ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

STATISTICAL AFAAN OROMO GRAMMAR CHECKER

ABEBE MIDEKSA DESALEGN

FEBRUARY 2015

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

STATISTICAL AFAAN OROMO GRAMMAR CHECKER

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES OF ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SCIENCE

BY

ABEBE MIDEKSA DESALEGN

FEBRUARY 2015

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

STATISTICAL AFAAN OROMO GRAMMAR CHECKER

BY

ABEBE MIDEKSA DESALEGN

Name and signature of Members of the Examining Board

Name	**Title**	**Signature**	**Date**
_____	Chair person	_____	_____
Ermias Abebe_____	Advisor(s)	_____	_____
Solomon Tefera(PhD) _____	Examiner	_____	_____
Martha Yifru(PhD)_____	Examiner	_____	_____

FEBRUARY 2015

DEDICATION

This thesis is dedicated to my son Moti Abebe, my wife Liya and my parents.

TABLE OF CONTENTS

CONTENTS

ACKNOWLEDGMENTS

First of all I would like to thank the Almighty God who blessed each and every minute of my life in doing all my work in his way. It is also my pleasure to express my deepest gratitude to my advisor Ato Ermias Abebe for his invaluable advice and guidance. He listened to all my problems I faced during this thesis and showed me the way to overcome them. He has been providing me constructive comments for the betterment of this study.

I would also like to thank Dr. Million Meshesha, Getachew Mamo, Tewodros Abebe for their support, constructive suggestion and encouragement at the very beginning of my research.

My special thanks also go to my family for their moral support and encouragement and my friend Getachew Gebru and Endashaw Morka for their financial support and constructive comments.

I would like to thank all my colleagues at Addis Ababa University, especially Asefa Bayisa, Kasahun Abdisa and others, for their ideal support.

Finally, I extend my heartfelt thanks and respect to all those people who were not mentioned here but their contributions have been significant for the completion of this work.

ABSTRACT

Natural Language Processing (NLP) is a research area that focuses on developing systems that allow computers to communicate with people using everyday language. In order to communicate through natural languages, grammatical correctness of a language is very significant. Therefore, it is very important to have natural language processing applications that recognize the grammatical errors that may occur in natural language texts.

The natural language processing application that recognizes the grammatical error of a language is called grammar checker. Different approaches can be used to develop a grammar checker for a language. These are rule based, statistical and hybrid approaches. In this study statistical Afaan Oromo grammar checker is developed and tested using a prepared dataset.

In the statistical approaches of grammar checking two techniques can be used for detecting the grammatical correctness of a given sentences. The first one is token n-gram, in which sequence of token are extracted and the second is tag n-gram, in which sequence of tag are extracted. In this study these two techniques of statistical approach are used and their performance is tested on 85 Afaan Oromo sentences.

The evaluation results show that the performance of token n-gram in identifying incorrect sentence is a recall 100%, precision of 78.1% and F-measure of 89.0% and the performance of tag n-gram technique in identifying incorrect sentences is a recall of 86%, precision of 82.6% and F-measure of 84.3%. On the other hand, the performance of token n-gram technique in identifying correct sentences is a recall 60%, precision of 100% and F-measure of 80% and the performance of tag n-gram technique in identifying correct sentence is a recall of 74.2%, precision of 78.2% and F-measure of 76.4%.

There are also some reasons that lead to the low performance of the two techniques. The first one is the issue related to the performance of sentence boundary detector, word splitter, POS tagger and morphological analyzer modules. Another reason is for the low performance of the two techniques is related to the quality of the corpus (spelling error, the spacing error). As a result this study recommends the following recommendation in order to increase the performance of the grammar checker. The first one is using spelling checker in order to increase the performance of POS tagger and Morphological analyzer. The other is using good quality corpus and good performing POS tagger and Morphological analyzer.

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

BNC	British National Corpus
LISGrammarChecker	Language Independent Statistical Grammar checker
LM	Language model
Ms-Word	Microsoft Word
NLP	Natural Language Processing
ORTO	Oromoia Radio and Televesion Organization
POS	Part Of Speech
SLM	Statistical Language Model
SAOGC	Statistical Afaan Oromo Grammar Checker
SOV	Sentence Object Verb
SVO	Sentence Verb Object
VOA	Voice Of America
XML	Extensible Mark-up Language

LIST OF LISTINGS

CHAPTER ONE

INTRODUCTION

1.1. Background

Natural Language Processing (NLP) is the branch of computer science which focuses on developing systems that allow computers to communicate with people using everyday language [1, 15]. NLP, also called Computational Linguistics, concerns itself with how computational methods can aid the understanding of human language [9].

There are different tasks of NLP used in understanding and processing natural language, such as speech recognition, named entity recognition, grammar checking. With the advancement in electronic and computer technology, there is an explosive growth in the use of these NLP techniques for processing information [15]. For instance, Grammar checkers, Speech synthesizers, text summarizers are the most widely used tools within language engineering.

Grammar checker is one of the major tasks in NLP, and it is mainly used in text proof reading. It determines the syntactical correctness of a sentence and, in many cases, suggests possible corrections. It is mostly used in word processors and compilers. Grammar checking for applications like compilers is easier to implement because the vocabulary is finite for programming languages, but for the natural languages, it is challenging because of the infinite vocabulary in human languages and for some exceptions that can be occur in natural languages [10]. For example in some natural languages their a sentences that can be constructed from one word.

Three main approaches are widely used for checking the grammar a language: syntax-based checking, statistical-based checking and rule-based checking [1, 10]. In the syntax-based grammar checking, a sentence is completely parsed to check the grammatical correctness [1, 2]. The sentence is considered incorrect if the syntactic parsing fails.

In the statistical approach, the part of speech (POS) tag sequences or token sequences are built from the prepared corpus, and the frequencies of these sequences are counted and their probability is calculated. The sentence is considered incorrect if the sequence probability is lower

than some threshold. The statistical approach essentially learns the rules from the tagged training corpus.

In the rule based approach an input sentences is matched against manually handcrafted rules [1, 2]. For instance, simple rules can be used to detect errors which are very easy to find, e.g. double punctuations.

A great number of diverse researches have been proposed and done on grammar checker for different languages, such as English, Bangla, Czech, Persian and Icelandic language [2,8,10]. The grammar checker appears to be highly dependent on the language it corrects [7]. Therefore, it may not be possible to use the grammar checker which is developed for one language for some other languages mostly for a grammar developed using rule based approach. This is because different languages have different morphological structure. Therefore, starting the implementation of grammar checker for languages like Afaan Oromo which is syntactically and morphologically complex is a major feat.

In this study, a prototype statistical Afaan Oromo grammar checker is developed and tested. The statistical approach does not need language resources like handcrafted grammatical rules, except for a tagged corpus to train the language model (LM) [10]. This is because if we want to check the grammar of a sentences by using their part of speech its necessary to prepare tagged corpus. In general, in this approach we can simply measure the probability of a sentence using n-gram analysis.

1.2. Statement of the problem

There are more than 80 languages in Ethiopia [17]. Afaan Oromo is one of the Cushitic languages with large number of speakers [6]. It is the third largest language in Africa by the number of speakers following Arabic and Hausa. There are closely 30 million people who speak Afaan Oromo mainly in Ethiopia, and the rest in Kenya, parts of Somalia and Tanzania [16].

Besides having a large number of speakers, Afaan Oromo is now serving as a working language in the local administration and courts in the regional government of Oromia. It is also a medium of instruction in primary and secondary schools, training institutes and colleges in the region.

Moreover, literature and folklore of Afaan Oromo language is being offered as a field of study in many universities in Ethiopia [16]. There are papers and web based media in Afaan Oromo.

For instance, Oromia Television and Radio (Web news), Kallacha Oromiyaa, Bariisaa, Yeroo, Voice of America (VOA) (web news), are some of the news and information media that use the language [26].

There is also wide use of the language in word processor such as Microsoft office and social media sites, such as blogs, facebook, twitter, etc. While using these media, the user of the language may create a text that contains a grammatical error. The content of the created text may get changed or remain meaningless because of grammatical error created by the user.

In general, Afaan Oromo is the language of education and research, language of administration and politics, language of ritual activities and social interaction [17]. Therefore, high performance grammar checker is very essential in generating error free Afaan Oromo texts.

In Ethiopia, there is an attempt to develop grammar checker for Amharic Language [15], the working language of the federal government of Ethiopia. The work which is done for Amharic language uses rule based and Statistical approach for checking simple sentence and complex sentence respectively. For Afaan Oromo, there is a research conducted to develop grammar checker by using rule based approach by Dabala Tesfaye[1]. The research has tried to detect some simple grammatical errors in Afaan Oromo [1]. But, the developed prototype does not detect some of simple sentences, compound and complex sentences which have grammatical errors.

To the knowledge of this researcher, there is no research conducted on developing grammar checker using statistical approach for Afaan Oromo language. Hence, it is the aim of this research to come up with Afaan Oromo grammar checker using statistical approach. Additionally, in developing grammar checker using this approach, both simple and complex sentences are taken in to account.

Therefore, in this study, the researcher attempts to answer the following research questions.

- ✓ What are the different types of grammatical errors occurring in written Afaan Oromo?
- ✓ What types of grammatical errors are difficult to identify in the sentences?
- ✓ What types of sentences are easily identified by the grammar checker?
- ✓ What is the best grammar checker algorithm for Afaan Oromo Language?

1.3. Objective of the Study

1.3.1.General objective

The general objective of this study is to explore the possibility of developing a statistical grammar checker that identifies grammatical mistakes occurring in the written text of Afaan Oromo language.

1.3.2.Specific objectives

The specific objectives of the research are:

- ✓ To review related literature on different natural language processing tasks that are helpful in developing grammar checker.
- ✓ To review related literature on grammatical structure of Afaan Oromo language.
- ✓ To compile corpus from different sources for training and testing the grammar checker.
- ✓ To select better technique in developing statistical grammar checker for morphologically rich language like Afaan Oromo.
- ✓ To design and implement a grammar checker for Afaan Oromo language.
- ✓ To evaluate the performance of the grammar checker on selected Afaan Oromo test data.
- ✓ To draw useful conclusion and forward recommendations for further work.

1.4. Methodology

1.4.1.Research methods

To conduct this study, two research methodologies are employed. The first one is the experimental setup method, and the second is quantitative method. The first method is used to undertake the experiment on the developed grammar checker and the algorithms used in it. The second method is used to analyze the result obtained during the experiment undertaken.

The researcher has performed the following procedures. The researcher starts by reviewing different relevant local and international journal articles, conferences papers, books and

resources from internet that are related to Afaan Oromo grammatical structure and grammar checker in order to have conceptual understanding and identify research gap on Afaan Oromo grammar checker.

The corpus is compiled from different sources that are written in Afaan Oromo language. Basically, the corpus is compiled from the official website of Oromiya National Regional State, Voice of America Radio Afaan Oromo language, International Bible Society official website, Oromiya Radio and Television Organization (ORTO) news and other Internet based sources.

The compiled corpus passes through preprocessing activities like tokenization, part of speech tagging and morphological analyzing. Using statistical approach of grammar checking, the probability of tri-gram sequence of token or tag is calculated. Different data with grammatical error are prepared for testing the checker. Finally, the manually checked text is compared with the system checked text and the precision, recall and F-measure of the checker are measured.

1.4.2. Development Tools and Techniques

For the development of grammar checker, the researcher uses Linux as an operating environment and uses Python 2.7.3 as programming language. The Linux operating system is used to run python program coded to develop the grammar checker. Linux environment selected as running environment because the POS tagger used for this study is written in a shell script.

Python is used as programming language to develop a prototype because, it is dynamic programming language that is used in a wide variety of application domains. It is simple, strong and involves natural expression of procedural code, modular, dynamic data types, and embeddable within applications as a scripting interface [5].

Two techniques of statistical approach are used and tested in the study. The first technique is token n-gram frequency. Here, the probability of token sequence is calculated for training and testing the checker. The second technique is tag n-gram frequency. In this technique, POS tag is assigned to all tokens, and the probability of the tag sequence is calculated for training and testing the checker.

1.4.3. Evaluation

After the prototype for grammar checker has been developed, it is trained using the prepared training dataset and tested using the dataset prepared for testing purpose. The performance of the checker during training and testing is evaluated manually. Furthermore, precision, recall and F-measure is calculated to evaluate the effectiveness of the developed system. Precision is the number of correctly flagged errors out of total flagged errors, whereas recall is the number of correctly flagged errors out of the total number of errors that occur in the text [15].

1.5. Significance of the study

The Afaan Oromo Grammar checker is hoped to have the following significance to Afaan Oromo Language speakers, students and others.

- ✓ It may improve the quality of Afaan Oromo text by detecting grammatically incorrect sentences.
- ✓ This research may also enable Afaan Oromo language users, specially the non-native to prepare different types of documents, letters, emails, etc effectively.
- ✓ It could very helpful for primary school students and language learners as they be able to become more independent and get their work done with the help of such a checker.
- ✓ It may also improve productivity as it saves the time users take for proof-reading.
- ✓ Additionally, this study can be used as a stepping stone for further work in this specific area.

1.6. Scope and Limitations of the Study

Errors which may appear in written text can be categorized as spelling errors, grammar errors, style errors or semantic errors [2]. Afaan Oromo text may contain the above types of errors. This study entirely focuses on developing a grammar checker that effectively determines the grammatical correctness of Afaan Oromo texts. The analysis of spelling errors, style errors and semantic errors are not included in the scope of this research.

The grammar checker is developed using Statistical approach of grammar checking. Therefore, no rule is constructed manually. Two techniques of N-gram analysis techniques are tested (Token n-gram check and Tag n-gram check) in order to determine the correctness of the sentences.

The developed Afaan Oromo grammar checker accepts Afaan Oromo sentence as an input and performs preprocessing activities such as tokenizing at word level and sentence level, POS tagging and morphological analyzing. After preprocessing, the probability of the sequence is calculated by using frequency of sequence in the corpus.. Based on the calculated values, the sentences are judged as correct or incorrect. If the sentence is incorrect, the checker tries to identify the type of error in the sentences.

Even though the statistical-based approach requires quite a large corpus size for a better performance [8], the grammar checker is trained with the corpus containing only 10000 words. This is due to constraints such as time, resource and training, testing and evaluating of large corpus which requires high processing speed and large computer memory.

1.7. Organization of the thesis

The thesis contains six chapters. The first chapter presents the background, statement of the problem, objectives, research methodology, significance, and scope and limitations of the study.

Chapter two presents the grammar of Afaan Oromo language including its morphology, writing system, word class, tag set and punctuation marks.

In chapter three, fundamental concepts related to grammar checking and various approaches that can be used to develop grammar checking are discussed. This chapter also presents summary of related researches conducted for Afaan Oromo and other languages.

In chapter four, the design and implementation of the grammar checker is discussed. Basically, this chapter discusses the statistical methods (techniques) used: Token N-gram and Tag N-gram with their architecture and modules used for each method.
The analysis of the results of the experiment is presented in chapter five. Additionally, the performance of two methods discussed in chapter four is evaluated in chapter five.

