ADDIS ABABA UNIVERISTY
SCHOOL OF GRADUATE STUDIES
COLLEGE OF NATURAL SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

Design and Development of Amharic Grammar Checker

Aynadis Temesgen Gebru

A Thesis Submitted to the School of Graduate Studies of Addis Ababa University in Partial Fulfilment for the Degree of Master of Science in Computer Science

March 2013

ADDIS ABABA UNIVERISTY
SCHOOL OF GRADUATE STUDIES
COLLEGE OF NATURAL SCIENCES
DEPARTMENT OF COMPUTER SCIENCE

Design and Development of Amharic Grammar Checker

Aynadis Temesgen Gebru

Signature of the Board of Examiners for Approval

	Name	Signature
1.	Dr. Yaregal Assabie, Advisor	
2.	Dr. Dejene Ejigu , Examiner	
3.		

Declaration

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

_____

Aynadis Temesgen Gebru

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

_____

Dr. Yaregal Assabie

To the memory of my father

ACKNOWLEDGMENT

I would like to thank my advisor, Dr.Yaregal Assabie, for his support, assistance, understanding and motivation throughout this thesis work. My special thanks goes to Michael Gasser for his willingness, help and guidance while trying to use HORNMORPHO morphological Analyzer. I sincerely want to thank very much Teshome, a language professional and PHD student in Addis Ababa University, who has helped me a lot regarding Amharic language and preparation of testing corpus. It is also my pleasure to express my gratitude to Professor Baye Yimam who kindly gave me reading material on Amharic language. My deepest gratitude goes also to my brother, Befkadu Temesgen, for his help with all he can from the very beginning of my thesis. The last but not the least is my friends, especially Genet Asefa for her understanding and support in my work.

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

ABBREVIATION

BNC	British National Corpus
CFG	context free grammar
COGrOO	Corretor Grammatical para OpenOffice
DTD	Document Type Definition
FDRE	Federal Democratic Republic of Ethiopia
HMM	Hidden Markov Models
NLP	Natural Language Processing
POS	Part Of Speech
SOV	Subject Object Verb
STAMGRAM	Statistical Amharic Grammar checker
SVO	Subject Verb Object
XML	Extensible Mark-up Language

ABSTRACT

Most human knowledge is recorded in natural language. The records are kept in computers or on paper to be manipulated and reserved for use in the future. Natural Language processing plays an important role in increasing computers capability to understand natural languages. Designing and implementing computer programs that can understand natural language is the aim of the works in the area of Natural Language Processing. In order to communicate through natural languages grammatical correctness is very crucial. Therefore, natural language processing applications should be enabled to recognize the grammatical errors of natural language texts. This process is known as grammar checking. This work introduces development and design of Amharic grammar checker.

Two grammar checker approaches have been used in this research. The first approach is a rule-based and it is tested for simple sentences. The rules are constructed manually and matched against the patterns of the sentence to be checked. The second approach is statistical approach and tested for both simple and complex sentences. In the statistical Amharic grammar checker, n-gram and probabilistic methods are used to check grammatical errors of Amharic sentence. The patterns and the corresponding 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 test set, is prepared from a manually part-of-speech text of the language.

The evaluation is made in two test cases. The first case is done on simple sentences. In this test case, 92.45% precision and 94.03% recall is obtained for the rule-based Amharic grammar checker. On the same test case, the statistical Amharic grammar checker (trigram) shows precision and recall of 67.14% and 90.38% respectively. The statistical Amharic grammar checker is tested using complex sentences in the second test case. In this test case, 63.76% of the errors are detected. The evaluation result shows that each approach is capable of detecting multiple errors from a sentence. 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 in both approaches.

Keywords: Statistical grammar checker, rule-based grammar checker, n-gram, POS tag sequence

TABLE OF CONTENT

CHAPTER ONE

INTRODUCTION

1.1 Background

Technology is rapidly changing our world all the time affecting many aspects of our daily life. Computers, one of the most prominent innovations that this advancement in technology has brought to the world, are very productive and efficient machines which make our personal and professional lives more simple and rewarding. Many researches have been conducted on computers applications which improve the existing ones or add some other capabilities to advance new applications. One of these researches is Natural Language Processing (NLP) and it has contributed a lot to this advancement. NLP is an area of research and application that explores how computers can be used to understand and manipulate natural language, a language that has evolved naturally as a means of communication among people, text or speech to do useful things [1, 2]. One of the good examples of NLP application is grammar checker.

Grammar checker is a software program or a part of other large programs that checks written texts for grammatical errors [3]. Grammar (or syntax) refers to a system of rules describing what correct sentences should look like [4]. Having these rules in mind, people can make a decision on correctness of the sentence. The history of grammar checker dates back to 1970's. English grammar checker was first started in with UNIX system known as Writer's Workbench [3]. It checks the grammar error and suggests the user to choose other better options. Most of the previous grammar checkers were based on style checking, checking uncommon words and complicated sentence structure, but now they are upgraded to high capacity grammar checking tool, not only as a part of other program but also as easy software to be installed in many operating system[3].

Rule-based checking, statistics-based checking and hybrid checking are the three widely used ways to implement a grammar checker [5, 6]. Rule-based checking has set of rules developed manually, match against the text. It has advantages such as sentence does not need to be complete to be checked, rules can be built incrementally, suggest error message with helpful comments, easy configuration and so on. However, it's very difficult to understand and include

all correct grammatical rules which can be used to check sentences, especially for complex sentences. On the other hand, in the statistic-based checking, Part Of Speech (POS)-annotated corpus is used to build the grammar rules automatically by identifying list of POS tag sequences and then those common sequences that occur often can be considered correct and the uncommon ones incorrect. However, it's very difficult to understand error message suggested by this checking system as there is no specific error message [6]. In hybrid grammar checking, the good qualities of the rule-based and statistical approach can be combined to develop the grammar checker.

Amharic POS tagger [7], morphological Analyzer [8], sentence parser [9], spell checker [10] are some of the attempts that have been made in Amharic language processing in the past years. However, to the best knowledge of the researcher there is no research conducted for Amharic grammar checker. Therefore, this research is aimed at filling the gap on Amharic grammar checker which assists users into eliminating mistakes in their Amharic writings.

1.2 Motivation

Amharic language, one of the Semitic languages, is the official language of the Federal Democratic Republic of Ethiopia (FDRE) [4]. Most of the official documents in governmental and private sectors in Ethiopia are written in Amharic. Amharic has its own syllabic alphabet or character ('ፊደል'/'*fidel*') that requires close orthographic treatment when forming words [4].

Many grammar checker researches have been done in different foreign language texts. English [6, 11, 12], Swedish [13], German [14], and Arabic [15] are some of the grammar checker researches conducted in the past years. The same research had also been conducted on Ethiopian language, Afan Oromo [5]. Even though Amharic language users can write Amharic texts in word processing applications, the users can not be sure whether the written texts are grammatically correct or not. Therefore, this research contributes a lot in indentifying grammatical errors from written texts. It also helps users in handling their grammatical errors while writing Amharic texts.

Amharic language, which is morphologically rich, has some unique features when compared with other languages like English [4].Unlike the English which is SVO (Subject, Verb, Object), the Amharic clause order is SOV. To show this with example, the Amharic sentence "አበበ በሶ በላ

"/ "'*abebe beso bela*" (where 'አበበ/'*abebe* is subject, በሶ/*beso* is an object and በላ/*bela* is verb) and while the English "Abebe ate besso" (where Abebe is subject, ate is a verb and besso is object). Besides, Amharic verbs can have a number of different forms compared with English verbs. For instance, the English word "drink" can have the form "drink, drinking, drinks, drunk" but when it comes to Amharic language ጠጣ/ *TeTa*, ጠጣች/ *TeTac*, ጠጡ/ *TeTu*, ሊጠጡ/ *liTeTu*, ከጠጡ/ *kTeTu*, አስጠጡ/ *'asTeTu*, አጠጡ/ *'aTeTu*, አጠጥተዋል/ *'aTeTitewal*, አስጠጥተዋል/ *'asTeTitewal*, ጠጣሁ/*TeTahu*, አስጠጣሁ/*'asTeTahu*, አልጠጣም/*'alTeTam*, አላስጠጣም/*'alasTeTam* and so on. A single verb may consist time (past, present, and future), gender (male and female), action (command, statement, invitation) and negation (not) which is expressed in a sentence in other languages .For example, the Amharic word "'aldeweleclacewum" refers to the English sentence "She did not call them." In which case, the Affixes አል/'*al* -, -ች/*c*-, -ላቸዉ/*lacewu*-, ም/*m* add additional meaning of various kinds to the main morpheme (root or stem) "ደወል"/"*dewel*".

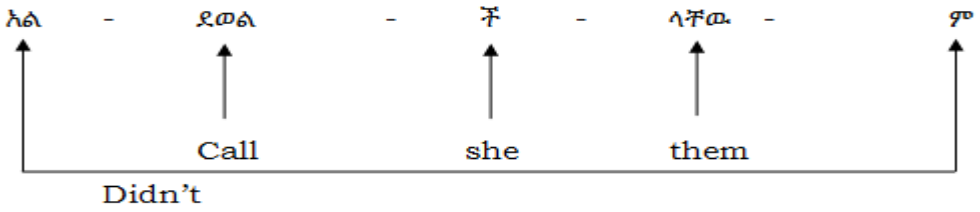


Figure 1.1 : Morphemes of Amharic word

The other different feature in Amharic is the second person 'you' which may take different agreements when referring 'plural', 'respect (politeness)','singular-female' or 'singulars-male' nouns[4]. All these cases should be handled or treated in different way as they represent different meaning. For instance,

አንተ መሄድ ትችላለህ /*'ante mehEd tclaleh* → you can go (Singular male)

አንቺ መሄድ ትችያለሽ/ *'anci mehEd tcyalex* → you can go (Singular female)

እናንተ መሄድ ትችላላችሁ/*'nante mehEd tclalacu* → you can go (Plural)

እርሶ መሄድ ይችላሉ/*'rso mehEd yclalu* → you can go (politeness)

As shown in the above examples and statements, Amharic language is morphologically rich and complex. Due to this richness and complexity, it is very difficult to adopt or customize

other language grammar checker for Amharic language.

Amharic users, specially the non-native ones, can possibly make a mistake while writing Amharic documents, letters and other texts. Common Amharic Grammatical errors include subject and verb mismatch, person, gender and number miss match. For example,

እሱ በሶ በላች/*ʾsu beso belac* (subject and verb miss match)

ልጁ እራቱን በልቶ እንቅልፉን ተኙ/ *lju ʾratun belto ʾnqlfun teNu* (number)

Taking the above mentioned issues in to consideration, it is necessary to conduct a research on Amharic grammar checker that will help users in generating error free texts and save the time that they waste for proofreading.

1.3 Statement of the Problem

Amharic texts can be represented on computers or written in word processing programs. These texts may contain grammatical errors. Grammar checker has applications in word processing program. It helps user in generating error free natural language texts. However, Amharic language lacks this capability. A great number of diverse researches have been proposed and done for grammar checker in different language. English grammar checker is one of the most commonly used and also available as a past of word processors like Microsoft word [12]. As Amharic is morphologically rich and has its own complex grammatical structure, other language grammar checker techniques are not capable of filling the gap for Amharic language.

This work tried to answer the following questions:

- Can grammar checker of other languages be customized and used for Amharic?
- What are the commonly used grammar checker approaches?
- Which technique can be applied for Amharic language?
- Which technique is appropriate for simple and complex sentence?

1.4 Objective

1.4.1 General Objective

The general objective of this research is to design and develop a system that handle possible grammatical errors from Amharic text and identify grammatically correct and incorrect sentences.

1.4.2 Specific Objective

The specific objectives of this project are to:

- study grammatical structure and rule of Amharic language, its Morphology and syntax;
- review literatures on grammar checker in different languages and other related researches conducted on Amharic language;
- collect a corpus to prepare train set to train the system and the test set to test the prototype developed
- develop algorithm for approaches used to build Amharic grammar checker system;
- develop a prototype for the system; and
- evaluate the system using collected test set corpus.

1.5 Scope and Limitation of the Study

This study only covers automatic grammar checking for Amharic language. Direct Amharic letter (fidel) is given as an input and the input text is automatically transcribed inside the Morphological Analyzer (HornMorpho). Due to the lack of large annotated corpora, the available corpus is automatically annotated using HornMorpho morphological analyzer. The training corpus is not grammatically checked. As finding corpus which contains possible grammatical errors is difficult, the testing set is prepared manually with the help of linguistic experts. This work tried to include morphological property of the language. The scope of this study is limited to detect common grammatical errors of Amharic language.

1.6 Methodology

1.6.1 Literature Review

Related works will be reviewed to get a deeper understanding about Amharic language grammatical structure, pre-processes for Amharic grammar checker system development and fundamental concepts related to this work. A review on different approaches of grammar checking systems will also be made to identify and understand the concept related to them. A discussion with linguistic experts will be made to recognize common Amharic grammatical errors and correct grammatical structure of Amharic sentence.

1.6.2 Data Collection

Different sample data will be selected for the rule-based and statistical grammar checker. The sample data used in the rule-based approach will be selected on the basis of researcher's knowledge about the language and the nature of the research aim. Rules will be extracted based on the part of sample data which contain Amharic grammatical errors. The rest of the sample data will be used to test the prototype.

For the statistical grammar checker, both the training and test set will be prepared using manually POS tagged corpus. The sentences for test set will be selected randomly from corpus. The selected sentences for test set will be prepared manually with the help of linguistic experts. The test set contains grammatically correct and incorrect sentences which were used to determine the performance of the system.

1.6.3 Prototype Development

To develop the prototype of the system, appropriate supporting tools is required to facilitate the study. These include POS tagger and morphological analyzer. The supporting tool will be used to assist the study. HornMorpho, Amharic morphological analyzer tool, will be used to give linguistic meanings to the morphemes in each word and annotate the words in the corpus based on the linguistic meanings. The implementation of the rule-based and statistical Amharic checker grammar checker will be done using python programming language (version 3.1.3). In the rule-based Amharic grammar checker, the rules in the language model will be written and saved in XML file. The XML file will have DTD for the elements that are used in the XML file. The DTD for the XML file is attached in Appendix 2 of this document.

1.6.4 Evaluation

After developing the two prototypes for Amharic grammar checker, prototypes developed will be tested using the test set prepared for this purpose. The evaluation of the performance of system will be determined by comparing the system result with the manually checked results. The precision and the recall will be calculated by identifying the correctly flagged errors, incorrectly flagged errors and the total number of errors in the sentences. Counting those errors identified by the system will be done manually by the researcher with the help of linguistic experts.

1.7 Application of Result

As this research is mainly concerned with grammatical error correction in Amharic writing, it has significances in every program that provide an interface to Amharic writing. This includes word processing programs such as Microsoft Office (word, excel, power point), E-mail, social network sites and the like. The greater advantage of having grammar checker is producing error-free work. This research also enables Amharic language users, specially the non-native ones, to effectively prepare official documents, letters, emails, etc. It also improves productivity as it saves the time users take for proofreading.

1.8 Organization of the Thesis

This section describes the organization of the rest of the thesis. Chapter two discuses fundamental concepts, different approaches and techniques related to grammar checking researches. It also discuses related works to this study. Chapter three presents Amharic grammar which includes Amharic morphology, sentence structure and tenses. Chapter three also describes the two approaches used in this research in detail. The experiment and results are discussed in chapter four. The last chapter, chapter five, presents the conclusion and recommendation based on the experiment and results.

CHAPTER TWO

LITERATURE REVIEW

This chapter is about fundamental theoretical concepts related to grammar checker and different approaches used in developing grammar checker system. Other systems, which are pre-processes to grammar checker, are also discussed in this chapter in terms their relations to this work. The first section presents about the fundamental concepts and approaches to grammar checking system development. Section two, section three, section four and section five discuss about corpora, tokenization, part of speech tagging, and morphological analyzer respectively. Related works to this research are also discussed in the last section of this chapter.

2.1 Grammar checker

2.1.1 Grammar, Grammar Checking and Grammar Checker

Dictionaries define grammar as the rules and explanations which 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). People speaking the same language communicate and understand each other while one writes and the other reads or one speaks the other listens. This communication is facilitated by the grammar rules of the language. Communicators understand each other because they know and follow same grammatical rule of the language. If one speaks or writes a grammatically incorrect sentence, without following the grammatical rules of the language, the other (listener or writer) may not understand the speaker or writer. Therefore, the grammar rules or correct grammatical structure are one of the important things for communication through natural languages.

One may create errors which oppose the grammar rules while writing or speaking. Those errors which contradict the grammar rule of the language are called grammatical errors; which are different from spelling errors, style errors and semantic errors in a sentence [6]. Grammar checker, one of the most widely used systems in the word processors applications, verifies or tests the grammatical correctness of the sentence [5; 16]. Identifying grammatical error from a text is one of the main activities done by the grammar checker systems.

When compared with spell checker that can be done using a dictionary and few rules, grammar checker is much more complex. Some grammatical errors may be obvious and easily identified, on the other hand most of them are very difficult to recognize as the errors seem correct.

