical properties. There are five lexical categories in Sidaamu Afoo. These are: nominal, verb, adjective, adverb, postpositions. But, Indrias Xa’miso et al., [78] stated that there are six lexical categories, they added Conjunction as additional sixth category. Each word class has a specific morphological and/or syntactic property and can be classify its own subclass. Verbs, nominal, adverbs, and adjectives can be simple or derived and may inflect for various grammatical categories [78]. Generally, Kawachi [8] presented these word classes in to two super classes i.e., open classes and closed classes. Nouns, verbs, adjectives, and adverbs are mainly classified under open classes. Pronouns and related forms, clitics, and interjections are classified under closed classes. In our study we can’t give more detain explanation about the difference of both classes, rather we will discuss some of word classes i.e. verb, noun, adjective, and its inflections.

2.5.3.1 Nominal

The morphology of nominal is categorized into three sub-classes namely simple nouns, derived nouns and pronouns [78].

A. Nouns

According to Anbessa Teferra [76], in Sidaamu Afoo, a noun in its citation form minimally consists of a noun stem, and a terminal vowel, i.e., the noun stem always ends with a vowel. Most nouns contain a singular marker which always follows the noun stem. All of Sidaamu Afoo nouns ends in a terminal vowel of **/a/, /e/, /o/** in citation form with its order of **noun Stem-Terminal vowel** or **noun stem-singulative.**

Nouns are marked for number, gender, and case grammatically.

Number distinction in Sidaamu Afoo is singular and plural. Singular nouns are unmarked morphologically and most singular nouns often take a singulative marker. Singulative is marked by **/- čo(*cho*)/** former occurs after stem final sonorants as in /***dar*- čo(*cho*)/** 'one single leaf' [76,78]. In some nouns stem-final consonant are totally assimilate with /č-ch/ singulative marker. e.g., /**gereb**-/+/č/→/**gerečo/** 'a sheep'. Some singulative nouns have an unmarked counterpart that is formed by suffixing either **/-o/ or /-e/** to a noun stem. e.g., /**farad**-o/- 'horse'. Plural marker on Sidaamu Afoo nouns are apply by deleting terminal vowel. Not only deleting, germination is takes place in stem-final consonant and then suffixing /-a/. e.g., **goga** (S)→gog**g-a** (PL) 'skin'. For plural marker, some noun takes plural suffix /-**uwa**/. It also has an allomorph /-**ubba**/ used with nouns.

e.g., **adde** (S)→**add-uwa** (PL), **siqqo**(S)→**siqq-ubba** (PL).

In additionally, there is so many plural markers for Sidaamu Afoo nouns and some are used for adjective marker sometimes. these are:

- /-**aasine**/ for nouns that ends with /-**aasinčo**/ and /-**aančo**/ in singular forms.
- /- **aano**/ for nouns which ends with /-**aančo**/ in singular form, /-**oota**/ for some nouns.
- /-**eeyye**/ for nouns that ends it singular form by /-**eessa**/(M) and /-**eette**/(F).
- /-**uulle**/ for some nouns. /**uulle/** and /**eeyye/** are more productive in adjectives than nouns.
- /-**oolle**/ for a few nouns. e.g., /moot-i-**ča/**→moot-**oolle** 'chief'

As Anbessa Teferra [76] states that, these plural suffixes cover most of plural-marking processes countable nouns in Sidaamu Afoo. As presented before there is so many plural suffixes, this tells that plural-formation in Sidaamu Afoo is not restricted to a single suffix,

but rather involves a number of arbitrary suffixes. Uncountable nouns neither inflected for plural nor take numerals as modifiers.

Gender in Sidaamu Afoo is distinguished in various ways. It can be marked either lexically or by means of gender-marking suffixes. Gender marker can be marked for feminine and masculine suffixes. There is different gender-marking suffixes. These are:

- Some kinship terms and common domestic animal names distinguish gender lexically. e.g., /**hando/** (M) 'ox'→ /**saa/**(F) 'cow', /**manna/**(M) 'men'→/**meento/** (F) 'women' /**wosiila/** (M) 'maternal uncle'→/**la'lama/** (F) 'maternal aunt'
- For some nouns, gender distinction is indicated via gender-marking suffixes. /-ča/ and /-**eessa**/ mark masculine but, /-**tte**/ and /-**eette**/ mark feminine.
  e.g., /moot-i-**ča/** (M)→/moot-i-**tte/** (F) 'lord/master'

Plural nouns do not make gender distinctions. Thus, for instance, /**og-eeyye**/ 'bonesetters' is a plural form for both the masculine and feminine. In a couple of kinship nouns, feminine is marked by /-**ee~e**/ while the masculine is unmarked or basic in form. e.g., /**ahaaho/**(M) 'grandfather'→/**ahaah**-e/(F) 'grandmother', /**aroo/**(M) 'husband'→/**ar-ee/**(F) 'wife'.

Sidaamu Afoo is a nominative-accusative language. **Case** can be morphologically marked by suprafixation, suffixation, or both [8]. The nominative, absolutive and genitive are three basic case types. Besides, there are a set of oblique cases such as dative, instrumental and ablative.

Nominative case:

In Sidaamu Afoo, nominative case is assigned to a subject NP of a finite or tensed clause. As Kawachi [8] states that, only masculine subject noun can be marked with the nominative case suffix, which substitutes for the final vowel of a noun stem, which is /e, a, or o/. The nominative allomorph /-u/ and /-i/ can be marked on the nominative of masculine nouns. But feminine nouns are not marked morphologically overtly for nominative and hence such nouns always remain as it is in their citation forms. The nominative allomorph /-u/ and /-i/ have grammatically conditioned distribution. Thus, /-u/ mostly marks the nominative of masculine subject noun when they appear alone, while /-i/ occurs elsewhere [76,80].

e.g., /**wee'n-i-č-u**/ '(colobus monkey- SG-nom)', /**beett-i**/ 'boy-nom'

Absolutive is a case which is taken by an object NP. As Anbessa Teferra [76] states, different researchers used different names for absolutive case, i.e., "base form", "simple case", "accusative", "absolutive", while authors decided to use the term "absolutive" rather than

"accusative". Absolutive is not grammatically marked case, it gives a noun in its citation form. Accusative is morphologically marked for other non-Ethiopian language as Anbessa Teferra [76] presented, but it is not marked in Sidaamu Afoo. Nouns occurring in the absolutive case have as the final vowel /o/, /a/, or /e/ and never /-i/ or /-u/. absolutive case is assigned to an object of a transitive verb.

Genitive Case:

According to Kawachi [8], the genitive suffix-marking pattern is similar to the nominative suffix-marking pattern except that /–**te/** is used on Unmodified, feminine common nouns. if the genitive structure is possessor + possessed, then the genitive case of possessor is marked by gender-sensitive suffixes. These are /-**u**/ 'masculine' (common noun), /-**i**/ 'masculine' (proper noun) and /-**te**/ 'feminine'. [76].

e.g., /**Daafurs*a****/*** *(proper noun)*→/Daafurs***i/*** (genitive case)

/G**aango/**→/**Gaango**-te/ 'mule-gen.(F.)'

Dative Case: dative is marked by a set of three postpositional suffixes which are added to the absolutive form of the object noun. These are: /-ho/ 'M. noun', /-te/ 'F. noun' and /-ra/ 'pronouns and proper names' [76]. As Kawachi [8] presents, it is also said to be Dative-Locative case. e.g., /**beetto**-ho/ 'child-DAT. M', /**nafara**-ho/ 'compound-LOC.M',

/**beetto**-te/ 'child-DAT.F', /**uulla**-te/ 'ground-LOC.F' when they are unmodified. If the noun is modified, it should be /**uulla**-ra/ 'ground-LOC' [80].

Instrumental and Ablative Cases marker:

Both cases are marked by suffix /-**nni**/ which is suffixed to a genitive base [8].

e.g., instrumental→/**dull**-**u**-***nni****/ 'club-gen.(M.)-with'*

*ablative→ /**dikko**-**te**-**nni**/ 'market-gen.(F.)-from'*

B. Personal pronouns

Personal pronouns are marked for nominative and absolutive cases. All nominative pronouns are marked by an unstressed final vowel. All singular persons (except 3SGF. that are marked suffix /-**e**/) are additionally marked by the suffix /-**i**/. But, plural person marked by a suffix /**ni**-**nke**/ 'we-1PL', /**ki**- 'ne/ 'you-2PL and 2POL', and /i-**nsa**/ 'they-3PL and 3POL' [76] . Whereas, in all absolutive pronouns the final syllable is stressed, but they are uninflected. All the absolutive singular pronouns (except 3SGF.) are distinguished from nominative counterparts in terms of final vowel. i.e., /**ane**/ 'me-1SG', /ate/ 'you-2SG', /**iso**/ 'him-3SGM.',

/**ise**/ ‘her-3SGF.’. Generally, the suffixed absolutive pronoun is the last item in conjugated verb as /**kaa’l**-i-e/ ‘help-3SGMPR-**me**→he helped me’. The suffixed absolutive pronouns are: /- **‘e**/ ‘1SG’, /-**he**/ ‘2SG’, /-**si**/ ‘3SGM.’, /-**se**/ ‘3SGF.’, /-**nke**/ ‘1PL’, /-‘**ne**/ ‘2PL’, /-**nsa**/ ‘3PL’. E.g., /kaa’l-i-‘ne/ ‘he helped you’, /kaa’l-i-tu-nke/ ‘she helped us’.

C. Possessive Pronouns

According to Kawachi [8], if the modified noun is masculine and in the nominative or genitive, the possessive pronominal suffix follows the nominative or genitive case suffix for modified masculine common nouns, ***-i***. But, if the modified nouns are feminine, the possessive pronominal can’t follow suffix –***i***. The possessive pronouns suffixes are, /-‘**ya**/ ‘my-1SG’, /-**kki**/ ‘your-2SG’, /-**si**/ ‘his-3SGM.’, /-**se**/ ‘her-3SGF.’, /-**nke**/ ‘our-1PL’, /-‘**ne**/ ‘your-2PL/POL’, /-**nsa**/ ‘their-3PL/POL’.

E.g., /mine-kki/ ‘your house’, /darga-nke/ ‘our place’.

D. Relative Pronouns

As Anbessa Teferra [76], there is no independent relative pronoun, but there are three relative pronoun suffixes which can be suffixed to nouns, adjective, and verbs. These suffixes are: /-**h**/ for masculine, /-**t**/ for feminine, and /-**r**/ for plural nouns. According to Anbessa [76], all relative pronouns, even a feminine relative are case-marked. As elsewhere, the high vowels mark nominative, thus, relative suffixes of Sidaamu Afoo.

Table 2.2 Relative Suffixes of Sidaamu Afoo

	Relative Suff	Nominative rel	Absolutive rel
M.	-h	-hu	-ha
F.	-t	-ti	-ta
PL	-r	-ri	-re

E. Derived Nominal

In Sidaamu Afoo, nominals are usually derived from three of word classes; simple verb stems, noun, and adjective through suffixation of various derivational formatives [76]. Nouns are derived from verbs in different ways: these are:

a. Adding a nominalizer suffix /-**aančo**/ to a verb stem to generate instrumental nouns.
   e.g. /fey-/ ‘sweep’+ /-**aan**čo/ ‘NML’→/feyaančo/ ‘broom’
b. Agent nouns are formed by adding nominalizer suffix /-aančo/, /-aasinčo/, /tiča/.
   e.g., /ros-**i-s**/ ‘teach’+/-**aan**čo/ ‘NML’→/rosisaančo/ ‘teacher’
   /daddal-/ ‘trade’ + /-**aasinčo**/ ‘NML/→/daddalaasinčo/ ‘merchant’.

/tum-/ ‘pound’ +/-**tiča**/ ‘NML’→ /tuntiča/ ‘blacksmith’

Hence, the nominalizer suffixes /-**aančo**/ and /-**aasinčo**/ are not limited to agent noun formatives, rather they can serve as nominalizer for noun also.

c. Abstract nouns are derived from a verb stem by adding nominative suffixes /-**(m)ma**/ and /-**(l)le**/. Note that in Sidaamu Afoo, during derivation and inflection the usual phonological rule must be apply for any word class.

e.g., /goww-/ ‘be foolish’ +/-**mma**/ ‘NML’→/gowwimma/ ‘foolishness’

/bax-/ ‘like/love’+ /-**lle**/ ‘NML’→/baxille/ ‘love’

/busul-/ ‘be clever’+ /-le/ ‘NML’→ /busulle/ ‘cleverness’

d. Some nominal that indicate the country of origin and ethnicity are derived from another noun by suffixing /-**aawiča**/ ‘M’, /-**aawitte**/ ‘F.’ for country of origin and /-**tiča**/ ‘M.’, /-**titte**/ ‘F.’ for ethnicity [75].

e.g., /Sidaama/ ‘Sidama’+ /-**tiča**/ ‘NML’→/**Sidaan**-**tiča**/ ‘a Sidaama man’

Although, several nominal suffixes that are not classified under above rules are used to derive a nominal from the verb stem [8]. These nominal suffixes are: **/-e/, /-a/, /-o/, /-te/, /-sa/, /-ša/, /-šša/, /-ana/, /-ano/, /-añe/** in /darš-**e**/ ‘swelling’, /birr-**a**/ ‘spring’, /duun-**o**/ ‘pile/stack’, /gal-**te**/ ‘spouse’, /(dh)ɓaan-**sa**/ ‘message’, /gatamar-**ša**/ ‘restoration’, /wi’l-i-**šša**/ ‘mourning’, /čuf-**ana**/ ‘door’, /got’-**ano**/ ‘sleep’, /bob-**añe**/ ‘bad smell’.

Besides, some of nominals in Sidaamu Afoo are derived from adjectives by suffixing nominal suffix /-**imma**/ and /-**naate**/ [75]. These nouns are:

/dančа/ ‘good-Adj’+/-**imma**/ ‘NML’→/dančimma/ ‘goodness’

/haaro/ ‘new- Adj’ +/-**imma**/ ‘NML’→/haarimma/ ‘newness’

/koliššo/ ‘black-Adj’ +/-**naate**/ ‘NML’→/koliššinaate/ ‘blackness’

F. Infinitives:

An infinitive is a word form which usually does not have an explicit subject and which is also devoid of aspect/tense markers [76]. Suffixing the deverbative suffix /-**a**/ to a verb stem is apply for infinitives in Sidaamu Afoo [75]. Though, an infinitive is verbal in base,

functionally and distributional it is a nominal and hence it can function either as subject or as object.

e.g., /kubb-**a/** 'to jump/jumping'→function as a subject.

/gan-**a/** 'to hit/hitting'→function as an object.

Sometimes Sidaamu afoo uses this suffix as a gerund form. But, it(**/-a/)** sometimes follows the suffixes like /-**nna/** 'during- conjugation' or **/-nni/** 'during- instrumental postposition'.

G. Negation of Nouns and Pronouns

Finally, the suffixes /-**weelo**/ 'without', and /-**no/** 'and/too' are the suffixes that added to a noun stems to imply the negative noun (we will discuss later about negative affixes).

2.5.3.2 Verb

A verb is one of the word classes which serves as a predicate in Sidaamu Afoo and appears sentence-final position. Verbs in Sidaamu Afoo can inflect for various grammatical categories such as aspect/tense, mood, and agreement. It also derives causative and passive verb by taking causative and passive morpheme respectively [78]. As Anbessa Teferra [76] presents, when the verbs are inflected for number and gender, a nominal and adjective is similar to verbs of Sidaamu Afoo. But verb differ morphologically from all other word categories while it inflected for person, tense/aspect and mood.

Structures (Canonical form) of the verb stem

All verbs that is occur in Sidaamu Afoo has verb stem and those stems always ends with consonant. These verb stems can be classified in to different type according to their corresponding Consonant Vowel (henceforth CV) structure and glosses [76]. These are:

1. Mono-Consonant-C: there is some only one-consonant words in Sidaamu Afoo, like, /**š**-/ → C- 'kill', /**y**-/ →C- 'say'
2. Mono-Syllabics: more Sidaamu Afoo verbs are mono-syllabic. Here also different structures are there.

   For instance,

VC- →/ag-/ 'drink'	CVC- →/dod/ 'run'
VC1C1-→/ass-/ 'do'	CV1V1C- →/kaad-/ 'deny'
VC1C2- →/irb-/ 'rub'	CVC2C3- →/sirb-/ 'sing
V1V1C1C1- →/uurr-/ 'stand'	C1VC2C2- →/kubb/ 'jump'

C1V1V1C2C2- →/coomm-/ ‘be sweet/tasty’

C1VC2C3C3- →/hayšš-/ ‘wash’

3. Disyllabic: disyllabic have two different sources: some are simple disyllabic which are not derived from other word class and the other one are the verbs that are derived from other word class by the process of reduplication or that are geminated [76]. Most basic ideophones are disyllabic [79]. These disyllabic are:

CV Structure	Verb stem	meaning	CV Structure	verb Stem	previous stem
VCVC-	/amad-/	catch	CV1V1C2C3VV	/laanšaab-/	
VC1C1VC-	/akkal-/	become worn	CVCV2V2C-	/šolool-/	
VCVC2C2-	/egenn-/	know	CVCVC3C4-	/godo'l-/	
CVCVC-	/dadil-/	worry	CVC2C3VC-	/gangan-/	/gan-/
C1VC2C2VC	/daddal-/	trade	CVC2C2VC-	/kukkub-/	/kubb-/
CVC2C3VC-	/gombis-/	overturn	CVC2C2VC-	/mudd-am/	/mudd-/
CV1V1CVC-	/diilal-/	be windy			

4. **Tri-syllabics**

The verb that are categorized under tri-syllabic has phonological limitation and it is limited in number [76]. e.g.,

C1VC2VC3VC4- /ɓuɓukat-/ ‘*crackle’*

C1VC2C2VC3VC4- /fokkifat-/ ‘*be ashemed’*

C1V1C2V2C3V3V3C4- /dugunuun-/ ‘*bow one’s head*’

C1VC2VCV3V3C4C4-/hamuruurr-/ ‘*be bruised*’

Types of verb

According to Kawachi [8], to classify a verb in to different types, there is two ways: based on the distinction between **transitive** and **intransitive** verbs, and other classification is based on the aspectual distinction between **action verbs** and **state-change** verbs.

Transitive verbs are a verb that must take objects (action receiver) to give complete message from a sentence, e.g., /mur/- ‘to cut’ this must take objects that what it cuts? For instance, /*ise haqq-i-cho mur*-**t-u**/ ‘3SGFNom tree (ACC) cut-3SGF-PRF3SGF’ ‘*she cuts the tree’* here the transitive verb /*murtu*/ ‘she cut’ must take an action receiver /*haqqicho*/ *‘tree.*

Whereas, intransitive verbs do not take objects, e.g., /*buus*/- ‘to cough’. For instance, /*is*-**i** *buus*-**i**/ ‘3SGM-Nom cough-PRFSGM’ has no need of any action receiver. Ambi-trasitive verbs that exist in English (i.e., *break* that can be both) are not found in Sidaamu Afoo.

As Kawachi [8] stated, several intransitive verb roots causative forms are transitive, whereas, passive form of a transitive verb root acts as an intransitive verb that expresses a state-change. Many languages have two types of verbs based on aspectual distinction. These are: dynamic verbs and static verb. Dynamic verbs are those that express events involving actions or change. In Sidaamu Afoo, there is no verb root that are categorized under a class of static verbs that express states. Majority verb stems are classified under dynamic verbs are also classified in to two of broad types that are express action or express state-change [8]. The verbs that tells ones feeling of mental or physical feeling are basically state-change verbs. e.g., /*dod-*/ 'to run' is used to express an action that are ongoing activity of running. Likewise, the word /*uurr-*/ 'to stand' is used to express state-change that tells immediate process of standing up.

Inflections of Verbs

The verbs in Sidaamu Afoo are inflected for tense/aspect, person, number, gender [76].

A. Tense/ Aspect

Sidaamu Afoo has no different formatives for tense and aspect, they are joined together and has the same formative functions both as tense and aspect marker. Verbs are classified in to perfect and imperfect of Tense/Aspect forms. Further, perfect is subdivided into simple perfect and present perfect. There are also additional conjugations known as present continuous tense and the con-verb or conjunctions [76]. These and other Sidaamu Afoo verb inflections are discussed below.

1 Simple perfect, Present perfect, and Imperfect

As Anbessa Teferra [76] described, the simple perfect is roughly corresponds the past tense in English and expresses a completed action. But it doesn't express that the action is completed; it could be a recent or a remote past (this is the same with Amharic perfect tense as in /ቦላ/ 'he ate'). E.g., /sirb-/ 'sing' +/-**ummo**/ '1SGMPRF'→/sirbummo**/** 'I sang'.

An event which have happened a relatively a short period of time is present perfect. An example of present prefect is: /sirb-/ 'sing' +/-**oommo**/ '1SGMPPRF/ →/sirboommo/ 'I have sung'. The imperfect marks an incomplete action. This used as present or future tense in English as Anbessa Teferra [76] stated. E.g., /**it**-/ 'eat' +/-**eemma**/ '1SGF.IMP'→/iteemma/ 'I eat/shall eat/ We will discuss more about suffixes that inflects present perfect in Section 2.5.4.