Finally, the last chapter, which is chapter six, presents the conclusions and recommendations based on the experiments conducted and results obtained.

CHAPTER TWO

OVERVIEW OF AFAAN OROMO

2.1. Background

Afaan Oromo is one of the languages that are widely spoken in Ethiopia and some neighboring countries like Kenya and Somalia [22, 23]. It is one of the five most widely spoken languages among the thousands of languages spoken in Africa [24]. According to the census taken in 2007, the number of people speaking Afaan Oromo as their mother tongue is 34.5% of total population of Ethiopia [25].

Currently, Afaan Oromo is used as a working language of the Oromia regional state of Ethiopia. Afaan Oromo is also used as medium of instruction in primary schools and given as an independent subject in secondary schools throughout the region. Moreover, a number of books, newspapers, magazines, educational resources, official documents, and religious writings are written and published in the language [6, 26, 27, 28].

2.2. Afaan Oromo Writing System

Afaan Oromo uses the Latin based alphabet called ***'Qubee'*** or ***'Qubee Afaan Oromo'***. It has been adopted as the official script of Afaan Oromo since 1991 [6, 29]. The ***'Qubee'*** writing system has a total of 33 letters of which 26 of them are similar with English letters and 7 of them are combined consonantal letters known as ***'qubee dachaa'*** (digraphs). The digraphs include '***ch', 'dh', 'sh', 'ny', 'ts', 'ph***' and ***'zy'*** [30,31].

There are five vowels in ***'Qubee Afaan Oromo'***. These are ***'a', 'e', 'i', 'o', 'u'***. These vowels may appear as short vowels or long vowels in the language. A vowel is said to be short if it is one. It is called a long vowel if it is two, which is the maximum. The use of long and short vowels can result in different meanings. Consider the following Example 1**:**

1. “lafa” : ground
2. “Laafaa” : soft

Example1

In the first word ***‘a’*** is short vowel and in the second word ***‘aa’*** is the long vowel which leads to change in meaning.

There are 28 consonants in ‘***Qubee Afaan Oromo’*** including digraphs. They are listed in table 3.2 with their sounds. Geminating (doubling a consonant) is also significant in Afaan Oromo because consonantal length can distinguish words from one another. Most Afaan Oromo consonants can be geminated except ***‘h’*** and digraphs [27, 30, 31]. Consider the following words in Example 2:

1. “damee” : branch
2. “dammee” : sweety
3. “badaa” : bad
4. “baddaa” : highland

Example 2

As we can see from the above example 2, geminating consonants changes the meaning of each word. Afaan Oromo vowels and consonants with their sound are shown in Table 3.1 and Table 3.2 respectively.

Vowels			
	Front	**Central**	**Back**
close	i /ɪ/, ii /iː/		u /ʊ/, uu /uː/
Mid	e /ɛ/, ee /eː/		o /ɔ/, oo /oː/
Open		a /ʌ/	aa /ɑː/

Table 2.1 Afaan Oromo vowels and their sounds (adopted from [23])

Consonants						
		Bilabial/ Labiodental	**Alveolar/ Retroflex**	**Palato-alveolar/ Palatal**	**Velar**	**Glottal**
Stops and affricates	Voiceless	(p)	T	ch /tʃ/	k	' /ʔ/
	Voiced	b	D	j /dʒ/	g	
	Ejective	ph /p'/	x /t'/	c /tʃ'/	q /k'/	
	Implosive		dh /ɗ/			
Fricatives	Voiceless	f	S	sh /ʃ/		h
	Voiced	(v)	(z)			
Nasals		m	N	ny /ɲ/		
Approximants		w	L	y /j/		
Rhotic			R			

Table 2.2 Afaan Oromo consonants and their sounds (adopted from [23])

2.3. Afaan Oromo Morphology

Morphology is the study of morphemes and their arrangements in forming words [16]. Morphemes are the minimal meaningful units which may constitute words or parts of words [37]. The two broad classes of morphemes are stems and affixes (prefix, infix and suffix). The stem is the main morpheme of the word, supplying the main meaning whereas affixes are used to add additional meaning to words [16, 37].

Like in a number of other African and Ethiopian languages, Afaan Oromo has a very complex and rich morphology [1, 31]. It has the basic features of agglutinative languages involving very extensive inflectional and derivational morphological processes [1]. In agglutinative languages like Afaan Oromo, most of the grammatical information is conveyed through affixes, (that is, prefixes and suffixes) attached to the root or stem of words [1]. Afaan Oromo words have some prefixes and infixes whereas suffixes are the predominant morphological features in the language [31].

In Afaan Oromo, words can be formed from morphemes in two ways [33]. These are inflectional morphology and derivational morphology. In inflectional morphology, words are formed by the combination of stem with a grammatical morpheme, usually resulting in a word of the same class as the original stem, and usually filling some syntactic function. Inflectional morphemes modify a word's tense, number, aspect, and so on.

‘singular	Plural
‘barsiisaa’ (teacher)	‘barsiisota’(teachers)
‘mana’ (house)	‘manneen’ (houses)
‘waraabeessa’ (hyena)	‘waraabeeyyii’ (hyenas)

Table 2.3 Afaan Oromo inflectional morphology

From the above Table 3.1, we can see that the plural nouns can be formed by adding suffix **‘-ota’, ‘-een’, ‘-yyii** to the stem of the words**.**

Derivational morphology deals with word formation from stem and grammatical morphemes, usually resulting in word of different lexical class. For example,

Words	Derived from
‘hojjetoota’ (workers)(noun)	‘hojjete ’(Work):verb
‘gammachuu’ (happiness)(noun)	‘gammadaa’(Happy):adjective
‘namummaa’ (humanity)(noun)	‘ nama’(Human):noun

Table 2.4 Afaan Oromo derivational morphology

2.4. Word Categories in Afaan Oromo

Words are the basic unit of a given language [33]. The combination of these words on the bases of the language gives us phrases, clauses and sentences. The meanings of these sentences depend on each word of the sentence and the way they are arranged. Afaan Oromo words can be placed in to different categories [34, 35]. These categories are Noun, Verb, Adjective, Adverb, Adposition, Pronoun, Conjunction and Interjection and numerals [34, 35].

2.4.1. Nouns

Afaan Oromo nouns are words used to name or identify any of the categories of things, people, animal, places or ideas [34, 35]. However, sometimes lexical classes like noun can be defined functionally (morphologically and syntactically) so that some words for people, places, and things may not be nouns [35].

In Afaan Oromo, nouns mainly occur at the beginning of a sentence. In the following examples, Afaan Oromo nouns are italicized and underlined.

‘*Dimaan* fillannoo kooti.’(Red is my preference.)

‘*Sareen* manatti olixxe.’(A dog entered into a house.)

Words that are categorized as nouns in a sentence can be a subject or object [35]. Subject mostly comes at the beginning whereas an object mostly comes after subject and before verb in a sentence. Consider the following example,

1. **‘*Warabessi harree* nyaate.’ (A hyena eats a donkey)**
2. **‘*Tolaan mana* ijare.’ (Tola built a house.)**

In the above two sentences, the underlined and italicized words, **‘Warabessi’ (A hyena)** and **‘Tolaan’(name of person)** are the subject of the sentences, whereas the two italicized words **‘harree’ (donkey)** and **‘mana’ (house)** are the objects of the two sentences respectively.

Afaan Oromo nouns are also inflected for number i.e. singular and plural [37, 43]. Singular nouns are nouns which are built without adding any affix. On the other hand, plural nouns are mainly formed by adding suffix or using numerals with single noun [29, 35].

The following are suffixes used to form plural nouns in Afaan Oromo. **‘-oota’, ‘-ota’, ‘-wwan’, ‘-een’, ‘-lee’,’-yyi’** etc. Table 3.5 presents plural noun formation in Afaan Oromo

singular	Plural	Plural marker
‘Mana’ (house)	‘Manneen’ (houses)	-een
‘Sangaa’ (ox)	‘Sangota’ (oxen)	-ota
‘Waraabeessa’ (hyena)	‘Waraabeeyyi’ (hyenas)	-yyi
‘Hojii’ (work)	‘Hojiilee’ (works)	-lee
‘Barsiisaa’ (teacher)	‘Barsiisota’ (teachers)	-ota
‘Sa’a’ (cattle)	‘Saawwan’ (cattle)	-wwan

Table 2.5 plural noun formation using suffix

Additionally, plural nouns can be formed by using numerals [35]. This can be done by writing singular noun followed by numeral. For example, the following are valid plural nouns in Afaan Oromo. Table 3.6 presents plural noun formation using numerals.

singular	**plural**
‘Mana’ (house)	**‘Mana sadi’(three house)**
‘Nama’ (man/woman)	**‘Nama lama’ (two man)**
‘Hoolaa’ (sheep)	**‘Hoola kudhan’(ten sheep)**

Table 2.6 plural noun formation using numerals

Afaan Oromo analyses masculine and feminine genders and most of the nouns belong to either of the two. However, there are some nouns used for both masculine and feminine [6, 35, 36]. For example, the noun **‘nama’(man or woman)** can be used for masculine and feminine.

Afaan Oromo nouns are also inflected for definiteness, but not for indefiniteness [37].The definiteness (the English ‘the’) in Afaan Oromo can be different for masculine and famine. For masculine **‘*-icha*’** is mainly used while **‘-ittii’** is used for feminine. Table 3.7 presents the definite formation of Afaan Oromo nouns

Noun	**definiteness**
‘nama’ (nama)	**‘namicha’ (the man)**
‘harree’ (donkey)	**‘harricha’ (the donkey)**
‘qaalluu’(priest)	**‘qaallitti’ (the priest)**

Table 2.7 definiteness form of nouns

Afaan Oromo nouns are also inflected for case. Case is a grammatical category of nouns that indicates the nature of their relationship to the verb in sentences [37].

Afaan Oromo nouns can be formed by adding derivational suffix to different categories of words such as noun, verb, adjectives etc. [37, 35]. Table 3.8 presents derivational noun formation

Noun	Derived noun
'bilisa'(free)	**'bilisummaa'(freedom)**
'fira' (relative)	**'firummaa' (relationship)**
'nagaa' (peace)	**'nageenya' (peaceful)**
'jabaa' (strong)	**'jabeenya' (strength)**
'nama' (man)	**'namooma' (humanity)**

Table 3.8 derivational nouns formation

In the above Table3.8, abstract nouns are formed from other nouns by adding ***'ummaa'*** or ***'eenya'*** or **'ooma'** suffix.

2.4.2. Pronouns

Afaan Oromo pronouns can be used for replacing nouns and noun phrases [35]. Like nouns, Afaan Oromo pronouns decline for number and gender [34]. Consider the following pronouns

'ishee': she, represents feminine noun and singular

'isa' : he, represents masculine noun and singular

'isaan' :they, represents plural nouns and either masculine or feminine.

There are different categories of pronoun in Afaan Oromo based on their functionality and meaning in a sentences. These are personal, possessive, demonstrative, reflective and reciprocal pronouns [37, 35].

Personal pronouns can be used to replace the subject and the object of a sentence. The following are Afaan Oromo personal pronouns used to represent subject of the sentence.

	First person	**Second person**	**Third person**
Singular	**'Ani'(I)**	**'Ati'(you)**	**'Inni'(he), 'ishiin'(she)**
Plural	**'Nuyi'/'nuti' (we)**	**'Isiin'(you)**	**'Isaan'(they)**

Table 3.9 personal pronouns that can represent subject

Consider the following two sentences.

‘Firomsaan leenca ajjeese.’ (Firomsa kills a lion)

‘Inni leenca ajjesse.‘ (He kills a lion)

The above two sentences have the same meaning. Even if in the second sentence the pronoun **‘inni’ (he)** replaces the subject **‘Firomsaan’(name of a person).**

A personal pronoun can also replace an object of a sentence in Afaan Oromo [35]. Table 3.10 presents personal pronouns that can replace objects of the sentence.

	First person	Second person	Third person
Singular	Ana (me)	si(you)	Isa(him), ishii(her)
Plural	nu (us)	Isiin(you)	Isaan(they)

Table 3.10 personal pronouns that can replace object

Consider the following example:

‘Leensaan barattota barsiiste.’(Lensa teaches students.)

‘Leensaan isaan barsiiste.’(Lensa teaches them.)

In this example, **‘isaan’(them)** replace the object **‘barattota’(students**)

Another category of Afaan Oromo pronoun is possessive pronouns which are used to indicate the ownership of something. Table 3.11 shows possessive pronouns of Afaan Oromo.

	First person	Second person	Third person
Singular	Koo/kiyya (mine)	Kee (yours)	Isaa(his),ishee(her)
Plural	Keenya (ours)	Keessan (yours)	(i)saan(their)

Table 3.11 Afaan Oromo Possessive pronouns

Consider the following example:

‘Qubeellaan kun kan *koo* dha.’ (This pen is mine)

‘Rifeensi *isaa* gurracha.’ (His hair is black)

The italicized and underlined words in the above sentences represent possessive pronouns in Afaan Oromo.

Demonstrative pronouns are pronouns that are used to refer to a thing that was known previously or mentioned earlier. It can also be used to refer to the objects which are in the speaker's mind [35]. Both proximal and distal demonstrative pronouns exist in Afaan Oromo. Proximal pronouns have masculine and feminine whereas distal do not have. However, plural and singular demonstratives are not distinguished. The proximal Afaan Oromo Pronoun are:

‘kana’/’kuni’ (this/these)(masculine)

‘tana’/’tuni’ (this/these)(feminine).

And the distal Afaan oromo pronouns are:

‘san’ (that)

’sun’ (those)

2.4.3. Verbs

Verbs are words or compound of words that express action, state of being in or relationship between two things [34,39]. In Afaan Oromo verbs mostly appear at the end of sentence. Consider the following example:

‘Guutaan kaleessa *deeme*.’ (Guta *went* yesterday.)

‘Caalaan kubbaa *dhiite.*’ (Chala *kicked* a ball.)

In this example, the words written in italic and underlined **‘deeme’(went)** and **‘dhiite’ (kicked**) are verbs of the sentence.

Afaan Oromo Verbs are inflected for number, gender and tense[35]. Additionally Afaan Oromo verbs can be categorized into main (transitive or intransitive) and auxiliary verbs [29, 39]. Intransitive verbs are main verbs which do not take object or complement in a sentence [35]. The following examples illustrate intransitive verb in Afaan Oromo.

‘Namichi *fige*.’ (the man runs)

‘Isaan Kaleessa *dhufan*.’ (they came yesterday.)

In the above example, the words written in italic and underlined are transitive verb. They do not transfer any message from subject to complement.

Transitive verbs are main verbs which transfer message to complement (objects). Consider the following two sentences:

‘Tolaan ulee *cabse*.’ (Tola broke a stick)

‘Caalaan muka *mure*.’ (Chala cut a tree)

In the above two examples the verb **‘cabse’(broke)** and **‘mure’(cut)** are transitive verbs. They interrelate subject and object in the sentences`.