2.1.2 Approaches to Grammar Checking

Researches on grammar checker have been conducted for number of foreign languages indifferent countries [5, 6, 16,]. These researches on grammar checker follow different approaches which are discussed in this section. The three commonly used approaches in grammar checker development is statistical, rule-based and hybrid grammar checker.

Statistical Grammar Checker

A statistical grammar checker is one of the approaches used for developing a grammar checker system. The approach assumes that a text can be corrected using a large amount text which is a collection of grammatically correct sentences [18]. In this method, the rules are generated automatically from corpus and large amount of data is required to train the system. The corpus may be built from articles, journals and other magazines. It is difficult for the users to interpret the system judgment unless the developer accesses the corpus [6]. There are two different ways of implementing statistical grammar checker. The first one is when an input text to be corrected, directly compared with the corpus. When this technique is used to check the grammar, the corpus needs to be visited directly in every text check. The other one, generating a grammar rule from the corpus and use those rules to check the input text. This one requires updating the rules if corpus is modified or some data is added to the corpus [18]. N-gram Based Statistical Grammar Checker for Bangla [17], language independent statistical grammar checker [18] and statistical grammar checker for English [19] are some of researches done using this approach.

Rule-based Grammar Checker

A rule-based grammar checker approach is the most common method of grammar checking [18]. It works in a way that the input text is checked against the rules which developed manually unlike the statistical method [5, 6, 11]. Manual work load is one of the disadvantages of this approach. However, the rules are easy to configure and can be managed individually [6].

These rules are easy to add, remove or modify. Rules can be written by a person who has no programming experience, like linguistic professional [11]. Rule-based systems can offer a detailed error message even stating or explaining the grammar rule. It can also be developed incrementally, staring with one rule and extending the system adding rule one by one. Input sentences can also be checked on fly i.e. the sentence doesn't need to be complete to be checked, it can be checked while writing the sentence. The other thing, which makes this approach so powerful, is it can cover almost all feature of a language [18]. There are a number of grammar checker researches using rule-based approach. These include a rule-based Afan Oromo Grammar Checker [5], style and grammar checker for English [6], dependency-based rule for grammar checking [11] and Punjabi Grammar Checker [16].

Hybrid Grammar Checker

There are also systems that use a hybrid system of both statistical and rule-based approaches. This helps the system to achieve high efficiency and robustness [20]. "Granska", one of the Swedish grammar checkers, is developed using this approach by Rickard Domeij, Ola Knutsson, Johan Carlberger and Viggo Kann [20]. The other hybrid structure is COGrOO. It's a Portuguese grammar checker based on CETENFOLHA a Brazilian Portuguese morphosyntactic annotated Corpus [21].

2.1.3 Common Grammar Errors

Grammar rules and regulation are required for most of the languages, and same is true for Amharic. If one knows rules of the language, it is very easy for others to understand what he/she writes. However, knowing all the rules may be very difficult for the user while writing specially for the non-natives ones. Therefore, people make different mistakes when they write Amharic texts. Most of the grammar errors groups discussed in this section are common with other languages. However, this section tries to illustrate the errors in Amharic texts. Grammar error groups are identified with the help of linguistic experts.

Subject-verb Disagreement

The subject and the verb in a sentence must agree in number, person and gender. The disagreement in all the cases creates misunderstanding or confusion for the reader of the sentence. To show each of the cases with example,

1. መደርደሪያው ላይ ያሉት ደብተሮች ተሸጠ። / *mederderiyaw lay yalut debteroc texeTe*።

2. እኔ በጣም ስለቸኮልኩ እየሮጥኩ ወደቤት ሄደች።/ *ʻnE betam sleckolku ʻyeroTku wedebEt hEdec*።

3. ዘፋኙ ኢትዮጵያዊ ነች።/ *zefaNu ʻityoPiyawi nec*።

The first sentence in the example above shows number mismatch in Amharic language. "ደብተሮች"/*"debteroc"* means "exercise books" is in a plural form, whereas the verb "ተሸጠ"/*"texete"* refers to singular form of verb for the subject. The second sentence has a person mismatch in it. "እኔ"/*"ʻnE"* 'I' (first person singular) and "ሄደች"/*"hEdec"* is a verb used for the third person feminine, which totally cannot go together. In the third sentence, "ዘፋኙ"/*"zefaNu"* "male singer" is a masculine whereas "ነች"/*"nec"* (feminine verb) is used in the sentence for masculine noun. After making the correction, the above three sentences can be corrected as follow.

1. መደርደሪያው ላይ ያሉት ደብተሮች ተሸጡ።/*mederderyaw lay yalut debteroc*

2. እኔ በጣም ስለቸኮልኩ እየሮጥኩ ወደቤት ሄድኩኝ።/*ʻnE beTam sleckolku ʻyeroTku wedebEt hEdkuN*

3. ዘፋኙ ኢትዮጵያዊ ነው።/*zefaNu ʻtyopyawi new*

Object-verb Disagreement

As subject and verb agreement must hold for number, person and gender, object and verb must also agree in these categories.

For example: this kind of object-verb mismatch can be seen in sentences

1. መደርደሪያው ላይ ያሉት እቃዎች ወድቀው ልጅቷን መቱት።/*mederderyaw lay yalut ʻqawoc wedqew ljtWan metut*

2. የሺ አንቺን ትወደኛለች።/*yexi ʻancin twedeNalec*

3. የሺ እነሱን ወደደችው።/*yxi ʻnesun wededecw*

The first sentence has a feminine object "ልጅቷን"/*"ljtWan"* "girl" which is incorrectly reflected

on the verb "መቱት"/*''metut''* "kicked him" as masculine; Gender mismatch. Second one constructed by second person object "አንቺን"/*''ancin''* that disagrees with the verb "ትወደኛለች"/*''twedeNalec''* which reflect first person object; person mismatch. The last sentence has plural object "እነሱን"/*''nesun''* and singular object reflection on the verb "ወደደችው"/*''wededecw''*; number mismatch. As shown in the example sentences, sentence may also contain object-verb disagreements other than subject-verb disagreements. The above three example sentence can be corrected as follows:

1. መደርደሪያው ላይ ያሉት እቃዎች ወድቀው ልጅቷን መቷት።/*mederdriyaw lay yalut ʻqawoc wedqew lejtWan mtWat*
2. የሺ አንቺን ትወድሻለች።/*yexi ʻancin twedshalc*
3. የሺ እነሱን ወደደቻቸው።/*yexi ʻnesun wededecacew*

Incorrect Word Order

This grammar rule, the word order, has different form in different language. This 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. English has an SVO sentence structure (subject verb object) whereas Amharic has SOV. Sometimes, in informal texts, it may occur as OSV. However, this OSV order can not be used in formal Amharic texts. Depending on the natural language used, the words in a sentence should be in their correct order. For instance, one of the commonly used Amharic language sentences "አበበ በሶ በላ"/*''abebe beso bela''* which means in English "Abebe ate beso". "አበበ"/*''abebe''* is a person name, "በሶ"/*''beso''* is one of the foods in Ethiopia and "በላ"/*''bela''* means ate.

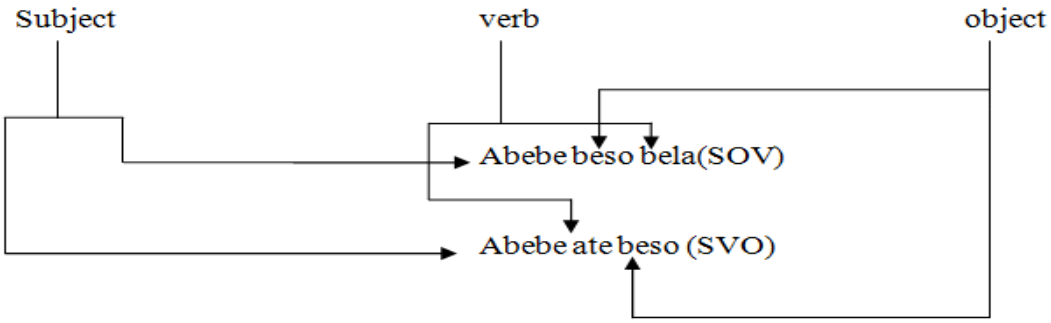


Figure 2.1 : General word order in Amharic sentences

As long as the placement of the words of the language is wrong, then sentence is considered grammatically incorrect. "አበበ በላ በሶ"/*"'abebe beso bela"*, "በላ አበበ በሶ"/*"bela'abebe beso"* or any of the other possible orders are incorrect order for Amharic. The subject of the sentence can be identified by its place in a sentence. The subject of Amharic sentences is mostly placed before the object. If the subject is incorrectly placed the meaning of the sentence may also be changed. For example, "በሶ አበበ በላ"/*"bela'abebe beso"* has a meaning "Beso ate Abebe" which does not make sense. This should be corrected as "አበበ በሶ በላ"/*"'abebe beso bela"* with an SOV order. If the noun has object marker, the OSV order can give a meaning. If the object marker '-ን'"-n" is added to 'በሶ'/"beso", then both "በሶውን አበበ በላ"/*" besown'abebe bela"* and "አበበ በሶውን በላ"/*"'abebe besown bela"* are meaningful Amharic sentence. Even through this OSV order seems meaningful, the sentence is still considered grammatically incorrect because it follows wrong Amharic word order.

The word order also includes the adjective and noun order, adverb and verb order, verb and sentence end and the other POS tags in a sentence. For example, Adjectives should appear before the noun that it modifies and adverbs also before the verb it modifies.

Adjective and Noun Agreement

Modifiers are words or phrases that are used in a sentence to elaborate something. Modifiers can be Adjectives which modify noun or adverbs that modify verbs. Amharic Adjectives appear before nouns that they modify. Amharic Adjective should agree with the noun it modifies in number and gender.

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. Amharic language has adjectives to modify nouns. These

adjectives may mark number (person and gender) and gender (feminine and masculine) of the noun they modify. If an adjective marks the number and gender of the noun, the marker should agree with the number and gender of the noun.

For example:

“ትልቋ ላም”/*”tlqWa lam”* “the big cow” is a noun phrase that contains adjective “ትልቋ”/*”tlqWa”* and noun “ላም”/*”lam”*. The noun “ላም”/*”lam”* has a feminine character and the adjective “ትልቋ” has a feminine gender marker in it. If this phrase is rewritten like “ትልቁ ላም”/*”tlqu lam”*, then 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. Therefore, to make the correction either the noun should be changed to masculine gender or the gender marker in adjective should be changed.

“ትልቅ ላሞች”/*”tlq lamoc”* “big cows” is grammatically incorrect as it contains a singular marker in the adjective and the noun it modifies is a plural noun. This can be corrected by adjusting number marker either on the adjective or the noun. “ትልልቅ ላሞች”/*”tllq lamoc”* “big cows” is grammatically correct with plural noun and plural marker in the adjective. Both the adjective and the noun it modifies have a plural marker and agree in number. “ትልቅ ላም”/*”tlq lam”* is also correct with singular marker in the noun and adjective.

Adverb and Verb Agreement (Correct Use of Tenses with Time Adverbs)

Another type of describing word or modifier is an adverb. It defined as a category which is to a verb what an adjective is to a noun. Amharic Adverb usually modifies the first verb that comes next to it. Like the other languages, adverbs also exist in Amharic language. The Amharic adverbs are few in number and can be found in primitive (as separate word that appears by themselves) and compound form (as combination of two words appearing as a word or detached words).As in many languages Amharic adverbs are classified into subclasses such as adverbs of time, place, circumstance etc. The time adverbs describe the time at which an event takes place. These adverbs may show a specific time at which a given action takes place or its duration. Amharic verbs indicate the time at which action takes place in relation with the adverbs. In Amharic (like many other languages), there are a set of forms taken by a verb to indicate time. These forms (tenses) can be grouped into seven [35, 37]. Table 3.1.1 explains the

seven types of tenses with example. The time adverb and the tense disagreement is one of the common Amharic grammatical errors. The correct type of adverb should be used for the verb and vice versa.

Example:

In the sentence "እህቴ በሚቀጥለው ሳምንት አገባች።"/*"htE bemiqeTlew samnt 'agbac"*, 'በሚቀጥለው ሳምንት'/*"bemiqeTlew samnt"* 'next week' which is in the future and the verb 'አገባች'/*'agbac"* expresses the action that is already performed in the past. Therefore, it should be corrected as either 'እህቴ በሚቀጥለው ሳምንት ታገባለች።'/*"htE bemiqeTlew samnt tagebalec*" or 'እህቴ ባለፈው ሳምንት አገባች።' /*"htE balefew samnt agebac*. In these two sentences, the adverb and verb agree in time.

2.1.4 N-gram Model

N-grams are sequence of N tokens in a sentence. Specifying N means specifying number of token. A token may be a letter, word, tag, punctuation mark or other symbol [18]. An n-gram model is probabilistic model which calculates the probability of occurrence of token with the previous N - 1 token sequence [30]. Based on N value, different names can be given for the sequences. Bigram, trigram, quadgram and pentagram are for 2, 3, 4 and 5 values of N respectively.

Given the number of token and the value of N, the maximum number of sequences for the tokens can be calculated by the formula:

$$Ns = Nt - (N - 1) \quad (2.1)$$

Where, Ns is the number of sequences, Nt is the number of tokens and N is the N value in the N-gram.

Suppose Ns is list of sequences for a given sentence. One of the sequences in Ns has W1 and W2 in it (this means the value of N is 2 this time). The probability of the sequence i.e. Probability of W2 given W1 is calculated as follows:

$$P(W2|W1) = \frac{COUNT(W1\ W2)}{COUNT(W1)} \quad (2.2)$$

Where, W2 is the second word and W1 is the first word in the sequence.

The probability of sequences for N = 3, trigram, can be calculated by counting the number of the three words occurring together over the number of the first and second words together. If W1, W2 and W3 are the first, second and third word in a given sequence, the probability of the sequence (which is the probability of W3 given W1 and W2 together) can be calculated as follows:

$$P(W3|W1W2) = \frac{COUNT(W1\ W2\ W3)}{COUNT(W1\ W2)} \quad (2.3)$$

The generalized formula of the probability of the sequence for N = n is given by:

$$P(Wn|W1W2 \ldots Wn-1) = \frac{COUNT(W1W2 \ldots Wn-1\ Wn)}{COUNT(W1W2 \ldots Wn-1)} \quad (2.4)$$

Probability of the whole sentence for a given N value can be calculated by multiplying probability of all the sequences in the sentence for that N value.

$$P(S) = P(Ns1) * P(Ns2) * P(Ns3) * P(Ns4) * \ldots P(Nsn) \quad (2.5)$$

Where, Ns1, Ns2, Ns3 ... Nsn the sequences in the sentence

2.2 Corpora

Oxford dictionary define corpus as "a body or collection of linguistic data which either be written or spoken text in machine readable form". The data of the corpus are typically digital, i.e. it is saved on computers and it's machine-readable. Corpus has components [18]. Components are:-

- The text data itself, this is the text in natural language
- Possibly Meta data which describe the text data,
- And linguistic annotations related to the text data. This may be a POS tag, morphological analysis of words and linguistics information about the letters, word,

sentences and the like

Many languages are under-resourced especially languages in developing countries like Ethiopia [7, 22]. Amharic language has also a shortage of sample annotated corpus. Due to that different Amharic researches suffer and spend much time in collecting the available corpora and analyzing to make it suitable to their own research use [23]. Statistical grammar checker requires grammatically correct texts to training the system. The main issue related to statistical approach is finding error free corpus to train the system. The training and test data set should also be annotated i.e. words in corpus needs to be POS tagged and also morphologically analyzed. A rule-based grammar checker performs the tagging and analysis as a pre-process on the text to be checked.

Walta information center corpus is an Amharic news corpus. It contains about 8067 sentences which are taken from walta information center Amharic news. Single news in the corpus contains title and content. The corpus has another copy which is manually POS tagged with part of speech. Corpora, including the manually tagged, are available on the web for download.

2.3 Tokenization

When input is given to natural language processing system, it may occur or be given as a paragraph or paragraphs. This input text needs tokenization process, i.e. input text to an individual occurrence of a linguistic unit, for further processing [5]. The tokenization processes may be splitting the input text to sentences and also to words if necessary. In the case of grammar checker, input texts should be broken into sentences and then sentences to words. The output words can be tagged and analyzed separately in the morphological analyzer. Both the training and test set should pass the tokenization process while using statistical grammar checker approach. The rule-based grammar checker also needs the tokenization on the input text to be checked. Like the statistical grammar checker, the input text should be split into sentences and then to words in order to be tagged and analyzed.

2.4 Part of Speech Tagging

Part-of-speech tagging describes the assignment of the appropriate word class and morpho-syntactic features to each token in a text. The process of tagging is performed by a tagger [18]. The most common POS include noun, pronoun, verb, adjective, adverb, preposition,

conjunction, interjection and the like [24]. Before naming the word with its POS tag, the POS tagger should look at what the word is doing in a specific sentence because some words can be used in several different ways. For instance, in Amharic sentence,

አሸናፊ ወደ ትምህርትቤት ሄደ።/"*'axenafi wede tmhrtbEt hEde"*

አሸናፊ የሚሆነው ቡድን ወደሚቀጥለው ዙር ያልፋል።/"*'axenafi yemihonew budn wedemiqeTlew zur yalfal"*

The word '*አሸናፊ*'/*'axenafi* spelled the same in the above two sentences but it functions differently in each of the sentences. In the first sentence, the word '*አሸናፊ*'/"*'axenafi"* is used as noun, name of a person. In the second sentence, it is used as an adjective and describes the noun '*ቡድን*'/*'budn'*. Other words also may be spelled the same and used differently in different sentences as verb and noun, verb and adjective and the like.