2 Present continuous tense

The present continuous tense consists a subordinate verb with a continuous suffix /-**anni**/ and a present perfect conjugation of the auxiliary verb /**no**/ '*is*' for singular persons and /**heeɓ-/** '*live*' for 1st and 2nd plural. E.g., /sirb-/ 'sing' +/-**anni**/ '1SGMPCON' /**no-oommo**/ '*is-1SGMPPRF*' →/sirbanni noommo/ 'I am singing'. For more affixes refer Chapter 2.5.4.

3 Converb (mood) conjugations and Copula

Conjugation means to give various inflectional endings of a verb, i.e. voice, mood, tense, number and person [5]. The converb is always used as a subordinate verb and hence precedes the main verb. Also, the subject of the converb and the main verb must be co-referential [76]. e.g., /**sirb**-/ 'sing'+ /-**é**/ '1SGMCONV' →/**sirbé**/ 'I singing, …'

[sagale it-**é,** waa ag-**i**] 'food eat-Conv, water drink-3SGMPR'- 'having eaten the food, he drank water'. Here the converb conjugation consists of a stem and subject suffixes /-é/.

According to Anbessa Teferra [76], the suffix /-**ho**/ 'SGM.', /-**te**/ 'SGF.', and /-**ti/** 'it is' are grammatically conditioned alternatives for the 'verb to be' or the copula in Sidaamu Afoo. The plural forms can take either of them except /-**ti**/, but the rule of selection is not predictable. But, /-**ti/** is restricted to genitive structures. Thus, the copula in Sidaamu afoo is not conjugated for any tenses. E.g., /Daafurs-i/ 'Daafursa-Nom' /ros-aančo-**ho**/ 'learn-NML-is M.'. Also, for its marker see later Chapter 2.5.4

4 verb of presence and Possession

The stem of verb of presence /**no-/** '*present'* is the only irregular one in Sidaamu Afoo. Its present perfect is conjugated on the basis of /**no-/** *'present'* while its simple perfect and imperfect forms are conjugated on the basis of the existential verb /heeɓ-/ 'live'. E.g., /no-**oommo**/ '*present-1SGPPRF'* →/noommo/ '*I am/was present'.* Possession in present time is expressed by the verb of presence /**no-/** and suffixing the object suffixes.

Note that Sidaamu Afoo does not have a past time expression which is equivalent to Amharic **ነበረኝ, ነበረህ,** etc. During concatenation of object suffixes with verb of present to express a possession, lengthening of /o/ takes place at stem final of verb of presence.

E.g., /**no-o-si**/ '*present-o-3SGMPOS'* →/**noosi**/ 'he has'.

5 Imperative-Jussive

The imperative verbs are command verbs [6] and it marks only second person subject markers. The affirmative imperative marked by the subject marker /-**i**/ '*SG'* and /–**e**/ '*PL'.* In plural

imperative (2PL IMP), the stem-final consonants are geminated. Imperative is always unstressed [76].

e.g., /sirb-/ 'sing' +/-**i**/ '*2SG IMP'* →/sirbi!/ 'sing!',

/kul-/ 'tel'+/-**e**/ '2PLIMP' →/kulle/ 'tell!' bnm

The jussive can be viewed as a mild imperative in 1st and 3rd person and it expresses that someone commands another one. Simply 'let X V', where, X is doer and V is action or Verb.

e.g., /**sirb**-/ '*sing*' + /-**o**/ '*1SG/3SGM JUSS*'→/**sirbo**/ 'let me/him sing'.

/**sirb-/** 'sing' **+/i-to/** '3SG.F/PL'→ /**sirbito/** 'let her/them sing'.

/**sirb-/** 'sing' **+/i-no/** '3POL/ 1PL'→ /**sirbino/** 'let him (POL)/us sing'.

So, the suffixes /**-o/, /-i-to/, /-i-no/** is marked for jussive object marker.

The difference between imperative and jussive is direct and indirect command [78].

B. Person and Gender

The markers of 1st person, 2nd person, and 3rd person singular feminine/ 3rd person plural are clear. But, the marker of 3rd masculine is problematic since it can also function as a tense/aspect marker. The person markers are: /-**mm-/** '1SG Indicative', /-**tt**-/ '2SG Inductive', /-**t-/** '2PL Jussive/ 3SGF/PL Inductive /3PLF. Jussive'.

The tense/aspect conjugations differentiate a masculine and feminine in the 1st and 2nd person singular by suffixing /-**o**/ 'M', and /-**a/** 'F'. In 3rd person /-**t/** marks 'feminine' while the 'masculine' is unmarked [76].

C. Derived stems

Earlier, we have discussed that nouns can derive from different word class like verb, noun, adjectives. Likewise, verb also can be derived from different lexical category (i.e., verb, nouns, adjectives). While, the verb that derived from another verb stem can't bring category of word class. Derived classes of verbs are discussed below.

1. Singular and Double (Factitive) causative

According to Anbessa Teferra [76], single causative is formed by adding the formative /-**s/** to a simple verb stem. It is always added to an intransitive verb stem and the process results that a transitive verb. For this reason, the formative /-**s/** is said to be transitivizer.

E.g., Daafurs-**i** k'aakk'o **(***qaaqqo***)** gox-**i-s-i '***Daafursa-Nom baby sleep-ep-CAS-3SGF.PR'*

Factitive is marked by doubling singular causative suffix /-s-ii-s/, the middle double /i/ is used to separate causative morphemes and it is usually present in many causative forms. There are

two types of verbs that take factitive; some verb first takes singular causative, then adds another one, and the other one takes the double causative at once without taking singular causative [76]. E.g., /it-**i-s-sù**/ '*eat –ep-CAS-3SGF.PRF'* and /it-**i-siis-sù**/ '*eat-ep-DCAS-3SGF.PRF'*-takes causative again after takes singular one.

E.g., /agar-**/** 'guard'+/-**siis**/ 'DCAS'→/agar-**siis-sù/** 'guard –DCAS-3SGFPRF'

2. Autobenefactive and Passive

The autobenefactive is also said middle voice (**-ɓ, -'r, -', -p', -t'**) expresses the notion 'for oneself' is formed by suffixing the autobenefactive formative /-**ɓ/** to simple verb stems. The epenthetic vowel /*i*/ is appear in the middle of a verb before ***-ɓ*** *or* ***–'r*** [8]. E.g., the autobenefactive stem /hayšš-**i-ɓ**/ '*wash oneslf'* is from the verb stem /hayšš-**/** 'wash'. Most a time autobenefactive are derived from basic verb stems, but some are derived from adjective stems. E.g., /**dib-/** 'cover' +/-**ɓ/ '**ABF'→/dip'/ 'cover oneself'

Kawachi [8] stated that passive as a reciprocal suffix and has the form as the passive suffix: /***-am/*** sometimes before dental it can be /***-an/***. E.g., /bat'-/ 'to love'+/-**am/** →/bat'am/ 'to love each other'. Passivization is a lexical process which usually operates on transitive verbs.

Passivization is not necessarily restricted to transitive verbs. There are some passive verbs without a corresponding transitive base. E.g., /k'arr-**am**/ '*be in trouble-3SGMPRF*'

3. Intensive (reduplication)

According to Anbessa Teferra [76], the action which is doing repeatedly several times are indicated as or said to be intensive verb. It duplicates some syllables from a certain verb stem. e.g., /kubb-/ 'jump' →/kukkubb-/ **'**jump repeatedly', /gan-**/** 'hit'→/gangan/ 'hit repeadly'. But there are some stems that occur only in a reduplicative form. E.g., /gungum-**/** 'murmur', /jajjar-**/** 'be confused', /gaggab-**/** 'stagger/stumble'

2.5.3.3 Adjectives and other word class

Adjectives

An adjective is one of a word class which is identified, and on the basis of its position in relation to a noun. Sidaamu Afoo adjectives have morphosyntactic properties distinct from those of other categories, but share some morphosyntactic properties with nouns and with some verbs. Some adjectives have all or most of the characteristic of nouns [8]. Both nouns and adjectives can be inflected for gender, number, and cases. Adjectives can normally be

used either predicating noun-phrase clitic that follows a predicative adjective or attributively. The predicating noun-phrase clitic that follows a predicate adjective agrees in gender with the subject noun, and an attributive adjective agrees in gender and case with the noun that it modifies. Some type of structures allows the occurrence of nouns in the position of the adjectives [76]. Sometimes nouns can also serve as an adjective for other nouns. The gender formatives /**-eessa/** 'M' and /**-eette/** 'F' are mostly restricted to adjective, but the only few nouns are present which appear with these suffixes. E.g., /dur-**eessa**/, /dur-**eette**/ 'rich'. The other morphological feature which distinguishes adjectives from nouns is their declension for diminutives. Only adjectives that take the diminutive /**-č'o/** implies that in adjective singulative marker.

e.g. /šiima/ 'Adj'→/šiim-i-**čč'o/ '**tiny Dimunitive'.

There are two types of adjective classes: free adjective and derived adjectives.

The type of adjective that usually expresses only some crucially important is said to be free/true adjectives. This type always implies that the word indicating dimension, color, age, and value. The free adjective list that have discussed above is;

/Lowo/ 'big'**, /**šiima/ 'small'**, /**koliššo 'black'**, /**waajjo/ 'white'**, /**haaro/ 'new'**, /**danča/ 'good'

On the other hand, the mostly used adjectives are derived from verbs, and the formative which change a verb into an adjective are **/-a/** (in most cases), /**-čo/**, and **/-ado/**.

E.g., /akkal**-/** 'become worn'+/**-a/→/**akkal**a/** 'old (for thing)-Adj'

/hara'm**-/** 'become short'+/**-čo/ →/**haran**čo/** 'short-Adj'

/šaal**-/** 'become thin'+ /**-ado/→/** šaal**ado/** 'thin-Adj'

In general Adjectives in Sidaamu afoo are inflected for case, gender, and number, and agree with the nouns which they modify [76]. Adjectives must agree with their head noun that they modify in cases. Adjectives also can be marked as nominative or absolutive. We have discussed so far about nominative and absolutive (absolutive is unmarked in Sidaamu Afoo). Adjectives are characterized by gender-marking suffixes. Masculine is indicated by suffixes such as **/-o/, /-ča/, (l)eessa/** while feminine adjectives are marked by /**-e/, /-tte/,** and **/-(l)eette/** respectively**.**

e.g., /atoot-**aam-o/** 'M', /atoot-**aam-e/** 'F' 'sinful', /ball-**i-ča/** 'M', **/**ball-**i-tte/** 'F' 'blind'

/dur-**eessa/** 'M', **/**dur-**eette/** 'F' 'rich',

/fayya-**leessa/** 'M', /fayya-**leette/** 'F' 'healthy, innocent'

Mostly adjectives like their nominal that they modify, are inflected for plural. Plural are marked by plural-marking suffix and reduplication of an adjective stem. These plural marker suffixes are: /**-(mm)ma/, /-yye/, /-oota/, /-uulle/,** etc.

E.g., /badd-**aamo/ '***SG*' →/badda-**aam-ma/** 'PL' – 'bald'

/boo'na-**leessa/** *'SG'*→/boo'na-**leeyye/** 'PL'- 'boaster'

/ball-**i-ča/ '***SG'* →**/**ball-**oota/** 'PL'- 'blind',

/buša/ '*SG*'→**/**buš-**uulle/** 'PL'- 'bad', /lowo/ '*SG*' →/low-**i-ddaanna/** 'PL'- 'big

As well as adjectives can be marked plural by reduplication of an adjective stems.

E.g., /jawa/ 'Stem'→/jajjabba/ 'PL'- 'big'

Sidaamu Afoo has no preposition rather only has postposition. Postpositions are closed word class in Sidaamu Afoo, and they don't take any morphological features such as number, gender, person, aspect, etc. Thus, **/-nni/** 'form/with/at/on/by'**, /-weelo/** 'without-NEG'**, /-ho** 'to/for –M.'**, -te** 'to/for-F.'**, -re** 'to-for'**/** can be used as some bound postpositions.

There is also another word classes that formative whose function is to connect two or more grammatical units such as words, phrases, and clauses is so called conjunctions. It also divided into two types: coordinating and subordinating. Coordination almost involves elements which have the same grammatical status. There are some formatives that its function as coordinating conjunctions. These are: **/-nna/** 'and'**, /-no/** 'too/also'**, /-na/** 'and'(interrogative)**, /-nso/** 'or'(interrogative)**,** etc [79].

E.g., Daafurs-i, Dikko ha'r-anno. At-i-**na?** 'Daafursa-NOM market go-2SG.M. IMP you-NOM-and' 'Daafursa goes to a market. Are you?

2.5.4 Affixes in Sidaamu Afoo

Morphemes in a language are composed of affixes and stems. An affix is a bound morpheme that is attached to a base (stem or root) [3]. Sidaamu Afoo affixes can be prefix, infix, suffix, and circumfix. Almost most affixes in Sidaamu Afoo are suffixes.

Prefix: prefix is an affix that is attached in front of a base or stem [14]. As Kawachi [8] stated, there are no prefixes in Sidaamu Afoo, but the only negative proclitic /***di-/*** is usually used as a verb prefix to indicate the negativity of the verb.

E.g., /**di-/** 'NEG' +/sirb-ino/ 'sing -3SGMPPRF' →**/*di****sirbino*/ 'he has not sung'

Suffix: Sidaamu Afoo mostly have suffix affixes. That means there are only one prefix and some countable infix (e.g., /*hii****kk'a****kk'i/* is derived from the stem /***hiikk'/*** by infixing /***-kk'a-/***

at the middle of the syllables) are there, but sometimes it can be taken as reduplication and concluded to be, there is no infix in Sidaamu Afoo. All Sidaamu Afoo Verbs, nouns, and adjectives are inflected through suffix. These inflection markers are:

Nominative dative or dative Postposition or copula marker

The dative marker is marked by a set of three postpositional suffixes which are added to the absolutive of the noun. These suffixes are **/-ho/** 'M. noun'**, /-te/ '**F. noun'**,** /**-ra/ '**nouns and pronouns'. We have discussed briefly about copula suffixes earlier.

Nominalizer suffixes

Sidaamu Afoo also has a set of nominals that are derived from verbs, nouns, and adjectives through suffixation of various derivational formatives. As we have discussed derived nominal under Chapter 2.5.3.1, there are so many types of suffixes that nouns can be derived from verb, noun, adjective. These suffixes are: *-aancho, -aasincho, -mme, - (m)ma, -(l)le, - ano, -anye, -ticha, -titte, -aawicha, -aawitte, -e, -a, -o, -te, -sa, -sha, -shsha, -ana, -naate*, etc.

Simple perfect subject marker suffixes

The suffixes that marks for simply perfect (we have used only PRF or perfect in our study mostly to represent a simple perfect) are listed below Table 2.3 with examples. **Source**: Anbessa Teferra [76].

Table 2.3 Perfect paradigm

Singular						Plural		
Person	Gender	Suffix	Example	Meaning		suffix	example	meaning
1st	M	**-ummo**	**sirbumm**	I sang				
	F	**-umma**	**sirbumm**	I sang		**-i-nummo**	**sirbinummo**	we sang
2nd	M	**-itto**	**sirbitto**	you sang	PL/			
	F	**-itta**	**sirbitta**	you sang	POL	**-i-tiní**	**sirbitiní**	you/you(POL) sang
3rd	M	**-í**	**sirbí**	he sang				
	F/PL	**-i-tu**	**sirbitu**	she/they sang	POL	**-i-ní**	**sirbiní**	He (POL) sang

Present Perfect Subject marker suffixes

The suffixes that present prefect (we have used PPRF in our study) marked is listed below Table 2.4 with the examples. **Source**: Anbessa Teferra [76].

Table 2.4 Present Perfect Paradigm

Singular						Plural		
Person	Gender	Suffix	Example	Meaning		suffix	example	meaning
1st	M	**-oommo**	**sirboommo**	I have sung		**-i-noommo**	**sirbinoommo**	we have sung
	F	**-oomma**	**sirboomma**	I have sung				
2nd	M	**-ootto**	**sirbootto**	you have sung		**tinoonni~ -tinooy**	**sirbitinoonni sirbitinooy**	you/you(POL) have sung
	F	**-ootta**	**sirbootta**	you have sung	POL			
3rd	M	**-ínó**	**sirbínó**	he has sung		**-i-nooníní ~ -nooy**	**sirbinoonni sirbinoy**	He(POL) has sung
	F/PL	**-i-tínó**	**sirbitínó**	she/they has sung	POL			

Imperfect verb suffixes

There are so many suffixes of imperfect subject markers that have not discussed so far. These are Table 2.5: **Source:** Anbessa Teferra [76].

Table 2.5 Imperfect Paradigm

Singular						Plural		
Person	Gender	Suffix	Example	Meaning		suffix	example	meaning
1st	M	**-eemmo**	**sirbeemmo**	I sing/shall sing		**-i-neemmo**	**sirbineemmo**	wesing/shall sing
	F	**-eemma**	**sirbeemma**	I sing/shall sing				
2nd	M	**-atto**	**sirbatto**	you sing/will sing		**-i-tinanní~ -tinay**	**sirbitinanní sirbitinay**	you/you(POL) have sung
	F	**-atta**	**sirbatta**	you sing/will sing	POL			
3rd	M	**-anno~-awo**	**sirbanno**	he sings/will sing		**-i-nanní ~ -nay**	**sirbinanní sirbinay**	He(POL) sings/will sing
	F/PL	**-i-tanno~ -tawo**	**sirbitanno**	she sings/they sing, she/they sing/will	POL			

Present Continuous subject marker: Present continuous suffixes differs structurally from other tense/aspect conjugations in being a complex form [76]. Some suffixes like similar with some of imperative suffixes. These and others are listed below.

The 1st person and 3rd person *SG* masculine marked for the suffix **/-anni/** →/sirb**anni/** 'I am/ he is singing'. The 2nd person and 3rd person *SG* feminine or plural marked for the suffix **/-i-tanni/** →/sirb**itanni/** 'you/ she/ they are singing'.

The suffixes that marked for 1st person plural and polite, and 2nd person plural and 2nd person polite **/-i-nanni/**→sirb**inanni** and **/-i-tinanni/**→sirb**itinanni/** respectively are the same with 3rd person polite imperfect and 2nd person plural/polite imperfect respectively. Thus, we have coded or set a marker code as the same, because it minimizes the decrements of accuracy or helps our machine learning predicts with a good performance in our study.

Possessive Object markers suffixes: The possession suffixes expressed by the verb of presence 'no' and follows object suffixes. The suffixes of marker that marks for possession is discussed below Table 2.6. Source: Anbessa Teferra [76].

Table 2.6 verbal Possessive object markers

Singular						Plural		
Person	Gender	Suffix	Example	Meaning		suffix	example	meaning
1st		**-e**	**nooe**	I have		**-nke**	**noonke**	we have
2nd		**-he**	**noohe**	you have		**-'ne**	**noo'ne**	you have
3rd	M	**-si**	**noosi**	he has				
	F	**-se**	**noose**	she has		**-nsa**	**noonsa**	they have

The suffix /***-e/*** is used for various purpose. For instance, the affirmative imperative is marked by the suffixes /**-i/** and **/-e/** 'PL'. Imperative marker suffix for 2nd person plural is /**-e/→/**sirb**e/** 'sing!' used as a command. It is hard to use this kind of suffixes separately by giving separate marker symbol or code as they are in our study. This decreases the accuracy of our system, during training no matter that system can learn as they are. But, when system predicts unseen classes during test, system can classify or predict by wrongly or classify mismatched class, if we use separate marker as they are. Thus, we have used one marker symbol or code that helps our system to predict accurately for these kinds of suffixes in our study.

On the other hand, there is also pronominal possessive object marker. The difference is that the verbal possessive follows the verb of presence '*no',* while pronominal possessive expressed by means of genitive pronoun suffixes. Here most suffixes are similar structurally with verbal and pronominal possessive, but it may differ its meaning that gives after suffixing it with the stems. There are only two suffixes are different, the other are similar. Let's discuss below Table 2.7. **Source:** Anbessa Teferra [76].

Table 2.7 Possessive pronoun object marker

Singular					Plural		
Person	Gender	Suffix	Example	Meaning	suffix	example	meaning
1st		**-ya**	**mine'ya**	my house	**-nke**	**minenke**	our house
2nd		**-kki**	**minekki**	your hose	**-'ne**	**mine'ne**	your house
3rd	M	**-si**	**minesi**	his house			
	F	**-se**	**minese**	her house	**-nsa**	**minensa**	their house

The suffix /***-kki/*** seems to be the same with the negative marker suffix /***-kki***/. But both are different in their meaning after fused to stem and suffixing position. The difference is, the possessive pronoun is always suffixed with the noun and the negative marker always suffixed with the verbal suffixes. If the suffix ***/-kki/*** is attached to the noun, it is pronoun possessive object marker. But, if it is suffixed with the verbal stem, it is negative marker.