Auxiliary verbs support the main verbs used in a sentence. The following are Afaan Oromo auxiliary verbs **‘dha’**, **‘ta,e’, ‘qabda’, ture, ‘jira’,** etc. In the following two examples the auxiliary verbs are written in bold and italic.

‘Isheen barattu cimtu *dha.*’ (she is a clever student.)

‘hojii kana hojjechuu *qabda.*’ (you have to work this job.)

In the above two sentences the word **‘dha’** and **‘qabda’** are auxiliary verbs.

2.4.4. Adjectives

Adjectives in a sentence are used to modify nouns to show the quality of things. i.e. it specifies to what extent a thing is distinct from something else [3,34,35].

Consider the following examples:

‘Leenseen *bareeddu* dha’ (lense is beautiful.)

‘Tolaan *dheeraa* dha’ (Tola is tall.)

In the above examples the word **‘barreeddu’(beautiful)** and **‘dheeraa’ (tall)** are adjectives.

Afaan Oromo adjectives can be formed from compound words [35]. For instance, **‘humna dhabeesssa’(weak),‘simbo qabeessa’(handsome)** are some of adjectives constructed from compound words.

Adjectives inflect for number and gender in Afaan Oromo language. Table 3.12 presents inflection of adjectives for number and gender.

singular	plural	masculine	Feminine
guddaa	gudguddaa	guddaa	Guddoo
jabaa	jaboota	jabaa	Jabduu
Ko’eessa	Ko’eeyyii	Ko’eessa	Ko’eettii
cimaa	ciccimoota	cimaa	Cimtuu

Table 3.12 adjective inflection for number and gender

2.4.5. Adverbs

Adverbs are words which are used to modify verbs [35, 38]. In Afaan Oromo adverbs come before the verb they modify. Afaan Oromo adverbs are categorized as adverbs of time, place and manner (condition) [34,35]

Adverbs of time show the time the action takes place. The following are the words that can be used as adverbs of time in Afaan Oromo language.

'amma'(now),'boru'(tomorrow),'kaleessa'(yesterday),'yoom' (when) ,'har'a'(today),'galgala'(tonight) etc.

Consider the following example.

'boonsaan *kalessa* dhufe.' (bonsa came yesterday.)

'Qananiisaan *boru* ni figa.'(Kenenisa will run tomorrow.)

In these examples the word '**kaleesa'(yesterday)** and **'boru'(tomorrow)** are adverbs of time. Mostly adverbs of time answer the question of when the action takes place.

Adverbs of place show the place where the action takes place. The following are the words that can be used as adverb of place in Afaan Oromo. **'as'(here), 'achi' (there), 'gadi'(below), 'gubbaa'(above), 'jidduu'(middle),'irra' (on**), etc. Consider the following example,

'Tolaan *mana* jira.' Tola is at home.

'Inni *konkolaataa irra* jira' he is on the car

Adverb of manner show how the action of the sentence is done. The following are Afaan Oromo words that can be used as adverb of manner **'ariitin'(quickly), 'suuta' (slowly), 'akka gaarii' (well) etc.**

Consider the following example,

'Inni *ariitin* figa.' He is running quickly.

'Calaan *baay'ee* cimaa dha.' Chala is very clever

In the above sentences the word **'ariitin'(quickly)** and **'baay'ee'(very)** are adverbs of manner.

2.4.6. Adpositions

The term adposition refers to words, which will have meaning only when they are attached or used together with other words such as nouns, verbs, pronouns and adjectives [27]. Adpositions are characterized by having no inflectional or derivational morphology and belong to the closed category [27]. The word or phrase that the preposition introduces is called the object of the preposition [3].

Some of the adpositions in Afaan Oromo include: **'akka'(as), 'eegasii' (since), 'hamma'(until),'gara'(to), 'gadi'(below), 'ol'(above), 'irra'(on)**, and so on.

Consider the following sentences.

'caaltuun *gara* mana barumsaa deemte' (Chaltu went to school)

'Toolaan konkolaataa *keessa* jira' (Tola is in a car)

In the above sentences **'gara'(to)** and **'keessa'(in)** are adpositions introducing **'mana barumsa' (school)** and **'konkolaataa'** (**car)** respectively.

Afaan Oromo adpositions can appear as simple adpositions that stand alone as separate words or can be attached to other words [27]. Consider the following examples,

'Abbabaa *walin' (*with Abebe)

'*Gara* mana' (to house)

'harka-an' (by hand)

'barataa-f' (for student)

In the first two examples, we have stand alone adpositions '**walin**' (**with**) and '**Gara'**(**to**) which are separated from other words. The third and fourth example show suffixed adoposition. These are '-**an**' (**by**) and '**-f**' **(for**).

2.4.7. Conjunctions

Conjunctions are words that are used to join words, phrases or sentences [34]. Conjunction can be categorized as coordinating and subordinating conjunctions. Afaan Oromo Coordinating conjunctions are used to join main clauses that are given equal emphasize by user [34]. Afaan Oromo coordinating conjunctions include **'akkasumas'(besides/in addition to), 'garuu'(but), 'haata'u malee'(however), 'ta'ullee'(even though),**

‘kanafuu’ (so/therefore), ‘moo’(or) and so on. Consider the following examples,

1. **‘Abdiin hin barannee *garuu* waan baay’ee beeka.’ (Abdi is illiterate but he knows everything.)**
2. **‘Ati magaala *moo* badiyyaa jiratta?’(where do you live urban or rural?)**

In the first sentence, two sentences **‘Abdiin hin baranne’(Abdi is illiterate’)** and **‘waan baay’ee beeka’(he knows everything**) are joined by coordinating conjunction **‘garuu’ (but).** In the second sentence, two words ‘ **magaala’(urban)** and **‘badiyyaa’(rural)** are joined by **‘moo’ or**. Therefore coordinating conjunctions can be used to join words, phrases ,clauses or sentences.

Subordinating conjunctions are those conjunctions that are used to join the main clause with the subordinate clause. A subordinating conjunction is always followed by a clause [38]. Afaan Oromo subordinating conjunctions include **‘yoo’ (if), ‘akka waan’ (asif), ‘wayta’/’yeenna’(when), ‘hamma’(until), ‘erga’(after), ‘dursa’ (before)** etc.

The following examples illustrate subordinating conjunction

‘Wayta innii dhufuu ani barressa jira.’ (When he was coming I was writing’)

‘Yoo haalaan hojjette qormaata ni dabarta.’(If you work hard you will pass the exam.)

‘Wayta’(when) and **‘Yoo’ (if)** in these sentences are used as subordinating conjunction. The **‘Wayta (when)** joins two subordinating clauses that are **‘inni dhufu’ (when he was coming)** and **ani barreessa jira,(I was writing.)**

2.4.8. Numerals

Numerals include words that refer to order or quantity of something [35]. Most of the time the position of numerals follows the category with which they form a syntactic unit [39]. Afaan Oromo numerals can be cardinal or ordinal. The following are the cardinal numerals in Afaan **Oromo: ‘tokko’(one),lama’(two)‘kudhan’(ten), ‘dhibba’(hundreds), ’kuma’(thousands),etc.**

On the other hand, ordinal numerals of Afaan Oromo can be formed from cardinal numerals by adding the suffix **'ffaa'** to them. Numeral in Afaan Oromo can be formed in compound forms [44]. For instance, **'kuma tokko' (one thousand),'dhibba sadi'(three hundred)** are valid Afaan Oromo numerals. Consider the following sentences:

1. **'Ibsaan konkolaataa *lama* qaba.'(Ibsa has two cars.)**
2. **'Jalanneen kutaa isiitii *tokkoffaa* baatee.'(Jalane stood first from her classes.)**

In the first sentence, the word **'lama' (two)** is cardinal numeral. It follows the word **'konkolaataa' (car)** and describes its quantity. In the second sentence, the word **'tokkoffaa' (first)** is the ordinal numeral.

2.5. Afaan Oromo Tags and Tagset

Tags are labels that add more information on each word in a sentence [34,35]. They are used represent the lexical categories of words. For example NN can be used to represent nouns and VV can be used to represent verbs in a given sentence. For this study the tagset developed by [35] are used. The complete list of the tag set is included on appendix II

2.6. Punctuation marks in Afaan Oromo

Punctuation marks used in both Afaan Oromo and English languages are the same and used for the same purpose with the exception of apostrophe [27]. The apostrophe mark (‘) in English mostly shows possession but in Afaan Oromo it is used in writing a glitch sound **'hudhaa'.** For example words like **'du'a' (death), 'har'a' (today), 'bay'ee', (many), etc** contain glitch the sounds. The following are the most commonly used Afaan Oromo punctuation marks.

1. Full stop(.): mostly used at the end of sentence and used in abbreviations
2. Comma (,): is used to separate listing in a sentence.
3. Question mark (?): used at the end interrogative sentences
4. Exclamation mark (!): used at the end of command and exclamatory sentences

CHAPTER THREE

GRAMMAR CHECKER

3.1. Introduction

Natural language is a way for humans to communicate using all written and spoken words and sentences. People speaking the same language communicate and understand each other while people speaking different languages cannot communicate [3]. Communicators understand each other because they know and follow same grammatical rule of the language they use.

Every Natural language has its own grammar. Grammar of a language deal with the forms and structure of words (morphology), their arrangement in phrases and sentences (syntax), and their classification based on their function (parts of speech) [3]. If there is a difference in syntax or morphology in two languages, these languages are not the same [5].

3.2. Grammar errors

Errors which may appear in written text can be categorized as spelling errors, grammar errors, style errors or semantic errors [2].

Spelling errors are errors that are caused due to the inappropriate arrangements of letters or characters in word making [2]. These errors are defined as an error which can be found by common spell checker software [2]. Spell checkers simply compare the words of a text with a large list of known words. If a word is not in the list, it is considered incorrect. Similar words will then be suggested as alternatives.

Style errors may happen because of Using uncommon words and complicated sentence structures which make a text more difficult to understand [2].

Semantic error is occurred when a sentence contains incorrect meaning of information in a text. It requires extensive knowledge to identify semantic errors in the text[2]. Therefore, these errors usually cannot be detected automatically.

Consider the following example adopted from[2]

MySQL is a great editor for programming!

This sentence is neither true nor false – it simply does not make sense, as MySQL is not an editor, but a database. This cannot be known, however, without extensive world knowledge.

People may create errors which may violate the syntactic rule of the language while writing or speaking. Those errors which violate the syntactic laws of the language are called grammatical errors [1, 2, 15].

In this study, the researcher only focuses on the analysis of grammar errors. The analysis of spelling errors, style errors and semantic errors are beyond the scope of this research. Every natural language has their own grammar rule and regulations that differentiate them from others languages. Different types of grammatical errors may occur during writing or speaking in Afaan Oromo.

3.2.1. Subject verb agreement

In Afaan Oromo, the subject of the sentence must agree with verb in three different forms. i.e. number, gender and person[3,44]. Let us illustrate with example.

3. * Baratootni gara mana barumsaa deeme.
The students to school went

4. * Ani kitaaba bitte.
I a book buy

5. * Mucayyoon barataa cimaa dha.
The girl student clever is

The first sentence above shows a number mismatch in Afaan Oromo. The subject **"Baratootni"** is plural form, whereas the verb **"deeme"** refers to singular form of verbs for the subject. The second sentence has a person mismatch in it. The subject **"Ani"** is first person singular, whereas the verb **"bitte"** is a verb used for the third person singular feminine. In the third sentence, there is gender mismatch between subject and verb. The subject **"Mucayyoon"** is feminine, whereas the verb **"cimaa"** is a verb used for masculine subject. The subject verb disagreement can be corrected by either changing the subject or the verb. For instance, changing the verbs in the above sentences corrects the sentences.

NB: * represents the sentences are grammatically incorrect.

1. **Baratootni gara mana barumsaa deeman. (The students went to school)**
2. **Ani kitaaba bite.(I buy a book)**
3. **Mucayyoon barattuu cimtuu dha.(The girl is clever student)**

3.2.2. Word Order

Different languages have different word order [1, 2]. Word order includes the general structure of the sentence, the modifier and modified word order (adjective and noun or adverb and verb) and the like. Afaan Oromo mainly follows SOV sentence structure [1]. For instance, in the sentence

"Toolan leenca ajjese."
Tola a lion killed
[S] [O] [V]

The word "**Toolan"** is the name of the person who is the subject of the sentence. The word **"leenca"**(**a lion**) which is the object, while the word **"ajjese" (killed**) is the verb.

As long as the placement of the words of the language is wrong, then the sentence is considered grammatically incorrect. The word order also includes the order adjective and noun , the order of adverb and verb, the order of verb and sentences end and the other word class order in a sentence. In Afaan Oromo, Adjectives should appear after the noun that they modify and adverbs should appear before the verb they modify.

3.2.3. Adjective and Noun Agreement

Modifiers are words or phrases that are used in a sentence to elaborate something [15]. Modifiers may be either adjectives which modify noun or adverbs which modify verbs. An adjective is used in a sentence to give a clear picture of the noun that it modifies. It is simply a word that tells more about the noun. Adjectives are inflected for number and gender of the noun they modify. If an adjective is inflected for number and gender of the noun, the number and the gender should agree with the number and gender of the noun. In Afaan Oromo, adjectives appear after nouns that they modify. For example:

mucoo gurratti
girl a black

is a noun phrase that contains adjective **"gurratti"(black)** and noun **"mucoo"**(**girl**). The noun **"mucoo" (girl**) has a feminine character and the adjective **"gurratti" (black)** has a feminine gender marker in it. If this phrase is rewritten like:

***mucoo gurracha**
girl a black

it would be grammatically incorrect as the gender marker (which is masculine marker) in the adjective does not match with the gender of the noun. To correct adjective noun disagreements in gender, either the noun should be changed to the gender of the adjective or the gender marker in the adjective should be changed to the gender of the noun.

Adjectives and nouns should also agree in number. Consider the example:

***baratoota cimaa**
Students clever

The above example shows adjective noun disagreement in number. It contains a singular marker in the adjective and the noun it modifies is a plural noun. The adjective and noun disagreement can be corrected by adjusting number marker either on the adjective or the noun. For instance, the example above can be modified as:

baratoota cicimoo or barataa cimaa
Clever students student clever

3.2.4. Adverb and Verb Agreement

In Afaan Oromo, adverbs usually modify the first verb that comes next to them [1]. As in many languages Afaan Oromo adverbs are classified into subclasses such as adverbs of time, place, manner etc.[3,34,35]

The time adverbs describe the time at which an event takes place. The place adverbs describe the place where events take place. The time adverb and the tense disagreement is one of the common grammatical errors in Afaan Oromo, especially for new language learners [3].
Consider the following sentence.

***Biqilaan kaleessa dhufa.**
Bikila yesterday will come

In this sentence, **"kaleessa"(yesterday**) is adverb of time which is in the past. But the verb **"dhufa"(will come)** expresses the action that will be done in the future. The above sentence can be corrected either by modifying the adverb or verb.

"Biqilaan Kaleessa dhufe." Or "Biqilaan boru dhufa."
Bikila yesterday came Bikila tomorrow will come

Here the adverb and verb agree in both sentences.