POS tagger or POS tagging is one of the important components in most of the grammar checkers developed in pervious researches [5, 6, 16, 17]. Learning about the parts of speech helps to understand grammar mistakes that can be made and figure out how to correct them. It's also very advantageous in grammar checking as it's very difficult to include each word of the language in the manually constructed rules. POS tagger is also used in statistical approach as it is almost impossible to find each word in an input text in the corpora. Therefore, the words in the sentence should be assigned to their part of speech tags and check the correctness of the sentence based on the assigned tag in the corpora. All these factors make POS tagging an important component for grammar checker development.

Some researches have been conducted on POS tagging Amharic text. The first attempt was made by Mesfin Getachew [23]. As Daniel Gochel Agonafer in [25] cited Mesfin's tagger [23], is an Amharic POS tagger developed in 2001 using Stochastic Hidden Markov (HMM) approach for the prototype development. About 23 tags were identified by Mesfin in his research [7]. The experiment was conducted on an Amharic text composed of 261 words and the results achieved were 97% accuracies on the training and 90% on the test sets [25]. The other was, which is a stochastic model using conditional random fields and five manually annotated news corpora used to train the system, developed by Sisay Fissaha Adafre in 2005 [7, 26, 27]. He identified 10 tags set. On his experiment, the size of the dataset affects the

performance of the system and he obtained 74% average accuracy [27].

There was also research which compared three tagging strategies; Hidden Markov Models (HMM), Support Vector Machines (SVM) and Maximum Entropy (ME), using manually annotated corpus (developed at the Ethiopian Language Research Center 'ELRCs') [22]. The average accuracies are 85.56, 88.30, and 87.87 for the HMM, SVM and ME-based taggers, respectively for the ELRC tag-set

Even through all these researches have been done for Amharic part of speech tagger, the systems are not available to be used in the pre-processing of Amharic grammar checker system. This is either researchers do not keep their work for a long time or they are not willing to give their work. For this reason, manually tagged news corpus is used both for the training and the test data set.

2.5 Morphological Analyzer

The term morphology comes from Greek word, morph- means 'shape, form', and morphology is the study of form or forms of something depends on which area its' used. When the Word comes to linguistics, it means the study of the formation of words and their internal structure [28]. The minimal unit of the morphology is known as morpheme, which includes the root word and the other meaningful part of the word [28, 29]. For example, the word 'ሄደች'/*'hEdec'* has the morphemes 'ሄደ'/*'hEde'* and '-ች'/*'-c'* which stand as the root word and the meaningful piece of the word respectively.

Morphological Analysis is the process of finding the morphemes of the word and providing grammatical information for the word based on the identified morphemes. For example, the word 'ሄደች'/*'hEdec'* has the morphemes 'ሄደ'/*'hEde'* and '-ች/*'c'*; 'ሄደ'/*'hEde'* is the root word and '-ች'/*'-c'* is a morpheme that is used to refer to a "feminine gender". The gender, that the word refers to, can be determined using a morphological analysis. 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.

Morphological Analysis is very important for many higher level natural language applications such as machine translation, speech recognition, information retrieval, grammar checker and the like. In morphologically rich languages like Amharic, the morphological analysis is more

crucial. For the completion and also to improve the performance of the grammar checker, morphological analysis should be performed as pre-process both on the training set and input text to be checked. If the word is taken as it is, the probability of occurrence on the training set may be zero for most of the words. This lowers the performance of the system. In the case of rule-based, rules can be written using the analysis which makes it easier than using words themselves.

HornMorpho is morphological analyzer done for three languages; Amharic, Tigrigna and Oromo by Michael Gasser [8]. To perform the analysis finite state machine, finite state transducers (FST), is used. The system accepts a word to be analyzed and shows the analysis result which includes the word POS group (Usually tagged as noun or verb) and its grammatical structure. It marks a word category for person (first, second, third), gender (feminine, masculine), number (singular, plural), definiteness (definite, indefinite), grammar (perfective, imperfective, gerundive, jussive) and the like. For Amharic and Tigrigna, it romanizes the word initially and the analysis process is performed on the romanized word. After the analysis, the output will be written/displayed as the same language as the input word [8].

2.6 Related Works

This section has tried to discus works related to this research in detail. As the researches have similarities to this work, the related works discussed in this section are researches in languages other than Amharic. A research in local language, Afan Oromo, is also discussed in this section.

2.6.1 Afan Oromo Grammar Checker

One of the NLP researches done for local languages in Ethiopia is Rule-based Afan Oromo grammar checker [5]. As the name implies, the research uses a rule-based approach for Afan Oromo language grammar checking. In Afan Oromo grammar checker, different 123 rules were constructed in order to identify the grammatical errors. Using these rules, the system detects and suggests grammatical errors from Afan Oromo text. The 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 module, assigns each word into its POS tag. The stemmer module, which is the third module, accepts the tagged word and

provides the root and affixes for the tagged word. Series of steps are used to remove certain type affixes by substitution rule. In order to apply the rules, certain conditions must hold true. One example of these conditions is the minimal length of the resulting stem. Most rules have a condition based on the so-called measure. The measure is the number of vowel-consonant sequences which are present in the resulting stem. Consecutive vowels or consonants are counted as one. This condition must prevent removal of letters which resemble suffix but actually part of the stem. The forth module, which is a grammatical relation finder, assigns grammatical relations between words. Words include subject and verb, subject and adjective, main verb and subordinate verb in terms of number gender and tense. The agreement between words is checked based on the 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 that 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 [5].

The rule-based Afan Oromo grammar checker was tested based on the number of errors correctly detected (precision) and the number of errors covered (recall) by the system. 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 final performance result shows that the rule-based Afan Oromo grammar checker has precision and recall of 89.89% and 80% respectively. The researcher pointed out the possible reasons for the false alarm. Incorrect POS tagger's output was the first reason the researcher mentioned on the paper. The POS tagger assigns incorrect POS tag for some words. The second reason was production of wrong affixes by the stemmer. The rule is incomplete (didn't not cover every case) and this also makes false alarm to occur in the grammar checking. As most of the rules are constructed for simple sentences, most of the false flags are on complex and compound sentences.

2.6.2 English Grammar Checker

Number of grammar checker researches has been conducted on English language including Microsoft office grammar checker. One of these researches is done by Daniel Naber [6]. The

aim of the paper 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 54 pre-defined grammar rules, which are simply a sequence of tokens to be matched. 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 16 errors in 75,900 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 42 errors whereas MS-word detects 49 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.

Grammar Checking with Dependency Parsing is a Possible Extension for style and grammar checker language tool. The research identifies a 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 [11].

For example,

To handle incorrect use of article, the style and grammar checker grammar checker language tool handles it using the rule shown in Listing 2.1.

```
<!—"a/an" article, then a plural noun -->
<pattern>
<token regexp="yes">a|an</token>
<token postag="NNS|NNPS"</token>
<message >Don't use indefinite articles with plural words.</message>
</pattern>
```

Listing 2.1 : Sample XML rule in style and grammar checker

The rule can only handle two token sequence, the first with a word ‘an’ or ‘a’ and the last token noun. This rule cannot catch errors from sentence or noun phrase that contain other words between article and the noun like in a phrase ‘a pretty little girls’. This incorrect phrase cannot be detected by the style and grammar checker using this rule, unless there is a modification on the rule to include all possible adjectives and other words that can probably occur. [11] considers this solution as insufficient and proposed another solution for this problem: grammar checking with dependency parsing.

Grammar checking with dependency parsing considers the nonlinear structure of the phrase. Dependency parser is used to obtain nonlinear structure of the phrase. The nonlinear structure of every sentence represented as parse tree by the dependency parser. Using this method, two words can be linked together without considering the other words between the two. The parse tree of the sentence can extend the syntax of the rule-based style and grammar checker language tool. Three sub-problems were identified in the grammar checking with dependency

parsing research: identifying the dependency parsing system, developing syntax for the dependency based rules and design rule matching algorithm. A two open source high quality parsing system, both have the same input and output format, was selected to solve the first problem. In developing the dependency based rules, the rules should provide syntactic means for three basic functions. First one is matching link between two given words. Second, checking word precedence, whether a word appears before or after another word. Third and the last is checking the absence of a given sub tree in the parse tree. The rule definition is split into chunk. Each chunk represents a separate subtree to be matched. This means each chunk of a rule is matched one by one. The depth-first-search routine is implemented as rule the matching algorithm to solve the third sub problem which is designing the rule matching algorithm.

N-gram based statistical method of grammar checker was introduced in [17]. It was done for the language Bangla and English. The method works in a way that checks the probability of the "N" words occurring together in the corpora. The probability of the sequence of words is calculated using equation (2.4) (See Section 2.1.4).

For example, when n= 2, which is bigram case, the system checks how often each pair occurs in the corpora. After obtaining the probability of each pair from the corpora, the probability of the sentence is calculated using the equation (2.5) (See Section 2.1.4). If the probability of the sentences is above some threshold, the sentence will be considered as grammatically correct.

In the bangle and English grammar checker model, if probability is greater than zero then it considers the sentence as 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 probability is zero, the sentence probability will become zero (property of zero multiplication).

To show how the system works with example, the sentence "He is playing" is taken from the paper [17]. To check whether the sentence "He is playing" is correct or not, using bigram, the probability of the sentence is calculated as follows.

$$\mathbf{P}\ (\text{“}\mathbf{He\ is\ playing}.\text{”}) = \mathbf{P}\ (\mathbf{He} \mid < start >) * \mathrm{P}\ (\mathrm{is} \mid \mathrm{He}) * \mathrm{P}\ (\mathrm{playing} \mid \mathrm{is}) * \mathrm{P}\ (. \mid \mathrm{playing})$$

The probability of each pair will be obtained form the corpus using the probability formula (2). Each pair probability will be multiplied to calculate sentence probability. If a single word does

not exist in the corpus the probability of the whole sentence will become zero (because the probability of the word pair is zero).Then, the sentence is considered incorrect even for the correct sentence. However, considering the probability of the word as whole may cause the sentence to be incorrect as every word may not exist in the corpus. To come across this problem, the researchers consider Part-Of-Speech (POS) tags instead of the words. Taking the above example again, "he is playing" will be replaced by the POS tags "pps bez vbg ." .

$$\mathbf{P}\ (\mathbf{pps\ bez\ vbg}\ .) = \mathbf{P}\ (\mathbf{pps} \mid < start >) * \mathrm{P}\ (\mathrm{bez} \mid \mathrm{pps}) * \mathrm{P}(\mathrm{vbg} \mid \mathrm{bez}) * \mathrm{P}(. \mid \mathrm{vbg})$$

Therefore, in the n-gram based statistical grammar checker for Bangla and English, first each word in a sentence assigns to its POS tag. Then, it uses the 'N' value to determine the sequence probability of the words. 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 63% (detected 545 sentences as correct, out of 866 correct sentences). Using manually tagged corpus, the grammar checker's performance is 53.7% (detected 203 sentences out of 378 correct sentences) in Bangla. The performance of the grammar checker also tested using 34 automatically tagged sentences for Bangla. Then, the grammar checker produced result with lower performance which is 38% correct result.

The researcher stated on the paper 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.

2.6.3 Portuguese Grammar Checker

One of the grammar checkers for Portuguese language is COGrOO [21]. It is a Portuguese grammar checker based on CETENFOLHA a Brazilian Portuguese morphosyntactic annotated

Corpus. Nominal and verbal agreement, nominal and verbal government, misuse of adverb and adjectives are some of the problem that the research designed to solve. There are two rule set in the system: local and structural error rules. Local rules includes the short sequence of words rules whereas the structural rules consists of the more complex rules such as nominal and verbal agreement, nominal and verbal government, misuse of adverb and adjectives. The input text is first broken into sentence in sentence boundary detector module. Words in the sentence will be morphologically tagged. Then chunker will chunk the tagged sentence to small noun and verbal phrases. The grammatical relation finder finds the relation between the noun and verbal phrases and their grammatical roles (subject, object and verb). The local error checker and the structural error checker check for the local and structural error respectively.

2.6.4 Punjabi Grammar Checker

Punjabi is a language spoken in Indian and Pakistan. It is a member of Indo-Aryan family of language. A Punjabi grammar checker is explained in [16]. It's a rule-based system. In Punjabi grammar checker, grammatical errors such as modifier and noun agreement, subject and verb agreement, noun and adjective, order of modifier of noun in a noun phrase, order of verb in a verb phrase and the like. To detect the errors the system passes through few steps or phases. Initially, pre-processing task is done on the input text which is tokenization. Then morphological analysis will be performed after the tokenization phase. The rule-based part of speech tagger is engaged to disambiguate in the tags. Then the text is grouped into phrases based on the phrase chucking rule. Finally, using the grammatical error checking rule, grammatical errors internal to the phrases and the sentences will be identified and correction suggested [16].

2.6.5 Language Independent Statistical Grammar Checker

A language independent research on grammar checkering – LISGrammarChecker - is also done by Verena Henrich and Timo Reuter. It uses statistical approach to detect grammatical errors. The grammar checker uses mainly n-gram approach for the grammar checking. A statistical database is built from a large amount text. Using the statistical data in the database, the grammatical errors in the text can be detected. The database can be updated when there is a change in the corpus. In addition to N-gram, LISGrammarChecker also includes morphosyntactic features to checker agreement between words i.e. adjective and noun, adverb

and verb.

In LISGrammarChecker, there are two different workflows. The first one is training the system and the other is grammar checking. The training mode accepts training corpus as an input. The input training corpus will be tagged for further processing. The important step in LISGrammarChecker training is data gathering. In this step, the statistical data is collected from the corpus and saved permanently in the statistical database. Two up to five consecutive token and tags (bigram to pentagram) are extracted from the text. The amount of occurrence of these n-grams will also be saved for every entry in the database. Figure 2.3 illustrates the workflow of LISGrammarChecker training mode.

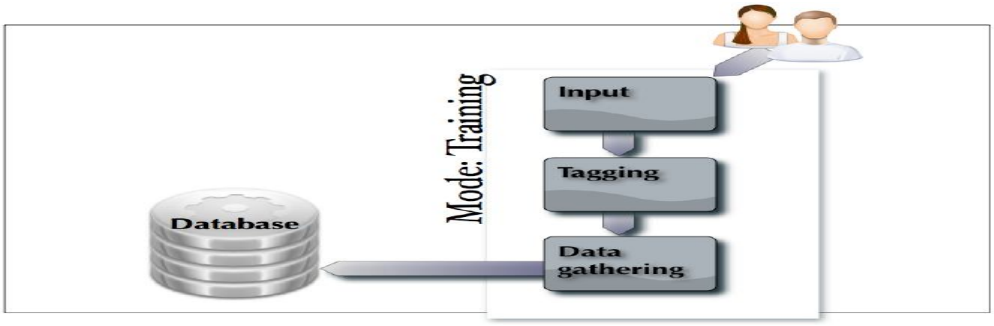


Figure 2.63 : Architecture of LISGrammarChecker training

The second mode, in LISGrammarChecker, is the grammar checking. The architecture of grammar checking phase is shown in Figure 2.4. The grammar checking is done using the stored data in the training mode. As in training mode, the input in grammar checking mode is text (text to be checked and possibly with errors). The input text is tagged in the tagging module. The grammar checking methods accepts the tagged text as an input and performs different grammar checking operation on the input tagged text. The grammar checking methods are token n-gram check, tag n-gram check, Internet functionality, Adverb-verb-agreement and adjective-noun-agreement. All methods use different statistical information. The Internet functionality is no complete checking method, but rather an extension for the token n-gram check which can be activated optionally. In this case, the statistical data is extended with n-gram search results from an Internet search engine. In order to check the grammar, the input text is required to be analyzed with respect -to the n-gram (bigram to pentagram). The extraction starts by pentagram, follows the quadgram and so on. The error in the input text is the smallest incorrect n-gram. All checking methods can detect distinct errors if any entry is

not in the database. All detected errors are only handled as error assumptions. A true error is defined when enough error assumptions occur. True errors are calculated in the error counting component. The error counting uses the detected error assumptions to calculate the over all error and define the true error. Each error assumption has an individual weight. The summation of all error assumption in a sentence is an over all error amount in a sentence. This error amount is compared with error threshold to decide on the correctness of the sentence. If the error amount is above the threshold, the sentence is grammatically incorrect. The sentence may have errors detected on it. However, if the error amount is less than error threshold, the sentence considered grammatically incorrect. This can possibly minimize the false positive.