Emphatic /Subordinate Clause suffix marker

The emphatic suffix /***-nku/*** occur between a verb in the simple perfect and the ablative instrumental suffix /**-nni/.** There is another pair of suffixes that shows the gender contrast, /**-nka/** 'M', **/-nta**/ 'F', can attach to certain adjectives to emphasize their meaning e.g., duuča**nka '**M', duuča**nta**'F'. These suffixes inflect for case (Accusative, Nominal, Genetive) [8]. Eg. Lame**nka** 'ACC.M.', Lame**nta** 'ACC.F', lame**nku** 'NOM.GEN.M', lame**nta** 'NOM.F', lame**nte** 'GEN.F'. On the other hand, subordinate clause ending in verb suffixes can be used for various types of adverbial meanings. Such suffixes include /**-ro/** 'if'. E.g., /gan-**t-u-ro/** 'hit-3SG.F.-PRF.3sg.F-if'. Sometimes the clitic that attached to an NP or an adverbial /**-no/ '**also' can be combined with the subordinating suffix /**-ro/** to express 'even if/though'. e.g., /xiss-i-he-**ro**-no/ 'cause sickness-3SG.M-PRF.3SG.M-2SG-if-even'. It can be used also unrealized or counterfactual condition, indirect question, a construction for 'wish'. it always attaches to the verb at the end of a conditional clause [8].

Negative Suffix marker

Unlike the negative proclitic **/di-/** which occur at the beginning of a constituent in the focus position and is used as a negative in various types of constituents, the negative suffix /**-kki/** is always attached to the verb of subordinate clause to negate it. Also, it can be used to negate the verb in a complement clause but Sometimes, it can be used to negate the verb in an adverbial clause followed by noun-phrase clitic **/-wa/** 'place indicator', and the suffix /**-ro/** 'if', **/-gede/ '**so that'**, /-wote/** 'when'**, /-daafira /-hura /** 'because' [8].

e.g., /mann-**u/** 'people-NOM.M' /af-a-nno-**kki-wa/ '**find-INF-3SGMIMP-NEG-place' /wor-i/ **'**put-2SGIPM'→ 'put it a place where people cannot see it.

According to Anbessa Teferra [76], there is also the negative imperative suffix of 2nd person singular /**-tooti/** and that of 2nd person plural/polite suffix /**-tinoonte/.** On the other hand, the negative suffix of all the jussive form is /**-nke/.**

e.g., /**sirb-i-tooti/** '2SG.NEG' 'do not sing!' /**sirb-tinoonte/** '2PL/POL' 'do not sing!'

/**sirb-oo-nke/ '**let me/him sing!'. Here the middle double /**o**/ is inserted.

Additionally, there is another pronoun/ noun negation suffix /**-weelo/** 'without'**.** This suffix sometimes used as postposition negating suffix. E.g., /**isi-weelo/** 'without he'. The affix /-kki/ is used for both the verb of subordinate clause and for 2nd person singular pronominal possessive, and the affix /-nke/ is also used for both the negative suffix for all jussive form

and for 1st person plural verbal possessive or pronominal possessive. But both suffixes can be different in the position by appending with the stem and by giving meaning.

2.5.5 Order of suffixes

According to Kawachi [8], any of Sidaamu Afoo words follows the order of root-derivational suffixes-inflectional suffixes. The different order of noun suffixes and their ordering relationship are discussed below.

ROOT	NML ABST NUM GEND	CASE (NOM/GEN)	POSS	CASE (DAT/LOC/ALLC/ ABL/ INST)

The right after the root, there are types of suffixes, these are nominalizing and abstracting suffixes which are derivational, and the number and gender suffixes which are inflectional. The nominalizing and abstracting suffixes each occur before any of the type of case suffixes. E.g., In /bun-ú su'n-iill-i buš-i./ 'coffe-GEN.M smell-NML-NOM.MOD.M become.bad-3SG.M.PRF', the nominalizing suffix /***-lle/*** of /*su'n-**iill-i***/ precedes the nominative suffix /**-i/** on it, and the abstracting suffix /**-imma/** of /keeraanč-**imma-te/** precedes the dative suffix /**-te/** on it. The gender/ number suffix also occurs before both the case suffix and the possessive pronominal suffix [8]. The number suffix precedes the case suffix. E.g., the plural suffix /**-uulle/** precedes the nominative case suffix /**-u/**, and the singular suffix /**-čo/** precedes the genitive case suffix /**-i/** and dative case suffix /**-ra/** in /k'aakk'**-uull-u/** and /wosin**-č-i-ra/** respectively.

It is only the nominative or genitive case suffix that can precede the possessive pronominal suffix within a word. All the other case suffixes follow the possessive pronominal suffix with in a word. E.g., /anga**-se-**nni/ **'**hand GEN.F-3SG.F. POSS-INST with her hand', the 3rd person singular feminine possessive pronominal suffix precedes the instrumental suffix /**-nni/**.

The order of Adjective suffixes also follows the following order:

ROOT	{	ADJ	NUM GEN	}	CASE

The order of adjective suffix precedes the number suffixes as:

		V-ADJ or N-ADJ	V-ADJ-PL or N-ADJ	
e.g.,	biif- 'to become beautiful'	biif-ado	biif-ad-da	'beautiful'

Both the adjectivizing suffix and the number suffix precede the case suffix [8]. In /biif-**ad-du/** the verb root is followed by a sequence of three suffixes: the adjectivizing suffix, the plural suffix, and then the nominative case suffix.

The order of Verb suffixes also places the derivational suffixes before the inflectional suffixes [8]. A derivational suffix may or may not occur, but at least one of the inflectional suffixes has to occur. When two or more types of derivational suffixes co-occur, they roughly follow the order.

ROOT	MID1/ VBLZ	PASS RECP	CAUS DBL.CAUS MID2

The middle1/ verbalizing suffix, which is the closest to the root and is required by the root that it affixes to, is glossed as [-MID] (the meaning of the middle-marked verb is used as a gloss for the root), whereas the middle 2 suffix is also glossed as –MID [8].

e.g., /ha-**'r-i-s**/ [go[-MID]-EP-CAUS-] 'to make somebody go'

/it-**am-s-/ [**eat-PASS-CAUS-] 'to make somebody sink'.

2.6 Summary

The first task in NLP application is morphological analysis. Morphological analysis studies the internal structure of a word, and it conducts the segmentation of words into their composed morphemes. We have discussed the concepts and terminologies of morphological analysis in detail. These terminologies are morphemes (the basic building blocks in morphology), allomorphs (departs from regular pattern with its pair), morpho-phonemics (the phonemic changes when a morpheme is inflected with stems), morphotactic (patterns or structures in which morphs are combined to form words), types of morphology (inflectional versus derivational and free versus bound morphology), computational morphology (theories and techniques for computational morphology analysis), and morphological analyzer (splitting words of surface level in to words of lexical level). Approaches to morphological analyzer is discussed with two types: corpus-based and rule-based. The rule based is works based on the rules that are prepared by an experts and corpus based is a machine learning that machine learnt by its own by taking the sample data, then predict the unseen example based on the corpus that was trained. Machine learning can be classified into supervised and unsupervised learning, we have chosen supervised learning for our study that uses memory-based learning. We discussed memory-based learning machine learning methods and various algorithms that fused in memory-based learning within TiMBL tools.

Finally, we have discussed a lot about morphology of Sidaamu Afoo. The detail discussion was about: the phonology of Sidaamu Afoo, morphophonemic rules, word classes (nominal, verb, adjective), and affixes that are inflected in Sidaamu Afoo and order of suffixes.

Thus, our work aims on Morphological analysis of verbs, nouns, and adjectives of Sidaamu Afoo.

3. Chapter Three: Related Work

3.1 Introduction

Morphological parsing is a necessary preliminary or complement to many kinds of computational analysis of words [22]. Developing morphological analysis has various uses for any languages. Researches on morphological analysis has been done in different languages by various researchers. During their morphological analysis study, researchers have used different methodology or techniques with different algorithms and finally they tested their system and analyzed the achieved accuracy. Most research was done by using machine learning and some was done by using rule-based methodology. Memory-based learning is applied to most of studies that has used machine learning methodology as a technique. For instance, the languages that are applied memory-based machine learning are Amharic [11], Tigrigna [6], and other foreign languages like Arabic [37], Dutch [44], etc. Their work, that was developing morphological analysis contributed as an input for future development of other NLP applications for that language. We have reviewed different papers that are related to our work. These papers are worked on different languages in Ethiopia as well as in foreign country. But there is no any work that have done in Sidaamu Afoo and our work is the first work and it should help as the starting idea for newly worked studies on Sidaamu Afoo language as well as our work is a contribution for a language. Thus, we motivated to use machine learning and apply memory-based learning for our study.

3.2 Morphological analysis for Ethiopian Languages

In Ethiopia, some researches are done for some languages. These are presented below:

Morphological analysis for Afan Oromo

Afan Oromo is one of the languages that spoken in Ethiopia and some part of Kenya and some part of Somali, Uganda, Tanzania and Djibouti [3]. Morphologically, Afan Oromo has concatenative form of suffixes and its roots of lexical words (nouns, verbs, adjectives and adverbs) end either in single or double consonants [81].

The Michael Gasser [13] presents about the HornoMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya. The author designed system architecture that consists analysis and generation. As a methodology the author used finite state transducer for each of these three languages. The data used are collected from the different sources that are

online lexicons. The system design has two functions: analysis of a single word and analysis of all words from a file. Additionally, for Amharic and Oromo, there are segmentation of a word and segmentation of all words in a file for the analysis function. For generation, there is only single function *gen* which takes a stem or root and a set of grammatical features, and returns the canonical form of the root or stem. Finally, author tests his work by manually, by peoples those are a familiar with the languages. Author has evaluated his work by using 200 verbs for Amharic and Tigrinya languages, and 200 nouns and adjectives for only Amharic, but the author haven't specified the amount of data that has used to evaluate the system of Afan Oromo. But as presented in study, for Afan Oromo author has used the Oromo lexicon for noun and verb from online sources. The system achieved better accuracy even if the data has taken is very little. The system achieved the accuracy of 96% for Tigrinya verb, 99% for Amharic verb, 95.5% for Amharic noun and adjective. For generation, system achieved 100% of accuracy for Amharic and 93% of accuracy for Tigrinya from 310 tests that has system generated. It is not evaluated for Amharic nouns and adjective for the generation. also, there is no specified amount of accuracy for Afan Oromo. As a reason, it is complicated by the great variation in the use of double consonants and vowels by Afan Oromo writers. In addition, the study was not done with similar word classes for all three classes, for instance, for Tigrinya only verb, and verb and noun for Amharic and Afan Oromo.

Abebe Abeshu and Dida Midakso [3], designed morphological synthesizer for Afan Oromo. Their system synthesizes Afan Oromo words automatically and authors have tested the system with manually prepared data. As authors stated the output or result of their works is serve as a suggestion list for spell checker. Authors started their work and developed algorithms that suitable for Afan Oromo language from scratch, because there is no any try to Afan Oromo before this work, in the area of morphological processing systems of NLP, else there is some linguistics work. This limitation makes their effort couldn't be comfortable with Afan Oromo. They have used rule-based approach with the rules that are prepared by experts. To evaluate the experiments of the study, the authors collect input data from different sources, and the collected stems are stored with their tags and subtypes for further use by the system. Although, to evaluate the algorithm, the error counting approach was adopted, and the output of synthesis is analyzed based on the comparison of data that are prepared by linguistics with generated output. For instance, if the generated word is matched with the manually prepared one, it

accepted as correct, else it is incorrect. Authors has tested their work with 10 verbs and 7 noun test stems and the result of each test has tested by experts. Thus, the average accuracy for verb stems are 96.28% are correctly generated words and the noun stems are 97.46% are correctly synthesized. As we reviewed the study, it is limited on very less data. For instance, 230 surface forms are generated from 7 noun stems and 230 generated words from 10 verb stems.

Getachew Mamo et al., [5] studied probabilistic and grouping methods to develop a morphological root identification model for Afan Oromo. The authors combine probabilistic and grouping methods after each method performs its own particular task in identifying morphological roots. The study was examined its performance in terms of derivational and inflectional boundaries. To evaluate the effectiveness of methodology, experiments are implemented and compared the methodology with some state-of-art methods, to make the experiments stronger. The data are randomly selected from a list prepared for a spell checker for Afan Oromo. The selected words are 2000 and some experts helps the authors by segmenting and tagging segmentation positions as root, derivational and inflectional suffix boundary. Finally, the authors evaluate their model in terms of precision and achieved 88.4%, then the system compared with others work and the achieved result is better.

Getachew Mamo [81] also tried OroRoots: rule-based root generation system for Afan Oromo. The author used rule-based methodology to generate the root for Afan Oromo. The system was evaluated with testing wordlist and has been experimentally shown that it improves the performance of state-of-the art methods for the language. The testing wordlist contains 2000 segmented unique lexical words and the evaluation has been taken through the number of correctly indicated root boundaries, and achieved 98.4% precision. Author compare their work with their previous work [5] 88.4% precision, the present work achieved better precision.

Morphological analysis for Amharic

Kibru Lisanu [15] developed a prototype of automatic morphological synthesizer for Amharic specially for perfective verb forms. The author used the combination of rule-based and neural network approaches to design and developed a prototype, and tested the study separately by using both approaches. For instance, rule-based approach generates all the roots successfully, whereas, the neural network predicts the type of roots in the test data set with an accuracy of 81.48% from totally 273 roots. For a new case consisting of 14 roots the neural network

identifies in different types that are divided into A, B, C. The accuracy for perfective verb that type A predicts 80%, type B predicts 25%, and type C predicts 100%. The author tries to use two different techniques; it wastes a lot time to prepare and test manually and in addition, it takes another time to test neural network. It is better to integrate both as one technique to reduce the time of learning and testing. Also, the author's work is limited on perfective verb; it is better even it covers whole verbs it can't cover whole word class.

Tesfaye Bayu [14] developed automatic morphological analyzer for Amharic by employing a combination of unsupervised learning and auto segmental analysis approaches. Corpus based learning are applied to the study, rather using supervised learning, author used unsupervised approach. The author used language independent system called Linguistica2001 system that are tested in other languages like English and French, as one of testing system in addition to their own system. However, Linguistica2001 can't handle morphological analysis task of stem internal for Amharic. For this reason, another approach is adopted based on the principle of auto segmental phonology. To apply this, author design algorithms and data structures, and develop a system called Amharic Stems Morphological Analyzer (ASMA). The author prepared data from different sources and used separate data for both systems, finally tested the study and achieved the precision and recall results from both systems. For instance, Linguistica2001 achieved 87% of words from the test data (433 out of 500 words) (i.e., 95% Precision and 90% recall), and ASMA has achieved 94% (241 out of 255 stems) from the test data. As author stated, the data or corpus that has used for study is too small that compared with other similar researches worked for other language and author concluded that the result is somewhat acceptable that got from both systems. It is not clear that if Linguistica2001 can't handle Amharic internal stem, why the author used it.

Wondiwosen Mulugeta and Gasser [10] has presented learning morphological rules for Amharic verbs using inductive logic programming. The authors have conducted supervised machine learning for their study. Inductive logic programming (ILP) has used and CLOG has implemented (a Prolog based ILP system that have used, and it learns rules as first order predicate decision list). The reason that they used ILP is, it can be used to fast-track the process of learning morphological rules. What they done is identifying simple affix and confronts one of the challenges in template morphology by learning the root-template extraction as well as stem-internal alternation rule identification. The authors have prepared the training data

manually according to the structure of the background predicates can be used for learning process. They collected 108 stem extraction and 19 root template extraction rules from data trained. Finally, the system evaluated on a test set containing 1784 Amharic verbs and achieved 86.99% accuracy. But their system is only conducted for subject marker, nothing about object marker, and it has not included in the learning process that system is practical for all the prefix and suffixes. Thus, they taken action as future work to handle this.

Saba Amsalu and Gibbon [12] tried to construct a generic morphological analyzer for different NLP applications, and they tried to construct a tool that can analyze automatically Amharic words from natural language texts. The authors used finite state technique and implemented finite state toolset. The data was collected from the Book of Matthew chapter 1-5 in holy bible, taken as a corpus. The evaluation was made for all word classes and tests them correctly. The authors have tested the system through 468 verbs and 650 nouns from corpus and achieved 94% and 85% accuracy respectively. They have concluded that increasing the size of corpus may increase an accuracy of implementation.

Mesfin Abate and Yaregal Assabie [11] developed Amharic morphological analyzer using memory-based approach. As authors stated a supervised machine learning with data-driven experimental approach was used. Memory-based supervised machine learning method was used to predict a new unseen class (that has not seen previously) based on an instance that are given for training. The manually prepared morpheme annotation that inflected Amharic words was prepared, and used in training phase (one of phase that learnt from a given instance and used as a classifier in analysis phase). The authors used TiMBL as learning tools with different algorithms. Among them, authors used IB1 and IGTREE algorithms for their study with TiMBL tool. In order to examine the system, the experiment was conducted by combining the noun and verbs. To evaluate a system, the authors collect totally 1022 words (841 verbs and 181 nouns), and generated 8075 instances (6719 from verb and 1356 from nouns) with 26 different class labels. The authors have used 10-fold cross validation for both of these algorithms to evaluate the performance of the model, and achieved the average accuracy of 93.6% by IB1 and 82.3% by IGTREE with optimized parameter, and achieved the accuracy of 92.02% and 76.27% for IB1 and IGTREE respectively with the default parameters. Also, authors have tested the system using default parameter setting for IB1 with leave_one_out testing method, and achieved 93.3% of accuracy. To increase the performance

of the system they have increased the size of training data. Thus, we have used the same techniques and tool with some customization and newly added prototype because, their achieved result is better besides of others that we have reviewed yet and the structure of Sidaamu Afoo is may fit with this methodology to prepare annotated data.

Morphological analysis for Ge'ez

Desta Berihu [4] tried to design and implement an automatic morphological analyzer for Ge'ez verbs. The author has used rule-based approach by combining CV-based and Two-Level morphology to design the model as well as to implement the prototype of the analyzer. By taking the morphological, morpho-phonological and orthographic properties into considerations, author has designed the algorithms starting from scratch. The data has been collected from Holy Bible of all twenty-seven New Testament, and prepared manually by domain expert. Evaluation was done by comparing the output that analyzer generated and manually prepared analyses of the same verb-set by the language experts. Accuracy was done with two-levels of verb sets: feature-of-verbs level and verb level. Thus, the system achieved 92.05% accuracy at feature level and 73.98% at verb level. But author has studied only the ቀተለ /qätälä/ category verb forms, and its derivations and inflections. It is better to do the generalized verb or even for additional word classes.

Yitayal Abate and Yaregal Assabie [6] Also developed morphological analyzer for Ge'ez verb by conducting Memory-based learning. Authors has used supervised machine learning approach by applying memory-based learning. The authors have prepared morphologically annotated data that are suitable for extracting features. The features are extracted from these annotated morphemes and given to memory-based learning tools (TiMBL). As authors stated, the system has two phases: the training and analysis phase. In the training phase, memory-based machine learning takes place, and in analysis phase, morphological analysis was done through making instance by extracting features. Totally 1105 words was collected that counted 12135 instances with 31 different class labels, and used 90% of those words for training and 10% of those words for testing. The authors have used TiMBL as developing tools that has different algorithms and parametric setting with default and optimized values. The authors developed system by using python language and used IB2 and TRIBL2 algorithms in TiMBL for implementation by using the default and optimized parameter setting. To evaluate the performance of the system 10-fold cross-validation technique have

used, and achieved overall accuracy of 93.24% and 92.31% for IB2 and TRIBL2 respectively with optimized parameters. Also, the overall precision, recall, and F-score achieved with optimized parameters of IB2 classifier was 55.6%, 56.3% and 59.95% respectively, and using TRIBL2 was 58.8%, 60.3% and 59.54% respectively. Finally, they have drawn the curve and concluded that, if the number of training dataset increase, the accuracy on unseen data can be increased. Thus, IB2 algorithms shows better result than TRBL2 algorithm for Ge'ez verb morphology.

3.3 Morphological Analysis of Non-Ethiopian Languages

Daelemans and Bosch [44] developed Memory-based morphological analysis for Dutch language by using supervised, inductive memory-based machine learning approach. The authors have classified a letter sequence as marking different types of morpheme boundaries. Inflections of verbs, nouns, and adjectives for Dutch as authors studied is mostly inflected by suffixation, like Sidaamu Afoo. They have used the corpus from CELEX lexical database and drew an instance for a training. CELEX is lexical database that contains 247,415 morphologically analyzed word forms and features of Dutch language. The instance was generated by using windowing method, each instance is focused on one letter, and consisting a fixed number of left and right neighbor letters. Thus, they generated 2727462 instances from these words with 2422 different class label. They have used TiMBL tool for their study, this tool is already developed by these authors for the morphological analyzer of Dutch, nowadays, it is helping as contributions for other languages like Arabic, Ge'ez, Amharic, Sidaamu Afoo. They have developed that tool by combining several algorithms with default and optimized parameters setting. So, they have used IB1-IG memory-based learning algorithm, it is the extension of IB1. The k-nearest neighbor was used for searching the best nearest match for the newly given instances from memory that has earlier learned and stored in memory. To evaluate the experiments, the authors used 10-fold cross-validation technique. It divides total instances into 10 subsequent partition and has stored 90% for training and 10% for testing, then resulting in 10 experiments. The evaluation of performance is achieved accuracies of: 84.6% correct fully-analyzed words (for 64.6% unseen words), and 96.7% correctly segmented and coarsely-labeled words (90% for unseen words). The precision and recall of fully-labeled morphemes are estimated in real texts to be over 93% and the prediction of syntactic classes of unknown word forms was 91.2% correct; the corresponding free-text

achieved 97.2% correctly-tagged word forms. Finally, the approaches that have used was compared with the previously used traditional approach, and achieve better result. As the authors stated, memory-based learning approach has various advantage: learning is automatic and fast, it is language independent (it only deals with morphologically analyzed corpus in any language), no need of more linguistic knowledge than explicitly present in the corpus used for training; this lead us to use MBL.