3.3. Approaches to grammar checking

Basically, four approaches are used for grammar checking in a language, i.e pattern matching approach, statistical approach, rule-based approach and hybrid approach [8].

3.3.1. Pattern Matching

In the a pattern matching approach common grammatical mistakes are stored together with their corrections, and a sentence, or part of it, is checked by matching it to some error entry in the data storage[8]. In case of a match, an error is identified and can be corrected with the stored correction. This method is quite effective for the patterns that are in the data storage. But there is a lack of generality and every small difference in grammar needs a new pattern. For example the wrong sentences,

* **`He sell`**. and

`* He tell.`

need separate entries in the database, even if they differ only in one character, the missing **'s'** for third person singular. As a result storage space is the main problem of this approach. This is because every error that may appear in the language needs its own entry in the lookup table to correct the error.

3.3.2. Rule-based approach

The rule based approach is a common way for checking the grammatical correctness of the sentences. In this approach, an input text is tested against manually handcrafted rules [1, 2, 21]. For instance, simple rules can be used to detect errors which are very easy to find, e.g. double punctuations. But, for more complex sentences the rules to detect errors become more complicated. Constructing the rules manually is one of the disadvantages in this approach [1, 8, 15]. Another disadvantages of this approach is, the rules constructed may not be complete. Therefore, this approach cannot solve both compound and complex sentences since it is difficult to set rules for compound and complex sentences [1, 15].

This method has several attractive features [1,15]. New rules can be easily added to the system; an existing rule can be modified or can easily be removed. Every rule can have a corresponding extensive explanation of error message which are helpful for the end users. The rules can be defined by linguists, with limited or no programming skills.

3.3.3. Statistical approaches

Another approach for grammar checking is the statistical approach. The main assumption in this approach is that sentence can be corrected by using large amounts of text containing grammatically correct sentences [10, 8, 15]. These texts form sequences of tokens or POS tags which are used to detect grammatical errors in the given sentences.

In the statistical approach, we can simply measure the probability of tokens or POS tags sequences in a sentence using n-gram analysis in order to determine the grammatical correctness of a sentence [10]. If the sequence of tokens or POS tags is common in the corpus and the probability of it is not zero then the text is considered as correct. If the sequence is not common or do not appear in the corpus, the probability becomes zero and the text is considered incorrect.

This approach can be implemented using n-gram analysis in two ways [8, 10].

3.3.3.1. Token N-gram Analysis

In the token N-gram analysis the probability of tokens appearing in a certain order is calculated. Based on the value of N (1, 2, 3, 4, …), N adjacent tokens are selected to calculate the probability of the sentence. For example, using bigram analysis the probability of the sentence

He is playing.

can be given by:

P (“He is playing.”) = P (He | <start>) * P (is | He) *
P (playing | is) * P (. | playing) *
P (.|<end>)

P is the probability of the sentence or probability of the token sequence in the sentence. For instance, **P(He|<start)** is the probability of the word coming at the start of the sentence. It can be calculated by count of **(<start> He**) divided by the total number of sentences in the corpus.

Now if any of these three words are not in the training corpus then the probability of the sentence will become zero because of multiplication. If we, therefore consider the words in this statistical method then we need a huge corpus that must contain all the words of the language.

3.3.3.2. Tag n-gram Analysis

In token N-gram analysis, the probability of the sentence may become zero because the exact word may not found in the training corpus [10]. To solve this problem, tag N-gram analysis can be used. In tag N-gram analysis, every word in both the training corpus and the testing sentence are assigned to their POS tag [8]. The POS tagging increases the availability of words in the corpus. This is because,two different words may belong to the same word class.

After POS tagging the probability of a POS tag coming after any other adjacent POS tag is calculated. Take again the sentence in the previous example:

He is playing.

After the POS tagging, the sentence becomes

"He/pps is/bez playing/vbg ./.".

where the token after the '/' is the POS tag of words.

Now we can use the tag sequence to calculate the probability of the sentence. To calculate the probability of the sentence there should be the sentence start(**<start>)** sentence end(**</End>**) to show the start and the end of the sentence**.**

P (pps bez vbg .) = P (pps | <start>) *P(bez | pps) * P(vbg | bez) * P(. | vbg) * p(</End>).

In the statistical approach, two different ways can be used to achieve the goal of correcting grammar [5]. The first one is done by comparing the sentence to be corrected with sentences in the corpus directly. The second way is driving a set of grammar rules from data stored in the corpus.

3.3.4. Hybrid Approaches

A Hybrid approach is the combination of rule based approach and statistical approach [15]. This approach helps the system to achieve high efficiency and robustness [18]. High efficiency and robustness are obtained by taking the good qualities of the rule-based and statistical approaches [15].

3.4. Tasks in Grammar checking

3.4.1.Tokenization

Tokenization is the process of splitting down given text into sentences and then into tokens if necessary. Tokens are every item including word, number, punctuation, abbreviation, etc in a sentence [8].

The main challenge of tokenization is the detection of the sentence and word boundaries. For example, in the English language, the period (.) can be used for several tasks. It can be used as sentence end marker, as a decimal delimiter, or to mark abbreviations [8]. Period in Afaan Oromo is also used for marking end of sentence, as a decimal delimiter or to mark abbreviations

This causes problems in determining the correct sentence end marker. To solve this problem a list with abbreviations can be used to distinguish abbreviations and the sentence end marker. But if we consider a sentence with an abbreviation at the end of a sentence, there is an ambiguity which needs to be resolved.

Another challenge in tokenization is the detection of word boundaries. Usually words are delimited by white space characters. There are some exceptions, e.g. multiword expressions, which need a different treatment. An example for this is the expression " to kick the bucket" taken from [8]. If the space character is used as the word delimiter, the result is four individual tokens that are tagged individually. But if we are looking for the idiomatic meaning (to die), the whole expression needs to be tagged as a verb.
In Afaan Oromo tokenizing compound words is challenging. This is because Afaan Oromo compound words are mostly words that are made up two different words separated by space. Since there is space between the two words the tokenizer considers them as separate word.

This is different in other languages like Chinese which use logographic symbols representing words. Chinese and Japanese use the period as a sentence delimiter, words are not necessarily delimited by white space unlike English and Afaan Oromo. One can see here that tokenization is not an easy task and has potential influence on the accuracy of any method, like tagging, which depends on this task.

Clitics can be interpreted as on its own, or it can be split and interpreted as two words during tokenization. For example, the word can't can be interpreted as one token (can't) or as two

tokens (can and not). There are several similar cases, where more than one possibility can make sense. One of such case is the handling of numbers. The format of numbers can vary greatly, e.g. large numbers can be written without any punctuation as 1124000.

3.4.2. Parts Of Speech Tagging

Part-of-speech (POS) tagging is the act of assigning each word in a sentence a tag that describes how that word is used in the sentences [34,35]. The most common POS include noun, pronoun, verb, adjective, adverb, preposition, conjunction, interjection and the like [15]. A POS tagger attempts to assign the corresponding POS tag to each word in a sentence, by taking into account the context in which this word appears [8].

For example, given the sentence

“Boontun kaleessa deemte
Bontu yesterday went

The tagging result of the sentence using a tag set used by [34] can be given as

“Boontun”\NN “kaleessa”\AD “deemte”\VV

where NN stands for noun tag, AD for adverb tag and VV for verb tag.

In the above example, the words in the sentence are tagged with appropriate lexical categories of noun(NN), adverb(AD) and verb(VV) respectively. The process of tagging takes a sentence as input, assigns a POS tag to the words or tokens in a sentence or in a corpus and produces the tagged text as output.

POS tagging has great advantage for grammar checking [8]. It helps to understand grammar mistakes that can be made and figure out how to correct them. It is also very advantageous in grammar checking as it is very difficult to include each word of the language in the manually constructed rules or corpus.

3.4.3. Morphological Analysis

Words are the basic elements of a language [16]. They are formed from smallest meaningful unit called morphemes which include the root and bound morphemes [15]. Afaan Oromo has a very rich morphology [1]. It has the basic features of agglutinative languages where all bound forms (morphemes) are affixes (prefix, suffix and infix) [1]. For example **“kufte”** has two meaningful parts (morphemes). **“kuf”** the root and **“te”** the bound morpheme.

Morphological Analysis is the process of finding the morphemes of the word and providing grammatical information for the word based on the identified morphemes [15, 16]. For example, the word **"kufte"** has the morphemes **"kuf"** and **"te". "kuf"** is the root word and ***"te"*** is a bound morpheme that indicates a feminine gender.

A program or system that performs morphological analysis is known as morphological analyzer. It detects the morphemes and provides grammatical information for the morphemes of an input word.

In morphologically rich languages like Afaan Oromo, morphological analysis is more crucial. Morphological Analysis is very important for many higher level natural language applications such as grammar checker, machine translation, speech recognition, information retrieval and the like[16]. For grammar checker, morphological analysis provides detailed grammatical information of words in a given sentence.

3.4.4. The statistical language model

A statistical approach does not need language resources like handcrafted grammatical rules, except for a tagged corpus to train the language model (LM)[15]. Statistical language modeling (SLM) is the attempt to capture regularities of natural language for the purpose of improving the performance of various natural language applications [41]. In statistical language modeling, large amounts of text are used to automatically determine the model's parameters [42]. A language model is used to estimate the probability distribution of various linguistic units, such as words, sentences, and whole documents [43]

Statistical language modeling is crucial for a large variety of language technology applications. These include speech recognition (where SLM got its start), machine translation, document classification and routing, optical character recognition, information retrieval, handwriting recognition, spelling correction, grammar correction and any more [41].

There are different techniques to develop the statistical language model. These include n-gram models, decision tree models, linguistically motivated models, Exponential models, adaptive models. In this study n-gram models is the only reviewed technique [41].

An n-gram language model is one of the major statistical language modeling methods used for different NLP applications [41, 42]. It is usually formulated as the distribution of probability (P)

over string (S) that attempts to reflect how frequently S occurs as a sentence [43]. The language is modeled by calculating the probability of subsequence of N tokens in sentence, where token can be words, phonemes, characters, morphemes, syllables and N can be range from 1 to many [19]. The subsequence of tokens can be formed by considering N adjacent or non-adjacent units extracted from the training text [8].

Consider the following Afaan Oromo sentence.

Baratotni hundinu qoruumsa darbuuf sirritti qo'atan
(All students studied hard to pass the exam.)

Considering the above sentence the two tokens (bi-gram) subsequences of the sentence can be formed are as follows

"Baratotni hundinu",
"hundinu qoruumsa",
"qoruumsa darbuuf",
"darbuuf sirritti"
"sirritti qo'atan".

In general, given the sentence with N_t number of tokens and N-gram model, the maximum number of subsequences occurring in the sentence can be calculated as:

$$N_s = N_t - (N-1)$$

where N_s is the total number of subsequence in the sentence, N_t is the number of token in the sentence and **N** is the number of tokens in the single subsequence [15].

Depending on the number of tokens used in the subsequence, different names are given to n-gram language model. Uni-gram, Bi-gram, Tri-gram, Quad-gram and Penta-gram for **N** is equal to 1, 2, 3, 4 or 5 respectively.

Let us introduce this model by considering the case **N**= 2; this model is called bigram language model. First, we note that for a sentence ***s*** composed of the words $w_1 \ldots w_m$, the bi-gram language model of the sentence *s* is given as [15, 42, 43].

$$P(s) = P(w_1 w_2) * P(w_2 w_3) \ldots . * P(w_{m-1} w_m) = \prod_{i=1}^{m} P(w_i w_{i+1})$$

Consider the sentence as an example

He is clever.

The bi-gram language model of the sentence can be given as:

$$P(He\ is\ clever) = P(He|<start>) * P(is|He) * P(clever|is) * P(<.>|clever) * P(<end>|.)$$

$P(W_2|W_1)$ **is** the probability of the sequence which is calculated by

$$P(W_2|W_1) = \frac{C(W_1W_2)}{C(W_1)}$$

where W_1 and W_2 are words and C is their respective count

An n-gram language model is basically used for Statistical grammar checker [42, 43]. This is to enable the statistical grammar checker find the probabilistic occurrence of sentences in the corpus.

3.5. Related works

3.5.1. Rule base checkers

Rule based is the most common grammar checking approach [1,15]. Grammar checkers for different languages have been developed using this approach. One of these grammar checkers is the rule based Afaan Oromo grammar checker by Debela[1]. Debela[1] used manually constructed rules in order to identify the grammatical correctness of a text. Totally 123 rules were constructed for detecting the errors and suggesting possible solution.

This grammar checker has five components. The first component is the tokenizer module, in which the input text is split into a sentence. The second module, part of speech (POS) tagger, assigns each word into its POS tag. The stemmer module is the third module which accepts the tagged word and provides the root and affixes for the tagged word.

The fourth module, which is a grammatical relation finder, assigns grammatical relations between word classes. These include subject and verb, subject and adjective, main verb and subordinate verb in terms of number gender and tense. The agreement between word classes is checked based on the prepared rules. The agreements are between subject and verb, subject and adjective, main verb and subordinate verb in number, tense, gender and other causes. The rules take the affixes those which are produced by the stammer module in order to check the

agreement between words. Grammatical information of the words is constructed and presented using affix-rules in the language. The last module is a suggestion creating module which suggests the correct sentence options.

The rule-based Afan Oromo grammar checker was evaluated using precision and recall evaluation metrics. The precision and recall was calculated based on the counting made on the number of errors in the text, the number of errors detected by the grammar checker and the number of errors correctly detected by the grammar checker. The researcher states that the grammar checker has 89.89% of precision and 80% recall.

Incorrect POS tags were identified as the first reason for the false alarm. The POS tagger assigns incorrect POS tag for some words. The second reason was production of wrong affixes by the stemmer. As most of the rules are constructed for simple sentences, most of the false flags are on complex and compound sentences.

Another grammar checker developed using rule based approach is by [2]. The aim of the work was to develop a grammar and style checker system, for English language, which can be used both as standalone and integrated in word processor systems.

The style and grammar checker described in the paper takes a text and returns a list of possible errors. To detect errors, each word of the text is assigned into its part-of-speech tag i.e. noun, verb, determiner, adjective, adverb and the like. Many words can have different POS tags depending on their context. If the number of POS tags increases, it will be more difficult to handle for the algorithm to find the right tag for a given occurrence of a word. For example a word may be tagged as a verb or a noun depending on the context of the sentence. After POS tagging, each sentence is split into chunks, e.g. noun phrases.

The grammar checker has 42 pre-defined grammar rules, which are simply a sequence of tokens. The text, after tagging and chunking, is matched against all these pre-defined error rules. The mal-rules are defined in XML. The rules describe errors as patterns of words, part-of-speech tags and chunks. A rule defines incorrect sequence of token. If a rule matches, the text is supposed to contain an error at the position of the match. Each rule also includes a description about the error.

The evaluation for the English style and grammar checker was not based on precision and recall values. As the researcher explained in the paper, a corpus of yet unedited text with all errors marked up is required to calculate for meaningful precision/recall values. However, the resource of such kind was not publicly available when the research was conducted. Therefore, to evaluate the system two corpora were prepared; a corpus that only contains sentences with errors and a corpus that contains very few errors (BNC's texts).