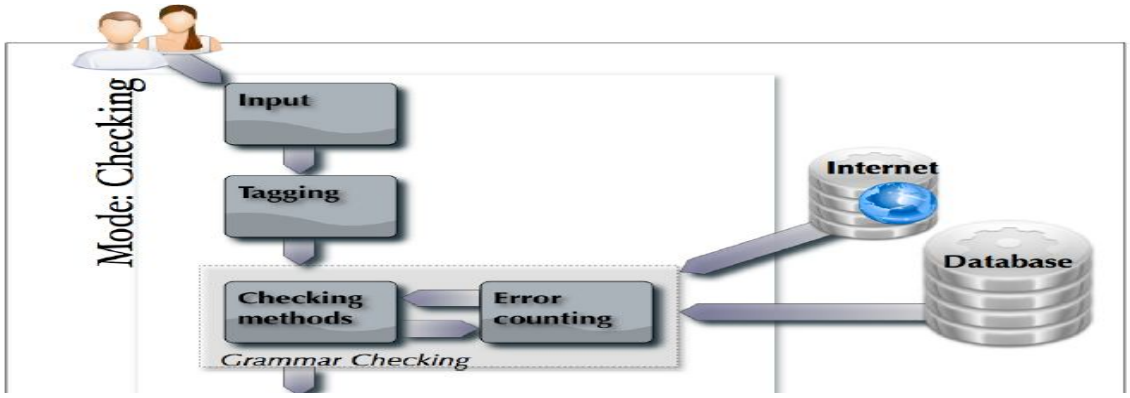


Figure 2.4 : Architecture of Grammar checking in LISGrammarChecker

The LISGrammarChecker is tested with five test cases. The first test case is used to evaluate different requirements using different test set. The test results reveal that one fourth, one third, three fourth of the errors are detected by the tag, hybrid (both token and tag),token n-grams respectively. In the false positives case, the best result is achieved by the tag n-gram check where only few false positives occur. Test case four is used to test that LISGrammarChecker can be used in different language. German language is tested to show the language independency. The tagset used in this case is not large in size. As German language is morphologically more complex and thus there can be more mistakes due to disagreements. Results show that it works and all errors in a sentence are detected.

CHAPTER THREE

AMHARIC GRAMMAR CHECKER

This chapter tries to briefly discuss the two approaches that have been used to check grammatical errors of Amharic sentences. The first approach is a rule-based Amharic grammar checker. The rule-based Amharic grammar checker is developed using manually constructed rules and the rules are tested on Amharic simple sentences. The statistical Amharic grammar checker is the second approach used to check both simple and complex sentences. N-gram and other probabilistic method are applied in statistical Amharic grammar checker. The first section of this chapter presents Amharic language grammar. The detailed discussion about rule-based and statistical Amharic grammar checker is also presented in second and third section respectively.

3.1 Amharic Grammar

This section explains Amharic language and its grammatical structure in terms of the word classes and their part in sentence formation. This section gives only the emphasis on Amharic grammar, the more detail on this area can be obtained on [33] and [34].

3.1.1 Background

Amharic is a Semitic language spoken in eastern Africa mostly in Ethiopia. It is also the official language in the countdsry, Ethiopia. Amharic has its own alphabet known as 'ፊደል' /*'fidel'* which is written from left to right. In each 'ፊደል'/*'fidel'* or alphabet, the vowel and consonants exist at the same time. In Amharic writing system, there are thirty-three consonants and seven orders (for each consonant) by combining each consonant with different vowels [4].

3.1.2 Amharic Morphology

The one of the main reasons that makes Amharic language complex is its morphology. Amharic language is one of the morphologically rich Semitic languages. A single word (verb) in Amharic may indicate the details of the subject and object of the sentence. A word in Amharic can also be considered as a sentence in other language like English. Example, the word 'አልነገረኝም'/*'alnegereNm'* in Amharic, which means 'he didn't tell me' in English, specifies the subject (he) and the object (me) in a word.

Amharic words grouped into eight main word classes i.e. noun, verb, adjective, adverb, Preposition, Pronoun, Interjection [33, 34]. By the affixation process or adding affixes to the words, grammatical class of the word can be changed i.e. noun to adjective and verb, verb to adjective and noun and the like. This process of forming words without changing the word class is known as derivational morphology [30]. Amharic word of one class can be derived from the class itself and different other classes. For example, '*ደግነት*'/*'degnet'* is noun which is derived from the adjective '*ደግ*'/*'degt'* and morpheme '*-ነት*'/*'-net'*. Morphemes or affixes can also be added to a word to mark syntactic function of the word by keeping its original class. This way of forming a word is called inflectional morphology [30]. Amharic noun and adjective can be inflected for number (plural, singular), gender (male, female), definiteness (indefinite, definite) and case (objective, subjective, possessive). Amharic verbs can be inflected for person (first, second, third), gender, number, cases and tense/aspect, and mood (command, request, negative). For example, the verb '*መጣ*'/*'meTa'* can be inflected as '*መጣች*' /*'meTac'* to mark third person, singular feminine subject.

Amharic words can be inflected by adding different affixes (prefixes, infixes and suffixes) to the words. These affixes placed in the words to mark definiteness, negativity, subject and object, conjunction and the like. The prefix and the suffix for Amharic verbs can be sited in four slots i.e. four slots before and four slots after the stem. Before the stem (see Figure 3.1), preposition or conjunction can be placed on the first slots of the prefix, then relative marker takes the second slot, negative marker at the third slot and on the forth slot the subject marker will be placed. When it comes to the suffix, the object or the definite marker, negative or auxiliary or accusative and conjunction in first, second, third and forth slots respectively.

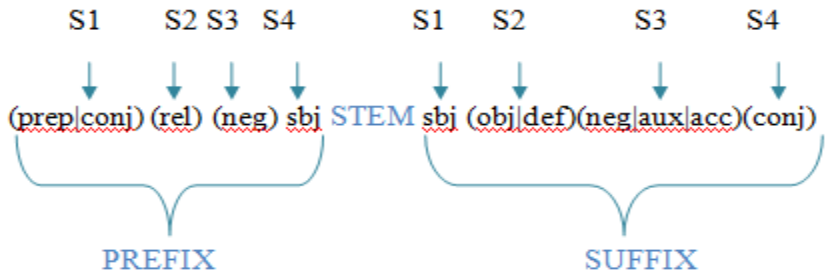


Figure 3.1 : Placement of affixes (prefix and infix) in a word

Awgichew and Meyer wrote on [35] that Tekle-Maryam [36] identified three tenses for

Amharic verbs as past, present and future. Awgichew and Meyer put these tenses with three special "cultures": indefinite, definite and continuous and get seven tenses. Table 3.1.1 summarizes the seven tenses by inflecting word for person, number and gender. Present and future tenses in Amharic use the same morpheme. Therefore, it is necessary to use the adverbs to distinguish those tenses. For example, the tense for the verb '*ይመጣሉ*' */'yemeTalu'* can not be distinguished unless an adverb that shows future or present time added to it: '*አሁን ይመጣሉ*'/ *"ahun ymeTalu'*, '*ነገ ይመጣሉ*' */'nege ymeTalu'* , '*ዛሬ ይመጣሉ*' */'zare ymeTalu'* and the like .

3.1.3 Amharic Sentence

A sentence is a collection of words expressing a judgment of the mind. A sentence should not be incomplete like a phrase to express something. Amharic sentences classified into two: simple and complex sentence. A simple sentence is a sentence which has only one verb phrase. Whereas a complex sentence is a sentence which is categorized as simple sentence and formed by complex phrases.

A general Amharic sentence structure is subject object verb (SOV). But sometimes sentences may occur as OSV. For example, the sentence '*አበበ ከበደን ገደለው*'/*'abebe kebeden gedelew'*, which is in SOV order, can also be written as '*ከበደን አበበ ገደለው*' /*'kebeden'abebe gedelew'* in OSV order. Taking these sentences into consideration, one can only be sure about the verb that appears/placed at the end of the simple sentences and the subject and the verb may exchange position. However, the OSV order can not be used in formal Amharic texts. The order of words in Amharic is not only bounded to subject, object and verb. Other word classes (like adjective, adverb) also have their own arrangement in a sentence. For example, an adjective should appear before the noun it modifies.

Depending on placement of words in a sentence, the sentence meaning can be changed unless the word (object of the sentence) has the object marker '*-ን*'/*'-n'*. For instance, '*ጅብ ውሻ ይበላል*' /*'jb wuxa ybelal'*and '*ውሻ ጅብ ይበላል*' /*'wuxa jb ybelal'*: the words used in both sentence are the same but different meaning. '*ጅብ*' /*'jb'* and '*ውሻ*' /*'wuxa'* are the subject of the sentences in the first and second sentence respectively. Amharic nouns do not have different subject marker or morpheme (affix) [34]. However, a subject can be identified from its place in a sentence. '*-ን*' /*'-n'* is suffix that is used as a sign for Amharic objects. A noun with this suffix can be easily identified as an object. Other than the order and placement of words in a sentence, the Amharic words must also agree with each other. For example, the subject and object with the verb (in

number, gender and person), adjective with noun (in number and gender), adverb with verb (in time) and so on [4, 34, 35].

Table 3.1 : Amharic tenses

Person	Number	Gender	Jussive (Command)	Perfective (simple past)	Near past	Gerundive (Remote past)	Imperfective (present/ future)	Near future	Past continuous
First	Singular		ልግደል / *lgdel*	ገደልኩ / *gedelku*	ገድያለሁ / *gedyalew*	ገድዬ ነበር / *gedyE neber*	እገድላለሁ / *'gedlalehu*	ልገድል ነው / *lgedl new*	እገድል ነበር / *'gedl neber*
	Plural		እንግደል / *'ngdel*	ገደልን / *gedeln*	ገድለናል / *gedlenal*	ገድለን ነበረ / *gedlen nebere*	እንገድላለን / *'ngedlalen*	ልንገድል ነው / *lngedl new*	እንገድል ነበር / *'ngedl neber*
Second	Singular	Feminine	ግደይ / *gdey*	ገደልሽ / *gedelx*	ገድለሻል /*gedlexal*	ገድለሽ ነበረ / *gedlex nebere*	ትገያለሽ / *tgeyalex*	ልትገድል ነው / *ltgedl new*	ትገድል ነበር / *tgedl neber*
		Masculine	ግደል / *gdel*	ገደልክ / *gedelk*	ገድለሃል / *gedlehal*	ገድለህ ነበረ / *gedleh nebere*	ይገድላል / *ygedlal*	ሊገድል ነው / *ligedl new*	ይገድል ነበር / *ygedl neber*
	Plural		ግደሉ / *gdelu*	ገደላቹ /*gedelacu*	ገድላችዋል /*gedlacwal*	ገድላቹ ነበረ /*gedlacu nebere*	ትገላላቹ /*tgelalacu*	ልትገድሉ ነው / *ltgedlu new*	ትገድሉ ነበር / *tgedlu neber*
	Polite		ግደሉ / *gdelu*	ገደሉ /*gedelu*	ገድለዋል / *gedlewal*	ገድለው ነበረ / *gedlew nebere*	ይገድላሉ / *ygedlalu*	ሊገድሉ ነው / *ligedlu new*	ይገድሉ ነበር / *ygedlu neber*
Third	Singular	Feminine	ትግደል / *tgdel*	ገደለች / *gedelec*	ገድላለች / *gedlalec*	ገድላ ነበረ / *gedla nebere*	ትገድላለች / *tgedlalec*	ልትገድል ነው / *ltgedl new*	ትገድል ነበር / *tgedl neber*
		Masculine	ይግደል / *ygdel*	ገደለ / *gedele*	ገድሏል / *ged lWal*	ገድሎ ነበረ / *gedlo neber*	ይገድላል /*ygedlal*	ሊገድል ነው / *ligedl new*	ይገድል ነበር /*ygedl neber*
	Plural		ይግደሉ / *ygdelu*	ገደለ / *gedele*	ገድለዋል / *gedlewal*	ገድለው ነበረ *gedlew nebere*	ይገድላሉ / *ygedlalu*	ሊገድሉ ነው /*ligedlu new*	ይገድሉ ነበር / *ygedlu neber*
	Polite		ይግደሉ / *ygdelu*	ገደለ / *gedele*	ገድለዋል / *gedlewal*	ገድለው ነበረ / *gedlew neber*	ይገድላሉ / *ygedlalu*	ሊገድሉ ነው / *ligedlu new*	ይገድሉ ነበር *ygedlu neber*

3.2 Rule-based Amharic Grammar Checker

The rule-based Amharic grammar checker is the first approach used to check whether a given Amharic sentence is grammatically correct or not. The rule-based Amharic grammar checker has a language model which contains grammar rules. This language model was developed using manually constructed rules and tested using Amharic text. To the best knowledge of the researcher, there is no Amharic grammar rule written by the linguists and can be used directly or with a simple modification for this research. For this reason, each rule in the language model

is developed manually by considering grammatically incorrect Amharic sentences. Amharic grammar books by Baye Yimam[34], Getahun Amare[33] and linguistic experts assistance have also great contribution on constructing the rules for the rule-based Amharic grammar checker. As developing manual rules for Amharic complex sentences is difficult, the rules on the rule-based Amharic grammar checker are constructed only for simple sentences. Due to this and other reasons stated in section 3.2.3, the rule-based Amharic grammar checker could not be fully designed and tested for complex sentences.

The rule-based Amharic grammar checker accepts Amharic text containing sentences in SOV order as an input. The input text split into sentences, then each sentence to words. Amharic subjects do not have subject marker. Therefore, the rule-based Amharic grammar checker identifies the first noun as the subject and the second noun as an object of the sentence. Then, every word will be analyzed using Amharic Morphological Analyzer (HornMopho). Based on the analysis, grammatical relation is built for each sentence. Grammatical relations (the patterns) will be matched against the rules in the language model. Each rule in the language model shows grammatically incorrect Amharic sentence structure. If any of the grammatical relations matches with one or more rules, the sentence with that grammatical relation will be identified as grammatically incorrect sentence. Those that are not matched against the rules will be considered as grammatically correct. Below, in section 3.2.1, the architecture of the rule-based Amharic grammar checker is described as follows.

3.2.1 Architecture of Rule-based Amharic Grammar Checker

The rule-based Amharic grammar checker accepts Amharic text as an input and identifies grammatically correct and incorrect sentences from the input text and shows the result as an output. The architecture of the rule-based Amharic grammar checker is shown in Figure 3.2 and the steps shown in the architecture are described one by one as follows:

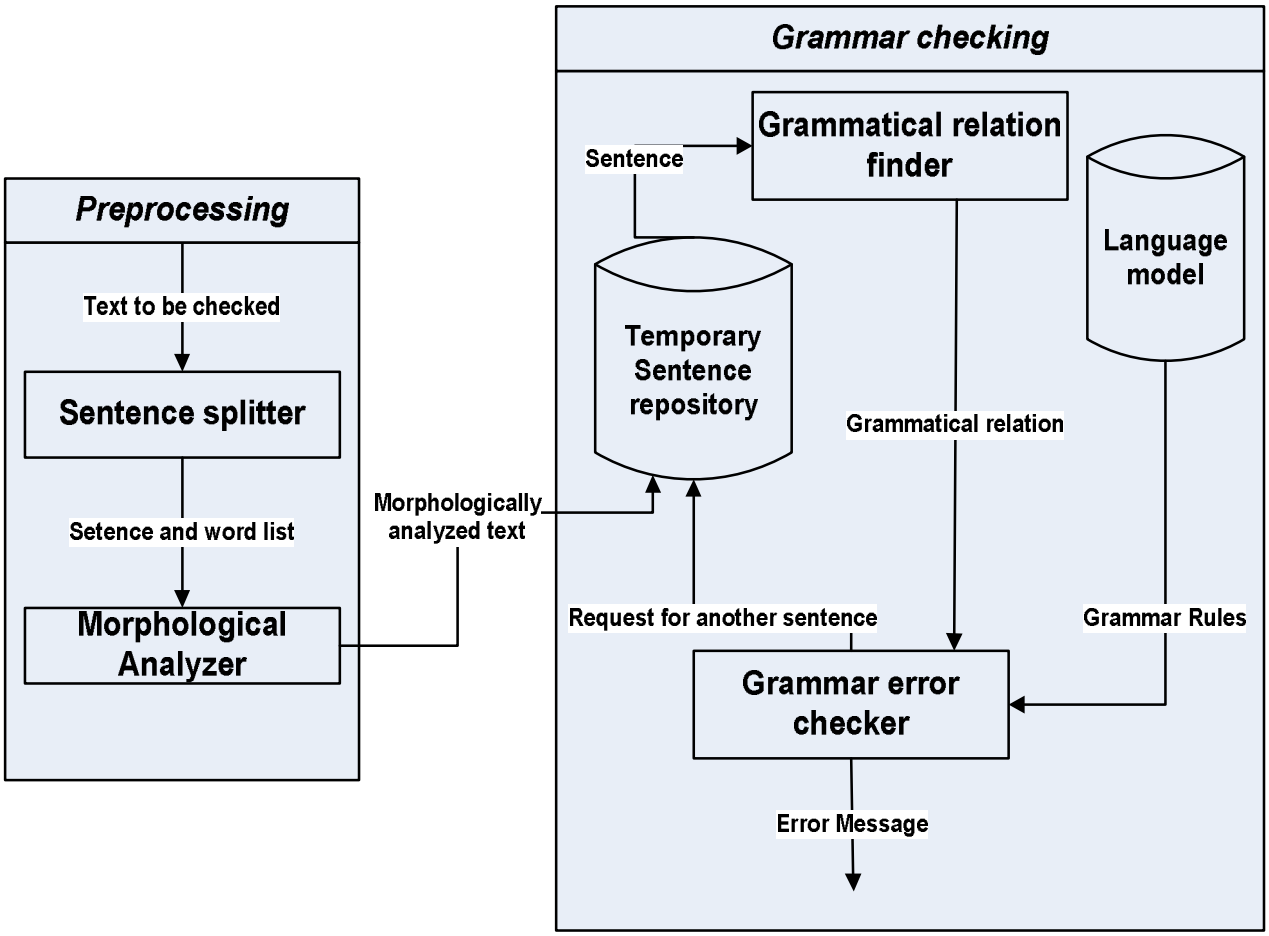


Figure 3.2: Architecture of rule-based Amharic grammar checker

Sentence Splitter

The sentence splitter module splits the input text into sentences and the sentence into words. The input text has punctuation mark at each end of the sentence. This punctuation mark is used in Amharic language to indicate sentence end. Using this punctuation mark as split indicator, input text will be split into sentences for further processing. After the input text is split into sentences, each sentence will be kept in temporary repository. From the sentence temporary repository, each sentence will be split into words to be processed in the morphological Analyzer.