Clark [56] discussed memory-based learning of morphology by using Stochastic Transducers approach by combining supervised learning of morphology, and trained using the algorithms Expectation Maximization and to maximize the conditional probability the joint descent algorithm has possibly used. The author has studied on different languages: English, German, Slovene, and Arabic. The author presented the algorithms for learning the finite-state transduction between pairs of uninflected and inflected words. The data has collected from different sources and used 28, 729 total datasets from different language (English past tense from another research 1394, English past tense from CELEX 5324, German from CELEX 16970, Arabic plural 859, Arabic broken plural 3261, and Slovene genitive noun 921). To evaluate the performance of the study, 10-fold cross validation has applied, and finally, tested using two approaches: conditional likelihood (CL) and using the MBL with unscaled conditional features. After testing, author achieved different accuracies for these languages: for English through both models 85.8% and 79.3% in MBL for different research gathered data, and 61.3% and 72.1% in CL for CELEX data, for Arabic through five models 16.7% in MBL and 1.6% in Cl, and for Slovene 98.9% in MBL and 63.6% in CL. Generally, for these all language, it achieved better results with the MBL approaches rather than CL approaches.
As Bosch et al, [82] applied the memory-based learning to morphological analysis and part-of-speech tagging of written Arabic data from Arabic Treebank1. The authors have used the Arabic Treebank 1 (ATB1), version 3.0 as their data source, and collected 185061 tokens in a corpus and extracted 16626 unique word type and 129655 analyses. By testing the analyzer, the authors have achieved 64% precision and 89.7% of recall. The authors have developed PoS tagger that trained and predict an unvoweled word in context, then a concatenation of the PoS tags of its morphemes. So, they tried to train on the full ten-fold cross-validation in analyses and tested on eleventh fold. Thus, training set contains 150966 words in 4601 sentences, and test set contains 15102 words in 469 sentences. Also, 358 unique tags occur in

the corpus, 947 words that are not found in training set are occur only in test set. The accuracy of the tagger on the above corpus is 91.9% correctly assigned tags, on the 14155 known words in the test set the tagger attains an accuracy of 93.1%; on the 947 unknown word it attains lower 73.6% accuracy. Finally, authors have merged both analyzer and PoS tagger, achieved a recall of the contextually appropriate analysis of 0.9% on test words, and also a low precision of 0.64% largely caused by over generation of invalid analyses.

Yet, the accuracies that achieved for Ge'ez verb [6], Amharic Verb [11], Dutch [44] , and Different language [56] is better than any other researches that we have reviewed. For those researches, the approach used is supervised machine learning by applying memory-based learning. Even if the language structures of these researches studied is not similar with Sidaamu Afoo, we decided to use these tools as well as algorithms by using default and optimizing setting with developing a prototype based on the morphological feature and structure of Sidaamu Afoo.

3.4 Summary

From Ethiopia, for different languages, there are different works was tried by several researchers. Especially, for Amharic there are so many researches done; some of related with our study are reviewed and discussed above. Ge'ez is ancient Ethiopian language and some papers are reviewed in our study: Desta Berihu's [4], and Yitayal Abate and Yaregal Assabie's [6] work. For Afan Oromo, there is limited researches done; also no more scholars are tried to develop direct morphological analyzer rather than developing morphological synthesis for Afan Oromo yet what we have reviewed, but there are papers that are done for root extraction for Oromo verb [81]. Though, Sidaamu Afoo language structure little similar with Afan Oromo than other Semitic language but nothing is tried on it. Mesfin Abate and Yaregal Assabie [11], and Yitayal Abate and Yaregal Assabie [6] has developed their system by applying MBL machine learning by combining different parameter and they have achieved good accuracy, 93.6% and 82.3 % for Amharic and 93.24% and 92.31% for Ge'ez. Likely, there is also different papers that has used MBL machine learning then achieved better accuracy. Thus, we proposed to use MBL machine learning to achieve a better accuracy for Sidaamu Afoo morphological analysis.

4. Chapter Four: Design of Morphological Analysis for Sidaamu Afoo

4.1 Introduction

In this chapter, we will discuss the design of morphological analyzer for Sidaamu Afoo with the implementation by using the tools and algorithms. Morphological analyzer analyses the internal structure of the words [20], and it is essential for computational processing of any language. Thus, we designed the architecture of a morphological analyzer based on the morphological properties of Sidaamu Afoo language that we have discussed in Chapter 2.

4.2 System Architecture

We have designed the architecture of a morphological analyzer for Sidaamu Afoo. Our architecture has training and morphological analysis phase. Training phase contains the morphophonemic analysis, process of morphologically annotating of words, stored words which are morphologically annotated that will feature can be extracted from by using a fixed length of windowing method, memory-based learning and its model that helps to train an instance which extracted from morphologically annotated words.

In Morphological analysis phase, the words given are analyzed their morphophonemic process and features are extracted from these words to destruct a word. After extracting features it could classify their class label, if direct match of instances was happened, else it could estimate the class label based on the instances stored in memory, and stem is extracted by reconstructing and inserting morpheme as boundary marker then finally gives the lexical form of word through morpheme with its function. Our architecture is show in Figure 4.1.

4.3 Morpheme Annotation

We have prepared annotated words starting from nothing because we haven't got prepared data from anywhere. The preparation is based on the morphological formation of the language for all word class that are considered in our study. We collected data from different sources like thesis, hardcopy dictionary, textbooks, and from peoples that are familiar with the language and analyzed morphophonemic process, then prepared manually these words as annotated words and checked by linguistic expert from a linguistic department of Sidaamu Afoo in Hawasa university. During morpheme annotation, morphophonemic analyzed for each word of collected. After analyzing morphophonemic insertion of boundary markers is

takes place. Insertion of boundary markers is based on identified class labels according to language's morphological structure.

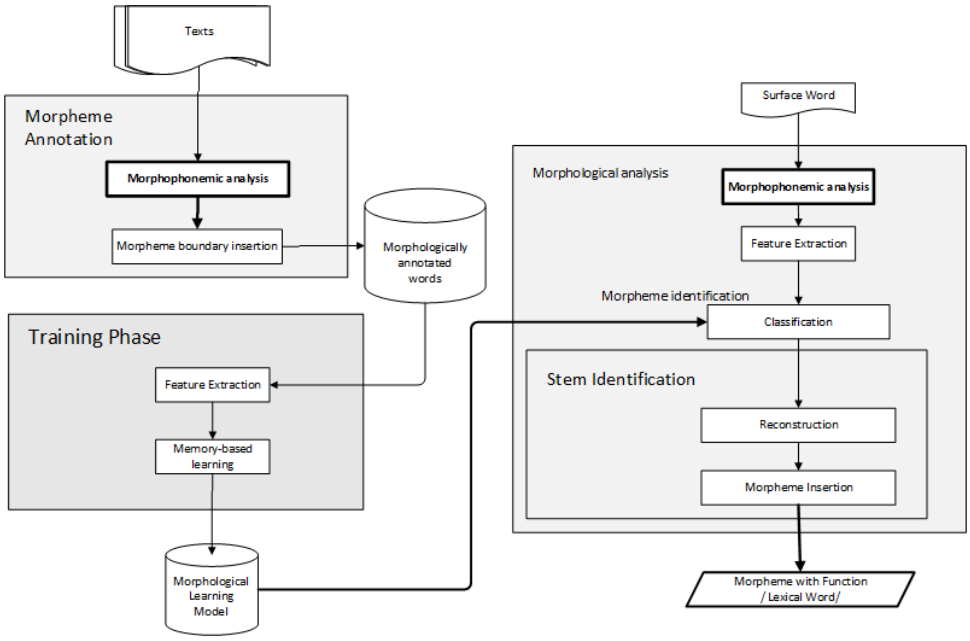

Figure 4.1 Architecture of Morphological Analyzer for Sidaamu Afoo.

4.3.1 Morphophonemic Analysis

As we have explained in Chapter 2, Sidaamu Afoo has its own Morphophonemic rules. These are, Epenthesis, Metathesis, and other assimilation rules. As we discussed further, the maximum consonantal sequence in Sidaamu Afoo is two. When consonant ended verb-stems are fused with consonant initial suffixes, it may break these rules; at this time epenthesis rule is takes place. Epenthesis means inserting /*i*/ between a successive consonant to break being more than two consonantal sequences. E.g., /*sirb-/+/-tu/*→/*sir**bt**u**/→/*sirb**i**tu/ 'she/they sang'*

Metathesis is a phonological process whereby two adjacent consonant phonemes are transposed. E.g., /*it-/+/-nummo*/ →/*i**nt**ummo/ 'we ate'*

*/fan-/+/- ɓ(dh)/ →/fa**'n**-a/ 'to open oneself'.* we have discussed all these rules in Chapter 2.

Before morphologically annotate our data, the morphophonemic rule must be taken place. Because if we annotate the word as it is, a machine can learn by assuming that affixes are correct and may classify affixes with mismatched boundary marker or classes. For example, the word /*i**nt**ummo*/ is composed by the verb /*it-*/ and the affix /*-nummo*/. This kind of words are generated when the stem final consonant is obstruent and 1PL affixes are fused together.

Some such kind of verb stems changes the character of stem final with affix's initial character when it takes these affixes. Also, sometimes /*n*/ could be changed into /*m*/ of 1PL affixes. For instance, the verb stem /*gib-*/ is merged with /*-neemmo*/ it gives the word /*gimbeemmo*/ 'we refused'. When we annotate these words without reordering phoneme by using rule, machine can learn /*int-*/ as verb and the /*-ummo*/ as affix, the same to the word /*gib-*/. Hence, there is no such verb stem in Sidaamu Afoo and incorrectly classified 1st person plural simple perfect affix /*-nummo*/ as 1st person singular masculine simple perfect affix /*-ummo*/. To handle this kind of problem, fist we have reordered the phoneme based on the language's morphophonemic rule. When we reorder the above example, it rearranged as the verb /*it-*/ and the affix /*-nummo*/ and it reordered the sequence of consonant as /*itnummo*/ and annotated as /*itVnummo5/*; V and 5 are boundary markers (see Appendix B).

The word that contains /'r/ when spoken for masculine is always changed into /-dh-/ at that position when it is spoken for feminine. E.g., */ha'r-e*/ 'he went' →/*hadh-e/ 'she went'.* Also, the affix /-n-mmo/ is attached with the stem final consonant of */-ɓ(dh)*/, it takes glottal stop before /*n-mmo/.* E.g., /*ha'noommo*/ is generated from the word /*haɓ(dh)/* and the affix /*noommo*/. During the preparation of morphologically annotated data /*ha'noommo*/ is rearranged into /*haɓVnoommo€*/ for further learning and classification; this also increases the general accuracy of our system. Some words that verb's stem final consonant is sonorant are geminate its stem final consonant and deletes the affix's initial consonant /*n*/ when it takes affixes /*-neemmo, -noommo, -nummo*/, /*-no*/, /*-ni*/. E.g., /fool-/+/*-neemmo*/→/*foolleemmo*/ and /*fool-*/+/*-no*/→/*foollo*/. In addition, the word /*waam-*/ is inflected in to /*waa'm-i*/, when it takes auto benefactive affix */-ɓ(dh)/.* Rather than classifying incorrect verb stem as /*waa'm*/ that not exist in a language, it is better to rearrange and classify correctly as verb stem /*waam*/ and affix */-ɓ(dh)/.* Some verb stems that ends with obstruent and takes affixes /*-tini*/ 2PL/POL PRF, /*-tu*/ 3SGF/PLPRF, and /*-tino*/3SGFPPRF is geminate the verb's final consonant and append the affix by deleting affix's initial consonant /*-t-*/.

E.g., /amad-/+/-tino/ → /amaddino/, …+/-tini/→/amaddini/, …. +/-tu/→/amaddu/.

/xib-/ +/-tino/→/xibbino/.

There are some Exceptions when these above rules are takes place in our system; these rules acquire additional linguistic study. In our system we have used it in both training and

morphological analysis phase; there is no more difference except process is applied by manually and automatically. In training phase, we apply it by manually through rearranging the phonemic of a word (if rearrangement is needed) during preparing morphologically annotated data but in morphological analysis phase, all these rearrangements are done by automatically by using the algorithm. The algorithms that works these rules are discussed below Algorithm 4.1.

Input: Surface word

Output: rearranged surface word

---

1. Define sonorant, obstruent phonetic
   Define 1PL, some affixes that always bring change during inflections
2. If word contain 1PL and 1PL initial consonant n is changed in to m, change m to n and swap its position with successive consonant
3. If word contain 1PL or some other affixes with the deletion of first consonant n and stem final consonant is geminated, delete one geminated consonant and add deleted affix initial consonant at that position
4. If word contain 1PL and 1PL affix initial consonant is swapped the position with stem-final consonant, swap their position
5. If sonorant or 1PL are followed by glottal stop except some usual allowed words, change glottal stop by '*ð(dh)*'
6. If r is followed by glottal stop, glottal stop+r is changed by '*ð(dh)*'
7. If other rules are existed, fix by rearranging based on suitable rule
8. If word is rearranged correctly, finish
   Else go to step 2

Algorithm 4.1 Morphophonemic Analysis Algorithm

4.3.2 Morpheme boundary insertion

To prepare annotated data, the tasks should be identified and performed is: *identifying inflected words, segmenting these words into prefix, stem, and suffixes, then inserting boundary marker between these segments, and finally describing those each marker with their functions*. As Sidaamu Afoo is concatenative suffix-based language, our annotated data preparation is more focused on segmenting suffixes with stem and with each other of suffixes. We have identified several suffixes and 1 prefix that helps us to prepare annotated word (see Appendix A). As we discussed in Chapter 2.5.5, the Sidaamu Afoo verbs, nouns and adjectives has their order of suffixes and ordering relationships. For all word classes there is only one position for prefix because there is only one prefix are there in Sidaamu Afoo.

The order of suffixes for verb is complex because one order may contain several segments. For instance, verb has three order of suffixes that follow the root, the word /***di-amman-nummo-nsa-kki-nni-ti-na***/ has not only three segment but it has 8 segment with three suffixes order (*prefix-verb Stem- person/gender subject- person/gender object-NegSuff-PostPos-RelNom-Conj*). So, segmenting this kind of suffixes is difficult for verbs. According to verb, noun's and adjective's segmentation is not difficult. We identified the classes (boundary marker) that helps us to prepare annotated words. During preparation, these identified boundary markers are inserted between each corresponding affixes of a given word. Finally, after all collected words are prepared by inserting boundary markers as annotated word, it stored in a database for further feature extraction. These classes represent different features and grammatical functions of the verbs, nouns, adjectives, etc. (Appendix B). These identified boundary markers may help machine learning to learn and predict unseen examples previously after instances are created by using sequences of feature values and corresponding classes for each morphologically annotated word. The morphemes that are attached to the stems of noun which filled with the appropriate grammatical feature for each segmentation are prepared as shown Table 4.1. Preparing these kinds of annotated lexicons are helped for suitable extraction of feature.

Table 4.1 Example showing annotation of nouns

Word form	Description	prefix	stem	suffix
adotenni	*'by milk'*	-	ado	-te(ψ)-nni(I)
dibunaho	*'it is not coffee'*	di(G)-	buna	-ho(Ø)
galte'yarano	*'for my wife too'*	-	galte	-'ya(Ω)-ra(D)-no (C)
digaltesi	*'not his wife'*	di(G)-	galte	-si(ϕ)

Similarly, we prepared morphologically annotated word list for verbs with its inflectional formation. Tokens are manually annotated from morphologically annotated word list for all verbs, nouns, and adjective. Then Instances are created from these tokens that are suitable for feature extraction. The annotated word list for some sample verbs are shown Table 4.2.

Table 4.2 Example Showing annotation of verbs

Prefix	Verb	Suffixes			word	Description
	agar	-i(3)			agari	*' he waited'*
di(G)	agar	-tu (8)			diagartu	*'she haven't waited'*
	agar	-umma(4)	-se(λ)		agarummase	*'I (F) waited she*
	agar	-umma(4)	-nke(ϕ)		agarummanke	*'I (F) guared for us*
di(G)	agar	-umma(4)	-'ne(()		diagarumman'ne	*'I (F) haven't waited you(PL)*
	agar	-umma(4)	-ne(()	-ro(J)	agarumma'nero	*' if I (F) waited you(PL)*

Also, for adjective we have prepared morphologically annotated word list with its inflectional forms like verbs and nouns.

4.4 Training

The morpheme that remains unchanged during morphological analysis that Sidaamu Afoo inflected is stem; any word inflects or derived through its stem or base of word. While words are inflected or derived on/from any words, phonological changes are takes place in stem. Therefore, we studied the morphological formation of Sidaamu Afoo for verbs, nouns, and adjectives. After we identified a common property of morphological formation for these word classes and grammatical features of morphemes, we prepared a morphological database. Hence, we prepared the data by manually annotating sample words with their pattern by inserting boundary marker and stored in database that is used for a training. After morpheme annotation, in training phase feature extraction extracts the features that helps to learn learning model.

4.4.1 Feature Extraction

As we discussed, After the analysis of morphophonemic rule for collected wordlist, the words are manually annotated from these lists stored in database. The preparation of manually annotated words should help for feature extraction. The features are extracted automatically from prepared dataset by the concept of windowing method to make instances. The concept of windowing works by dividing windows into left and right context to store fixed-length string of instances in both sides, at middle it contains the focus letter and the right most side

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 are no any symbols used as boundary marker, 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 4.2 Feature Extraction Algorithm

it contains the label of classes for each instance. As Algorithm 4.2 (Adapted from [5]) shows instances are created by sliding each character down as a focus letter by preceding left context and following right context characters. Each example in feature extraction is focuses on character based by including fixed length of right and left neighbor character using 1-12 to 12-1 windows which yields twenty-six features. The input character in focus, including the twelve preceding and twelve following characters are placed in the windows. From the morphologically annotated database, features are extracted through this method by sliding windows over each character from word in the lexicon to be suitable for memory-based learning. Sidaamu Afoo writing system is usual exist in independently with consonant and vowel order. This makes memory-based learning and feature extraction to be simple and no need of additional study for orthography like Amharic [11] and Ge'ez [6].

When the algorithm with the character-based representation of the Sidaamu Afoo word '**ammannummonsakkinniitiro'** described as follows:

From the Table 4.3 the equal sign (=) is used as a symbol which shows there is no character at that position. From a given word, 24 instances are created with associated eight classes. For instance, the thirteenth instance class is ')', it indicates the morpheme ending with 'nsa' is a suffix represents the *3rd person plural verbal possessive or pronominal possessive object marker* of verbal stem 'amman' (firth instance). Thus, the character-based representation of

words exhaustively transcribes their deep structure of phonological process and segments each character one at a time.

The cause that extracting feature like this is that we have used machine learning to develop a morphological analyzer for Sidaamu Afoo in verbs, nouns, and adjectives. Hence, we may not handle all productive classes for inflections and derivations of all verbs, nouns, and adjectives, because of it may very broad and the lack of time or it is impossible to cover all classes in our study, it could be covered in our future works. To be learned and trained for machine learning, it is mandatory to extract feature and preparing annotated data for extraction. The database for the learner consists of description of instances in terms of fixed number of feature values.