The BNC's texts have been proof-read. When the checker is evaluated with this text, it claimed 11 errors in 74,900 sentences. However, most of these errors are false alarms. The researcher stated that the reason for the false alarm might occur: the text was incorrectly split into sentences, a word has been assigned an incorrect part-of-speech tag, or the rule simply is not strict enough and triggers too often.

The checker also evaluated using another corpus which contains sentences with errors and compared with Microsoft Word 2000 checker. The grammar checker detects 22 errors whereas MS-word detects 29 errors. But, this style and grammar checker identifies errors in the same sentence when the errors are independent of one another. This kind of error could not be detected in MS-word 2000. MS-word 2000 can not specify more than one error per sentence.

Furthermore, Grammar Checking with Dependency Parsing [21] is another grammar checker developed using rule based approach. It is a possible extension for style and grammar checker. In this research the researcher identifies the gap in the style and grammar checker tools and tries to fill the gap. In the style and grammar checker language tool, the rules do not consider the words between any two token sequences [21]. But in[21] it is tried to consider the words between any two token sequence.

3.5.2. Statistical grammar checker

There are different grammar checkers developed using this approach. Amharic grammar checker is the one [15]. In the attempt by [15], two grammar checker approaches have been tested. This grammar checker has been tested for both simple and complex sentences.

The statistical Amharic grammar checker uses n-gram and probabilistic methods to check a grammatical correctness of Amharic sentence. The patterns and their probabilities of occurrence are automatically extracted from the training corpus and stored in a repository. Sentence probability can be calculated using these patterns and probabilities. Then, probability of the

sentence and specified threshold are used to determine the correctness of the sentence. The corpus, both for training and testing the grammar checker are compiled from a manually tagged Amharic language.

The performance of the grammar checker is evaluated using two test cases. The first test case is done on simple sentences. In this test case, the performance of the grammar shows 90.38% % of precision and 17.12% of recall.

The statistical Amharic grammar checker is tested using complex sentences in the second test case. In this test case, 13.71% of the errors are detected. The false alarms are due to the incomplete grammatical rules and quality of the statistical data. The accuracy of morphological analyzer also affects the grammar checking result.

N-gram based Statistical Grammar Checker for Bangla and English [10] is another grammar checker developed using the statistical approach. The method checks the probability of the N words occurring together in the corpora. For example, when N= 2, which is bigram case, the system checks how often each pair of words occurs in the corpora and calculated the probability. Using the calculated probability of pair of words in the probability of the sentence to be tested is calculated. If the probability of the sentences is above some threshold i.e zero in this study, the sentence will be considered as grammatically correct. Probability of a sequence becomes zero when two or more consecutive tags cannot be fit together (or in other word they are incompatible). If one of the sequences probabilities is zero, the sentence probability will become zero (property of zero multiplication).

To avoid zero probability, in n-gram based statistical grammar checker, first all the words in a sentence are assigned to their POS tag. Then, determine the sequence probability of word tags. Sentence probability will be calculated based on the word sequences probability. Finally, if the sentence probability is above a given threshold, the sentence is correct otherwise it is incorrect.

The performance of n-gram based grammar checker is tested using manually and automatically tagged corpus. The paper does not clearly state which corpora is used to train and test the system. Using manually tagged corpus, the performance of the grammar checker for English is 13% (detected 424 sentences as correct, out of 811 correct sentences). Using manually tagged corpus, the grammar checker's performance is 43.7% (detected 203 sentences out of 378 correct sentences) in Bangla.

The performance of the grammar checker also tested using 32 automatically tagged sentences for Bangla. Then, the grammar checker produced result with lower performance which is 38% correct result.

The researcher stated that one of the reasons for the low performance of the grammar checker is the performance of POS tagger. Most of the errors that could not be detected by the grammar checker are agreement errors (i.e. a mismatch in number, person…).
To overcome this problem, a tag set with agreement feature is required. The grammar checker works well on simple sentence than on compound. The grammar checker shows lower performance on corpus which contains large compound sentences.

LISGrammarChecker: Language Independent Statistical Grammar Checking is another grammar checker developed using the statistical approach[8]. The researchers consider a database containing all bi, tri, quad, and penta grams of tokens and of tags of a language. This database is built up in a training phase. When the database is built up, statistical information was gathered. This statistical information is used for detecting errors and proposes error corrections.

Every sentence is checked for all n-grams of tokens and n-grams of tags and different error points are given if a specific n-gram is wrong. The researcher explained this approach with an example using trigrams. Consider the following sentence

"All modern houses are usually very secure."

In the LISGrammar checker the researcher first extracts every trigram sequences out of the above sentence.

"All modern houses",
"modern houses are",
"houses are usually",
"are usually very",
"usually very secure"

These are the trigram sequence that can be extracted from the sentence. Then these trigram sequences are saved into the database for training.

For checking the grammatical correctness the sentence **“All modern houses is usually very secure.”** which contain wrong verb tense, first the trigram sequence of sentence is extracted and the existence of every trigram sequence in the database is checked. In this case, the trigram **“houses is usually”** is not found in the database, and thus an error is assumed. Additionally, this trigram analysis is also done for trigrams of tags, not only for the trigrams of the tokens themselves.

LISGrammarChecker aims to be completely language independent. It achieves the language independence by the opportunity to train the database for every natural language. The checking is then done for the specified language. This grammar checker only uses statistical data for grammar checking. This facilitates all languages to be used in the same program. And this program allows also languages where it is impossible or where the language is not spread enough to write a set of rules for grammar checking.

3.5.3. Hybrid grammar checker

Granska is a hybrid grammar checking system that uses surface grammar rules to check grammatical constructions in Swedish. The system combines probabilistic and rule-based methods to achieve high efficiency and robustness, which is a necessary prerequisite for a grammar checker that runs in real time in direct interaction with users [18]. Using error detection rules, the system can detect a number of Swedish grammar problems and suggest corrections for them.

CHAPTER FOUR

DESIGN AND IMPLEMENTATION

4.1. Introduction

There are different approaches to grammar checking. Rule based, Statistical based and Hybrid approaches are the most widely used ones [8, 10, 15]. In this study Afaan Oromo grammar checker is developed using the statistical approach.

The statistical approach generally checks grammar of a language by calculating the probability of a sentence based on word sequence statistics. The probability is calculated by comparing the sequence of words in an input text in which errors should be detected with the sequence of words in the text stored in the corpus. To allow the comparison, both the input text and the text in the corpus need to be analyzed with respect to their n-grams (bi-gram, tri-gram, etc).

Based on the calculated value of the probability, the sentence is categorized as correct or incorrect. If the probability of a sentence is greater than a certain threshold settled, the sentence is considered to be correct. Otherwise, the sentence is considered as grammatically incorrect.

In statistical approach, two n-gram techniques can be used [8]. These are Token based n-gram check and POS tag based n-gram check. In this study, we have implemented the two techniques for checking the correctness of a given Afaan Oromo sentence and compare their performance in detecting grammatically incorrect sentences. In the following sections the two techniques of statistical aproach are discussed in detail.

4.2. The Token n-gram technique

The token n-gram technique is one of the statistical approach techniques used in this study for checking whether a given Afaan Oromo sentence is grammatically correct or not. This technique uses a sequences of tokens stored in the corpus for checking the grammatical correctness of input sentence. This means, probability a sequences of tokens in the input sentence are calculated by using a sequences of tokens in the training corpus. If the probability of the input sentence is greater or equal to the threshold settled in order to determine the correctness of a sentence, the sentence is considered as correct. Otherwise the sentence is incorrect [8,10].

Generally, two basic workflows (modules) are performed for checking the grammatical correctness of Afaan Oromo sentence in this technique. The first one is the training module, which is used to train the system by filling the database with statistical data from the prepared corpus for training the system. The second is the testing module, which is used for checking grammatical correctness of a given Afaan Oromo text. The system architecture is presented in Fig 4.1.

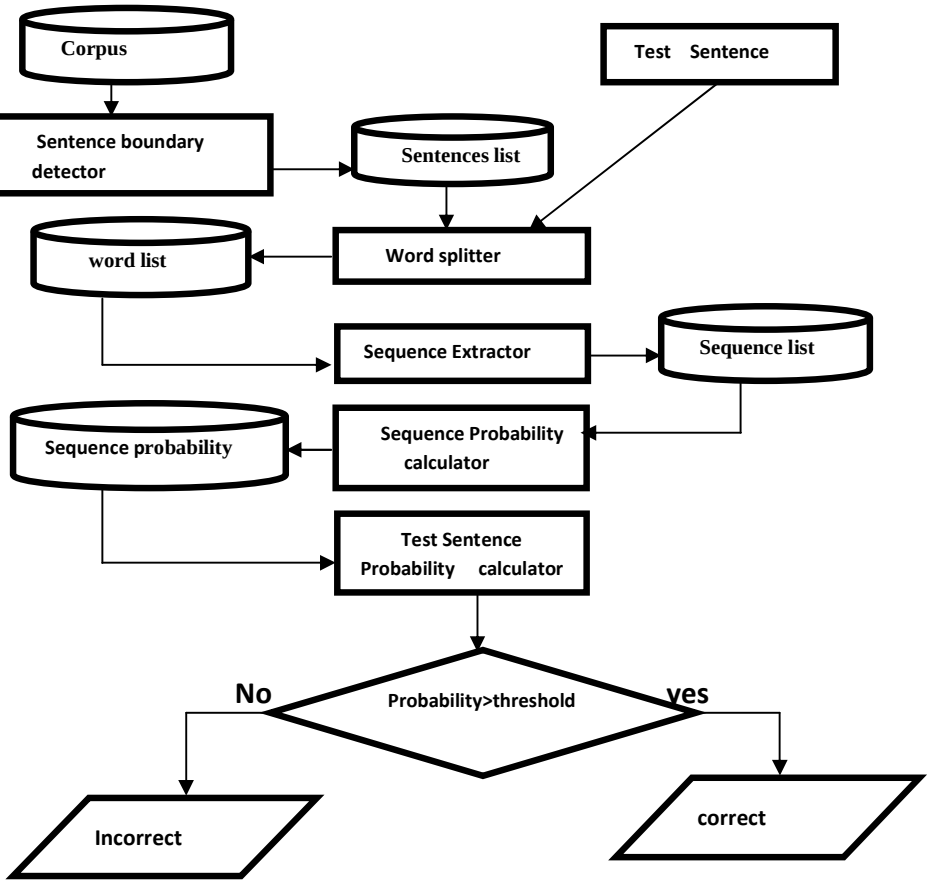

Fig 4.1 Architecture of Token N-gram check

4.2.1 Training Module

The training module is used to train the system by filling the database with statistical data from the given corpus. The training module has the following components

A. **Sentence Boundary Detector**

The sentence boundary detector splits the text in the corpus into sentences by using punctuation marks used to end the sentences in Afaan Oromo language. Three punctuations marks may be used to end sentence in Afaan Oromo [27,41]. These are full stop (.), question marks (?) and exclamation mark (!). Using these punctuation marks as split indicators, the texts in the corpus are split into sentences. The listing 4.1 presents the sentence boundary detector algorithm.

```
Input: Training corpus
Algorithm:
Open the corpus
  Sentence =empty
  Read word in the corpus at a time
       If the word ends '.' or '?' or '!'
            Sentence=Sentence +word
            Add the sentence to repository
            Sentence =empty
       Else
            Sentence=Sentence+ word
Output: list of sentence in a repository
```

Listing 4.1 Sentence boundary detector algorithm

B. Word boundary detector

After the input text is split into sentences, each sentence is kept in a temporary repository. The word boundary detector accepts a single sentence from the repository at time and then split the sentence to a set of words (tokens). The word splitter uses space as word delimiter to split the sentence. The splitted sentences are then stored to repository for further processing. The listing 4.2 presents the word boundary detector algorithm. Splitting compound word is not covered in this thesis.

Input: A list of sentence
Algorithm: *For all sentences in the repository* *Read sentence from the repository* *Sentence =sentence·split()* *Add the words to repository*
Output: A list of words in the repository

Listing 4·2 word boundary detector algorithm

C. N-gram Sequence extractor

N-gram sequence extractor is used to extract N sequence of tokens in the sentence. The value of N can be 2 or 3 for bigram sequences and trigram sequences respectively. For the purpose of this study SRILM language model is used in order to extract the sequence [44]. The N-gram sequence extractor accepts one sentence at a time from sentence repository and extracts the bi-gram or tri-gram sequence of tokens for the sentence. The bi-gram sequences of sentence are extracted by taking two successive tokens in that sentence. The N-gram sequence extractor adds <S> and </S> for all sentences to show the start and the end of the sentence before extracting the sequence.

For example the bigram(N=2) sequence list of the sentence

“Ani kaleessa mana barumsaa deeme.”(I went to school yesterday.)

First the sequence extractor adds sentence start and end to the sentence like the following.

“<S> Ani kaleessa mana barumssaa deeme.</S>”

By taking the two adjecnet tokens in the sentence, the bigram sequence list will be

[‘<S> Ani’, ’Ani kaleessa’, ‘kaleessa mana’,

’mana barumsaa’ ,’barumsaa deeme’,’deeme </S>]

The tri-gram sequences are extracted in the same way as bi-gram sequence extraction. The only difference is in the case of tri-gram sequence extraction three successive words are considered. Again take the above sentence as an example, the trigram sequence list of the sentence can be extracted as follows.

“Ani kaleessa mana barumsaa deeme.”(I went to school yesterday.)

First the sequence extractor adds sentence start and end to the sentence.

"<S> Ani kaleessa mana barumssaa deeme</S>"

Taking the three adjecent tokens in the sentence, the trigram sequence list will be:

'<S> Ani kaleessa', 'Ani kaleessa mana', 'kaleessa mana barumsaa',

'mana barumsaa deeme' ,'barumsaa deeme </S>'

Both the tri-gram sequences and bi-gram sequences are stored in the temporary repository for further processing. The algorithm for N-gram sequence extractor is given below in Listing 4.3.

```
Input: list of words sentences
Algorithm:
 Read sentence in the repository
   Trigram_sequence_list=empty, Bigram_sequence_list=empty
   j=0
   For  sentence in the repository
     Add <S> and </S> at start and end of the sentence respectively
     i=0,x=0
    For N-2 number of words  in the sentence
        Trigram_sequnce=sentence[j][i] + sentence[j][i+1] + sentence[j][i+2]
        Add Trigram_sequence to Trigram_sequence_list
        i=i+1
     For N-1 number of words  in the sentence
        Bigram_sequence= sentence[j][x]+ sentence[j][x+1]
        Add Bigram_sequence to Bigram_sequence_list
        x=x+1
    j=j+1
   Add Trigam_sequence_list to repository
   Add Bigram_sequence_list to repository
Output: A list of Bigram and Trigram sequence in the repository
```

Listing 4.3 N-gram sequence extractor algorithm

D. Sequence probability calculator

Sequence probability calculator is the end task of the training module. It starts by accepting the trigram and bigram sequence of the sentences from the repository. To calculate the probability, the trigram and bigram sequence vocabulary is created. That is non repeatable sequences for both trigram and bigram are stored in the repository differently. For each sequence of tokens in the vocabularies the occurrence of the sequence is counted and stored temporarily for probability

calculation. The probability calculation for this study is based on trigram `language model. The trigram language model is a probabilistic model that calculates the probabilistic occurrence of a token with the previous 2 tokens sequence for a given sentence [15]. The probability of the given trigram sequence is given by:

$$P(W_3|W_1W_2) = \underline{Count(W_1W_2W_3)}/Count(W_1W_2)$$

P is the probability of $W_1W_2W_3$ sequence given the sequence W_1W_2 and W_1, W_2, W_3 are the first, second and third words(tokens) respectively.