Morphological Analyzer

Morphological analyzer module accepts the split words from the sentence splitter and assigns linguistic meanings to each word. Each affix (prefix, infix, and suffix) or morpheme in word refers to different linguistic meaning such as number, person, gender, definiteness and the like. Morphological analyzer should be used to know what linguistic meaning that a given affix refers to.

In the rule-based Amharic grammar checker, each word in the input text to be checked is analyzed using HornMopho Amharic morphological analyzer [8]. The HornMorpho recognizes two POS tags i.e. noun and verb. If the word is a noun, linguistic meanings such as number, person, gender that noun belongs will be determined using the HornMorpo. This will be done for both the subject and the object. On the other hand, the verb should be analyzed for both the subject and the object markers because number, person and gender of the subject and object of the sentence are reflected on the verb as well. Therefore, when the verb analysis is done using HornMorpho the subject and object marker on the verb will be defined in terms of number, person and gender.

For example,

In the sentence '**እሱ እሷን መታት**' /*"su 'sWan metat'*, **እሱ**/*"su'* is subject (singular in number, third person masculine). **እሷን**/*"sWan'* is object (singular, third person feminine). This detail of the subject and object must agree with the subject and object reflection on the verb. The verb '**መታት**' /*'metat'* has the same number, person and gender markers with its subject and object preceding it.

Grammatical Relation Finder

Grammatical relation finder module assigns grammatical relation between words in a sentence such as the subject and verb, the object and verb and so on. The grammatical relation is constructed based on the analysis result of the Morphological analyzer. Grammatical relation finder assigns subject-verb and object-verb relation in terms of number, person and gender. The relation between the words is represented in the form of word pattern. Numbers of Patterns are constructed for the sentence. Each pattern has sentence start and sentence end marker. The constructed patterns (grammatical relations) are used to be matched with the rules in the

language model. The following example explains how grammatical relation is built for a given sentence in terms of number, person and gender.

For example:

The sentence ‘እሱ እሷን መታት’ /*”su ‘sWan metat’* can be put in three different word patterns for number, person and gender. The following three lines show how the pattern for the rule-based grammar checker appears for the sentence. Line one, line two and line three are for number, person and gender respectively.

'SStart NSng NSng VSbSng VObSng SEnd',

'SStart NP3 NP3 VSP3 VOP3 SEnd',

'SStart NM NF VSbF VObM SEnd'

To identify the beginning and the ending of the sentence, each word pattern includes the string “SStart” and “SEnd” for sentence start and end respectively. The strings “NSng”, “NP3” and “NM” which appear right after the sentence start markers refer to the subject of the sentence ‘እሱ’ /*”su’*. “NSng”, “NP3” and “NM” represent linguistic meanings that the subject ‘እሱ’ /*”su’* is noun singular, third person and masculine respectively. The string after the subject description “NSng”, “NP3” and “NF” is for the object of the sentence ‘እሷን’ /*” ‘sWan’*. It means the object is third person singular feminine. The last two strings before the sentence end refers to the verb in the sentence i.e. “VSbSng VObSng”, “VSP3 VOP3” and “VSbM VObF”. The first one “VSbSng VObSng” defines the number and represents linguistic meaning that the verb has singular subject and object marker. The second “VSP3 VOP3” is about person and states that both the subject and the object marker in the verb are third person. The last one “VSbM VObF” defines the gender marker in the verb and it’s feminine for the subject and masculine for the object. This way, the grammatical relation finder constructs grammatical relations for a sentence. The grammatical relations for a sentence will be ready to be sent as an input to grammar rule checker.

Language Model

The language model is a storage that holds the grammar rules. The grammar rules in the model are manually constructed rules. Rules describe what must not occur in Amharic sentences or incorrect grammatical sentence structure in the form of words pattern. This means, if a sentence matches with any of rules then it will be considered as grammatically incorrect sentence for each of unmatched rule. The rules are regular expression and written in XML. The root element for the rule set is "rules". The root element "rules" has two children named "rule" and "rulegroup". The rule has three sub elements: "pattern", "message" and "example". "rulegroup" has set of rules inside with rulegroup id.

- <rules> is the root element for the elements inside the XML that contains the grammar rules.

- <rule> is the sub element for the root element. It defines each rules pattern with its message and examples. It has an attribute id and name. The id uniquely identifies each rule and helps in turning on and off rules separately when required. The name attribute describes what the rule is. Both the id and the name are used for internal use.

- <pattern> is sub element for the <rule>. It defines the word pattern for grammatically incorrect sentence structure. The pattern is simply the different expression for the linguistic meanings of words in the sentence. The expression for the word may be based on number, gender or person. The pattern element contains regular expression to be matched with the grammatical relation or pattern of the sentence to be checked. The word pattern for the sentence to be checked is extracted in the grammatical relation module. The extracted patterns will be matched with the regular expression in the <pattern> element.

- <message> is another sub element for the <rule> and it contains description about the current rule declared as parent of the <message>. <message> element includes text value which describe the rule that the message belongs.

- <example> is third sub element for the <rule> element. It has 'type' attribute which has a value "correct" and 'incorrect' for correct and incorrect example respectively. In this element, additional description can be kept about the error with two different type

examples of the same sentence

Listing 3.1 shows one sample rule in the rule-based Amharic grammar checker. The rule is represented in XML and it contains the above XML elements.

```
<rule id="_NpVSbS" name="Subject/Verb agreement in number (subject plural and verb singular)">
<pattern >
SStart NPlr (NPlr|NSng) VSbSng (VObPlr|VObSng) SEnd
</pattern>
<message>
The plural subject should have a plural marker on the verb
</message>
<example type="correct”> በሬዎቹ ልጅቷን ወጓት/’berEwocun ljtWan wgWat
</example>
<example type="incorrect"> በሬዎቹ ልጅቷን ወጋት/”Na ‘sWan ‘wedatalehu’
</example>
</rule>
```

Listing 3.1 : An Example of how rules in the language model represented in XML

The sample rule in listing 3.1 contains subject-verb agreement rule. The rule element has sub elements that are used to describe the rule defined. The rule defines subject-verb agreement in number i.e. when the subject is plural and the subject marker on the verb refers to singular subject. The <pattern> element defines incorrect word patterns in a sentence using regular expression. <message> gives explanation about the rule that matches the pattern in the <pattern> element. The <example> element contains two type of the same sentence. One contains error of the current rule and the other correct Amharic sentence. By observing those examples, one can get addition description about the rule to understand the errors in the sentence.

Grammar Rule Checker

Grammar rule checker module matches the patterns extracted in the grammatical relation finder module against the rules in the language model. The grammatical errors on the sentence will be checked based on the rules in language model. If extracted pattern for the sentence matches against the pattern in the language model, the sentence will be considered as grammatically incorrect. On the other hand, the sentence will be grammatically correct if the pattern for the sentence does not match with the any of the rules.

Example:

Consider the sentence "እኛ እሷን እወዳታለሁ"/*"Na 'sWan 'wedatalehu'* as sentence to be checked using the rule-based Amharic grammar checker.

The morphological analyzer will analyze each word and assigns linguistic meaning to each word. The subject in the example sentence is plural in number and the object is singular. On the other hand, the subject marker and object markers on the verb is singular.

In the grammatical relation finder module, the number relation is constructed based on the morphological analyzer result. The number relation for the sentence is kept in the form of word pattern. The number pattern for the sentence "እኛ እሷን እወዳታለሁ"/*"Na 'sWan 'wedatalehu'* is 'SStart NPlr NSng VSbSng VObSng SEnd'. This pattern matches with the rule pattern in Listing 3.1. Therefore, the sentence is identified as grammatically incorrect because it has subject-verb disagreement in number.

Sentence: እኛ እሷን እወዳታለሁ /*"Na 'sWan 'wedatalehu'*

Message: the plural subject should have a plural subject marker on the verb

Example: Correct -በሬዎቹ ልጅቷን ወጓት/*'berEwocun ljtWan wgWat*

Incorrect - በሬዎቹ ልጅቷን ወጋት/*"Na 'sWan 'wedatalehu'*

3.2.2 Agreement Checking in the Rule-based Grammar Checker

The rule-based Amharic grammar checker system accepts subject, object and verb of a sentence i.e. sentence in SOV order and check for their agreement in number, person and gender. The agreement is checked for both subject-verb and object-verb in all the three cases (number, gender and person). The next two sections explain how subject-verb and object-verb disagreement checking works.

I. Subject- verb Disagreement Checking Using Rule-based Amharic Grammar Checker

As stated above, the system first accepts sentences that contain subject, object and verb of the sentences. After sentence and word splitting, every word will be analyzed. During the words analysis, every word will be assigned with its own analysis result based on number, person and gender. The nouns and verbs are analyzed in different form. Nouns should be analyzed for number, person and gender whereas verbs should be analyzed both for the subject and object marker based on number, person and gender. After each word is assigned based on the analysis, then the subject label of the sentence will be matched with the subject analysis part of the verb in the sentence. The next example explains how subject-verb agreement checking works.

Example: በሬዎቹ በጉን አባረርከው/*'berEwocu begun 'abarerkew'*

In this sentence, two disagreements can be identified. The first one is number mismatch between the subject and the verb. The subject '**በሬዎቹ**' /*'berEwocu'* is a singular noun and the subject marker on the verb marks plural. The other one is a subject-verb disagreement on person. The subject '**በሬዎቹ**' /*'berEwocu'* is a plural third person whereas the subject marker on the verb "**አባረርከው**"/*''abarerkew'* marks the second person "**አንተ**"/*''ante'* "you".

In rule-based Amharic grammar checker, after the sentence and word splitting, the system will run the following two modules.

Grammatical Relation Finder

In the grammatical relation finder module, the sentence will be assigned into three different the patterns: number, person and gender.

'SStart NPlr NSng VSbSng VObSng SEnd',

'SStart NP3 NP3 VSP2 VOP3 SEnd ',

'SStart NM NM VSbM VObM Send'

In the first line, the pattern describes the sentence based on number. The first noun, which is the subject, is plural noun 'NPlr'. The second is singular 'NSng', the object of the sentence. The last two, 'VSbSng' and 'VObSng' are for the verb of the sentence, describe the subject singular and object singular marker on the verb.

The second line contains the person pattern for the sentence. 'NP3' and 'NP3' are the subject and the object nouns respectively. Both the subject and object is third person noun. 'VSP2' and 'VOP3' explains about the verb in the sentence: the subject marker in the verb 'VSP2' is second person whereas the object'VOP3' is third person.

The last line holds gender information about the sentence. The pattern tells the system that both subject and the object have a masculine gender: 'NP3' and 'NP3'. The verb has masculine object and subject marker: VSbM VObM.

Grammar Rule Checker

The grammar rule checker module matches the pattern produced in the grammatical relation finder module with the rules kept in the language model. If the pattern matches with the rule, the system considers the pattern as an error. Therefore, for the above sentence, the first two patters matches with the manually constructed rules in the language model and the gender pattern does not match with the rule. As the first, second and third pattern describe the number, person and gender respectively, the sentence has subject-verb mismatch on the number and person.

II. Object–verb Disagreement Checking Using Rule-based Amharic Grammar Checker

To check the object- verb disagreement the system passes all the steps explained in the subject-verb disagreement checking part (I). To explain this with example,

Example: The sentence ‘በፊው በጉን ወጋት’/*’berEw begun wegat’*

This sentence contains object-verb disagreement in gender. The object ‘በጉን’/*’begun’* has a masculine marker in it but the verb has feminine object marker which disagrees with the object in the sentence.The sentence first split into words, then after each word will be analyzed using a HornMorpho Amharic morphological analyzer. Based on the analysis result the following modules will be done.

Grammatical Relation Finder

The grammatical relation finder marks the start and end of the sentence and discribs the sentence in three different patterns based on number, person and gender. These three lines describe the number, person and gender pattern of the sentence in the example above.

'SStart NSng NSng VSbSng VObSng SEnd',

'SStart NP3 NP3 VSP3 VOP3 SEnd',

'SStart NM NM VSbM VObF SEnd'

Grammar Rule Checker

Patterns identified in the grammatical relation finder module are the input for Grammar Rule Checker module which checks whether the patterns exist in the language model or not. As stated above, the language model contains list of patterns which considered as error in Amharic grammar. In short, if pattern matches with the rules in a language model, the pattern is considered as grammatically incorrect.

3.2.3 Problems on this approaches

1. To the best knowledge of the researcher, Amharic grammar rules is not described by linguistic experts in the way that can be used as either a direct input or with a simple modification to this research. There are Amharic grammar books written by linguists. The books explains Amharic letter sounds, word formation, phrase formation, different types of sentences and their sentence formation but do not clearly state what makes Amharic sentences grammatically incorrect. As rule-based approach requires manually developed rules, unavailability of such rules or structures make this approach difficult to complete.

2. The other problem which affects the rule-based Amharic grammar checker completion is the Amharic POS tagger. Amharic POS tagger is one of the tool requires for the development of grammar checker (See Section 2.4). High performance Amharic POS tagger, which can identify number of POS tags, is not available to be used with this research. HornMopho Amharic Morphological Analyzer only identifies nouns and verbs: even adjectives are considered or analyzed as Nouns.

3. It's very difficult to deal with the rules of Amharic complex sentences. Amharic complex sentences are usually very long and they contain numbers of verbs and connectors such as "ቢሆንም"/*'bihonm'*, "ስለዚህ"/*'slezih'*, "ስለሆነም"/*'slehonem'*, and the like. Therefore, rules for this kind of sentence are difficult to develop manually. The problem listed in (1) makes the rule construction for complex sentences even harder.

3.3 Statistical Amharic grammar checker (STAMGRAM)

As the rule-based Amharic checker could not go further to include complex sentences, the n-gram based Statistical Amharic grammar checker is implemented and tested for both simple and complex sentences. In this system, two major tasks are performed. The first one is training the system by extracting patterns with different 'N' values (bigram and trigram) and the second one is checking grammatical error using the extracted pattern set. In extracting the patterns, the extracted patterns with their probability of occurrence are kept in repositories i.e. different repository for different N values. The patterns that are saved in repositories are accessed / used when grammar checking is performed. The n-gram model is implemented to check incorrect POS tag sequences and agreement in number, person and gender of Amharic sentence. Adverb/ tense agreement, which is discussed in section 3.3.5, is checked using probabilistic method. The two major tasks in n-gram method are discussed in detail in section 3.3.3 and 3.3.4. Section 3.3.1 presents fundamental concepts in n-gram based statistical Amharic grammar checker.

3.3.1 Fundamental Concepts

Pre-processing and sentence splitting – texts in the corpus, which are not required for implementation of the system, should be identified and separated from those that are required. In pre-processing, the input corpus first processed to remove unnecessary texts. Unnecessary texts include the elements other than the POS tags i.e. <Document>, <title> and other elements

in the corpus. After text elimination, the corpus includes sentences that have words and their POS tags. In order to make the corpus suitable for further processing, sentences are kept line by line. Therefore, the pre-processed text has a sentence in a line to make further processes simple.

POS tagging – POS tagging is the process performed by a POS tagger. POS tagger is one of the NLP applications which assign words to its part of speech (See Section 2.4). In order to maximize the performance of grammar checker applications, it's very important to include POS tagger as a pre-process. Each word in a sentence (both in training and test set) should be assigned to its own POS tag. This is very helpful as processing all the words are very difficult to handle. The reason for this is, a word may not occur at all in the corpus or occur only once or very few times which makes calculating probability difficult and unreliable. Finding the exact word match in a given corpus may also be difficult. In grammar checking, each word in a sentence will be processed/checked using its POS tag. In STAMGRAM, the words (including punctuation marks) in the training and the test set are tagged manually. Due to unavailability of Amharic POS tagger system (See Section 2.4), POS tagger is not used as a sub system. For this reason, The POS tagging process for the STAMGRAM is done manually i.e. a manually tagged corpus is used for the training and test set. The training set is taken from manually tagged news corpus containing about 7964 sentences. 31 different POS tags are used to tag the corpus. A list of these POS tags and their description can be found in Appendix 2. The test set is also prepared from this manually tagged corpus.