*Table 4.3 Character based representation of the word /*ammannummonsakkinniitiro/

No of	Left Context												Focus	Right Context												Class
1	=	=	=	=	=	=	=	=	=	=	=	=	a	m	m	a	n	n	u	m	m	o	n	s	a	**0**
2	=	=	=	=	=	=	=	=	=	=	=	a	m	m	a	n	n	u	m	m	o	n	s	a	k	**0**
3	=	=	=	=	=	=	=	=	=	=	a	m	m	a	n	n	u	m	m	o	n	s	a	k	k	**0**
4	=	=	=	=	=	=	=	=	=	a	m	m	a	n	n	u	m	m	o	n	s	a	k	k	i	**0**
5	=	=	=	=	=	=	=	=	a	m	m	a	n	n	u	m	m	o	n	s	a	k	k	i	n	**V**
6	=	=	=	=	=	=	=	a	m	m	a	n	n	u	m	m	o	n	s	a	k	k	i	n	n	**0**
7	=	=	=	=	=	=	a	m	m	a	n	n	u	m	m	o	n	s	a	k	k	i	n	n	i	**0**
8	=	=	=	=	=	a	m	m	a	n	n	u	m	m	o	n	s	a	k	k	i	n	n	i	i	**0**
9	=	=	=	=	a	m	m	a	n	n	u	m	m	o	n	s	a	k	k	i	n	n	i	i	t	**0**
10	=	=	=	a	m	m	a	n	n	u	m	m	o	n	s	a	k	k	i	n	n	i	i	t	i	**5**
11	=	=	a	m	m	a	n	n	u	m	m	o	n	s	a	k	k	i	n	n	i	i	t	i	r	**0**
12	=	a	m	m	a	n	n	u	m	m	o	n	s	a	k	k	i	n	n	i	i	t	i	r	o	**0**
13	a	m	m	a	n	n	u	m	m	o	n	s	a	k	k	i	n	n	i	i	t	i	r	o	=	**)**
14	m	m	a	n	n	u	m	m	o	n	s	a	k	k	i	n	n	i	i	t	i	r	o	=	=	**0**
15	m	a	n	n	u	m	m	o	n	s	a	k	k	i	n	n	i	i	t	i	r	o	=	=	=	**0**
16	a	n	n	u	m	m	o	n	s	a	k	k	i	n	n	i	i	t	i	r	o	=	=	=	=	**G**
17	n	n	u	m	m	o	n	s	a	k	k	i	n	n	i	i	t	i	r	o	=	=	=	=	=	**0**
18	n	u	m	m	o	n	s	a	k	k	i	n	n	i	i	t	i	r	o	=	=	=	=	=	=	**0**
19	u	m	m	o	n	s	a	k	k	i	n	n	i	i	t	i	r	o	=	=	=	=	=	=	=	**I**
20	m	m	o	n	s	a	k	k	i	n	n	i	i	t	i	r	o	=	=	=	=	=	=	=	=	**{**
21	m	o	n	s	a	k	k	i	n	n	i	i	t	i	r	o	=	=	=	=	=	=	=	=	=	**0**
22	o	n	s	a	k	k	i	n	n	i	i	t	i	r	o	=	=	=	=	=	=	=	=	=	=	**W**
23	n	s	a	k	k	i	n	n	i	i	t	i	r	o	=	=	=	=	=	=	=	=	=	=	=	**0**
24	s	a	k	k	i	n	n	i	i	t	i	r	o	=	=	=	=	=	=	=	=	=	=	=	=	**J**

Each character from extracted feature is used as once as Focus **(F),** left (**L1-L12),** and right (**R1-R12)** characters as shown Table 4.4. Each Feature is separated by commas to be used as input for the learner. The tool that we used, TiMBL use different format for training and testing file including, Compact, C4.5, ARFF, Columns, etc. From these we have used C4.5 as it is a default format, it represents the feature value are separated by comma and the last feature value denotes the class of the instances (see Appendix C for more sample extracted features). Therefore, it requires the feature values with their corresponding classes.

*Table 4.4 Feature extraction of the noun word /*adatero*/*

L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	F	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	C
=	=	=	=	=	=	=	=	=	=	=	=	**a**	d	a	t	e	r	o	=	=	=	=	=	=	0
=	=	=	=	=	=	=	=	=	=	=	a	**d**	a	t	e	r	o	=	=	=	=	=	=	=	0
=	=	=	=	=	=	=	=	=	=	a	d	**a**	t	e	r	o	=	=	=	=	=	=	=	=	N
=	=	=	=	=	=	=	=	=	a	d	a	**t**	e	r	o	=	=	=	=	=	=	=	=	=	0
=	=	=	=	=	=	=	=	a	d	a	t	**e**	r	o	=	=	=	=	=	=	=	=	=	=	ψ
=	=	=	=	=	=	=	a	d	a	t	e	**r**	o	=	=	=	=	=	=	=	=	=	=	=	0
=	=	=	=	=	=	a	d	a	t	e	r	**o**	=	=	=	=	=	=	=	=	=	=	=	=	J

The class of third instance is ‘N’ in Table 4.4 that tell the word ‘ada’ is noun, not indicate the morpheme ends with ‘a’ is a noun like Ge’ez verb’s instance [6]. Rather, the class of fifth instance ‘Ψ’ indicate the morpheme ending with ‘te’ implies *singular feminine nominative dative*. The classes that indicated by ‘0’ is the default category and it indicates the endings of focus character that present in that row indicate nothing class. The features that are supported by learner tool must be separate each character by comma and top row is unnecessary. So, the above feature looks like Table 4.5 when it supports by learner. We have prepared our morphological annotated word lists with total number of sixty-seven classes (boundary markers) (Appendix B). But our class would be increase when our study covers additional affixes of Sidaamu Afoo verbs, nouns, and adjectives inflections and derivations that are not covered in our study by the case of different limitations.

Table 4.5 Feature format that supported by learner tool

```
=,=,=,=,=,=,=,=,=,=,=,=,a,d,a,t,e,r,o,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,a,d,a,t,e,r,o,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,a,d,a,t,e,r,o,=,=,=,=,=,=,=,=,N
=,=,=,=,=,=,=,=,=,a,d,a,t,e,r,o,=,=,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,a,d,a,t,e,r,o,=,=,=,=,=,=,=,=,=,=,ψ
=,=,=,=,=,=,=,a,d,a,t,e,r,o,=,=,=,=,=,=,=,=,=,=,=,0
=,=,=,=,=,=,a,d,a,t,e,r,o,=,=,=,=,=,=,=,=,=,=,=,=,J
```

4.4.2 Memory Based Learning

Memory-based learning has its defining characteristic that it stores in memory all available instances of a task, and that it extrapolates from the most similar instances in memory to solve problems for which no solution is present in memory [34]. When new instances are presented to a memory-based learner, it searches for the best matching instances in memory, according to different metrics. When it has found the best matches (the nearest neighbors), it returns their solution (classification) to the new instance [37]. The tool that are many machine learning

approaches are applied to solve NLP problem are used for our study, TiMBL. It is developed by using C and C++ languages and has Python extension. Our study used TiMBL without extension. As we have discussed in Chapter 2, TiMBL implements several learning algorithms like **IB1**, IB2, IGTREE, **TRIBL**, and TRIBL2. As a common behavior all algorithm stores some instance in a memory for further prediction of unseen examples. When it tests, the classification process is takes place; if there are direct matched instances in memory from previously learned instance, it returns the classes label which it corresponds. But if there is no direct match in a memory, the extrapolation is processed through predicting the most similar stored cases by using k-nearest neighbor methods. The difference of these algorithms are incorporated in TiMBL lie in: the definition of similarity, the way of the instances is stored in memory, and the way the search through memory is conducted [55].

The features (parameters) that are tuned in TiMBL is various: it is Fast, decision-tree-based implementation of k-nearest neighbor classification, implementation of various algorithms that discussed above, fast leave-one-out and internal cross-validation testing, similarity metrics: overlap, MVDM, cosine, etc., feature weighting metrics: information gain, gain ratio, chi squared, shared variance, and etc. [55]. From these features we have selected **IB1** and **TRIBL** algorithms and optimized parameters setting with **IB1**: modified valued value difference similarity metrics (MVDM: '-mM'), shared variance weighting metrics ('-w4'), Inverse distance class voting metrics ('-dID'), and number of nearest neighbors used for extrapolation as 5 ('-k5'), and optimized parameter setting with **TRIBL:** Information gain weighting metrics (-w2) as feature weighting metric.

IB1 is usually, but not always lead to more accuracy at the cost of more memory and slower computation in the trade-off between generalization accuracy and efficiency [70]. **IB1** is the simplest instance-based learning algorithm [71] and classified into lazy classifier for storing all of training instances and does not really work until the classification process runs [72]. As Oswin R. et al., [72] described, IB1 relatively accurate in classifying but require a large amount of storage space for storing all the training instances. IBL algorithms classify instance by comparing it with training data (predefined class) so that a similar instance will have a similar classification using similarity function. The similarity function (3) used in IB1 is:

$$\text{Similarity}(x, y) = -\sqrt{\sum_{i=1}^{n} f(xi, yi)} \qquad (3)$$

Where the instances are described by *n* attributes with i^{th} feature in instances X and Y. It defined that $f(x_i,y_i) = (x_i - y_i)^2$ for numeric-valued attributes and $f(x_i,y_i) = (x_i \neq y_i)$ for Boolean and symbolic-valued attributes.

IB1 is identical to the nearest neighbor algorithm except that it normalizes its attributes' range, processes instance incrementally, and has a simple policy for tolerating missing values. By default, k-NN algorithm with overlap matric is called IB1, when k =1. IB1 algorithm computes the similarity between a test item and each training item in terms of different optimized parameters. When using IB1 algorithm, there is a different metrics for influencing the definition of similarity to increase the evaluation result of a learning system. For instance, with the *modified value difference metric* (MVDM), each pair of values of a particular feature is assigned a value difference. In IB1, similarity of a new instance to stored instances is used to find the nearest neighbor instances are then used to predict the class of the new instance. The classes associated with the nearest neighbor instances are then used to predict the class of the new instances. All features are assigned the same relevance.

As [71] described, IBL algorithms do not construct extensional concept descriptions. Instead, concept descriptions are determined by how the IBL algorithm's selected similarity and classification functions use the current set of saved instances. These functions are two of the three components in the following framework that describes all IBL algorithms:

1. Similarity Function: This computes the similarity between a training instance *i* and the instances in the concept description. Similarities are numeric-valued.
2. 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*.
3. 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.

But, the concept description changes for IB1 requires how we can use the similarity and classification of IB1 to yield an extensional concept description (CD), although it is not

```
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 classification ← incorrect
         4. CD← CD U {x}
            End if
            End for
      End for
```

Algorithm 4.3 IB1 Algorithm

necessary to produce the extensional CD. To improve this all, there is an algorithm that IB1 works, see Algorithm 4.3 **Source** ([71]).

TRIBL: most algorithms have their own draw back, e.g., IB1 consumes large amount of storage space for storing instance, so, this is one of IB1 algorithms' drawback. To handle this, the evolution of another algorithm in addition to these algorithms is one of the solutions. Hence, TRIBL is hybrid generation of IGTREE and IB1. TRIBL allows to exploit the trade-off between optimization of search speed (as in IGTREE), and maximal generalization accuracy (as in IB1). To achieve this, a parameter ('-q') is set to determining the switch from IGTREE to IB1. During search, the normal IGTREE search algorithm is used, until the case-base nodes are reached, in which case regular IB1 nearest neighbor search is used on this sub-case-base. This TRIBL seems the same with the updated version TRIBL2. But the difference is, it has no parameter which determines a switching point from IGTREE to IB1 like TRIBL. As with IGTREE, it does not make sense to use TRIBL without feature weighting, so do not combining TRIBL with (-w 0): thus, we used -w2.

The 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

Input:

- A training set *T* of instances with their classes (start value: a full instance base),
- An information-gain-ordered list of features (tests) F_1, …F_n (start value: F_1, ... F_n).

Output: A subtree

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) F_i *in* F_i…F_n, and construct a new node *N* for feature F_i, and as default class *c* (the class occurring most frequently in *T*).
   - Partition *T* into subsets T_1…T_m according to the values v_1…v_m which occur for F_i in *T* (instances with the same value for this feature in the same subset).
   - **for each** j ε {*1,…,m*}:
     if not all instances in T_j map to class *c*, BUIL-IGTREE (T_j, F_{i+1}…F_n),
     Connect the root of this subtree to *N* and label the arc with v_j.

Algorithm 4.4 Building IGTREE Algorithms

on one of the features) and a class label (representing the default class at that node). These algorithms are discussed Algorithm 4.4 and 4.5 **Sources** [74].

Input:

1. The root node *N* of a sub tree (start value: top node of a complete IGTree),
2. An unlabeled instance *I* with information-gain-ordered feature values f_i…f_n (start value: f_1…f_n).

Output: A class label.

1. **If** *N* is a leaf node, output default class *c* associated with this node.
2. **Otherwise**, if test F_i of the current node does not originate an arc labeled with f_i, 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-IGTREE (*M*, f_{i+1}…f_n)

Algorithm 4.5 Algorithm for searching IGTREE

Thus, TRIBL is combination of these above three algorithms. Both of IB1 and TRIBL algorithms are applied for our study. The training phase that used memory-based learning model is used when morphological analysis phase is applied during classification.

4.5 Morphological Analysis

The morphological analysis phase implements all processes by incorporating with training phase. The classification is takes place after the memory-based learning is learned from extracted instances of surface word. When classifying the new instances, extrapolation is takes place if not direct match in a memory for a new instance and used to morpheme identification and stem is extracted by reconstructing and inserting these identified morphemes, finally lexical words are produced by giving morpheme with its function. The morphophonemic process that are present in morphological analysis phase is the same with that presented in training phase which is discussed previously in training phase.

4.5.1 Feature Extraction

Instances are a sequence of features separated by comma. These new instances are classifier by machine learner tool by using stored instances into memory previously during training data. When a new word is given to be analyzed by analyzer, first morphophonemic process checks is it rearrangeable or not and processes it, then it extracts feature (deconstruct) as instances to make similar representation with previously learned instances and stored in memory instances. During extraction the class label is filled with unknown character that has not used with in our class identification and we have used question mark (**?)** in last index of right most, what the feature extraction is different from described in training phase. When previously unseen word needs to be segmented, this word is deconstructed and represented as instances using the same information. Table 4.6 shows how system accept word '*difixxoommona*' and extract feature as sample.

Table 4.6 Instances to be Classified

12	11	10	9	8	7	6	5	4	3	2	1	F	1	2	3	4	5	6	7	8	9	10	11	12	Class
=,	=,	d,	i,	f,	i,	x,	x,	o,	o,	m,	m,	o,	n,	a,	=,	=,	=,	=,	=,	=,	=,	=,	=,	=,	?

4.5.2 Morpheme Identification

After instances are extracted from a rearranged surface word, the next task is identifying morphemes. The instances are made before it is given to a system to be classified by classifier. During classification, if there is an exact match in the memory with the classifying instances, it returns the classes of corresponding instance which is already stored in memory. But if there is no direct match with any of stored instances, classifying by analog task is taken place. Extrapolation is performed to assign the most likely neighbor class with its morphemes based

on their boundaries. This is done based on the similarity metric and k-nearest neighbor applied on the training data. For instance, to find the classes for instance that shown in Table 4.6, the instance is compared with each and every instance in the training set, recorded by the memory-based learner. The classifier finds the most closely similar instance from stored instances previously in a memory during a training. It is eleventh instances from a given word, as they share almost all features (**L9, L5, L4, L3, L2, L1, F, R2**). The memory-based learner then extrapolates the '<' class of this training instance and predicts it to be the class of the new instance as shown Table 4.7. Only instances that are misclassified by the classifier will be stored in the memory-based model for further analysis.

Table 4.7 Instances after classes predicted

12	11	10	9	8	7	6	5	4	3	2	1	F	1	2	3	4	5	6	7	8	9	10	11	12	Class	Predicted
=,	=,	d,	i,	f,	i,	x,	x,	o,	o,	m,	m,	o,	n,	a,	=,	=,	=,	=,	=,	=,	=,	=,	=,	=,	?,	<