The probability of the sequence is always between zero and one inclusive. After the probabilities of all trigram sequences are calculated they are stored in the repository with their respective sequences. The algorithm for sequence probability calculator is presented in Listing 4.4.

```
Input: a list of sequence
Algorithm:
        Trigram_Sequence_Vocabulary=empty
        Bigram_sequence_vocabulary= empty
         Read the bigram and trigram sequence from the repository
                If   bigram _sequence in Bigram_sequence_vocabulary
                     Read another bigram_sequence
               Else
                        Add bigram_sequence to Bigram_sequence_vocabulary
         If   trigram _sequence in trigram_sequence_vocabulary
                     Read another trigram_sequence
         Else
                     Add trigram_sequence to trigram_sequence_vocabulary
        Read trigram sequence from trigram_sequence_Vocabulary
        P(trigramsequence)=count(trigramsequence)/count(bigramsequence)
        add the probability and trigram sequence to the repository
Output: a list trigram sequence and their probability
```

Listing 4.4 sequence probability calculator algorithm

Where, count of the sequences is obtained from the SRILM language model in the sequence extractor module.

4.2.2. Testing module

The testing module is the second basic component in token n-gram technique. It is used to check whether a given Afaan Oromo sentence is grammatically correct or not. This module starts by accepting a single sentence from user as input. It has different sub components that work different function for checking the grammatical validity of input sentence. Most of the components used in the testing module are similar to the component used in training module. For example the word boundary detectors, sequence extractors are similar in both training and testing module.

A. Sentence probability calculator

The sentence probability calculation module calculates the probability of input sentence based on tri-gram language model. The probability calculator accepts the trigram sequence lists of input sentence stored by n-gram sequence extractor in the testing module and the trigram sequence list and their probabilities stored by sequence probability calculator in the training module.

The probability for the trigram sequence list of the input sentence is calculated by matching the sequence with the sequence of training sentences stored in the repository. If a match is found for the sequence, the probability of that matched sequence is assigned to the sequence of input sentence. Otherwise, the probability of the sequence is assigned to the probability of unknown words. The probability of input sentence is calculated by multiplying the probability of all trigram sequences in the sentence. List 4.5 presents the sentence probability calculator algorithm.

```
Input:  ·A list of triagram sequence from the repository
        ·A list of triagram sequence of the input sentence
Algorithm:
        Read Trigram_sequence_list of input sentence
       Read Trigram_sequence_list and their probability in the corpus
       For all Trigram_sequence in the Trigram_sequence-list of input sentence
           if triagram sequence of input sentence found in the repository
                Trigram_sequenceprobability= probability of the sequence in the corpus
          Else
                Trigram_sequenceprobability=1/vocabulary size
              Probability of sentence=multiplication of tri-gram sequence probability
Output: probability of input sentence
```

Listing 4·5 sentence probability calculator algorithm

The following example shows how the probability of a test sentence is calculated.

Example: Assume the following statistical data exist in the training corpus

Trigram sequence	Count	Bigram sequence	Count
(<S> Tolaan hiriyaa	2	(<S>Toolan	4
Tolaan hiriyaa isaa	1	Tolaan hiriyaa	2
hiriyaa isaa waliin	2	hiriyaa isaa	5
isaa waliin taphachaa	1	isaa waliin	4
waliin taphachaa jira	3	waliin tapachaa	5
tapachaa jira </S>	1	taphachaa jira	2

Table 4.1 a sample statistical data taken from training corpus for calculating a probability of sentence

The probability of the sentence

'Tolaan hiriyaa isaa waliin taphachaa jira.' can be calculated as:

P(<S>Tolaan hiriyaa isaa waliin tabachaa jira </S>)=

***P(hiriyaa|<S> Tolaan) * P(isaa |hiriya Tolaan) ***

***P(waliin |isaa hiriyaa) * P(taphachaa|waliin isaa) ***

P(jira|taphachaa waliin) * P(</S>| jira taphachaa)

Where P is the probability of sequence which is given by

$$P(\text{hiriyaa}|\text{<S> Tolaan}) = \frac{\text{Count(<S> Tolaan hiriyaa)}}{\text{Count (<S>Toolan)}} = \frac{2}{4} = 0.5$$

$$P(\text{isaa }|\text{hiriyaa Tolaan}) = \frac{\text{count(Tolaan hiriyaa isaa)}}{\text{Count(Tolaan hiriyaa)}} = \frac{1}{2} = 0.5$$

$$P(\text{waliin }|\text{isaa hiriyaa}) = \frac{\text{count(hiriyaa isaa waliin)}}{\text{Count(hiriyaa isaa)}} = \frac{2}{5} = 0.4$$

$$P(\text{taphachaa}|\text{waliin isaa}) = \frac{\text{count(isaa waliin taphachaa)}}{\text{Count(isaa waliin)}} = \frac{1}{4} = 0.25$$

$$P(\text{jira}|\text{taphachaa waliin}) = \frac{\text{count(waliin taphachaa jira)}}{\text{Count(waliin tapachaa)}} = \frac{3}{5} = 0.6$$

$$P(\text{</S>}|\text{ jira taphachaa}) = \frac{\text{count(tapachaa jira </S>}}{\text{Count(taphachaa jira)}} = \frac{1}{2} = 0.5$$

Finally the probability of this sentence is by the product of the above sequence probability

P(<S>Tolaan hiriyaa isaa waliin tabachaa jira </S>)=

0.5*0.5*0.4* 0.25*0.6*0.5=0.0075

B. Grammar checking

The main aim of this module is to determine whether the input sentence is grammatically correct or not based on the result of sentence probability. The sentence is grammatically correct, if the probability of the sentence is greater than a threshold. Otherwise the sentence is considered incorrect.

4.2.3 Problem in Token N-gram check

1. Token N-gram check needs huge corpus for checking the grammar of language [8]. But, there is no well organized corpus that contains a list of all Afaan Oromo sentences.
2. This technique does not work well if the sequences of tokens in a sentence do not exist in the training corpus [8]. This is because if the tokens in the sequence do not exist in the training corpus they are considered as Out of Vocabulary (OOV), which decreases the probability of the sequence. Therefore, some grammatically correct sentence may be adjudged to be incorrect if the sequence of token in the sentence is not found in the corpus.
3. It is difficult to identify what is wrong with incorrect sentence [8]. Because, no linguistic information about the word is used or it difficult to know the grammatical relations among words.

4.3. Tag N-gram technique

Tag n-gram check is the second technique used to check the grammatical correctness of Afaan Oromo sentences. As explained in section 4.2.3 the token n-gram method does not work well if the sequence of tokens in a sentence does not exist in training corpus. To overcome the problem in token n-gram method, the tag n-gram method can be used. In this method POS tag and linguistic information of each token are used instead of token for checking the grammatical correctness of Afan Oromo sentences.

The Architecture of tag n-gram check technique has two main modules similar to token N-gram check. These are the training module and the testing module. The training module is used to train the system by using training set prepared for training the system.

The testing module is used to check whether the sentences in the test set are correct or not. Both the training and the testing modules in tag n-gram check are discussed below.

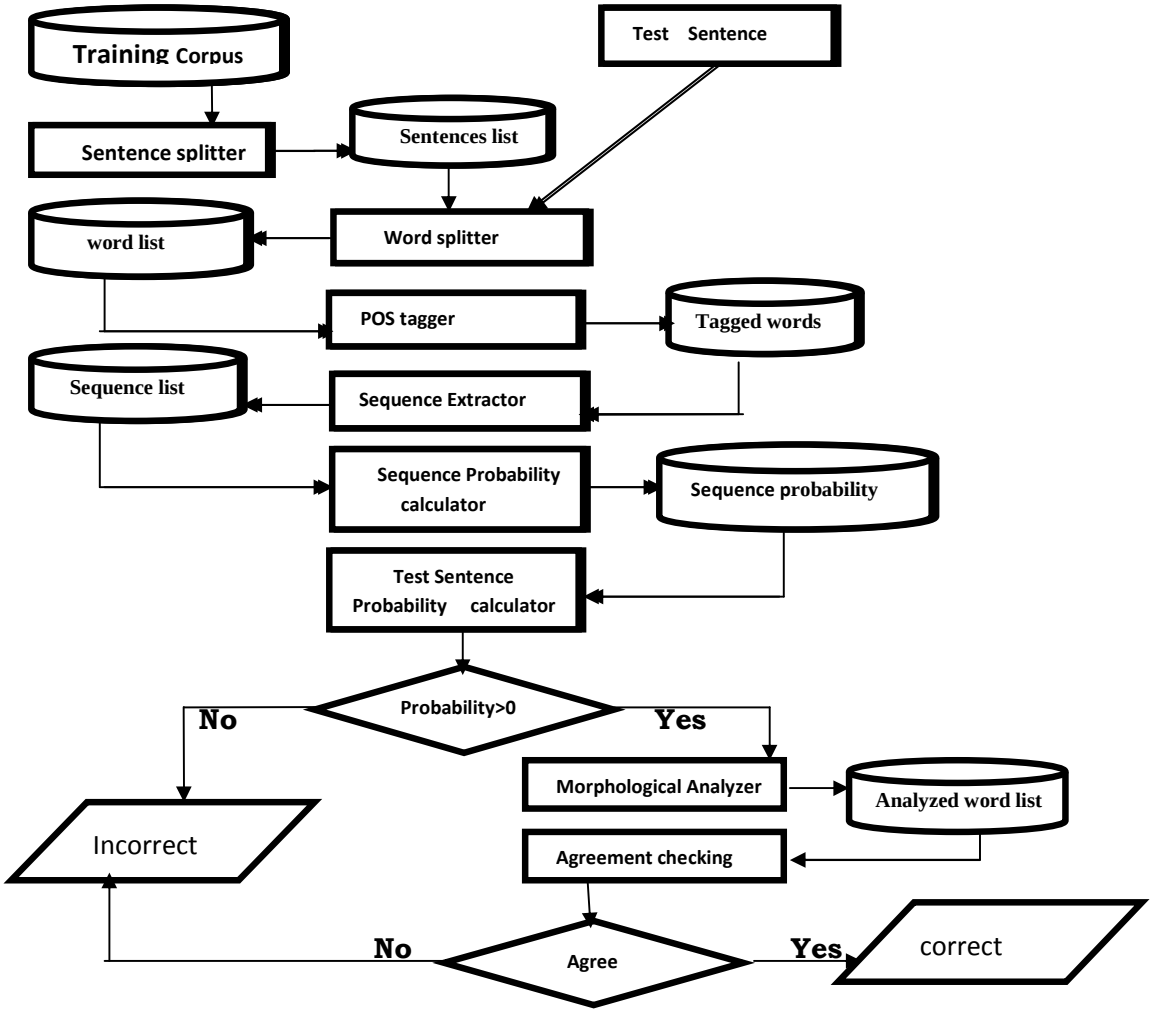

Figure 4.2 Architecture of tag N-gram check

4.3.1. Training Module

The training module in tag n-gram method is similar with the training module in token n-gram method. It is used to fill the database with statistical data by accepting the prepared Afaan Oromo training dataset. In the following subsections, the algorithms not covered elsewhere are presented.

A. POS Tagger

POS tagger is one of the NLP applications which assign words to its part of speech [1, ,34, 35]. In order to maximize the performance of grammar checker applications, it is very important to include POS tagger as a pre-process [15]. In this study each word in both training and test dataset is assigned to its POS tag.

This is very helpful as processing all the words are very difficult to handle [8]. The reason for this is, a word may not occur at all in the corpus or occur only once or very few times. These make calculating probability difficult and unreliable.

In this study, the researcher uses TnT POS tagger to assign words to their part of speech. TnT is the short form of *Trigrams 'n' tags.* It is a very efficient statistical part of speech tagger that is trainable on different languages and any tag set can be used [20]. Tnt tagger requires manually tagged corpus for training the tagger. For this study, the researchers have used the manually tagged corpus by Getachew Mamo [35] as the training data for the tagger. The Corpus contains 168 sentences and 18 tags in the tag set. Tnt tagger incorporates several methods of smoothing and handling of unknown words and it is optimized or speed [20].

After each word is assigned to their part of speech using Tnt tagger they are stored in the temporary repository for further processing.

B. Sequence extractor

Sequence extractor is a module used to list the POS tag sequences for all sentence used in both training and testing dataset based on language model used. The language model used for SAOGC is trigram language model. This language model requires the tri-gram and bi-gram sequence for modeling the language. Therefore, the bi-gram and tri-gram sequences of POS tags are extracted from the repository created in section 4.3.1 A.

The sequence extractor first accepts tokens and their POS tags from the repository for a single sentence. The extractor then separate tokens from their POS tag and store them in the temporary repository. The extractor repeats these two steps for all sentences in the corpus. From the temporary repository the bi-gram and tri-gram sequence of token information is extracted for each sentence and stored in the repository.

The algorithm for sequence extractor in tag n-gram method is the same with the sequence extractor in training module of token n-gram method. The only difference is the element in the sequence. In token n-gram check the sequences contain collection a words or tokens whereas in tag n-gram methods the sequences are collection POS tags.

C. The sequence probability calculator

The sequence probability calculator calculates the probability of each trigram sequence based on tri-gram language model. It reads one trigram sequence at time and calculates the probability for that sequence by counting the occurrence of that sequence and its preceding two sequences in the repository.

The calculated probability and the sequence are stored in the repository for testing module. The probability calculator then selects the second trigram sequence and check if the probability of the sequence is calculated or not. If the probability is calculated for the sequence it jumps to the next sequence. Otherwise, the probability is calculated for the sequence. This continues until the end of the sequence list.

```
Input:  A list of POS tag
Algorithm:
    Trigram_Sequence_Vocabulary=empty, Bigram_sequence_vocabulary= empty,
    Trigram_sequence_count=empty, Bigram_sequence_count=empty
     Read the bigram sequence from the repository
            If   bigram _sequence in Bigram_sequence_vocabulary
                 Read another bigram_sequence
             Else
                    Add bigram_sequence to Bigram_sequence_vocabulary
                     Add the count of the sequence to Bigram_sequence_count
    Read the trigram sequence from the repository
          If   trigram _sequence in trigram_sequence_vocabulary
                Read another trigram_sequence
           Else
                    Add trigram_sequence to trigram_sequence_vocabulary
                  Add the count of the sequence to trigram_sequence_count
    Read trigram sequence from trigram_sequence_Vocabulary
        P(trigramsequence)=count(trigramsequence) / Count(bigramsequence)
          add the probability and trigram sequence to the repository
Output: A list of tag N-gram sequence in a repository
```

Listing 4.6 tag N-gram sequence extractor algorithm

The following example shows how to calculate the trigram probability of a tag sequence. Reading the sequence ***' <S><NN><VV> '***

```
P('<S><NN><VV>')=Count('<S><NN><VV>')/ Count('S><NN>')
```

The probability of the sequences are calculated like this and stored in the repository for testing module.