Morphological Analyzer – using a morphological analyzer, linguistic meaning of morphemes in a word can be defined. Each affix (prefix, infix, and suffix) or morpheme in word refers to different linguistic meaning such as number, person, gender, definiteness and the like(See Section 2.5).As Amharic is morphological rich language, this NLP application is a very crucial sub part/system for most of other higher level NLP applications like Amharic grammar checker. In STAMGRAM, the manually tagged words in training and test set are further analyzed using Amharic morphological analyzer named "HornMorpho". Each tagged word in corpus is given as an input to this Amharic morphological analyzer [Gasser, 2011]. HornMorpho analyzes noun, adjectives and verb. For the reason that nouns and adjectives have identical morphological structure, HornMorpho identifies and analyzes adjective as a noun. In the analysis process, POS tagging of the HornMorpho is not used. However, the adjectives are

analyzed as noun and the noun and the adjective can be identified from the manually tagged corpus. To give the words to HornMorpho as an input, from the 31 tags, 18 tag set are selected and grouped into two.

1) **Group noun:** this group includes nouns, pronoun and adjectives of different forms. Noun and adjectives are kept in the same group as both treated in similar way in HornMorpo. After the analysis the adjective and the nouns can be identified from the POS tag taken from the manually tagged corpus. The number, person, gender of each word in this group identified for the tagged words. The tags included in this group are NP, N, NC, PRONC, PRON, NPC, PRONP, VN, ADJP, ADJ and ADJPC. The description about the tag symbols can be found in appendix 1.

2) **Group verb:** this group has different forms of verbs in it. Amharic Verbs has inflected in different way with noun and adjectives. Verb can be analyzed for both subject and object that it refers to. The tag sets that are included in this group are VP, V, VC, VREL and VPC.

The other tags other than the tags grouped above cannot be analyzed. These tags are tags for punctuation marks, number and the like. ENDPUNC, PUNC, NUMCR and NUMPC some of the tags that not included the two groups above.

3.3.2 STAMGRAM Word Representation

To check grammatical errors in a sentence, words require linguistic information other than the POS tag. The information about the words can be retrieved from the morphemes of each word. To know what information that the morpheme refers to, a morphological analyzer should be used (See Section 2.5). These linguistic meanings include number (plural and singular), person (first, Second and third), gender (female and male), definiteness (definite and indefinite) and so on. After the linguistic information is assigned to each word, one can check whether the information about one word in the sentence agrees with the information about the other words in the sentence. As explained in Section 4.2.1, the agreement checking in STAMGRAM was done with the help of Amharic Morphological Analyzer known as HornMorpho.

Agreement checking includes number, person and gender agreement between words i.e. noun-verb agreement, adjective-noun, verb-verb agreement and so on. This agreement can be

checked after the morphological analysis of the words. Word classes (noun, verb and adjective) are grouped into two to make it suitable to the HornMorpho. The two groups are noun group and the verb group. Grouping the word class is required because noun and verbs have different morphological structure and should be analyzed in separate way. The first group, which is the noun group, includes nouns and adjectives of different form because noun and adjective have similar morphological structure and treated in the same way in HornMorpho. The detail about the two groups discussed on fundamental process (See Section 4.3.1).

Both the training and testing corpus are morphologically tagged using HornMorpho Amharic morphological analyzer. The analysis is done for all POS tags in the noun and verb group. After the analysis result retrieved from the HornMorpho, the result will be assigned for word next to the POS tag in three different slots. Information about the word is kept in four different slots including the first slot that contains the POS tag. The four slots are located next to the word inside the angle brackets "<" and ">" . The first slot contains the POS tag, the second the number description (singular and plural) about the word, the third refers to person (first, second and third) description and the last slot contains gender (masculine and feminine) information about the word preceding it. The slots information value is different for noun and verb group (see Figure 3.4 and Figure 3.5).

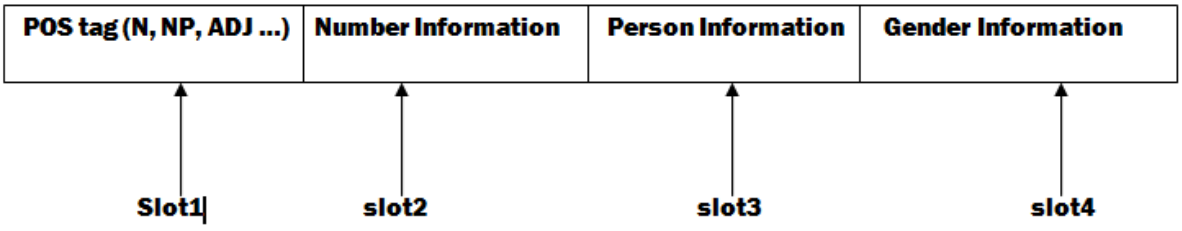


Figure 3.3: The placement the analysis result in the four slots

The three slots exist for both noun and verb group in the corpus. However, the slots contain different type of values. Information that is included in each slot is discussed as follows:

Slot 1:

This slot contains the POS tag information about the word preceding it. The system gets this tag from the manually tagged corpus. 18 tags from the 31 tags (see Appendix 1) are selected for

the analysis process. Other than correct POS tag sequence checking, this slot is required to check all the three agreement in number person and gender. This slot contains particular POS tag for word even when word is analyzed.

Slot 2:

This slot holds number information about the word i.e. whether the word is plural or singular. It has three different values in the case of noun group; P, S and NN. P, S and NN stand for plural, singular and NOT KNOWN. If the word is verb group, this slot will have four other different values. These are SS^OS, SS^OP, SP^OS and SP^OP.'^' sign in each the values act like 'and'. The first value "SS^OS" represents that verb has a singular subject marker and singular object marker. The second "SS^OP", the third "SP^OS" and the fourth "SP^OP" represent verb with singular subject and plural object marker, plural subject and singular object marker and both plural subject and object marker respectively. The NOT KNOWN value is given for this slot if the number information could not be identified by the HornMorpho.

Slot 3:

This slot contains person information about the word preceding it i.e. first person, second person and third person. The slot has four different possible values for the noun groups; P1, P2, P3 and NN. P1 stands for person one, P2 for person two. P3 for person three and NN is same as slot two which stands for NOT KNOWN. The words in verb group have different values than noun groups. The words in verb groups may belong to nine values. The slot values may also contain 'NN' value if the person information about the verb not known or cannot be identified by the morphological analyzer. Possible values for the third slot in the case of verbs are listed as follows:

SP1^OP1: first person subject and first person object marker

SP2^OP2: second person subject and second person object marker

SP3^OP3: third person subject and third person object marker

SP1^OP2: first person subject and second person object marker

SP2^OP1: second person subject and first person object marker

SP3^OP1: third person subject and first person object marker

SP1^OP3: first person subject and third person object marker

SP2^OP3: second person subject and third person object marker

SP3^OP3: third person subject and third person object marker

Slot 4:

This slot is the last slot and contains gender information about the word. Gender information means whether the word is masculine or feminine. This slot also has three different values for noun group like the previous two slots. The values are M, F and NN for masculine, feminine and NOT KNOWN respectively. For the verb group, the slot may contain four different values: SM^OF, SF^OM, SM^OM and SF^OF. The first value "SM^OF" indicates subject masculine and object feminine verb,"SF^OM" for subject feminine and object masculine verb, the last two" SM^OM and SF^OF" are for subject masculine and object masculine verb and subject feminine and object feminine respectively.

For example:

1) The Amharic noun ፕሬዚዳንቱ/*'prezidantu'* can be analyzed and assigned to the four slots as follows:

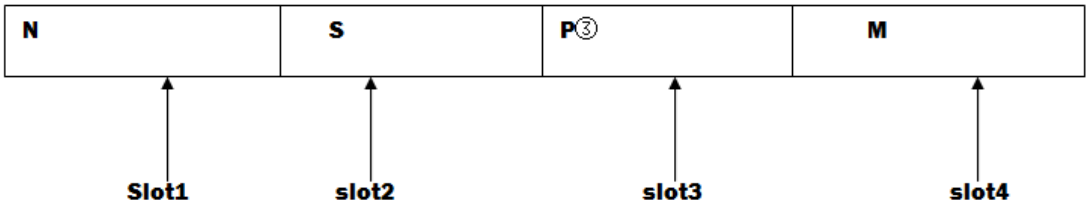
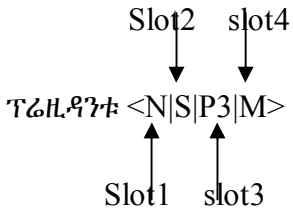


Figure 3.4 : Analysis of the noun "ፕሬዚዳንቱ"/*'prezidantu'* and its placement in the four slots

After assigning the values to the slots the word and slots values are kept in the corpus next to each other as follow.



2) The verb በደረሰበት/*'bederesebet'* can be analyzed using HornMorpho and assigned to the four slots as follows:

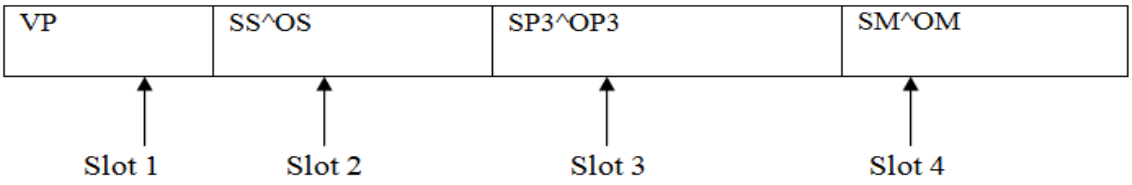
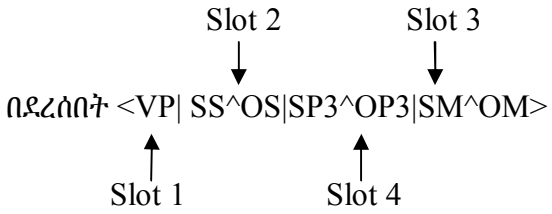


Figure 3.5 : Analysis of the verb "በደረሰበት"/*'bederesebet'* and its placement in the four slots

Finally the slot values are kept in the corpus as follows:



3.3.3 Training in STAMGRAM

In STAMGRAM, probability of n-gram of POS tag sequence calculated from the corpus for a given N value i.e. N=2 and N=3. These N-grams are then saved in repository and used in grammar checking process to determine whether a given sentence is grammatically correct or not. N-grams in the repository support the system in finding the incorrect POS patterns that make the sentence grammatically incorrect and checking agreement between words. The unique sequences and probability of occurrences are determined from the corpus, which is used to train the system. The probability calculation is based on the given N value in the N-gram. The value of N determines the number of POS tag sequences occurring together in a corpus. Figure 3.6 shows the workflow of the STAMGRAM training.

The training process in STAMGRAM starts by accepting the training corpus and the N value as an input. After accepting the input text and the N value, the input text split into sentences and kept in temporary sentence repository. The sentence splitting is done in sentence splitter module. The sentence temporary repository saves the sentences until all the sentences passes through the pattern extraction process. Each sentence from the temporary sentence repository is

forwarded to sequence extractor module. In sequence extractor, possible patterns (POS tag, number, person, gender) can be extracted. The extracted pattern from the current sentence is again temporarily saved in temporary sequence repository. Before calculating the probability of each extracted pattern, the system first checks its existence in the repository. If the pattern exists, it means its probability is already stored in the repository, the system passes to the next pattern in the temporary sequence repository. Therefore, it's not necessary to calculate the probability as it already exists. If the pattern does not exist in the repository, probability calculator module calculates its probability. The calculated probability and the pattern itself (as key) are forwarded and stored in the repository. Then, the pattern and its probability will be permanently stored in repository. When the sequence for the first sentence completed, then the system goes back to the temporary sentence repository to get the second sentence. The whole steps, which have been performed for the first sentence, will be repeated for the second sentence, third sentence, forth sentence...until the last sentence. Each and every activities done in STAMGRAM training process is explained with example as follows. The example discussed below only shows the POS tag sequence extraction. The agreement extraction is discussed in Section 3.3.5.

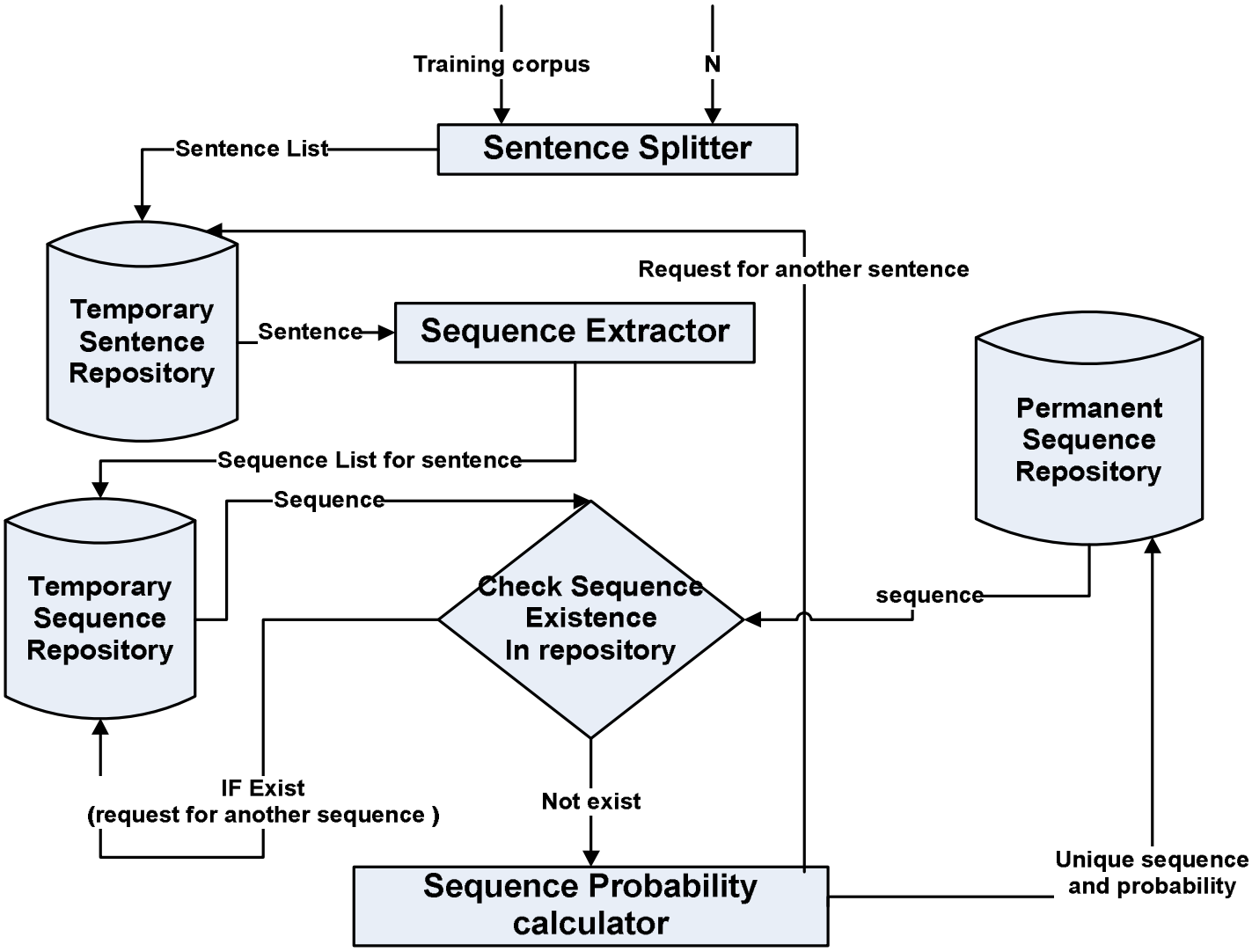


Figure 3.6 : Workflow of training part in STAMGRAM

Sentence Splitter - As explained in section 4.2.1, each sentence is kept in a line within the corpus. When the training process starts by accepting the training corpus, this module splits the training corpus into sentences and keeps the sentences in the temporary sentence repository to process each sentence one by one.