4.5.3 Stem Identification

After classification is applied, it is simple to extract stem by reconstruction and insertion of boundary marker in the middle of segmented morphemes. Reconstruction is constructing extracted features of individual instances into a meaningful (their original word forms), and insertion of boundary marker is takes place in their segmentation point. The algorithm 4.3 (Adapted from [6]) shows that how the steps of stem extraction process is implemented.

```
Input: - Sidaamu Afoo word (rearranged)
Output: - Stem with morpheme functions/ Lexical words
      De-construct the given word
      For each instance given
            Read IB1/ TRIBL tree
            If similar /related instances exist in memory
            Assign the right most part to the new instances as class
                  If more than one related instance exists
                        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 4.6 Stem Extraction Algorithm from Sidaamu Afoo word

System searches similar class from previously stored instances in a memory for that new instances extracted from word. Similar metrics is used to find nearest neighbor when system

doesn't get similar match from the memory. The modified value difference metric (MVDM) is distance metric method that determines the values of feature similarity through looking at co-occurrence of the values with the target classes. For the distance between two values v_1, v_2 of feature, the difference of the conditional distribution of classes $C_{1\ldots n}$ for these values are computed [34] as shown Equation below.

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

Where, v1 and v2 are feature values, $C_{1\ldots n}$ (Ci) are class labels of the feature-values and $P(Ci/v_1)$ is an example of a conditional probability which, expressed in words, means the probability of class C, occurring, given feature value v.

For example, the reconstruction and morpheme insertion of the feature extracted instances for word '*difixxoommona*' is shown in Figure 4.2. Here, four non-null classes are predicted by system in classification step, then these predicted classes are inserted between the morphemic segments are concatenated. Finally, these morphemes are outputted with their function and the stem are separated by differentiating the class that contains the word class of Sidaamu Afoo. From the classified class, the stem is identified as /*fixx*-/, because it predicted as V (verb) and the others are classified as one prefix and two suffixes.

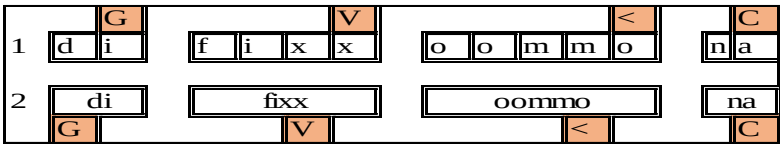

Figure 4.2 Reconstruction and insertion of morpheme for word 'Difixxoommona'

4.6 Summary

Our proposed architecture has two phases: training phase and morphological analysis phase. To implement both phase we used TiMBL tools with different algorithms by using default and optimized parameters setting. In training phase, the words are reordered its phonemic, annotated and stored as annotated word list in somewhere as our dataset, each annotated wordlist are extracted instances to be supported by learning tool in machine learning by sliding fixed length windowing method. But, in morphological analysis phase, words are extracted without class label (to be predictable, the right most places occupied by non-class symbol) after it reordered its phonemic and morpheme identified by classifying the instance's class. Finally, stem extraction task is processed by reconstruction and inserting morpheme boundary between identified morpheme with concatenated morphemic segment.

5. Chapter Five: Experiment

5.1 Introduction

In previous Chapter we have discussed the design of our system. As we have presented, our system has two phases: training and morphological analysis. In training phase, we have used memory-based learning to learn the given instances for further classification. While, morphological analysis phase extracts stem and retrieve morpheme with their functions. In this chapter we have presented the tool used, the corpus, algorithms, evaluations using algorithms and techniques, and generally experiment result that are obtained from the experiments.

5.2 Corpus Collection

We have collected data from different sources as we have discussed further. These data are divided into two after morphologically annotate. These are training and testing; the training data is used to train by a classifier for further classification of testing data. The classifier learns from the training data by storing instances in to a memory for further prediction of a class for a new, previously unseen instances. But to do this, properly prepared data with suitable format by separating comma or other format that helps the classifier to learn and/or test these data is needed. TiMBL stores instances during learning phase and tests the data based on previously stored instances and calculates the accuracy of classification [63]. We have named our training and testing datasets as Sama.train and Sama.test files respectively. These both files contain instances of morphologically annotated data that automatically extracted with comma separations. We divided data with 90% of training and 10% of testing as Sama.train and Sama.test respectively. When we divide these data, none of any data are existed in both files. The Corpus has prepared with 2231 total words of 1272 Verbs, 880 Nouns, and 79 Adjective stem and 89 Adjectives derived from verbs, nouns and adjective itself, and from these words totally 20,216 instances are generated with 67 class labels (see Annex B).

5.3 Implementation

As we have described, our system uses memory-based learning model for learning and classification, this model is fused into TiMBL tool. This tool is software package and it is developed and maintained by Introduction to Linguistic Knowledge (ILK) group at Tilburg

University and with the increasing use of MBL at the CNTS (now CLiPS) research group of the University of Antwerp. TiMBL was the result of combining ideas from a number of different MBL implementations, cleaning up the interface, and using a whole bag of tricks to make it more efficient [55]. TiMBL is a core component of various NLP software systems such as MBT (memory-based tagger generator), Frog (Dutch morpho-syntactic analyzer) [63]. It can be run either as a one-shot command line or as a server with different extensions. There are packages for Debian, Ubuntu and Arch Linux of TiMBL. We have installed it on Ubuntu using Virtual machine and trained and tested our corpus on it after some preparation. It has different versions through a time. We have used the latest version of TiMBL 6.4.14 by selecting the algorithm with default and optimized parameter setting. we used 18.04 Linux as working environment, NotePad++ editor for writing C++ code, GCC 7.5.0 Compiler for compiling a code, and Apache NetBeans 11 (incubating) for text tokenization, automatic feature extraction, data preprocessing, and morphophonemic rule process. Also, we have used java (Apache NetBeans 11) to write the system's prototype as a general with GUI. Furthermore, we have used IB1 and TRIBL algorithms with default and optimized parameters and Morphophonemic process algorithm.

After preparing all appropriate corpus we have run the tool by giving both Sama.train and Sama.test file with all algorithms either default or optimized parameter setting. The processes are divided into three phases as Figure 5 below. Phase 1 is Reading the data file examining upon it; first this phase checks the training file is present or not, then check the correct format that helps learner to learn is present or not. In this phase statistics made on the learning file during learning is processed. These statistics contains the lines of data, entropy, number of classes, Information gain, Gain ration. Phase 2: Building multi-index on datafile for later learning and testing.

Phase 3 is Learning from datafile of Sama.train, and tests data after learning. During learning it stores the instances in a memory according to the algorithms that have used (IB1 used large space). If the testing data file is not given it gives the statistics and results of yet and terminate. Else, if the test file contains the correct format, a new file is created (by appending the given test file, parameter setting, user given data, and finally .out) which is identical to the given test file by adding one column at the right most for storing predicted class label. After classifier classify the class label and writes the results into a new file, then gives the general report with

general accuracy by comparing previously given and predicted class label. The general report contains the algorithm has used, metric, the file name that written output files, weighting ration and so on as shown Figure 5.1 below.

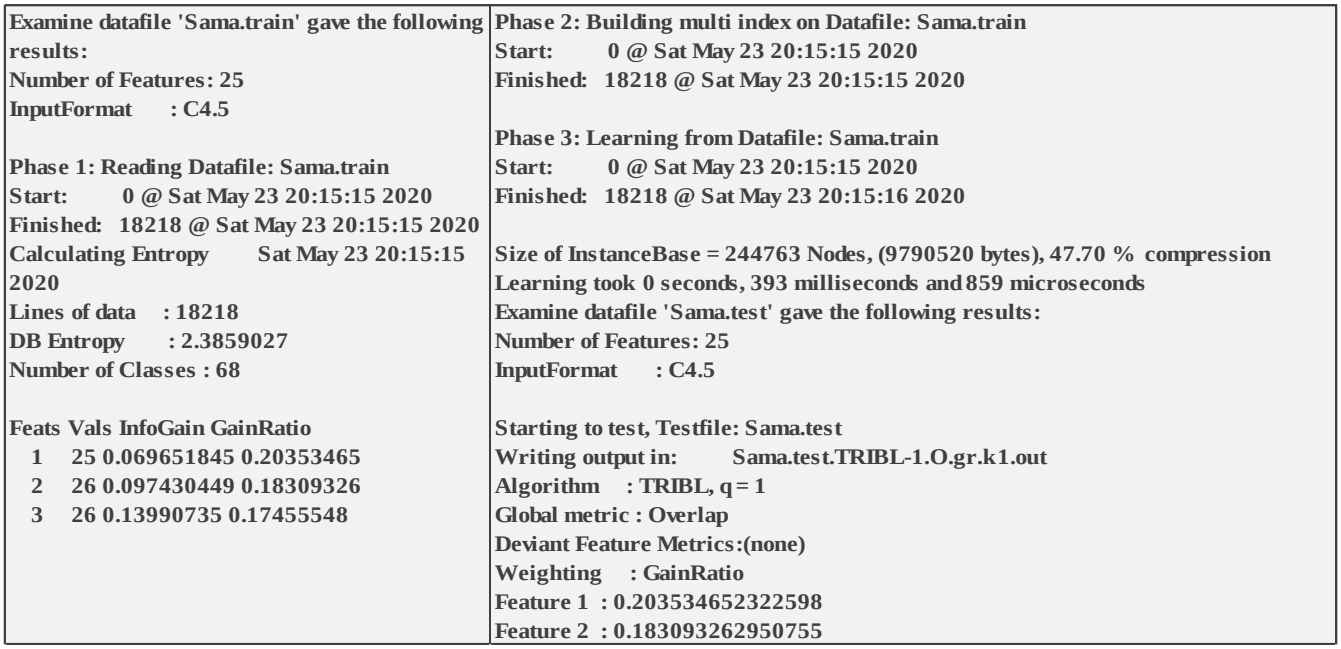

Figure 5.1 The Phases of TiMBL

5.4 Algorithm Optimization

During experimentation we have tested repeatedly our data by using default and optimized parameter and we have chosen the algorithm that achieve a better accuracy. Optimizing the parameter increases the achievement and the optimized parameter that we have used is discussed below.

The Distance Metrics:

This determines which distance metrics are used for each feature. There are so many types of distance metric such as overlap, MVDM, dot product, cosine distance, etc. [55]. The metrics that is Overlap and IG Overlap, are limited counting exact matches between feature values (i.e., that all values of a feature are seen as equally dissimilar). But an imaginary task that some phoneme is similar with each other than others. For this reason, Modified Value Difference Metric (MVDM) was defined. It determines the similarity of the values of a feature by looking at co-occurrence of values with target classes.

$$\sigma(v_1, v_2) = \sum_{i=1}^{n} |P(C_i|v_1) - P(C_i|v_2)| \quad 5.1$$

For the distance between two values *v1, v2* of a feature, it computes the difference of the conditional distribution of the class's C_i for these values. For computational efficiency, all pairwise $\sigma(v_1, v_2)$ values can be precomputed before the actual nearest neighbor search staff. MVDM differs considerably from Overlap based metric in its composition of the nearest neighbor set. Overlap causes an abundance of ties in nearest neighbor position. Increasing the value of *k* from 1 to 3 and 5 leads to deterioration of generalization accuracy with the Overlap function, it leads to improvements with MVDM. The absence of feature weighting leads to the lowest scores with the overlap function, and the highest score with MVDM and k=5. With the MVDM metric, however, the nearest neighbor set will either contain patterns which have the value with the lowest δ(v1,v2) in the mismatching position, or MVDM will select a totally different nearest neighbor which has less exactly matching features, but a smaller distance in the mismatching features. Thus, we have used MVDM metric as distance metric optimized parameter with IB1 algorithm.

Feature weighting metrics:

It chooses between feature-weighting possibilities. The weights are used in the metric of IB1 and in the ordering of the IGTREE. There are so many values of feature weighting metrics are there: no weighting, Gain ration, Information Gain, Chi-square, shared variance and standard deviation weighting. It is possible to use either chi-square (x^2) values as feature weights or can explicitly correct for the degrees of freedom by using the shared variance measure.

$$SV_i = \frac{X_i^2}{N \times (min(|C|,|vi|) - 1} \quad 5.2$$

Where |C| and $|V_i|$ are the number of classes and the number of values of features i, respectively, and N is the number of instances. The chi-square distribution generally becomes poor if the expected frequencies in the contingency table cells become small. A common recommendation is that the χ^2 test cannot be trusted when more than 20% of the expected frequencies are less than 5, or any are less than 1. Shared variance weights of *numeric* features are computed via a discretization preprocessing step (also used with computing IG and GR weights). Values are first discretized into a number (20 by default) of equally-spaced intervals between the minimum and maximum values of the feature. These groups are then used as

discrete values in computing shared variance weights [65]. Thus, we have used shared variance and Information Gain weights for our feature weighting metrics optimized parameter with IB1 and TRIBL respectively.

Nearest Neighbor:

The number of nearest neighbors used for extrapolation. The default value for kNN is 1, but it can be optimized by increasing it with odd numbers. Especially with the MVDM metric it is often useful to determine a good value larger than 1 for this parameter (usually an odd number, to avoid ties). Note that due to ties (instances with exactly the same similarity to the test instance) the number of instances used to extrapolate might in fact be much larger than this parameter. Optimizing of NN representing the number of nearest distances in which memory items are searched. Using a larger value of k can often lead to higher accuracy for discreate classes and larger training set. Hence, we have trained and tested our corpus with the value of k up to 5 to examine the effect of the k values when it classifies the new instances class label. If the value of k is 1 (default), our examined valued is:

```
 =,=,=,=,a,s,s,a,a,n,c,h,o,=,=,=,=,=,=,=,=,=,=,=,=,=,Z,Ø { Ø 1.00000 }
# k=1, 1 Neighbor(s) at distance:  0.42077802361611
# =,=,=,=,=,=,s,h,a,a,n,a,h,o,=,=,=,=,=,=,=,=,=,=,=,=,{ Ø 1.00000 }
```

Here the value of k=1, the classifier is looking the nearest neighbor and finds only instance *=,=,=,=,=,=,s,h,a,a,n,a,h,o,=,=,=,=,=,=,=,=,=,=,=,=,=,{ Ø 1.00000 }* as the class Ø. The reason that classifier classify the class as Ø is that the characters 'aancho' is a morpheme boundary marker with symbolized Z which is nominalizer. But the class predicted is Ø by classifier because classifier cannot get any other options that has nearest neighbor. If we increase the value of k is 3 the class label be correctly classified.

```
 =,=,=,=,a,s,s,a,a,n,c,h,o,=,=,=,=,=,=,=,=,=,=,=,=,=,Z,Z { N 1.00000, Ø 1.00000, A 1.00000, Z 2.00000 }
# k=1, 1 Neighbor(s) at distance:  0.42077802361611
# =,=,=,=,=,=,s,h,a,a,n,a,h,o,=,=,=,=,=,=,=,=,=,=,=,=,=,{ Ø 1.00000 }
# k=2, 2 Neighbor(s) at distance:  0.52806976081801
# =,=,=,=,=,=,=,=,m,a,n,c,h,o,=,=,=,=,=,=,=,=,=,=,=,=,=,{ N 1.00000 }
# =,=,=,=,=,=,h,a,r,a,n,c,h,o,=,=,=,=,=,=,=,=,=,=,=,=,=,{ A 1.00000 }
# k=3, 2 Neighbor(s) at distance:  0.56856353277790
# =,=,=,d,i,m,b,a,a,n,c,h,o,=,=,=,=,=,=,=,=,=,=,=,=,=,=,{ Z 1.00000 }
# =,=,=,f,i,i,y,a,a,n,c,h,o,=,=,=,=,=,=,=,=,=,=,=,=,=,=,{ Z 1.00000 }
```

Here there are four neighbors with different distances which is N (noun marker), Ø (SGM nominative dative/copula marker), A (adjective marker), Z (nominative marker). As we have

seen Z has 2.000000 distance than others and it is assigned as predicted class. When we increase the value of k, the classifier predicts the exact class label for the testing instances.

Class Voting Weight

The type of class voting weights that are used for extrapolation from the nearest neighbor set value can be of: normal majority voting, Inverse distance weighting, inverse linear weighting and exponential decay. 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. The voting process of the *k*-NN classifier as an attempt to make an optimal class decision, given an estimate of the conditional class probabilities in a local region of the data space. The radius of this region is determined by the distance of the k-furthest neighbor. Sometimes, if k is small, and the data is very sparse, or the class labels are noisy, the "local" estimate is very unreliable. The weight of the nearest and furthest neighbors and the slope between them depend on their absolute distance to the query. This assumes that the relationship between absolute distance and the relevance gradient is fixed over different datasets. This assumption is generally false; even within the same dataset, different feature weighting metrics can cause very different absolute distances. Generally, the distance weighting functions assign highly differing weights for close neighbors, and less differing weights for more distant neighbors. ID assigns very high votes (distance weights) to nearest neighbors at distances approaching 0.0 - in effect it assigns absolute preference to exact matches [55].