4.3.2.2 Testing Module

The testing module is used to test system by using the test set prepared for testing the system. The testing module accepts one sentence at a time and checks whether that sentence is grammatically correct or not. Different activities are performed for checking the sentence correctness or incorrectness. Most of the activities are similar to the activities in training module. Therefore it is not necessary to explain those activities.

A. Word class agreement checker

The word class agreement checker is the last sub module in tag n-gram technique. The word class agreement checker checks the agreement between word class in number, gender, etc. The word class agreement checker starts agreement checking if the probability of the sentence is greater than the threshold. Otherwise, the sentence is determined as incorrect and the incorrectness of the sentence can be adjudged as incorrect word order.

However, if the sentence probability is greater than the threshold some of the words are analyzed using the morphological analyzer in order to check the agreement between the words. Since there is no available Afaan Oromo morphological analyzer the researchers tried to develop a rule based simple morphological analyzer for the language. The developed morphological analyzer can only analyze words that are noun, verb, pronoun, adjective part of speech. It reads one word and its POS tag at a time from the repository and analyze the word if it's POS is either of Noun, verb, pronoun or adjective.

In general the developed morphological analyzer analyze the above words in their number description, gender description and person descriptions. The number description result may be plural, singular or unknown. The gender description result may be masculine, feminine or unknown. The person description is only for verb and pronoun. The result of person description may be first person, second person or third person.

After the words are analyzed in terms of the above description the word class agreement checking module checks the agreement between word class in number, gender and person. If the words are agree in all descriptions the sentence is considered to be correct. Otherwise, the sentence is incorrect and the disagreement is the incorrectness of the sentence.

CHAPTER FIVE

EXPERIMENTAL RESULTS AND DISCUSSION

In this chapter we will present the result of experiments undertaken to evaluate the performance of the two statistical approach techniques (Token N-gram and Tag N-gram) used in this study. The experiments are conducted by training the techniques on the training dataset and testing them on the prepared testing dataset. The performances of these techniques and algorithms used in each technique are also evaluated.

5.2. Data collection

Natural language processing using statistical approach requires data in the form of corpora [41]. Corpora can be divided into two categories: annotated corpora and unannotated corpora. Unannotated corpora are simply large collection of raw text, whereas annotated corpora is a large collection of text which contains text with their additional information such as phonetic transcription, part-of-speech tags, parse trees, etc [41]. In this study the researcher has used both annotated and unannotated corpora for different purposes.

An unannotated corpus for this study is collected and compiled from different sources. Specifically it is collected from Oromiya Radio and Television Organization (ORTO) news, Challacha Oromia newspaper and Internet based sources like the Oromo section of Wikipedia. An unannotated corpus in this research is used for both training and testing the grammar checker.

An annotated corpus, which contains words with their part of speech tag, is also used in this study. This corpus is taken from manually tagged text used by [35]. The corpus is selected because there is no other available Afaan Oromo annotated corpus that contain words with their part of speech tags publically. The main use of this corpus in this study is for training the POS tagger module.

In general, the collected data are categorized in to two broad class of dataset purposively. The first class is training dataset which is used to train the system and the second class is the testing dataset which is used to test the performance of the grammar checker.

5.1.1. The Training dataset

The training dataset is compiled from the unannotated corpora collected from different sources indicated in section 5.1. The prepared training dataset contains a total of 508 sentences with a total of 9152 words. All the sentences included in the training dataset are grammatically correct sentences and are more of simple and complex sentences. The correctness of the sentences is checked with the help of linguistic experts of the language. These dataset is used to fill the database with statistical data. The statistical data are then used by the grammar checker to check grammatical correctness of the sentence in the test data set.

5.1.2. The Testing dataset

The testing dataset is another collection of data compiled from different sources and sentences created with the help linguistic experts purposively. The testing dataset contains a total of 85 Afaan Oromo sentences out of which 50 of the sentences are grammatically correct and 35 grammatically incorrect sentences. The incorrect sentences contain different types of error including word order and word class agreement. Additionally the sentences included in this dataset are simple and complex sentence.

Some of the sentences in this test dataset are taken directly from the training dataset purposively. This is to check the performance of grammar checker on identifying correct sentences. The test dataset are used for testing the two statistical techniques used in this study.

5.2 Performance evaluation

After the grammar checker has been developed it is good to evaluate the performance of the checker. To evaluate the performance Afaan Oromo grammar checker the following procedure is employed.

- ✓ The numbers of correct and incorrect sentences in the test dataset are counted manually.
- ✓ The test dataset is given to the grammar checker
- ✓ The number of errors found by the grammar checker and the number of false positives generated by the grammar checker were counted manually.
- ✓ Using these numbers performance evaluation metrics such as precision, recall and F-measure of the system were calculated.

Precision measures how many of the sentences flagged as incorrect by the checker are indeed erroneous. Mathematically it is given by

$$\text{Precision} = \frac{\text{Number of correctly flagged error}}{\text{Total number of flagged error}} * 100$$

Recall measures how many of the errors in a text are found, i.e. how complete the grammar checker is. Mathematically it is given by

$$\text{Recall} = \frac{\text{Number of correctly flagged error}}{\text{Total number of error that occur in the text}} * 100$$

F-measure is the average of precision and recall and it can be calculated as

F-measure = 2 * Precision * Recall / Precision + Recall

5.2.1. Performance of the algorithms

In this part of the study, the performances of the algorithms used in developing statistical Afaan Oromo grammar checker are discussed. As it is indicated in chapter four, two statistical techniques are used to develop Afaan Oromo grammar checker. These are Token N-gram and Tag N-gram techniques. The two techniques use a set of algorithms in order to check the grammatical correctness of Afaan Oromo sentences.

Since the performance of grammar checker significantly depends on the output of those algorithms, evaluating the performance of those algorithms is essential [41]. Therefore, before evaluating the performance of the overall system the performance of the algorithms was evaluated. In this study the performance of the following algorithms is evaluated.

A. Sentence Boundary detector(SBD)

The sentence boundary detector accepts training dataset as an input and returns a list of sentences as an output. The performance of sentence boundary detector is not evaluated on the test dataset since the grammar checker accepts one sentence at a time as an input for testing a grammatical correctness of a sentence. Table 5.1 presents the performance of sentence boundary detector on the two training dataset.

dataset	Number of sentences	Correctly detected	Incorrectly detected	Total detected	precision	recall	F-measure
Training	508	495	24	519	95.3%	97.4%	96.3%

Table 5.1 Performance of sentence boundary detector

As it is shown in Table 5.1 the performance of SBD on the training dataset is a recall of 97.4 %, a precision of 95.3% and F-measure of 96.3%. In other words, 95.3% sentence boundaries have been found, and 97.4% of the positions detected as a sentence boundary are indeed sentence boundaries. The average performance of sentence boundary detector on this training dataset is 96.3%.

The sentence boundary detector detects some of the sentences incorrectly due a certain reasons. The first one is the sentence boundary markers are also used for other purposes in this dataset. For instance, dot ('.') is used to represents a shorten forms of words. Example 'mil.' and 'Qar.' to represent **'milliona'(million)**and **'Qarshii'(money)** respectively**.** The second reason is some of the sentences in the training dataset are not separated from one another so that the sentence boundary detector considers them as one sentence which is not incorrect sentences.

B. Word boundary detector

The word boundary detector takes a list of sentences detected by sentence boundary detector or a single sentence from testing dataset as an input and displays a list of words as an output. Table 5.2 presents the performance of word boundary detector on the training dataset and testing dataset

Dataset	Number of words	Correctly splitted	Incorrectly splitted	Total splitted	precision	recall	F-measure
Training	9152	8894	309	9203	96.6%	97.1%	96.8%
Testing	373	368	21	389	94.6%	98.6%	96.6%

Table 5.2 Performance of word splitter

As it shown in Table 5.2 the performance of word boundary detector is evaluated on two dataset. These are the training and the testing dataset. The performance result of word boundary detector on the training dataset is a recall of 97.1%, a precision of 96.6% and F-measure of 96.8%. This means that 97.1% of all words have been splitted correctly, and of all detected positions that are supposed to be a word boundary 96.6% are indeed word boundaries.

The performance result of word boundary detector on the testing dataset is a recall of 98.6%, a precision of 94.6% and F-measure of 96.6%. In other words, 98.6 % of all words have been splitted correctly, and of all detected positions that are supposed to be a word boundary 94.6% are indeed word boundaries.

The false alarm of the word boundary detector is due two main reasons. The first one is related to compound words in the language. Most of the time words in Afaan Oromo sentences are separated by space. Therefore the word boundary detector module uses space as word boundary to split the words. But in Afaan Oromo, compound words are words that are constructed mostly from two words separated by space. In this study the word boundary detector never treat compound words in the language.

The second reason for the low performance of the word boundary detector is, in some the sentences there is no space between different words in both training and testing dataset so that the word boundary detector considers the words as a single word.

C. POS Tagger

The POS tagger used in this study accepts a list of words as an input and assign part of speech tag to each word. The tagger used in this study is known as TnT tagger which is a language independent POS tagger [20]. The tagger is trained using 1893 words that are manually tagged by [35]. The performance of the tagger is measured in terms of its accuracy in assigning a correct POS tag to words. Table 5.3 presents the accuracy of TnT tagger on both training and testing datasets.

Dataset	Number of words	Correctly tagged	Incorrectly tagged	Accuracy
Training	9152	6347	2805	69.3%
Testing	373	305	68	81.7%

Table 5.3 Performance of POS tagger

As we can see from table 5.3 the performance of the POS tagger on the training dataset is an accuracy of 69.3%. In other words of 69.3% of words in the training dataset are assigned to their correct POS tag whereas 30.7% % of the words are assigned incorrect POS tag.

Table 5.3 also shows that the performance of POS tagger on the testing dataset is, an accuracy of 81.7%. This means that 81.7% of words in testing dataset are assigned to their correct POS tag whereas 18.3 % of the words are assigned incorrect POS tag.

Different reasons that lead to low performance of TnT tagger are identified. The first one is the size of training data used to train the tagger. If the size of training dataset to train the tagger is

small we may not find the words (tokens) to be tagged in the training dataset so that the TnT tagger consider the words(tokens) as unknown tokens and it is difficult to assigned the correct POS tag for them.

Another reason for the low performance TnT tagger is related to the low performance of word splitter on compound words. Since the word splitter considered the compound word as two different words, the tagger assigns different POS tag for both words even if they are a single word.

Furthermore, there are also spelling errors in the corpus. Therefore, the POS tagger considers these words as an unknown words and it may difficult to assign a correct POS tag.

D. Morphological Analyzer

The morphological analyzer accepts a word from a sentence to be checked and returns the words and their analysis as an output. In this study the morphological analyzer only analyses a word classes which are used to check agreement errors discussed in section 2.2. Specifically, the morphological analyzer is used to analyze noun, pronoun, verb, adverb and adjectives word classes. Table 5.4 presents the performance of morphological analyzer.

Dataset	Number of words	Correctly analyzed	Incorrectly Analyzed	Accuracy
Testing	253	149	104	58.8%

Table 5.4 Performance of morphological analayzer

As we can see from table 5.4 the performance of the morphological analyzer on the test dataset is an accuracy of 58.8%. In other words of 58.8% of words in the test dataset are analyzed correctly whereas 42.2 % of the words are incorrectly analyzed.

The false analysis of the morphological analyzer is due to different reasons. The first one is the incorrect POS tag assigned to the words. If the words are not assigned to their correct POS tag it is difficult to analyze them because different word class may analyzed in different ways.

The second reason is the rule used to develop the morphological analyzer is not complete. Therefore there are words that do not match the rule specified and analyzed incorrectly. The third one is related to compound words. Since compound words are not handle by the word splitter module the morphological analyzer analyze the two words separately.

5.2.2. Performance of grammar checker

In this subsection the performance of SAOGC trained on the training dataset is evaluated and analyzed on the testing dataset. The performance of SAOGC is evaluated in two ways on the test dataset.

The first one is the performance evaluation of the grammar checker in flagging incorrect sentences in the test dataset and the second one is the performance of the grammar checker in flagging correct sentences in the test dataset. Table 5.5 presents the performance of the SAOGC **in flagging incorrect** sentences and Table 5.6 presents the performance of SAOGC **in flagging correct** sentences respectively.

Techniques	Total sentence	Incorrect as Incorrect	Correct as Incorrect	Total as in Incorrect	Actual incorrect	precision	Recall	F-measure
Token N-gram	85	37	15	52	50	71.1%	74%	72%
Tag N-gram	85	43	9	52	50	82.6%	86%	84.3%

Table 5.5 Evaluation result of Afaan Grammar checker in detecting incorrect sentences

From the above table we can see that the performance of token N-gram technique in flagging incorrect sentences is a recall of 74%, precision of 71.1% and F-measure of 89.0%. This means that 74% of incorrect sentences have been flagged, and 71.1% of the sentences flagged as incorrect sentences are indeed incorrect sentence. The average performance of token n-gram technique is 72%.

The token n-gram technique has detected 74% of the incorrect sentence as incorrect sentences. This means that the grammar checker identifies 37 incorrect sentences in testing dataset as incorrect sentences. Furthermore, the token n-gram technique has detects 15(42.8)% of correct sentences as incorrect. This is because most of the sequence in these sentences does not exist in the training dataset and considered as Unknown words (Out Of Vocabulary) during the language model. This may leads to the probability of the sentences to be less than the settled threshold

On the other hand, from the above table we can see that the performance of tag n-gram technique in flagging incorrect sentences is a recall of 86%, precision of 82.6% and F-measure of 84.3%. This means that 86% of incorrect sentences have been flagged, and 82.6% of the sentences flagged as incorrect sentences are indeed incorrect sentence. The average performance of tag n-gram technique is 84.3%. The tag n-gram technique has also detected 25.7% of correct

sentences as incorrect sentences. In this technique most of the sentences with incorrect word orders are flagged as incorrect sentences. But some of the sentences with different word class agreement are challenging for flagging as correct or in correct.

The low performance of tag N-gram technique on flagging incorrect sentences is due to the low performance of the POS tagger and morphological analyzer module. If words are assigned to incorrect tags the extracted sequence of tag may be found in the training database. So that the sentence can be considered as correct even if the sentence is incorrect.

In this study, the performance of the grammar checker in flagging correct sentences in the testing dataset is also evaluated. Table 5.6 presents the performance of the SAOGC **in flagging correct** sentences.