Sequence Extractor - The sequence extractor accepts a sentence at time and processes each sentence individually. Possible sequences for each sentence in the corpus are extracted based on a given N value. This N value is provided to the system as an input to the training module. To explain the sentence extractor with example, there random short sentences are taken from the corpus. These sentences are considered as corpus which is given to the training task as an input.

```
READ sentenceList, N
 FOR sentence in sentenceList
    IF N = 2 THEN
        FOR word in sentence
            IF first word in a sentence THEN
                currenTagtSequence =  <START> < currentWordSlot1>
                currentNumberSequence =  <START> <currentWordSlot1 + currentWord Slot2>
                currentPersonSequence = <START> <currentWordSlot1 + currentWordSlot3>
                currentGenderSequence =  <START> <currentWordSlot1 + currentWordSlot4>
            ELSE
                currenTagtSeq =  <currentWordSlot1> <nextWordSlot1>
                currentNumberSeq  =  <currentWordSlot1  +  currentWordSlot2>  <nextWordSlot1  +
                nextWordSlot2>
                currentPersonSeq  =  <currentWordSlot1  +  currentWordSlot3>  <nextWordSlot1  +
                nextWordSlot3>
                currentGenderSeq  =  <currentWordSlot1  +  currentWordSlot4>  <nextWordSlot1  +
                nextWordSlot4>
    ELSE
        FOR tag in sentence
            IF first tag in a sentence THEN
               currenTagtSeq  =  <START> < currentWordSlot1> <nextWordSlot1>
               currentNumberSeq = <START> < currentWordSlot1 + currentWordSlot2>  <nextWordSlot1 +
               nextWordSlot2>
               currentPersonSeq  =  <START>  <currentWordSlot1  +  currentWordSlot3>  <nextWordSlot1  +
               nextWordSlot3>
               currentGenderSeq  =  <START> <currentWordSlot1 + currentWordSlot4> <nextWordSlot1 +
               nextWordSlot4>
            ELSE
               currenTagtSeq =  <currentWordSlot1> <nextWordSlot1> < nextOFNextWordSlot1>
               currentNumberSeq  =  <currentWordSlot1  +  currentWordSlot2>  <nextWordSlot1  +
               nextWordSlot2> <nextOFNextWordSlot1  +  nextOFNextWordSlot2>
               currentPersonSeq  =  <currentWordSlot1  +  currentWordSlot3>  <nextWordSlot1  +
               nextWordSlot3> <nextOFNextWordSlot1 + nextOFNextWordSlot3>
               currentGenderSeq  =  <currentWordSlot1  +  currentWordSlot4>  <nextWordSlot1  +
               nextWordSlot4> <nextOFNextWordSlot1 +  nextOFNextWordSlot4>
        ADD currenTagtSeq in tagSequenceList, ADD currentNumberSeq in numberSequenceList
        ADD currentPersonSeq in personSequenceList, ADD currentGenderSeq in genderSequenceList
```

Listing 3.2 : Algorithm for sequence extractor module

For instance: Suppose these three sentences are input training corpus

1. ማእከሉ በምርምር ያገኛቸው ሶስት የበቆሎ ዝርያዎች ተቀባይነት አገኙ።/*'ma'kelu bemrmr yagNacew sost yebeqolo zryawoc teqebaynet 'ageNu'*

2. ለገጠር ልማት ስትራቴጂው ስኬታማነት ለብዝሃሃብትዊ ህይወት ትኩረት መስጠት ኢንደሚገባ ተጠቆመ ።/*'legeTer lmat stratEjiw skEtamanet lebzhahabtwi hywet tkuret mesTet 'ndemigeba teTeqome'*

3. የድህነት ቅነሳ ስልት ሰነድ ላይ በክልል ደረጃ ነገ በዘጠኝ ክልሎች ምክክር ይጀመራል ።/*'ydhnet qnesa slt sened lay bekll dereja nege bezeTeN Kllloc'*

The sentences are tagged in a corpus as follows i.e taking only the first slot to check the POS tag sequence:

ማእከሉ/*'ma'kelu* <N> በምርምር/*'ma'kelu* <NP> ያገኛቸው/*bemrmr* <VREL> ሶስት/*sost* <NUMCR> የበቆሎ/*yebeqolo* <NP> ዝርያዎች/ *zryawoc* <N> ተቀባይነት/*tqbaynt* <N> አገኙ/*'agNu* <V> ። <ENDPUNC>

ለገጠር / *legeTer* <NP> ልማት/ *lmat* <N> ስትራቴጂው / *stratEjiw* <N> ስኬታማነት/ *SkETamaNeT* <N> ለብዝሃሃብትዊ / *LeBzhahaBTwi* <ADJ> ህይወት/ *hywet* <N> ትኩረት / *tkuret* <N> መስጠት / *mesTet*<VN> እንደሚገባ / '*ndemigeba* <VP> ተጠቆመ / *teTeqome* <V> ። <ENDPUNC>

የድህነት / *yedhnet* <NP> ቅነሳ / *qnesa* <N> ስልት / *slt* <N> ሰነድ / *sened* <N> ላይ / *lay* <PREP> በክልል / *bekll* <NP> ደረጃ / *dereja* <N> ነገ / *nege* <NP> በዘጠኝ / *bezeTeN* <NUMP> ክልሎች / *klloc* <N> ምክክር / *mkkr* <N> ይጀመራል / *yjemeral* <V> ። <ENDPUNC>

In the above sentences, each word appears with its POS tag. If N = 2, the bigram can be extracted as follows:

Sequence Extractor

<START> is not a POS tag in the sentence but a tag that the algorithm assigns to the sequence to show sentence beginning and <ENDPUNC> is the tag that is used to indicate the sentence end or the sentence end punctuation mark " ።". The three sentences have the following possible bigram sequences.

<START><N>, <N><NP>, <NP><VREL>, <VREL><NUMCR>, <NUMCR><NP>, <NP><N>, <N><V>, <V><ENDPUNC>

<START><NP>, <NP><N>, <N><N>, <N><N>, <N><ADJ>, <ADJ><N>, <N><N>, <N><VN>, <VN><VP>, <VP><V>, <V><ENDPUNC>

<START><NP>,<NP><N>,<N><N>,<N><N>,<N><PREP>,<PREP><NP>,<NP><N>,<N><NP>,<NP><NUMP>,<NUMP><N>, <N><N>, <N><V>, <V><ENDPUNC>

Sequence Probability Calculator

Sentence probability calculator calculates the probability for each unique POS tag sequence. Calculating the probability of the tag sequence follows sequence extraction of each sentence. Sequence probability calculator calculates the patterns for the first sentence first, then second sentence second, then third and so on and so forth until the last. The probability calculation is based on the probability equation (2.4).

```
READ sequences, N, dictionary
FOR sequence in sequences
        IF sequence NOT IN dictionary THEN
          READ tag in sequence
          IF N = 2 THEN
                numberOfAllTags = COUNT(tag 1 tag 2 )
                numberOfGivenTag = COUNT(tag 1)
          ELSE
                numberOfAllTags = COUNT(tag 1 tag 2 tag3)
                numberOfGivenTag = COUNT(tag 1 tag2)
                sequenceProbability = numberOfAllTags / numberOfGivenTag
        ADD sequence , sequenceProbability in dictionary
```

Listing 3.3 : Algorithm for probability calculator module

Two example sequences are given below to show steps on how the N gram (probability of occurrence of each sequence) is calculated.

Probability of <N> given <START>, the occurrence of <N> in the beginning of the sentence:

$$\mathrm{P}(< N > | < \mathrm{START} >) = \frac{\mathrm{COUNT}(< START >< \mathrm{N} >)}{\mathrm{COUNT}(< START >)}$$

Where, **COUNT (<START><N>)** is the number of <N>s that occurs in the start of the sentence and **COUNT (<START>)** is the number of sentence in the corpus.

Form the above example sentences, $\mathrm{P}(< NP > | < \mathrm{N} >) = \frac{1}{3} = 0.333333333333$

i. Probability of <NP> given <N>

$$\mathrm{P}(< NP > | < \mathrm{N} >) = \frac{\mathrm{COUNT}(< N >< \mathrm{NP} >)}{\mathrm{COUNT}(< N >)}$$

$$\mathrm{P}(< NP > | < \mathrm{N} >) = \frac{2}{14} = 0.142857142857$$

Probability of all the other tag sequences in all the sentences will be calculated like the example in (i) and (ii) above. If the tag sequences already exist in previously calculated list in the dictionary, it will not be calculated again. After tracing all the sentences in the corpus for the given N value, probability of every unique sequence will be saved to a dictionary. The probability of the possible two gram sequence for the three sentences above will be:

<NP><VREL> = 0.166666666667 <VREL><NUMCR> = 1.0

<NUMCR><NP> = 1.0 <NP><N> = 0.666666666667

<N><N> = 0.357142857143 <N><V> = 0.142857142857

<V><ENDPUNC> = 1.0 <START><NP> = 0.666666666667

<N><ADJ> = 0.0714285714286 <ADJ><N> = 1.0

<N><VN> = 0.0714285714286 <VN><VP> = 1.0

<VP><V> = 1.0 <N><PREP> = 0.0714285714286

<PREP><NP> = 1.0 <NP><NUMP> = 0.166666666667

<NUMP><N> = 1.0

3.3.4 Grammar Checking

After the possible tag sequences are stored in repository (See Section 4.3.3), the N-grams sequences and the probabilities in the dictionary will be accessed to check the errors in a given sentence. The probability of the sentence to be checked and threshold set are the two main units to decide on the correctness of the sentence. If the probability calculated for the sentence is less than or equal to the threshold, the sentence will be incorrect otherwise correct.

After the input text to be checked and the threshold are given as an input, the input text will be split into sentences by the sentence splitter. The temporary sentence repository temporarily saves the list of sentences spitted sentence splitter. The possible POS tag sequence for each sentence from the repository is extracted in the same way as explained in the training process (See Section 4.2.2). The sequence extractor extracts possible tag sequences for given N gram value. Then after, for each extracted sequence, probability of the sequence will be retrieved from the repository. If the extracted sequence does not exist in the repository, the probability of the sequence/pattern will be set to ZERO. If exist, the probability for the sequence will be retrieved from the dictionary to calculate the sentence probability (probability of the whole sentence). The sentence probability is calculated by sentence probability calculator i.e. by taking the probability of the POS tag patterns and multiplying all probability together. Figure 3.7 shows the workflow of the STAMGRAM grammar checking phase.

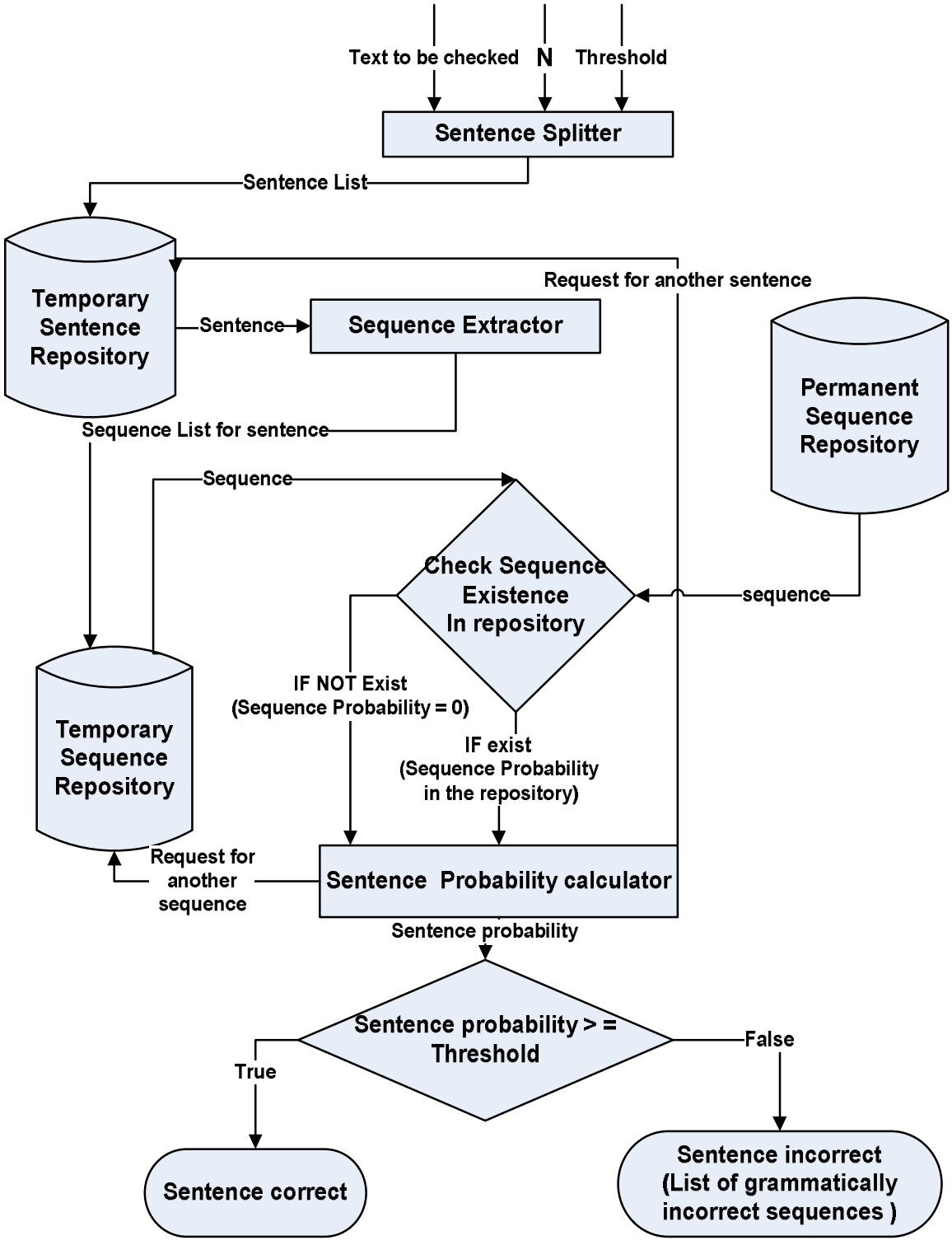


Figure 3.7 : Workflow of grammar checking in STAMGRAM

Sentence Splitter

The sentence splitter module does the same activity as the sentence splitter in the in training process (See Section 4.3.3). It splits the input text to be checked into sentences and keeps the sentence in temporary sentence repository.

Sequence Extractor

Sentence extractor does the same activity as the sentence extractor in training process. The difference is in the training process the input text is the training corpus while in the grammar checking the text to be checked. Sentence Extractor analyzes the sentence to be checked in order to extract the possible tag sequences for a given n gram value. The sequence starts with start tag (<START>), which indicates the start of the sentence, and move until the end of the sentence tag (<ENDPUNC>), which points to sentence end or sentence end punctuation mark in Amharic ('።') . Based on the given N, the sequences will be automatically created from start to the end which includes every tag in between.

For instance:

The probability of the sentence "ኮሚሽኑ በተጠርጣሪ የቀድሞ ባለስልጣናትና ፣ ነጋዴዎች ላይ መሰረተ ክስ ።"/ *komixnu beteTerTari yeqedmo baleslTanatna ፣ negadEwoc lay meserete ks ።* can be calculated as follows:

The sentence is tagged in this way in the corpus:-

ኮሚሽኑ / *komixnu* <N> በተጠርጣሪ / *beteTerTari* <NP> የቀድሞ / *yeqedmo* <ADJP> ባለስልጣናትና / *baleslTanatna* <NC> ፣ <PUNC> ነጋዴዎች / *negadewoc* <N> ላይ / *lay* <PREP> መሰረተ / *meserete* <V> ክስ / *ks* <N> ። <ENDPUNC>

For this tagged sentence possible sequence can be extracted for any N value in N-gram. If N=2 is calculated for the sentence, ten sequences can be identified. This can be checked using n-gram formula 2.1. The totalNumberOfTag refers all the POS tags including the start and end tags. Number of sequences that can be extracted from a sentence is the difference between total numbers of tag (totalNumberOfTag) and one minus the N value in the n gram.

numberOfSequence = totalNumberOfTag – (N -1)

numberOfSequence = 11 - (2-1) = 10

All ten sequences for the above are ['<START><N>', '<N><NP>', '<NP><ADJP>', '<ADJP><NC>', '<NC> <PUNC> ', <PUNC> <N> ', '<N><PREP>', '<PREP><V>', '<V><N>', '<N><ENDPUNC>']

Retrieving Sequence Probability

The probability of different tag sequences for different N values is kept in dictionary while training process of the system is performed. This probability of the tag sequence will be retrieved for each tag sequence extracted from the sentence to be checked. Before extracting the sequence probability, the system first check whether the POS tag sequence exists or not. If exist, the probability for the tag sequence will be retrieved from the dictionary and sent to the sentence probability calculator. If the tag sequence is not in the dictionary, ZERO probability will be assigned for the sequence and this probability will be sent to the sentence probability calculator.

```
READ sequences, dictionary
FOR sequence in sequences
    IF sequence NOT IN dictionary THEN
        sequenceProbability = 0
    ELSE
        READ probability FROM dictionary
        sequenceProbability = probability
    ADD sequence, sequenceProbability in sequenceProbabilityList
```

Listing 3.4 : Algorithm for retrieving sequence probability

For instance: the tag sequences extracted from the above sentence in sequence extractor. Considering the three sentences in the training process (See Section 4.3.2) as corpus, the following probability can be retrieved for tag sequences in the sentence "ኮሚሽኑ በተጠርጣሪ የቀድሞ ባለስልጣናትና ፣ ነጋዴዎች ላይ መሰረተ ክስ ።" / komixnu beteTerTari yeqedmo baleslTanatna ፣ negadEwoc lay meserete ks ። above.

'<START><N>' = 0.201657458564, '<N><NP>' = 0.156987992994,

'<NP><ADJP>' = 0.00672808825312, '<ADJP><NC>' = 0390590324012,

'<NC><PUNC>' = 0 '<PUNC><N>' = 0.28680159142016953,

'<N><PREP>' = 0.95777613384, '<PREP><V>' = 0,

'<V><N>' = 0.04855593183466, ‘<N><ENDPUNC>‘= 0

Zero probabilities for the tags '<NC><PUNC>’, '<PREP><V>‘and <N><ENDPUNC> is given because the tag sequences do not exist in the text that is used as a corpus. All the other probabilities, other than the zero, are calculated from the corpus.