```
 =,=,=,=,=,a,s,s,i,t,u,r,e,=,=,=,=,=,=,=,=,=,=,=,=,=,R,R { λ 1.00000, R 2.00000 }
# k=1, 1 Neighbor(s) at distance:  0.24425436797083
# =,=,=,=,=,=,a,s,s,i,t,u,r,e,n,a,=,=,=,=,=,=,=,=,=,=,{ R 1.00000 }
# k=2, 1 Neighbor(s) at distance:  0.52806976081801
# =,=,=,=,=,s,i,r,b,i,t,u,r,e,=,=,=,=,=,=,=,=,=,=,=,=,{ R 1.00000 }
# k=3, 1 Neighbor(s) at distance:  0.58959293261798
# =,=,=,=,=,x,a,',m,i,t,u,s,e,=,=,=,=,=,=,=,=,=,=,=,=,{ λ 1.00000 }
```

Here the classifier gets three solutions and extrapolates the class of the test instance *(=,=,=,=,=,a,s,s,i,t,u,r,e,n,a,=,=,=,=,=,=,=,=,=,=,{ R 1.00000 }*) the k-NN classifier is looking for its nearest neighbors and finds two neighbors of two R (PL.ABS Genitive marker) and one with λ (3SG.F verbal possessive marker). The classifier classified R class because it is majority class in the NN-set. However, we have tried to use different voting metrics but in our

experiment ID with different global metrics in IB1 achieves a good accuracy by predicting most exact match than others. ID weighting implies that weights, inversely proportional to the distance from the query instance, are awarded to the neighboring instance [6].

5.5 Evaluation and Results

In this section we explained the evaluation techniques, testing results, and comparing our work with other work which are done using MBL approaches.

5.5.1 Evaluation Technique

Fast leave-one-out testing and internal cross-validation is the features of TiMBL. The most common technique that memory-based classifier has been trained and tested is 10-fold cross validation technique. The dataset is divided into 10 equal parts and each can be used as a testing file at once and the rest 9 are used as training data by concatenating each divided data. Cross-validation is computer intensive technique that use all available examples as training and testing examples. It mimics the use of training and test sets by repeatedly training the algorithm N times with a fraction 1/N of training examples left out for testing purposes. 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 n-fold cross-validation. In this form the data is first partitioned in to *n* equally (or nearly equally) sized segments or folds. Subsequently n 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 *n-1* folds are used for learning [6]. An n-fold cross-validation experiment is performed on the basis of n files (e.g. 1/n partitioning's of an original data file). The names of these n files need to be in a text file (one name per line) which is given as argument of -f. In each fold f = 1...n, file number f is taken as test set, and the remaining n−1 files are concatenated to form the training set. In 10-fold cross-validation experiments, the system is trained and tested on approximately 90% and 10 % respectively. This is repeated 10 times through all these 10 folds, and within each repetition, one divided data is used as a test data but the other is concatenating to be used as a training data. All instances in the corpus are entirely included in either the test corpus or the training corpus. The Figure 5.2 below shows our 10-fold cross

validation evaluation. The whiter sections of data are used for training and the darker sections of the data are used for testing data (Validation).

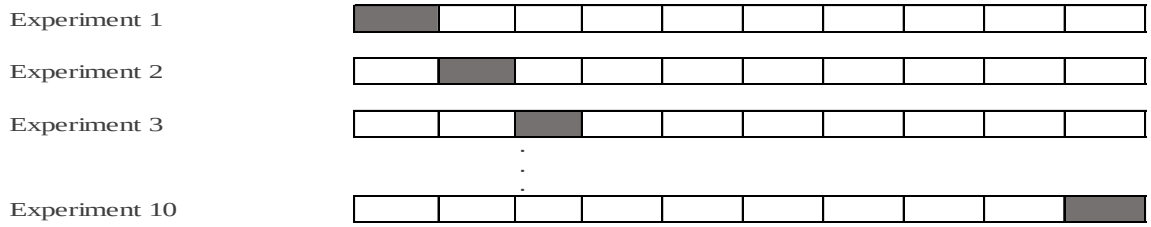

Figure 5.2 Procedure of 10-fold Cross-validation

We also used *leave-one-out* cross-validation for IB1 algorithm, which uses all available data except one (n-1) example as training material. It tests the classifier on the one held-out example by repeating it for all examples. In Leave-one-out no test file is read, but testing is done on each pattern of the training file, by treating each pattern of the training file in turn as a test case (and the whole remainder of the file as training cases). Only applicable in conjunction with IB1 (-a0).

Table 5.1 Default and Optimized Parameter with their values and Descriptions

Parameter	Default Value	Optimized Value	Description
Nearest Neighbor	-k1	-k5	K=1 number of nearest neighbors used for extrapolation
Feature-weighting	-m0	-mM	The distance between two patterns is simply the sum of the differences between the features (default--> Overlap, M--> MDVM)
Distance metrics	-w0	-w4	By default no weight. All features have the same importance. But -w4 has shared variance metrics
Class Voting	-dZ	-dID	Normal majority voting-all neighbors have equal weight. But it can be optimized ID inverse distance metrics.

5.5.2 Testing Results

Testing was done using the parameters that have specified above Table 5.1 by tuning both parameters setting. Based on the default values of parameters each algorithm (IB1 and TRIBL), we achieved the following results shown in Table 5.2 and 5.3 for 10 -fold cross validation and leave-one-out training and testing experiments. The performance for each experiment is discussed in the Table 5.2 and 5.3 for both experimental techniques. The leave-one-out technique is only applicable in IB1.

Table 5.2 10-fold CV Experiment on IB1 with Default parameter setting

No of experiment	1	2	3	4	5	6	7	8	9	10
No of instances	18192	18150	18206	18177	18165	18218	18186	181277	18234	18139
Seconds taken	0.406	0.395	0.514	0.396	0.394	0.4	0.406	0.407	0.392	0.396
Size of instances base (byte)	9101520	9028920	9127760	9054760	9024720	9067400	9082120	9174520	9143280	8966480
Compression (%)	51.47	51.68	51.25	51.54	51.63	51.56	51.42	51.16	51.26	51.96
Accuracy	93.4	93.4	93.9	93.04	94	94.5	94.2	92.6	93.5	93.9

In Table 5.3 q=1, 1 is the TRIBL offset, the index number of the feature (counting from 1) after which TRIBL should switch from IGTREE to IB1. It is only applicable in conjunction with TRIBL (-a2).

Table 5.3 10-Fold CV Experiment on TRIBL with Default parameter setting, q=1

No of experiment	1	2	3	4	5	6	7	8	9	10
No of instances	18192	18150	18206	18177	18165	18218	18186	181277	18234	18139
Seconds taken	0.552	0.494	0.495	0.496	0.502	0.508	0.523	0.493	0.494	0.491
Size of instances base (byte)	9961200	9900600	9998760	9921160	9912880	9790520	9889720	9872080	9957200	9847240
Compression (%)	46.88	47.01	46.6	46.9	46.87	47.7	47.1	47.45	46.92	47.24
Accuracy	95.85	95.4	96.3	95.6	96.8	96.7	95.9	95.5	96.1	96.1

The average performance is the average of total sum of each fold experiment for 10-fold experiment and for leave-one-out technique it is simply used one data corpus for self-test of it. The average result of the two algorithms with default parameter and optimized parameter setting for both techniques are described in Table 5.4 and Table 5.7 respectively.

Table 5.4 Average Performance of 10-fold CV and LOO experiment with Default parameter setting

Evaluation Method	Algorithm	Compression (%)	Time Taken (seconds)	Size of Instance base (byte)	Accuracy (%)
Leave one out	IB1	51.56	0.398	9067400	93.4
10 fold CV		51.493	0.4106	9077148	93.65
	TRIBL	47.067	0.5048	9905136	96.03

The average accuracy for default parameter is better for TRIBL algorithm than IB1 in both techniques. But it takes much more storage and takes more time than IB1 (in both techniques). IB1 takes less time to process and used less storage of memory than TRIBL.

To obtain a better performance, it is obvious to use optimized parameter setting than using default parameter. Repeatedly we have tested our corpus by using different optimized parameter setting. Then finally we have chosen the parameter that achieves the better performance. Thus, we have chosen for IB1; nearest neighbor: k=5, feature weighting metrics: shared-variance weighting metrics (-w4), distance metric: modified value difference metric (MVDM) (-mM), class voting metric: inverse distance (-dID). By using this optimized parameter, we have tested the corpus through both evaluation techniques in IB1 algorithm but in TRIBL we have tested only by using 10-fold cross-validation evaluation technique because LOO is only applicable in IB1. Hence, we have used only Information Gain (IG) feature weighting metric in TRIBL (-w2); these previously discussed optimized parameter have used only in IB1 except Information gain. The achieved result is discussed in Table 5.5 and Table 5.6 for default and optimized parameter respectively.

Table 5.5 10-fold CV experiment on IB1 with optimized parameter setting

No of experiment	1	2	3	4	5	6	7	8	9	10
No of instances	18192	18150	18206	18177	18165	18218	18186	18127	18234	18139
Seconds taken	0.489	0.488	0.493	0.501	0.497	0.494	0.507	0.499	0.499	0.495
Size of instances base (byte)	9101520	9028920	9127760	9054760	9024720	9067400	9082120	9174520	9143280	8966480
Compression	51.47	51.68	51.25	51.54	51.63	51.56	51.42	51.16	51.26	51.96
Accuracy	96.94	96.7	96.9	96.6	97.4	97.3	96.8	96.3	96.9	96.5

Table 5.6 10-fold experiment on TRIBL with optimized parameter setting and q=1

No of experiment	1	2	3	4	5	6	7	8	9	10
No of instances	18192	18150	18206	18177	18165	18218	18186	18127	18234	18139
Seconds taken	0.537	0.502	0.502	0.512	0.512	0.513	0.512	0.504	0.514	0.504
Size of instances base (byte)	12432080	12402440	12418320	12380120	12383000	12429640	12394520	12438960	12436560	12356120
Compression (%)	33.71	33.62	33.67	33.74	33.63	33.6	33.71	33.78	33.7	33.79
Accuracy	96.6	96.9	97.3	96.7	98	97.6	96.9	96.3	97.1	96.9

Table 5.7 Average performance of 10-fold CV and LOO with optimized parameter setting

Evaluation Method	Algorithm	Compression (%)	Time Taken (seconds)	Size of Instance base (byte)	Accuracy (%)
Leave one out	IB1	51.56	0.525	9067400	96.6
10 fold CV		51.493	0.4962	9077148	96.83
	TRIBL	33.7	0.5112	12407176	97.03

When it compares the default and optimized parameter setting's result, optimized parameter setting value is better than the default parameter setting's value. For instance, when we see the performance of IB1 with default and optimized parameter setting is 93. 65% And 96.83% respectively by 10-fold cross-validation techniques and it achieved 93.4% and 96.6% respectively by leave-one-out technique. Similarly, the general performance of TRIBL algorithm with default and optimized parameter setting is 96.03% and 97.03% respectively. However, TRIBL algorithm achieved better general performance but, in both cases, TRIBL occupies large amount of storage and consumes more time to process than IB1. From both parameter setting (default and optimized), we can discuss about the obtained result by using leave-one-out technique from IB1 classifier. The IB1 algorithm with optimized parameter setting is achieved better result of 96.6% than the default parameter setting result of 93.4% by LOO. The storage and compression are the same but optimized parameter setting takes more time than the default one. Aside with the general performance that are correctly classified test instances, we have used some more evaluation metrics that are 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 [6]. The precision, recall and F-score for unknown- words with default and optimized parameters described in Table 5.8.

Table 5.8 10-fold CV and LOO result of average precision, recall, and F-score with default and optimized parameter setting

Experiments	Algorithms	With default parameter				With optimized parameters			
		Precision	Recall	F-score	Size of instances (byte)	Precision	Recall	F-score	Size of instances(byte)
LOO	IB1	65.4	82	72.8	9067400	88.9	85.5	87.2	9067400
CVV		69.7	83.7	76.1	9077148	91.2	90.9	91.0	907714
	TRIBL	81.7	84.2	82.9	9905136	89.5	90.1	89.8	12407176

The result achieved in both parameter setting described in Table 5.8; the optimized parameter setting achieved better result than default parameter setting. The result is obtained from the corpus of 2231 words that doesn't included all inflections of verbs, nouns, and adjectives. Our

dataset contains more inflections even it doesn't contain complete derivations and achieved good result. These results are acceptable because some of results are almost achieved better and it gives hope to implement the system with the large and complex examples.

To increase the general performance of the system in memory-based learning, it is must to increase the amount of corpus. However, the size of the training data matters the learning performance of the algorithm. We have systematically increased the size of training data up to total available data of 2007 by performing a series of experiments. When drawing a learning curve, in most cases, the learning can be measured by fixing the number of test data against which the performance is measured. The learning curve of the system is drawn as Figure 5.3. We have divided our training and testing data in to 9 equal or nearly equal parts and tested our system. These results achieved by increasing the number of words is discussed in Table 5.9 below. The learning curve drawn on the morphological analyzing test set of Sidaamu Afoo, with increasing the number of instances in the training set. Number of training words are labeled in X-axis and the performance is labeled in Y-axis. We have divided our training data in to 9 equal intervals of as 223 words and tested it. But the data used for training is 90% (2007) of totally collected data and 10% (223) was used for testing. The curve shows that when the amount of data increase, it grows up by increasing the performance. Thus, we concluded from learning curve, increasing the amount of data gives the better generalization.

Table 5.9 Generalization Accuracy with increasing the number of words

No of instances	1883	3799	5873	8214	10435	12383	14376	16373	18341
Training words	224	447	670	893	1116	1339	1562	1785	2007
Performance (%)	90.7	91.89	92.5	93.1093	93.1097	93.0469	93.4544	93.6175	93.7

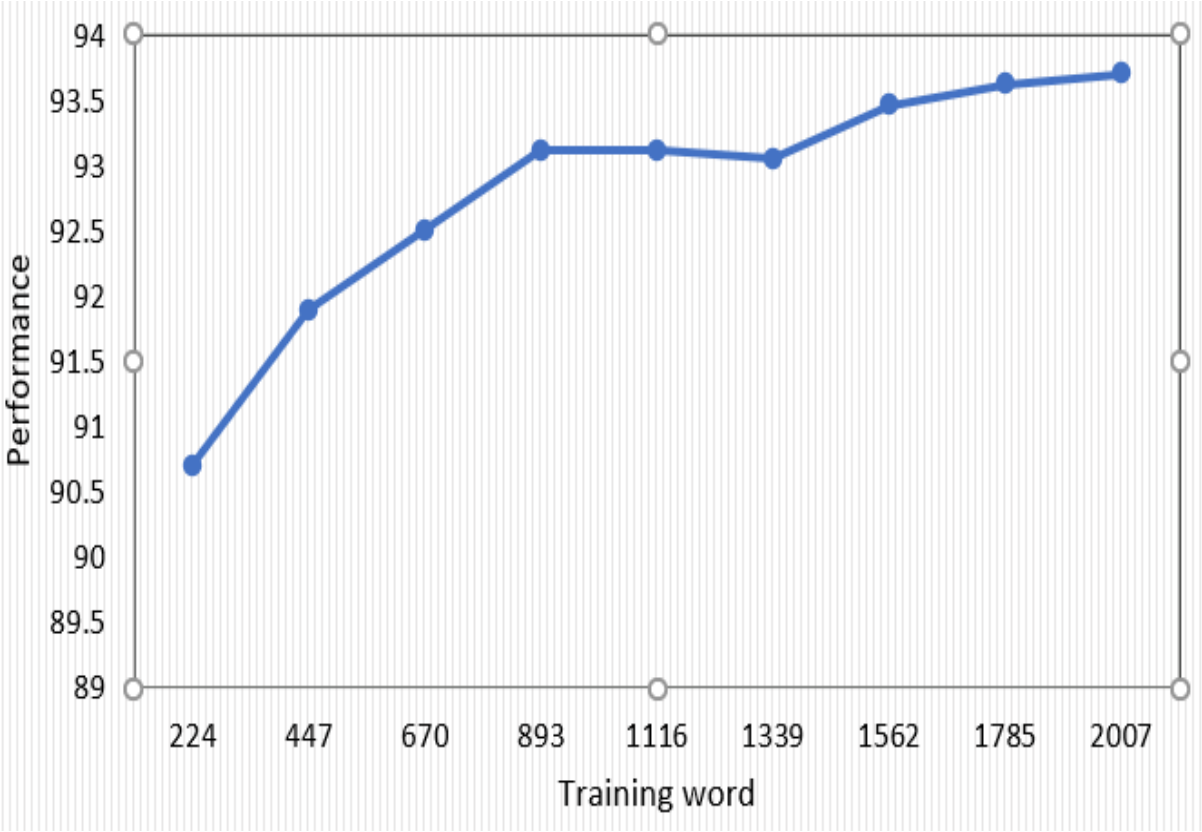

Figure 5.3 Learning Curve of the System

5.5.3 Comparison with Related works

As nothing is tried on a morphological analyzer of Sidaamu Afoo, we have tried to compare our work with the others work on different languages. These works are done by using memory-based learning through 10-fold cross-validation and leave-one-out experimental technique. Our study is the first work on morphological analysis as well as other NLP application for Sidaamu Afoo and we have tried to model memory-based machine learning for all category of verbs, nouns, and adjectives. We used 10-fold cross-validation and leave-one-out techniques and the experiment's general performance of IB1 and TRIBL is 96.83% and 97.03% respectively by using 10-fold CV and general performance of IB1 algorithm is 96.6% by using LOO.

Table 5.10 Comparison of Others work with Morphological analysis of Sidaamu Afoo

Morphological analysis of	Evaluation Techniques	Size of Corpus used	Word Class covered	Algorithm	Accuracy(%)
Dutch Language	10-fold	247415	Verbs, Nouns, Adjectives	IB1	91.2
Arabic Language	10-fold	16626	Verbs, Nouns, Adjectives	IB1	16.7
Amharic	10-fold	1022	Verbs, Nouns, Adjectives	IB1	93.59
				IGTREE	82.26
	LOO			IB1	93.3
Ge"ez Language	10-fold	1105	Verbs	IB2	93.24
				TRIBL2	92.31
Sidaamu Afoo Language (our work)	10-fold	2231	Verbs, Nouns, Adjectives	IB1	96.83
				TRIBL	97.03
	LOO			IB1	96.6

As shown in Table 5.9, most techniques have used is 10-fold CV and some are used leave-one-out technique. As others work, our work addressed similar word classes and uses similar evaluation metric and achieved better results than others. We have trained and tested on small number of corpus and we haven't considered all derivations and inflections of word classes due to the non-existence of previously prepared inflected words and limitation of time to prepare all of them. Thus, the result that our study achieved is acceptable to implement the system in large scale of examples and word with more complexity.

5.6 Summary

The study was analyzed on total corpus of 2231 words. These words are divided into 90% for training and 10% for testing. The experiment was done by using 10-fold cross-validation and leave-one-out evaluation techniques. The experiment is evaluated in terms of general accuracy, the storage amount of memory, the time taken to process, and precision, recall and f-score for IB1 and TRIBL algorithms by tuning default and optimized parameter setting. The general accuracy of the model with default setting is 93.65% and 96.03% for IB1 and TRIBL algorithms respectively by using 10-fold cross-validation and 93.4% for IB1 by using leave-one-out technique. By adjusting optimized parameter setting (-k5, -mM, -w4, -dID for IB1 and only -w2 for TRIBL) we obtain better result than using the default parameter setting. Its achieved result is 96.83% and 97.03 for IB1 and TRIBL respectively by using 10-fold CV and 96.6 for IB1 by using LOO. In contrast, TRIBL achieved better performance than IB1 even it takes much more time and storage, but the difference is that much not varied. Accepting lower achieved accuracy to minimize the storage of memory and the time taken is not recommended. Because the aim of morphological analysis is classifying accurately new example's class label. There for TRIBL classifier classifies better than IB1.

The experiment also evaluated by using advanced statistics metric techniques: precision, recall, and f-score by using 10-fold CV and LOO. The IB1 classifiers with default parameter setting achieved the precision, recall, and f-score of 96.7%,73.1%, and 83.7% respectively by using 10-fold CV technique and 65.4%, 64.2%, and 82% respectively by using LOO. Likewise, TRIBL with default parameter setting evaluated and achieved the precision, recall, and f-score of 81.7%,84.2, and 86.3 respectively. Also, when optimized parameter setting is tuned, IB1 achieved the result of precision, recall, and f-score is 88.9%, 85.5%, 92.7% respectively by using LOO, and 91.2%, 90.9%, and 96.2% respectively by using 10-fold cross-validation. Similarly, TRIBL algorithm with optimized parameter setting is achieved the precision, recall, and f-score of 86.5%, 90.1%, and 86.7% respectively. Finally, we have drew the graph to show the increment of performance is raised with the increment of training data. When the amount of data is increased, the system's general performance is also increased. Thus, the results achieved is acceptable for a morphological analyzer for Sidaamu Afoo and these algorithms are suitable for developing the morphological analysis.

6. Chapter Six: Conclusion and Future work

6.1 Conclusion

This work studies the morphological analysis of Sidaamu Afoo. Sidaamu Afoo is an official working language in Sidama National Regional State. It is under-resource language; there is no lexical linguistic studies for this language and there are no NLP applications that are tried to solve problems for this language. The absence of these all may consequent so different problems; the problem may occur during spell checker, information retrieval, information extraction, machine translations and so on. To solve these problems developing morphological analyzer is needed. Morphological analysis is the base of other NLP application. In the study supervised memory-based learning technique is applied. TiMBL tool is used as a learning tool by tuning IB1and TIMBL algorithm with default and optimized parameter setting, and achieved good results.

We have prepared the corpus starting from scratch, because of there are nothing researches are tried on the area of NLP for Sidaamu Afoo language and collected totally 2231 words of verbs, nouns, and adjectives. These collected surface words are processed morphophonemic rules of the Sidaamu Afoo to prepare morphologically annotated words. These corpuses are annotated morphologically to extract features automatically by using fixed length windowing method. Totally 20216 instances are automatically extracted from these datasets with 67 class labels. We have divided our dataset into 90% for training and 10% for testing. The experiment was evaluated by adjusting the default and optimized parameter setting of IB1 and TRIBL algorithms and optimized one is achieved better result. Both classifiers achieved different results through default parameter of 93.65% and 96.03% for IB1 and TRIBL respectively by using 10-fold Cross-Validation and 93.5% for IB1 by using Leave-one-out. By adjusting optimized parameter setting it achieved the result of 96.83% and 97.03 for IB1 and TRIBL respectively by using 10-fold CV and the result of 96.6% for IB1 by using LOO. TRIBL achieved better result rather than IB1 even to takes more storage and takes more time to process. The system evaluated through different evaluation metrics, and in additionally, it concluded that increasing the size of training data raises the performance of system by drawing the graph. Finally, we have compared our work with the others work that used similar methodology with us, and our work achieved better results than these works.

6.2 Future Work

The task that we have not handled in our work is recommended as a future work. We have used small size of data for training and testing our model due the shortage of time and the absence of previously stored data. So, it is better to use large amount of data to increase the performance of the system. When we process the morphophonemic rule of the language, it is so complex with some exceptions and studying these exceptions to develop full-fledged morphological analyzer for Sidaamu Afoo is optionless. But studying these additional exceptions with linguists is time consuming and takes extra study. We have tried to use memory-based learning approach, and achieved better results than others. To achieve more better results, it is recommended that integrate our technique (MBL) with the other methodology of machine learning like neural network, and so on. Finally, by using our system's output it is possible to develop another NLP application like spell checker and etc.

References

[1] "http://towardsdatascience.com/Natural Language Processing inApache Spark using NLTK (part 1/2)," *accessed on Oct 20, 2018.*

[2] D. Jurafsky and J. H., "Speech and Language Processing: An introduction to speech recognition," *natural language processing, and computational linguistics, Prentice-Hall, Inc,* 2000.

[3] A. A. and D. M. , "Analysis of Rule Based Approach for Afan Oromo Automatic Morphological Synthesizer," *Science, Technology and Arts Research Journal,* pp. 94-97, 2013.

[4] D. B. Weldegiorgis, "Design and implementation of automatic morphological analyzer for Ge'ez verbs," *Unpublished, Computer Science, Addis Ababa University,* 2010.

[5] G. M. Wegari, M. M. and S. T. , "Probabilistic and Grouping Methods for Morphological Root Identification for Afaan Oromo," *6th International Conference - Cloud System and Big Data Engineering (Confluence),* 2016.

[6] Y. A. and Y. A. , "Morphological Analysis of Ge'ez Verbs Using Memory Based Learning," *Unpublished Master's Thesis, Computer Science, Addis Ababa University,* 2014.

[7] "http://joshuaproject.net/people_groups/14630/ET," *Accessed on,* July 10, 2018.

[8] K. K. "A grammar of Sidama (Sidamo), a Cushitic language of Ethiopia," *University at Buffalo, the State University of New York,* 2007.

[9] J. B. Roly, L. K. Judith , A. Frank and A. Mark, "Computer Mlethods for Morphological Analysis," *I.B.M Thomas J. Watson Research Center, YorkTown, NewYork 10598, SUNY/ Stony Brook, Stony Brook, New York,11794.*

[10] M. G. Wondwossen Mulugeta, "Learning Morphological Rules for Amharic Verbs Using Inductive Logic Programming," *Workshop on Language Technology for Normalisation of Less-Resourced Languages,* 2012.