Techniques	Total sentence	Correct as Correct	Incorrect as Correct	Total as in correct	Actual correct	precision	Recall	F-measure
Token N-gram	85	20	13	33	35	60.6%	57.1%	58.5%
Tag N-gram	85	26	7	33	35	78.7%	74.2%	76.4%

Table 5.6 Evaluation result of Afaan Grammar checker in handling correct sentences

The Table 5.6 also shows that training the token N-gram technique yields a recall of 57.1%, precision of 60.6% and F-measure of 58.5%. This means that 57.1% of correct sentences have been flagged, and 60.6% of the sentences flagged as correct sentences are indeed correct sentence. The average performance of token N-gram technique is 58.5%.

On the other hand, the tag N-gram technique yields a recall of 74.2%, precision of 78.2% and F-measure of 76.4%. In other words, 74.2% of correct sentences have been flagged, and 78.2% of the sentences flagged as correct sentences are indeed correct sentence. The average performance tag N-gram technique is 76.4%.

The incorrect flags of token n-gram technique are mostly due to the size of corpus, the low performances of the algorithms and quality of corpus used to train the system and the quality of the corpus used for testing the system. The size used to train the grammar checker does not include all the words in the language so that the sequence of token (tag) in the testing dataset may not found and assigned unknown words which decreases the probability of the sentences.

The quality of the corpus is also another reason which leads to the low performance of token n-gram technique. Both the training and testing data set contain the spelling errors so that the words may not match with the statistical database. The incorrect flags of tag n-gram on the other hand are due to the low performance of word splitter, POS tagger and morphological analyzer on both training dataset and test dataset.

CHAPTER SIX

CONCLUSIONS AND RECOMMENDATIONS

6.1. CONCLUSIONS

In this study, a statistical Afaan Oromo grammar checker has been designed, developed and tested on 85 Afaan Oromo sentences. Two techniques of statistical approach (token n-gram and tag n-gram) are used in this research. The token N-gram technique checks the grammatical correctness of Afaan oromo sentence by extracting the tri-gram sequences of tokens in both training and testing dataset and calculating the probability these sequences. Using the probability of the sequences the sentence probability in the test dataset is calculated. If the probability of the sentence is greater than a threshold the sentence is considered as correct. Otherwise, the sentence is assumed incorrect.

The performance of the two techniques is evaluated in two different ways. The first one is the performance the techniques in identifying incorrect sentence and the other is the performance of the techniques in identifying correct sentences.

The performance of token N-gram technique in flagging incorrect sentence is a recall of74%, precision of 71.1% and F-measure of72 %. On the other hand the performance of this technique in flagging correct sentence is a 57.1%, precision of 60.6% and F-measure of 58.5% which needs more improvement.

Similarly, the tag N-gram technique checks the grammatical correctness of Afaan oromo sentence by calculating the probability of tri gram sequence of tag in both training and testing dataset.

The performance tag n-gram technique in flagging incorrect sentence is a recall of 86%, precision of 82.6% and F-measure of 84.3%. On the other hand the performance of this technique in flagging correct sentence is a recall of 74.2%, precision of 78.2% and F-measure of 76.4%.

There are also some reasons that lead to the low performance of the two techniques. The first one is the issue related to the performance of sentence boundary detector, word splitter, POS tagger and morphological analyzer modules. Another reason is for the low performance of the two techniques is related to the quality of the corpus (spelling error, the spacing error).

In general, in this study the researchers have identified different types of sentences errors occurring in Afaan Oromo languages such as word order, agreement error and punctuation mark errors. The study also shows word order errors and punctuation mark errors are easily identified by the grammar checker than word class agreement errors.

6.2. RECOMMENDATIONS

It is believed that the developed Afaan Oromo grammar checker in this research needs more improvements. Therefore, the following recommendations should be looked at in the future so that effective Afaan Oromo Grammar checker can be developed to help users in identifying their grammatical mistakes.

- The size of the training corpus used for this research was limited in size and doesn't have good quality. These limitations affected the accuracy POS tagger and Morphological analyzer. Therefore, some work should be done with large and high quality corpus to increase the performance of the grammar checker.

- The developed grammar checker does not check the spelling correctness of words in both training and testing dataset. This in one way minimizes the accuracy of POS tagger in such way that words may assigned to incorrect POS tag due spelling error. The incorrect POS tags and spelling errors of words in another way affect the accuracy of morphological analyzer. Therefore the Afaan Grammar checker should be integrated with spelling checker in order to maximize the performance.

- The morhopoloical analyzer do not perform well so it good to use high performance morphological analyzer to increase the performance of the grammar checker.

- Combination of the rule-based and statistical approach can help the system to achieve high efficiency and robustness. This can be possible by mixing the good qualities of the two approaches to detect high level grammatical errors.

References

[1] Debela,T. *"A Rule-Based Afan Oromo Grammar Checker"*. Jimma Institute of Technology. Ethiopia: Vol. 2, No. 8, 2011.

[2] Daniel, N. *"A Rule-Based Style and Grammar Checke"r*. Diplomarbeit. Technische Fakultät Bielefeld. 2003.

[3] Gutema, E. *"Afaan Oromo Text Retrieval System"*. Master Thesis, Department of Information science, Addis Ababa University, Addis Ababa.Ethiopia, 2012.

[4] Simon Ager, *"Oromo Language"*,‖ *Online edition*, 2012. [Online]. Available: www.sas.upenn.edu/African_Studies/Hornet/Afaan_Oromo_19777.html. [Accessed: 25-Oct- 2013].

[5] Python Software Foundation, *"About Python"*,‖ 2012. [Online]. Available: http://www.python.org/about/. [Accessed: 12-Oct-2013].

[6] Gumii Qormaata Afaan Oromoo. *"Caasluga Afaan Oromoo Jildi I"*, Komishinii Aadaaf Turizmii Oromiyaa, Finfinnee, Ethiopia, (1995)

[7] R. Goebel, J. Siekmann, andW.Wahlster. *"Lecture Notes in Artificial Intelligence"*.2009

[8] Henrich, Verena; Reute, Timo. *"LISGrammarChecker: Language Independent Statistical Grammar Checking"*. Hochschule Darmstadt & Reykjavík University: 2009.

[9] Peter Jackson and Isabelle Moulinier, *"natural language processing for online applications: text retrieval, extraction and categorization*",1984.

[10] M.J. Alam, N. UzZaman and M. Khan. *"Ngram based statistical grammar checker for Bangla and English"*. Proceedings of 9th International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh .2001.

[11] Tilahun, G. *"Qube Afaan Oromo: Reasons for Choosing the Latin Script for Developing an Oromo Alphabet"*, The Journal of Oromo Studies .1993.

[12] Mylanguage.org, *"Oromo Numbers"*,‖ 2011. [Online]. Available: http://www.mylanguages.org/learn_oromo.php. [Accessed: 12-Jan-2012].

[13 Tsuruga, Ikki-machi, Aizu-Wakamatsu. *"Dependency-Based Rules for Grammar Checking with LanguageTool"*. Maxim Mozgovoy. University of Aizu. IEEE. Japan, 2011.

[14] Daniel,B. *"Afaan Oromo Information Retrieval (CLIR): a corpus based approach"*. Master Thesis, Department of Information Science, Addis Ababa University, Addis Ababa. Ethiopia, 2005.

[15] Aynadis T. *"Design and Development of Amharic Grammar Checker*". Master Thesis, Department of computer science, Addis Ababa university, Addis Ababa. Ethiopia, 2013.

[16] Assefa W. "*Development of morphological Analyzer for Afaan Oromo"*. Master Thesis, Department of Information Science, Addis AbabaUniversity, Addis Ababa. Ethiopia,2005.

[17] Etnologue, "*Show Language*",‖ Online edition, 2009. [Online]. Available: http://www.ethnologue.com/web.asp. [Accessed: 01-Dec-2012].

[18] Domeij, Rickard; Knutsson, Ola; Carlberger, Johan; Kann, Viggo. " *Granska: An efficient hybrid system for Swedish grammar checking*". proceedings of the 12th Nordic conference in computational linguistic, Nodalida- 99. 2000.

[19] Bethlehem Mengistu.*" N-gram-Based Automatic Indexing for Amharic Text"*. Master Thesis, Department Information Science , Addis Ababa university, Addis Ababa. Ethiopia, 2002.

[20] Throsten Brants.*"TNT A statistical part of speech tagger"*. Saarland University, computational linguistics .23 May 2000.

[21] Tsuruga, Ikki-machi, Aizu-Wakamatsu. *"Dependency-Based Rules for Grammar Checking with LanguageTool"*.Maxim Mozgovoy. University of Aizu. IEEE. Japan, 2011.

[22] Abera, N. "*Long Vowels in Afaan Oromo: A Generative Approach*". Master Thesis. School of Graduate Studies, Addis Ababa University, Ethiopia 1988.

[23] Kula, K., Varma, V. and Pingali, P. "*Evaluation of Oromo-English Cross-Language*" Technologies Research Center. Information Retrieval IIIT, Hyderabad, India (2008).

[24]. Grage, G. and Kumsa, T Oromo Dictionary. African Studies Center, Michigan State University . (1982).

[25] Summary and Statistical Report of the 2007 Housing Census: Population Size by Age and Sex, Addis Ababa, 2008

[26] Oromo Language. http://en.wikipedia.org/wiki/Oromo_language; Last accessed on August 25, 2013.

[27]. C. Griefenew-Mewis, "*A Grammatical Sketch of Written Oromo*", Druckerei Franz Hansen, Bergisch Gladbach, Germany, 2001.

[28] Oromo Language: Encyclopedia, http://en.allexperts.com/e/o/or/oromo_language.htm, Last accessed on April 10, 2010.

[29]. Diriba M. "*An automatic sentence parser for oromo language using Supervised learning technique"*. Master Thesis, Department of Information Science, Addis Ababa University, Addis Ababa. Ethiopia, 2002.

[30]. Mandefro L, *"Named Entity Recognition for Afan Oromo"*, Master's Thesis, Department of computer Science , Addis Ababa University, Ethiopia,2010.

[31]. Tesfaye G, *'Afaan oromo search engine'* Master's thesis, School of graduate studies, Addis Ababa University, Ethiopia, 2010

[33]. Debela T., *"Designing a Stemmer for Afan Oromo Text: A hybrid approach"*, Master's thesis, School of graduate studies, Addis Ababa University, Ethiopia, 2010

[34]. Mohammed H. "Part-*of-speech tagging for afaan oromo Language using transformational error driven Learning (tel) approach*",2010

[35] Getachew M *,"part of speech tag for afaan oromo language"*, Master's thesis, School of graduate studies, Addis Ababa University, Ethiopia, 2009.

[36]. R.j.Hayward and Mohammed Hassen. "*The Oromo Orthography of Shayky Bakri*" ,Bulletin of the School of Oreintal and African Studies, University of London,1981

[37]. Abebe A, *"automatic morphological synthesizer for afaan oromoo"*, Master's Thesis, Department of computer Science , Addis Ababa University, Ethiopia, June 2010.

[38]. Abebe K., "*Thesis: Case System in Oromo*", Addis Ababa University, 2002

[39]. Irresoo Nagi *"Caassefama Afan oromo"*, Kuraz International, Addis Ababa, 2006.

[40]. Baye Yimam .*"The phrase structure of Ethiopian Oromo"*. University of London, School of Oriental Studies .PhD Dissertation, London 1986

[41. Girma D., "Afan Oromo news text summarizer", Master's Thesis, Faculty of Informatics , Addis Ababa University, Ethiopia,2013.

[42] Ronald Rosenfeld. *"Two decades of statistical language modeling: where do we go from here?"*School of Computer Science,Carnegie Mellon University, Pittsburgh, PA 15213,USA,1994

[43]. Stanley F. Chen and Joshua Goodman. *"An Empirical Study of Smoothing Techniques for Language Modeling"*,August 1998

[44] . Berlin Chen. "*Introduction to SRILM Toolkit*",Department of Computer Science & Information Engineering National Taiwan Normal University

APPENDIX I

Sample code

```
def word_spliter(check):
 if check=='0':
  sentencelist=open('short','r')
  sentence=sentencelist.read()
  sentence=sentence.split()
  w=open('word_list.txt','a')
  for words in sentence:
   if words.endswith('.') or words.endswith('?') or words.endswith('!'):
    leng=len(words)
    z=words[leng-1]
    x=words.partition(z)
    w.write(x[0] +"\n")
    w.write(x[1] +"\n")
   else:
    w.write(words +"\n")
  else:
   sentence=sentence.split()
```

```
def pos_taggeer():
 if check=='0'
  import os
  os.system('./tnt c  word_list.txt> tagged_list.tts')
  y=open('tagged_list.tts','r')
  w=y.readlines()
  print(len(w))
  y.close()
  tag=open('taggedlist.txt','a')
  for lis in w:
     if lis.startswith('%'):
       print
     else:
        tag.write(lis)
  tag.close()
```

APPENDIX II

Tag set

Tag	Description
NN	Tag for all types of noun that are not joined with other categories in a sentence
NP	Tag for all nouns that are not separated from postposition
NC	Tag for all nouns that are not separated from conjunctions
PP	Tag for all pronoun that are not joined with other categories
PS	Tag for all pronouns that are not separated from postposition
PC	Tag for all pronouns that are not separated from conjunctions
VV	A tag for all main verbs
AX	A tag for all auxiliary verbs
JJ	A tag for all adjectives that are separated from other categories
JC	A tag for adjectives that are not separated from conjunctions
JN	A tag for numeral adjectives
AD	A tag for all adverbs
PR	A tag for all preposition /postposition
ON	A tag for all ordinary numerals
CC	A tag for all conjunction
II	A tag for all introjections
PN	A tag for all punctuation

APPENDIX III

Sample testing dataset

Baacaan eessa jirata? kitaaba meeqa dubbiste? inni boru deeman. fardeen shan du'e. Tolaan bay'ee gurraatti dha. isaan kaleessa dhufan. isaan boru dhufe. nuuyi hunduu halluu dimaa jallana.

baratootni barsiisaa isaaniif badhasa kennan. Tolosaan kitaaba Caalaadhaaf ergise fudhate.

Tullun seenaa abbaa isaaa nutti hime. Dabalaan xalayaa barreessite.

biftuufi Daraaraan mana barumsaa deeman. magartuun bishaan waraabde.

sooreettiin bishaan waraabdee gara manaa debite. Guutaan badhasaan gammadee of wallale.

Caalaan mana barumsaa dhaqee gale. Guulaan waan bay'eee qo'ateef, qormata darbe.

dureettiin waan dukkubsatteef mana yaalaa dhaqte. isaan uffata bitaani galle.

nuyi uffata bittanne galan. sammuun lammi yaadu kamiyyuu biqiltoota hin mancaasu.

Bara dabre dhaabatan biqiltuleen kun. barsiisaan keenya baay'ee guddoo dha.

inni kaleessa dhufa debi'e. gammachuun mana citaa lama ijarte.

daa'imaniif nyaata madaalamaa keennuun barbachisa dha. itiyoophiya keessatti namoonni shan duute.

dulte ishiin bara darbe. isaan uffata gaarii bitate.

DECLARATION

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

_____

Abebe Mideksa

The thesis has been submitted for examination with my approval as university advisor.

_____

Ato. Ermias Abebe