[11] M. A. and Y. A. , "Amharic Morphological Analysis Using Memory Based Learning,," *Springer International Publishing Switzerland* , pp. 1-13, 2014.

[12] S. A. and D. G. , "Finite State Morphology of Amharic," 2014.

[13] M. G. "HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya," *Indiana University,* 2009.

[14] T. B. BATI, "Automatic Morphological Analyzer for Amharic an experiment employing unsupervised learning and autosegmental analysis approaches," *Unpublished Masters Thesis, Department of Information Science, Addis Ababa University,* 2012.

[15] K. L. WUDINEH, "DESIGN AND DEVELOPMENT OF AUTOMATIC MORPHOLOGICAL SYNTHESIZER FOR AMHARIC PERFECTIVE VERB FORMS," *Unpublished Masters Thesis, Department of Information Science, Addis Ababa University,* 2012.

[16] A. K. M, "Morphology Based Prototype Statistical Machine Translation System for English to Tamil Language," *Unpublished PhD Thesis, CENTER FOR EXCELLENCE IN COMPUTATIONAL ENGINEERING and Networking, AMRITA SCHOOL OF ENGINEERING, India,* 2013.

[17] Y. Assabie, "Morphological analysis," Unpublished Lecturer Note, Department of Computer Science, Addis Ababa, 2017/18.

[18] L. J. Whaley, "Chapter 7: Morphemes," in *Introduction to typology: The unity and diversity of Language*, SAGE Publications, Inc, 1997.

[19] P. Williams, " On the notion of “Lexical Related’ and “Head of a word”," *Linguistic Inquiry ,* vol. 12(2), p. 245 – 274, 1981.

[20] M. K. "Morphological Analyzer and Generator for Russian and Ukrainian Languages," *international conference on analysis of images, social networks, and texts.*, pp. 320-332, 2015.

[21] A. &. D. W. van den Bosch, "Memory-based morphological analysis," *In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, ACL '99, University of Maryland, USA*, no. College Park, MD: ACL., pp. 285-292, june 20-26, 1999.

[22] N. S. "Morphological analysis of historical languages," *Institute of classical studies University of London,* 2016.

[23] S. R. Spiegler, "Machine Learning for The Analysis of Morphologically Complex Languages," *Merchant Venturers School of Engineering, University of Bristol,* 2011.

[24] G. R. G. R. a. A. B. Stave Pulman, "Computational Morphology of English," *Technical reports, University of Cambridge Computer Laboratory are freely available via the Internet:http://www.cl.cam.ac.uk/TechReports/http://www.cl.cam.ac.uk/TechReports/,* 1989.

[25] C. P. a. P. Sheridan, "Multilingual information access, In ESSIR ’00:," *Proceedings of the Third European Summer-School on Lectures on Information Retrieval Revised Lectures, Springer-Verlag, ,* 2001 .

[26] K. Koskenniemi, "Two-Level Morphology: A general Computational Model for Word-form recognition and production," *University of Helsinki, Department of General Linguistics, Helsinki,* vol. Publication 11, 1983.

[27] D. &. M. S. Kazakov, "Unsupervised Learning for Word Segmentation Rules with Genetic Algorithms and Inductive Logic Programming.," 2000.

[28] S. Kotsiantis, "Supervised Machine Learning: A Review of Classification," *Department of Computer Science and Technology University of Peloponnese, Tripolis GR,* no. 22100, 2007.

[29] A. B. a. T. Dietterich, "Reinforcement Learning and its Relationship to Supervised Learning. University of Massachusetts Amherst, MA and Oregon StateUniversity," 2003.

[30] S. Marinal, Machine Learning in Document Analysis and Recognition, Springer Science & Business Media , 2008 M01 10-433.

[31] B. S. Sara and A. B. , "Uncovering discourse Relations to Insert Connectives between the Sentences of an Automatic Summary," *Springer International Publishing Switzerland ,* pp. 249-261, 2014.

[32] A. k. a. L. E. Samer Al Moubayed, "Automatic Prominence Classification in Swedish Centre for Speech Technology, Royal Institute of Technology (KTH)," *Lindstedtsvägen 24,* no. SE-10044, Stockholm, 2011.

[33] E. M. S. a. M. H. A. A. F. Alajmi, "Hidden markov model based Arabic morphological analyzer," *International Journal of Computer Engineering Research,* vol. 22, pp. 28-33, March 2011.

[34] W. D. and A. v. d. B. , "Memory-Based Learning," *Computational Linguistics and Natural Language Processing Handbook,* 2009.

[35] W. D. S. B. and J. V. , "Memory-Based Shallow Parsing," *arXiv:cs/9906005v1,* 1999.

[36] W. D. and A. v. d. B. , "Memory-Based Language Processing," *University of Antwerp and Tilburg University,* 2005.

[37] E. M. a. A. S. Antal van den Bosch, "Memory-based morphological analysis and part-of-speech tagging of Arabic," *Dept. of Language and Information Science, Faculty of Arts, Tilburg University and Ecole Nationale de l'Industrie Min´erale, Rabat, Morocco,* 2014.

[38] W. D. and A. v. d. B. , "Memory based morphological analysisfor Dutch language," *computational linguistics, Tilburg University,* 1999.

[39] W. D. and J. Z. , "MBT: A Memory-Based Part of Speech Tagger-Generator," *Computational Linguistics and AI, Tilburg University, Center of Dutch Language and Speech, University of Antwerp,* 2012.

[40] M. S. and E. A. , "Fine-Grain Morphological Analyzer and Part-of-Speech Tagger for Arabic Text," *School of computing, University of Leeds, LS2 9JT, UK,* 2008.

[41] S. Buchholz, "Memory-based grammatical relation finding," *Ph. D. thesis, Tilburg,* 2002.

[42] W. D. and A. v. d. B. , "Memory Based language processing," *CLiPS Research Group, University of Anterp, ILK/ Tilburg centre for Creative Computing, Tilburg University, the Netherlands,* 2005.

[43] S. C. and A. v. d. B. , "A memory-based shallow parser for spoken Dutch.," *Selected papers from the Thirteenth Computational Linguistics in the Netherlands Meeting, Antwerp, Belgium,* pp. 31-45, 2004.

[44] A. v. d. B. and W. D. , "Memory based morphological analysis for Dutch language," *computational linguistics, Tilburg University.*, 1999.

[45] B. D. J. D. W. D. and P. W. , "Memorybased phoneme-to-grapheme conversion," *Computational Linguistics in the Netherlands,* 2002.

[46] E. M. B. B. W. D. V. H. M. R. and A. v. d. B. , "Combining information sources for memory-based pitch accent placement.," *In Proceedings of the International Conference on Spoken Language Processing,* pp. 1273 - 1276, 2002.

[47] A. v. d. B. W. D. K. S. J. H. and S. C. , "Memory-based semantic role labeling: Optimizing features, algorithm, and output," *Proceeding of Eighth Conference on Computational Natural Language Learning (CoNLL-2004), Boston, MA,* 2004.

[48] J. V. A. v. d. B. S. B. W. D. and J. Z. , "Memory-based word sense disambiguation," *Computers and Humanities,* 2000.

[49] R. M. "Instance Based Learning with Automatic Feature Selection Applied to Word Sense Disambiguation," *Department of Computer Science, University of North Texas*, 2012.

[50] D. M. and W. D. , "Memory-based named entity recognition using unannotated data," *Proceeding of CoNLL-2003, Seventh Conference on Natural Language Learning, Edmonton, Canada,* pp. 208-211, 2003.

[51] J. Z. and W. D. , "Memory-based learning: Using similarity for smoothing," *In Proceeding of the 35th annual meeting of ACL, Madrid,* 1997.

[52] A. v. d. B. and V. H. , "A Modular approach to learning dutch coreference .In C.Johansson(Ed.)," *Proceeding of the workshop on anaphora Resolution Mjfjell, Norway* , p. 97, 2007.

[53] W. D. and J. Z. , "Text Mining, Theoretical Aspects and Applications, Chapter Feature-rich Memory-based classification for shallow NLP and information extraction," *Springer Physical-Verlag,* pp. 33-54, 2003.

[54] J. S. V. Jyh-Han Lin, A Theory for Memory-Based Learning, Pittsburgh, PA, USA: Fifth annual ACM conference on Computational learning theory , 1992.

[55] W. D. J. Z. K. v. d. S. and A. v. d. B. , "TiMBL: Tilburg Memory Based Learner, version 6.4, Reference Guide.," *ILK Technical Report XX-XX Available from http://ilk.uvt.nl/downloads/pub/papers/ilkxxxx.pdf,* 2012.

[56] A. C. "Memory-Based Learning of Morphology with stochastic transducers," *Proceeding of the 40th annual meeting of the association for computational linguistics (ACL), Philadelphia,* pp. 513-520, 2002.

[57] W. D. and A. v. d. B. , "Memory based language processing," *CLiPS Research Group, University of Antwerp, ILK / Tilburg centre for Creative Computing, Tilburg University, the Netherlands.*, 2009.

[58] S. G. a. G. D. W. Daelemans, "The acquisition of stress," *a dataoriented approach. Computational Linguistics*, vol. 20(3), pp. 421-451, 1994.

[59] W. D. a. A. V. d. Bosch, "Language-independent data-oriented grapheme-to-phoneme conversion," *In J. P. H. Van Santen, R. W. Sproat, J. P. Olive, and J. Hirschberg, editors, Progress in SpeechProcessing*, no. Springer-Verlag, Berlin, pp. 77-89, 1996.

[60] A. V. d. Bosch., "Learning to pronounce written words," *A study in inductivelanguage learning. Ph.D. thesis, Universiteit Maastricht,* 1997.

[61] J. Z. P. B. a. S. G. W. Daelemans, "mbt: A memory-based part of speech," *In E. Ejerhed and I. Dagan, editors, Proc. of Fourth Workshop on Very Large Corpora* , no. ACL SIGDAT, pp. 14-27, 1996b.

[62] A. N. "A machine learning Approach to Anaphora Resolution including Named Entity Recognition, PP Attachment Disambiguation, and Animacy Detection," 2009.

[63] K. v. d. S. A. v. d. B. W. D. P. B. M. v. G. T. W. and J. Z. , "https://languagemachines.github.io/timbl/," *Language Machines, Center of Language and Speech Technology*.

[64] S. C. and M. v. G. , "Python timbl, Tilburg University," *http://github.com/proycon/python-timbl/,* 2006.

[65] W. D. K. v. d. s. J. Z. and A. v. d. B. , "TiMBL: Tilburg Memory Based Learner, version 6.1, Reference Guide," *ILK Technical Report 07-xx, Available from http://ilk.uvt.nl/downloads/pub/papers/ilk0703.pdf,* 2007.

[66] D. R. W. and T. R. M. , "Reduction Techniques for Instance-Based Learning Algorithms," *Machine Learning Kluwer Acadamic Publishers, Netherlands,* vol. 38, pp. 257-286, 2000.

[67] D. C. S. Z. Z. D. Y. Z. and M. Z. , "kNN Algorithm with Data-Driven k Value," *Springer International Publishing Switzerland*, pp. 499-512, 2014.

[68] D. H. M. M. and P. S. , "Principles of Data Mining," *The MIT Press,* 2001.

[69] S. Z. X. L. M. Z. X. Z. and D. C. , "Learning k for kNN Classification," *ACM Transactions on Intelligent Systems and Technology* , no. Article 43, p. 19, 2017.

[70] W. D. J. Z. K. v. d. S. and A. v. d. B. , "TiMBL: Tilburg Memory Based Learner, version 4.1, Reference Guide," *ILK Technical Report 01-04, Available from http://ilk.kub.nl/downloads/pub/papers/ilk0104.ps.gz,* 2001.

[71] D. A. J. K. and J. A. , "Instance based Learning Algorithms," *Machine Learning,* vol. 6, pp. 37-66, 1991.

[72] O. R. H. G. V. and A. R. , "Implementation of IB1 Algorithm to the sentiment classification of film reviews," *Informatics Engineering Department Faculty of Information Technology UKDW Yogyakarta* , 2015.

[73] D. W. A. D. K. and M. K. A. , "Instance-Based ALgorithms," *Machine Learning, Kluwer Academic Publishers, Boston.* , pp. 37-66, 1991.

[74] W. D. A. v. d. B. and T. W. , "IGTree: Using Tree for Compression and Classification in Lazy Learning Algorithms," *Artificial Intelligence Review*, vol. 11, pp. 407-423, 1997.

[75] T. G. S. "The SocioLingustic and Grammatical study of personal names in Sidaamu Afoo," *Unpublished, Masters thesis, Department of lingustics, Addis Ababa University* , 2015.

[76] A. T. "A Grammar of Sidaama," *A Dissertation for the Doctor of Philosophy, Hebrew Univeristy, Jerusalem,* 2000.

[77] E. T. W. "Sound Correspondences and change: A comparative study of Burji, Hadiyya, Kambaata, Sidaama and Gedeo," *Unpublished Masters thesis, Department of Linguastics and Philology,* 2010.

[78] I. X. S. W. Y. L. and D. G. , "Sidama-Amharic-English Dictionary," *In cooperation with Ethiopian Languages Research Center, Addis Abeba University,* 2007.

[79] G. T. "Ideophones in Sidaamu Afoo documentation and description," *Unpublished Master's thesis, Department of Linguistics, Addis Ababa University,* 2013.

[80] K. K. and A. A. T. , "Modification within a Noun Phrase in Sidama," *Proceeding of the thirty-fourth annual meeting of the BERKELEY LINGUISTIC SOCIETY, Berkeley, CA, USA,* 2008.

[81] G. M. Wegari, "OroRoots: Rule-Based Root Generation System for Afan Oromo," *International Journal of Scientific and Engineering Research* , vol. 8, no. 3, 2017.

[82] E. M. A. v. d. B. and A. S. , "Memory-based morphological analysis generation and part-of-speech tagging of Arabic," *Proceedings of the ACL 2005 workshop on computational approaches languages to semitic languages,* pp. 1-8, 2005.

Appendix A: Sample Manually annotated wordlists

adaNkki∑raD

addaNsiϕ

addaNsiϕi3tiW

addeN

adoNteΨnniInnaC

aganaNhoØroJnnaC

agarVumma4'ne(roJ

agVa∞nnaC

annaN'yaΩnaC

arrishshoNteΨnaC

assVamH

baattoNnkeφriT

baattoNnsa)reR

babbVa∞

bicVootta©nkeφhuK

biifVa∞

bodirVa∞nniInoP

buddeenaN'yaΩnaC

bulleeNteΨnniI

bunaNhoØnoP

bunaNuUnniInnaC

burqaN

burqaNhoØ

cee`mVa∞leessaA

ciiggaAteΨnoPnaC

cimeessaN

cu`moNteΨ

cubboNaataamoA

cufVanaZi3nsa)raD

daafurVi3'ne(taS

daafurVi3weeloG

daddaloNhoØnkaJ

dadilleNhoØnoP

daguncho NhoØnoP

danchaA

danchaAteΨnaC

danqVi3tteAteΨ

daroNteΨnsoC

darsheN

darsheNteΨni9

darVa∞nkuJ

darVanni[

darVdhYamH

dayV

dayVeemmo@

dayVi3kkiG

dewee'lVi3neemmo*

dewee'lVinoα

diGegennVaamoAhoØ

diGkeeshshVi3tinoβ

diGrosVanno$

dimbVi3tu8taS

dodVamHa∞kki∑

dogVummo1

duu'mVa∞

duunoNnnaC

egennVoMkki∑nniItiW

ereeroN

e'Vnoommo€

faayyaAhoØnkaJ

fanVtu8nkeφroJ

farashshoN

farroN

fayyoN

fiiyVaanchoZ

fixxVi3tu8nsa)

fokkifatVteΨnsa)

fugVi3

fulVeemmo@

giirVi3
habVw{oommo<seλ
hagiidhVi3nanni|
hasaawaNhoØnaC
hayishshVa∞
hayxeNhoØnaC
hirVsiisFi3
honqooqichaAhoØnkaJ
hoogVi3roJ
hosVootto£roJ
hudeNteΨnoP
hurVsQi3tinoβ
iillVa∞kki∑
ilaalaNuUnniInaC
ilVi3
kaajjVa∞
keeshshVi3
kolishshoAteΨnaC
labbaaN
labbaaNhaLnnaC
labbaaNhuKraD
la'Veμ
la'VeμsiϕlekkaNi3weeloGhoØnaC
lekkaNkki∑nniInoP
lekkaNnsa)nniInoP
maganoNhoØ
maganoNhoØnaC
malawoNhoØ
mannaNuUnniIhoØnaC
masaalVi3
mesaneNteΨnoP
minVeμ
minVi3kki∑tiW
minVi3tiW
minVeμnnaC
mixashshoN
mixashshoNi3weeloG
mooteNi3chaZ
mooteNi3chaZhoØ
mundeeNnniIhaLnoP
muddVi3
muuzeNteΨ
muuzeNteΨnniI
odooN'ne(nniI
odooN'ne(noP
ogeessaNuUnniIhuK
ogeessaNuUnniIhuKnaC
qaaqqoNteΨ
qaaqqoNteΨnsoC
qarraNsiϕi3tiWnaC
qasVi3noP
qasVamHinoα
qeelVamHinoα
qiidaNhoØnoP
qiidaNi3reR
qiidVa∞hoØ
qolchVa∞'yaΩ
qolchVi3toE
seemoN
seemoNimmaA
shaaladoAhoØ

Appendix B: Classes (Boundary Marker symbols)

1 -1st person singular masculine simple perfect/ subject marker

4 -1st person singular feminine simple perfect /subject marker

5 -1st person plural simple perfect/subject marker

2 -2nd person singular masculine simple perfect / subject marker

6 -2nd person singular feminine simple perfect /subject marker

7 -2nd person plural/ polite simple perfect /subjective marker

3 -3rd person singular masculine simple perfect, nominative, epenthesis/ subject marker

8 -3rd person singular feminine /plural simple perfect / subject marker

9 -3rd person polite simple perfect /subject or interrogative marker

@ -1st person singular masculine imperfect /subject marker

-1st person singular feminine imperfect /subject marker

***** -1st person plural imperfect /subject marker

& -2nd person singular masculine imperfect / subject marker

% -2nd person singular feminine imperfect /subject marker

^ -2nd person plural/ polite imperfect/ present continuous / subject marker

$ -3rd person singular masculine imperfect /subject marker

! -3rd person singular feminine /plural imperfect /subject marker

| -3rd person polite imperfect/ 3rd person polite, 1st person plural present continuous / subject marker

[-1st person singular, 3rd person singular masculine present continuous subject marker

] -2nd person singular, 3rd person singular feminine/plural present continuous subject marker

< -1st person singular masculine present perfect / subject marker

> -1st person singular feminine present perfect / subject marker

€ -1st person plural present perfect /subject marker

£ -2nd person singular masculine present perfect /subject marker

© -2nd person singular feminine present perfect /subject marker

/ -2nd person plural/ polite present perfect /subject marker

α -3rd person singular masculine present perfect /subject marker

β -3rd person singular feminine /plural present perfect /subject marker

π -3rd person polite present perfect /subject marker

Ω -1st person singular pronominal possessive /object marker

∑ -2nd person singular pronominal possessive /object or Negative marker

μ -1st person verbal possessive / 2nd person plural imperative object or metathesis, nominalizer/ Adjective marker

θ -2nd person singular verbal possessive /object marker

ϕ -3rd person singular masculine verbal possessive /pronominal possessive /object marker

λ -3rd person singular feminine verbal possessive /pronominal possessive /object marker

φ -1st person plural verbal possessive /pronominal possessive/ object marker

(-2nd person plural verbal possessive/possessive /object marker

) -3rd person plural verbal possessive / pronominal possessive /object marker

Ø -singular masculine nominal dative/ dative postposition /copula marker

Ψ -singular feminine nominative dative/ dative postposition /copula marker

I -nominative ablative/ instrumental case marker /postposition marker

Z -nominalizer marker

Y -auto benefactive marker

X -intensive /reduplicative marker

Q -simple causative marker

F -double causative/ factitive marker

H -3rd person singular masculine object marker /passive marker

K -singular masculine nominative genitive/ relative marker

W -singular feminine nominative genitive/ relative /copula marker

T -plural nominative genitive/ relative marker

L -singular masculine absolutive genitive/ relative marker

S -singular feminine absolutive genitive /relative marker

R -plural absolutive genitive/ relative marker

U -nominative case subject marker

M -3rd person masculine, 1st person singular jussive, or Nominative marker

E -3rd person feminine singular/plural jussive marker

P -3rd person polite/ 1st person plural jussive, or conjugation marker

∞ -infinitive identifier/ present tense, nominalizer, Adjective marker

G -Negative marker

C -conjugation marker

N -noun marker

A -adjective marker

B -Plural marker

V -verb marker

J -subordinate marker

D -nouns dative case marker

{ -epenthesis, metathesis, other assimilatory process

Appendix C: Sample Features Extracted

=,=,=,=,=,=,=,k,o,l,i,s,h,s,h,o,h,o,n,a,=,=,=,=,=,0
=,=,=,=,=,=,k,o,l,i,s,h,s,h,o,h,o,n,a,=,=,=,=,=,=,0
=,=,=,=,=,k,o,l,i,s,h,s,h,o,h,o,n,a,=,=,=,=,=,=,=,0
=,=,=,=,k,o,l,i,s,h,s,h,o,h,o,n,a,=,=,=,=,=,=,=,=,A
=,=,=,k,o,l,i,s,h,s,h,o,h,o,n,a,=,=,=,=,=,=,=,=,=,0
=,=,k,o,l,i,s,h,s,h,o,h,o,n,a,=,=,=,=,=,=,=,=,=,=,Ø
=,k,o,l,i,s,h,s,h,o,h,o,n,a,=,=,=,=,=,=,=,=,=,=,=,0
k,o,l,i,s,h,s,h,o,h,o,n,a,=,=,=,=,=,=,=,=,=,=,=,=,C
=,=,=,=,=,=,=,=,=,=,=,=,l,a,l,o,=,=,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,a,l,o,=,=,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,a,l,o,=,=,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,a,l,o,=,=,=,=,=,=,=,=,=,N
=,=,=,=,=,=,=,=,=,=,=,=,l,a,',e,s,i,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,a,',e,s,i,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,a,',e,s,i,=,=,=,=,=,=,=,=,V
=,=,=,=,=,=,=,=,=,=,=,=,l,a,',e,s,i,=,=,=,=,=,=,=,=,µ
=,=,=,=,=,=,=,=,=,=,=,=,l,a,',e,s,i,=,=,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,a,',e,s,i,=,=,=,=,=,=,=,=,=,φ
=,=,=,=,=,=,=,=,=,=,=,=,l,e,k,k,a,i,w,e,e,l,o,h,o,0
=,=,=,=,=,=,=,=,=,=,=,=,l,e,k,k,a,i,w,e,e,l,o,h,o,n,0
=,=,=,=,=,=,=,=,=,=,=,=,l,e,k,k,a,i,w,e,e,l,o,h,o,n,a,0
=,=,=,=,=,=,=,=,=,=,=,=,l,e,k,k,a,i,w,e,e,l,o,h,o,n,a,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,e,k,k,a,i,w,e,e,l,o,h,o,n,a,=,=,N
=,=,=,=,=,=,=,=,=,=,=,=,l,e,k,k,a,i,w,e,e,l,o,h,o,n,a,=,=,=,3
=,=,=,=,=,=,=,l,e,k,k,a,i,w,e,e,l,o,h,o,n,a,=,=,=,=,0
=,=,=,=,=,=,l,e,k,k,a,i,w,e,e,l,o,h,o,n,a,=,=,=,=,=,0
=,=,=,=,l,e,k,k,a,i,w,e,e,l,o,h,o,n,a,=,=,=,=,=,=,0
=,=,=,l,e,k,k,a,i,w,e,e,l,o,h,o,n,a,=,=,=,=,=,=,=,0
=,=,l,e,k,k,a,i,w,e,e,l,o,h,o,n,a,=,=,=,=,=,=,=,=,G
=,l,e,k,k,a,i,w,e,e,l,o,h,o,n,a,=,=,=,=,=,=,=,=,=,0
l,e,k,k,a,i,w,e,e,l,o,h,o,n,a,=,=,=,=,=,=,=,=,=,=,Ø
e,k,k,a,i,w,e,e,l,o,h,o,n,a,=,=,=,=,=,=,=,=,=,=,=,0
k,k,a,i,w,e,e,l,o,h,o,n,a,=,=,=,=,=,=,=,=,=,=,=,=,C
=,=,=,=,=,=,=,=,=,=,=,=,l,o,o,s,o,o,m,m,o,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,o,o,s,o,o,m,m,o,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,o,o,s,o,o,m,m,o,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,o,o,s,o,o,m,m,o,=,=,=,=,=,=,V
=,=,=,=,=,=,=,=,=,=,=,=,l,o,o,s,o,o,m,m,o,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,l,o,o,s,o,o,m,m,o,=,=,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,o,o,s,o,o,m,m,o,=,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,o,o,s,o,o,m,m,o,=,=,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,o,o,s,o,o,m,m,o,=,=,=,=,=,=,=,=,=,<
=,=,=,=,=,=,=,=,=,=,=,=,l,u,k,k,i,c,h,o,h,o,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,u,k,k,i,c,h,o,h,o,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,u,k,k,i,c,h,o,h,o,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,u,k,k,i,c,h,o,h,o,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,u,k,k,i,c,h,o,h,o,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,u,k,k,i,c,h,o,h,o,=,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,u,k,k,i,c,h,o,h,o,=,=,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,u,k,k,i,c,h,o,h,o,=,=,=,=,=,=,=,=,=,N
=,=,=,=,=,=,=,=,=,=,=,=,l,u,k,k,i,c,h,o,h,o,=,=,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,l,u,k,k,i,c,h,o,h,o,=,=,=,=,=,=,=,=,=,Ø
=,=,=,=,=,=,=,=,=,=,=,=,m,a,g,a,n,o,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,g,a,n,o,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,g,a,n,o,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,g,a,n,o,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,g,a,n,o,=,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,g,a,n,o,=,=,=,=,=,=,=,N
=,=,=,=,=,=,=,=,=,=,=,=,m,a,g,a,n,o,u,n,n,i,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,g,a,n,o,u,n,n,i,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,g,a,n,o,u,n,n,i,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,g,a,n,o,u,n,n,i,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,g,a,n,o,u,n,n,i,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,g,a,n,o,u,n,n,i,=,=,=,=,=,=,N
=,=,=,=,=,=,=,=,=,=,=,=,m,a,g,a,n,o,u,n,n,i,=,=,=,=,=,=,U
=,=,=,=,=,=,m,a,g,a,n,o,u,n,n,i,=,=,=,=,=,=,=,=,=,0
=,=,=,=,m,a,g,a,n,o,u,n,n,i,=,=,=,=,=,=,=,=,=,=,0
=,=,=,m,a,g,a,n,o,u,n,n,i,=,=,=,=,=,=,=,=,=,=,=,I
=,=,=,=,=,=,=,=,=,=,=,=,m,a,n,a,a,d,o,n,n,a,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,n,a,a,d,o,n,n,a,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,n,a,a,d,o,n,n,a,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,n,a,a,d,o,n,n,a,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,n,a,a,d,o,n,n,a,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,n,a,a,d,o,n,n,a,=,=,=,=,=,=,0
=,=,=,=,=,=,=,=,=,=,=,=,m,a,n,a,a,d,o,n,n,a,=,=,=,=,=,=,A
=,=,=,=,=,=,m,a,n,a,a,d,o,n,n,a,=,=,=,=,=,=,=,=,=,0
=,=,=,=,m,a,n,a,a,d,o,n,n,a,=,=,=,=,=,=,=,=,=,=,0
=,=,=,m,a,n,a,a,d,o,n,n,a,=,=,=,=,=,=,=,=,=,=,=,C

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: Desta Legese Lalego

Signature: _____

Date: _____

Confirmed by advisor:

Name: Yaregal Assabie (PhD)

Signature: _____

Date: _____