Sentence Probability Calculator

Two important factors, the probability of the sentence and threshold, are used to determine the correctness of the sentence. The threshold is given as an input before the execution of the grammar checker module and it ranges between the values 0 and 1.Whereas the probability of the sentence can be calculated from the probability of the sequences in the sentence (See Equation 2.5).

> *Probability of sentence is the product of the probability of the TAG sequences in the sentence.*

```
READ sequenceProbabilityList
SET SentenceProbability = 1
FOR sequenceProbability in sequenceProbabilityList
        SentenceProbability = SentenceProbability * sequenceProbability
```

Listing 3.5 : Algorithm for sentence probability calculator

For instance:

Probability of the following sentence from the example in the sequence extraction section can be calculated as follows:

S = ኮሚሽኑ / *komixnu* <N> በተጠርጣሪ / *beteTerTari* <NP> የቀድሞ / *yeqedmo* <ADJP> ባለስልጣናትና / *baleslTanatna* <NC> ፣ <PUNC> ነጋዴዎች / *negadewoc* <N> ላይ / *lay* <PREP> መሰረተ / *meserete* <V> ክስ / *ks* <N> ። <ENDPUNC>

The sequences list ➔ ['<START><N>', ‘<N><NP>', '<NP><ADJP>', ‘<ADJP><NC>', '<NC> <PUNC> ', <PUNC> <N> ‘, '<N><PREP>', '<PREP><V>', '<V><N>',

‘<N><ENDPUNC>‘]

$$\mathbf{P(S)} = \mathrm{P}(< N > \mid < START >) \times \mathrm{P}(< NP > \mid < N >) \times \mathrm{P}(< ADJP > \mid < NP >) \times \mathrm{P}(< NC > \mid < ADJP >) \times \mathrm{P}(< PUNC > \mid < NC >) \times \mathrm{P}(< N > \mid < PUNC >) \times \mathrm{P}(< PREP >) \mid < N >) \times \mathrm{P}(< V > \mid < PREP >) \times \mathrm{P}(< V > \mid < N >) \times \mathrm{P}(< ENDPUNC > \mid < N >)$$

$$\mathrm{P(S)} = 0.201657458564 \times 0.156987992994 \times 0.00672808825312 \times 0.0390590324012 \times 0 \times 0.28680159142016953 \times 0.95777613384 \times 0 \times 0.04855593183466 \times 0$$

$$\mathrm{P(S)} = 0$$

If any of the sequences does not exist in the corpus, the sequence probability will be assigned to ZERO. This time, the sequence '<NC><PUNC>’, '<PREP><V>‘and <N><ENDPUNC> do not exist in the repository and its probability is ZERO. Then after, the probability of the whole sentence will be ZERO due to multiplication property of ZERO. Therefore, the sentence probability for the above sentence is ZERO (0).

There is a problem in calculating the probability for long sentences. The main reason for this is the tendency to ZERO when the product of fraction is done.

Checking the Correctness of the Sentence

After the sentence probability is calculated, the decision on the correctness of the sentence is made based on the threshold given as an input.

For example:

If we set the threshold to ZERO, the above sentence is considered as grammatically incorrect and '<NC><PUNC>’, '<PREP><V>‘and <N><ENDPUNC> are the sequence that makes the sentence grammatically incorrect.

3.3.5 Agreement Checking in STAMGRAM

The agreement between words in Amharic sentence can also be checked using n-gram approach. To do this, words in the training corpus are grouped into noun and verb group for morphological analysis. The other POS tag types, other than the tags in the two groups, are not

considered for the analysis. Some words which have POS tags in the two groups may not be recognized by the HornMorpho. These words have the POS tag only as the HornMorpho could not recognize them.

After words in the two groups are analyzed using hornMorpho, the analysis result will be assigned to each word in four different slots (See Section 4.3.2). After assigning the analysis result to the words, the sequence of slot set is extracted based on the given N value. Each slot sequence (slot2 for number, slot3 for person and slot 4 for gender) probability is calculated and kept in repository. Unlike the POS tag sequences extraction (first slot sequence extraction), the others slot sequence extraction cannot stand alone. When second slot, the third and the fourth probability is calculated, the first slot is required. The number, person and gender alone do not give a meaning unless the POS tag is stated together with them. For example, if one say a word is singular, it's impossible to know that whether the word is noun, adjective or any other POS type. It should be expressed as "singular noun", "singular adjective" or the like. Therefore, when the system extract the number, person and gender sequences the POS tag for each word will precede in each case.

The number sequence from the tag can be extracted using the first slot (POS tag) and the second slot (the number). Using this two slots number of sequences can be extracted for a given sentence. In the case of person sequence, the first slot and the third slot (person) taken to extract the sequence. Finally, the gender sequence can be extracted by using the first and the forth or the last slot which contain the gender information about the word. The following example show how extracting sequence works for the three types of agreement checking (number, person and gender).

For example:

"ልዩ ዞኑ ያስገነባቸው ፕሮጀክቶች 50ሺ ነዋሪዎችን ተጠቃሚ አደረጉ ። / *lyu zonu yasgenebacew projektoc 50xi newariwocn teTeqami 'aderegu ።* " :- this sentence is taken from the corpus as an example. The words (only the noun and verb group) in a sentence can be analyzed using HornMorpho. If the word is a part of the two groups, the word will be analyzed else the POS tag is kept as it is. The sentence can be analyzed and assigned as follows:

ልዩ / *lyu* <ADJ> ዞኑ / *zonu* <N|S|P3|M> ያስገነባቸው / *yasgenebacew* <VREL|SS^OP|SP3^OP3|SM^ONN> ፕሮጀክቶች / *projektoc* <N|P|NN|NN> 50ሺ / *50xi* <NUMCR> ነዋሪዎችን / *newariwocn* <N|P|P3|NN> ተጠቃሚ / *teTeqami* <ADJ|S|NN|NN> አደረጉ / *'aderegu* <V|SP|SP3|SNN> ። <ENDPUNC>

Step 1:

Sequence Extraction

Step 1(sequence extraction) is the same for both training the system and grammar checking module. Taking the N value 3(N=3); the number, person and gender sequence extraction can be done by taking the slots that it they require.

Number sequence extraction:-

1. <START><N|S ><VREL|SS^OP>
2. <N|S><VREL|SS^OP><N|P>
3. <VREL|SS^OP><N|P><NUMCR>
4. <N|P><NUMCR><N|P>
5. <NUMCR><N|P><ADJ|S>
6. <N|P><ADJ|S><V|SP>
7. <ADJ|S><V|SP><ENDPUNC>

Person sequence extraction:-

1. <START><N| P3><VREL| SP3^OP3>
2. <N| P3><VREL| SP3^OP3><N| NN >
3. <VREL| SP3^OP3><N| NN ><NUMCR>
4. <N| NN ><NUMCR><N| P3>

5. <NUMCR><N| P3><ADJ| NN >
6. <N|P3><ADJ| NN ><V| SP3>
7. <ADJ| NN ><V| SP3><ENDPUNC>

Gender sequence extraction:-

1. <START><N| M ><VREL| SM^ONN >
2. <N| M ><VREL| SM^ONN ><N| NN >
3. <VREL| SM^ONN ><N| NN ><NUMCR>
4. <N| NN ><NUMCR><N| NN>
5. <NUMCR><N| NN><ADJ| NN >
6. <N|NN><ADJ| NN ><V| SNN >
7. <ADJ| NN ><V| SNN ><ENDPUNC>

Step 2:

Step two is different in training and grammar checking module. Step two is calculating the probability of the sequence from the corpus in the training system module whereas it is retrieving the extracted sequence from the dictionary in the case of grammar checking module.

Probability calculation for a given slot sequence depends on the N value in the n gram. The N value for the probability calculation of the above sequences is three. For example the first sequence probability of the number sequence can be given as follows.

Probability of relativized verb with subject singular and object plural marker (<VREL|SS^OP>) given noun singular and sentence start (<START><N|S >) is the same as number of the three tags occurring together (<START><N|S><VREL|SS^OP>) over the number of noun singulars come at the sentence start.

$$\mathrm{P}(< VREL|\mathrm{SS\hat{}OP} > |< START >< \mathrm{N|S} >) = \frac{\mathrm{COUNT}(<START><\mathrm{N|S}><\mathrm{VREL|SS^{OP}}>)}{\mathrm{COUNT}(<\mathrm{START}><\mathrm{N|S}>)}$$

The second sequence probability is again calculated in the same way.

$$\mathrm{P}(< N|\mathrm{P} > |< N|\mathrm{S} >< \mathrm{VREL}|\mathrm{SS\hat{}OP} >) = \frac{\mathrm{COUNT}(< \mathrm{N}|\mathrm{S} >< \mathrm{VREL}|\mathrm{SS\hat{}OP} >< \mathrm{N}|\mathrm{P} >)}{\mathrm{COUNT}\ (< \mathrm{N}|\mathrm{S} >< \mathrm{VREL}|\mathrm{SS\hat{}OP} >)}$$

All the other sequences probability including the person and gender sequences can be calculated the same way as the previous two probabilities calculated. After calculating the probability, each unique sequence and the probability will be sent to the repository.

As explained above, step 2 is different for the training and grammar checking module. The training is calculating the probability from the corpus as explained in the right above example. In the grammar checking case, whereas, each sequence probability will be retrieved from the dictionary. If the sequence does not exist in the dictionary, the probability of the will be assigned to zero. If the probability exists, the probability will be retrieved and used for the sentence probability calculation. Sentence probability calculation for the agreement checking is the same as the tag sequence probability calculation in section 4.2.3(See Equation 2.5)

$$\mathbf{P}\ (\mathbf{sentence}) = \mathrm{p}\ (\mathrm{sequence}\ 1)\ *\ \mathrm{p}\ (\mathrm{sequence}\ 2) * \ldots \mathrm{p}\ (\mathrm{sequence}\ \mathrm{N})$$

Where, N is the last sequence of the sentence.

In the agreement case the sequence (sequence 1, sequence 2….sequence N) is the sequence of the slots (number, person and gender). Therefore, the product of the sequences for a given N value is the probability of a given sentence.

After calculating the sentence probability for a given sentence, the system can decide on the correctness of the sentence based on the threshold given.

3.3.6 Adverb-tense Agreement Checking

Amharic adverbs usually come before the verb they modify. When adverb appears in the sentence it usually modifies the next verb that comes after it. There may be number of other words in between (the adverb and the verb) other than the two words. However, the modified verb appears next to the modifier before any other verb in the sentence. There should be agreement between Amharic adverb (usually the time adverbs) and different verb forms (tenses).

To check this agreement, a simple algorithm is developed using probabilistic method. As Amharic adverbs are few in number, first list of adverbs are extracted from the corpus. Then only time adverbs are picked from the extracted adverb list. These time adverbs are stored in list. About 51 time adverbs are extracted from the corpus. To check the time agreement between the adverb and the verb, the tense for the verb that the adverb modifies should be identified (whether it refers to past or present/future). Four different types of tenses can be recognized using HornMorpho. These are perfective, imperfective, jussive/ imperative and gerundive. The probability of time adverbs with each tense type is calculated from the corpus and stored in repository. Appendix 3 shows the list of time adverbs with their calculated probability for each tense type. The listing 3.6 shows the Algorithm for finding the probability of the adverbs with each tense type.

```
READ adverbList , sentenceList
For adverb in adverbList
        SET perfective = 0, imperfective = 0, jussive = 0, gerundive = 0
        FOR sentence in sentenceList
                IF adverb IN sentence THEN
                        nextVerb =  get the next verb after the adverb
                        tenseType = analyze _word(nextVerb)
                        IF tenseType == "perfective" THEN
                                perfective = perfective + 1
                        ELSE IF tenseType == "imperfective" THEN
                                imperfective = imperfective + 1
                        ELSE IF tenseType == "jussive" THEN
                                jussive = jussive + 1
                        ELSE IF tenseType == "gerundive" THEN
                                gerundive = gerundive + 1
        totalNumberAdverb = perfective + imperfective  +  jussive  +  gerundive
        perfective = perfective/ totalNumberAdverb
        imperfective = imperfective/ totalNumberAdverb
         jussive = jussive/ totalNumberAdverb
         gerundive = gerundive/ totalNumberAdverb
        ADD adverb, [perfective, imperfective, jussive, gerundive] in adverbDictionary
```

Listing 3.6 : Algorithm for finding the probability of the adverbs

Whenever these time adverbs is found in the sentence to be checked, the tense type of the next verb will be identified by using HornMorpho. If the tense type is matched with tense with the highest probability than the other three types, the adverb and the tense are considered to have correct adverb/verb agreement. If tense does not match for the first highest probability, the tense is matched against the tense with the second highest. If both the first and second highest probability do not match, the adverb and the tense type are considered to have disagreement in time.

Example

ትላንት / *tlant* <ADV> ብቻ / *bca* <N> 64 <NUMCR> ማመልከቻዎች / *mamelkcawoc* <N> ይቀርባሉ / *yqerbalu* <V> ። <ENDPUNC>

In the this sentence, the time adverb ትላንት / *tlant* refers to the past and ይቀርባሉ / *yqerbalu* is imperfective (present or future). When the adverb ትላንት / *tlant* is identified by the system, the tense type of the next verb will be determined. Now, the verb is ይቀርባሉ / *yqerbalu* and it is imperfective. Then, the probability of occurrence of the time adverb ትላንት / *tlant* with different tense types will be retrieved from the adverb probability list (See Appendix 3). The highest probability of ትላንት / *tlant* is with the tense type Perfective (simple past). The imperfective tense type has zero probability. Therefore, ADVERB-VERB DISAGREEMENT IN TIME exists between ትላንት / *tlant* and ይቀርባሉ / *yqerbalu*.

CHAPTER FOUR

EXPERIMENT

To evaluate the performance of the two approaches used in this study, test cases have been prepared. The testing is carried out using total of 164 sentences. Most of the sentences in the test set include one or more grammatical errors. The test set is self-made corpus with the help of linguistic experts. This chapter gives a detailed explanation about experimental result and evaluation of the rule-based and statistical Amharic grammar checker. The test cases and testing corpuses used is also discussed in this chapter.

4.1 Test Cases Description and Results

The performance evaluation is made using two test cases. The test cases contain real world Amharic texts. In order to calculate the performance for each test case the number of errors in the texts, number of errors found by the Grammar checker and the number of false positives generated by the grammar checker were counted manually. These numbers were then used to calculate the precision and recall of the system. In addition to sentences with grammatical errors, the test cases also try to include error-free sentences to check if the sentences are properly recognized as correct.

Precision refers to the percentage of flagged grammatical errors and that are in fact grammatical errors.

Recall refers to the percentage of grammatical errors occurred in the text and that are in fact flagged.

The precision and recall can be determined:

$$\text{Precision} = \frac{\text{number of correctly flagged error}}{\text{total number of flagged error}} \times 100 \quad (5.1)$$

$$\text{Recall} = \frac{\text{number of correctly flagged error}}{\text{total number of error that occur in the text}} \times 100 \quad (5.2)$$

Test Case One

The first test case is prepared to compare the two approaches (rule-based and statistical Amharic grammar checker). Self-made corpus, that contains grammatical errors, is used in this test case. As the rule-based approach only works for simple sentence, this test case includes only Amharic simple sentences. In this test case, the test set contains total of 50 simple sentences (combination of grammatically correct and incorrect sentences). Some sentences, in this test set, contain more than one error per sentence.

Precision and recall values are calculated using equation 5.1 and 5.2 respectively. The required values for the calculation are counted manually. The performance for the STAMGRAM is done for two N values (bigram and trigram) Table 4.1.1 tries to compare the rule-based and statistical Amharic grammar checker based on precision and recall values.

Table 4.1 : Performance results for rule-based and statistical Amharic grammar checker

Approach		Incorrect flags	Correct flags	Total Number of flags	Total number of errors in the test set	Precision	Recall
Rule-based		4	49	53	52	92.45 %	94.23%
Statistical	Bigram	29	43	72	52	59.72%	82.69%
	Trigram	23	47	70	52	67.14%	90.38%

Form the evaluation result, the rule-based Amharic grammar checker has detected 92.45% of the errors correctly. The error coverage of the rule-based Amharic grammar checker shows better result than the error detection correctness i.e. 94.03%. Only 5.77% grammatically incorrect sentences are wrongly identified as grammatically correct. The rule-based grammar checker also wrongly identifies 30% of the correct sentences as grammatically incorrect. This shows that the rule-based has few false alarms on the correct sentence. Most of the false alarms occurred due to the fact that the language model did not cover all grammatical rules of the language. The other reason is accuracy of the morphological analyzer.

The statistical grammar checker evaluation is conducted for two N values i.e. bigram and trigram. The precision and recall results for trigram and bigram show that the STAMGRAM has better performance result on error coverage than the error detection correctness. In the

trigram case, the percentage of wrongly identified incorrect sentence (actually correct) is the same as rule-based Amharic grammar checker. Both bigram and trigram shows large number of false alarms when compared with the rule-based Amharic grammar checker. In this test case, sentences which contain more than one error also checked. Results show that both rule-based and STAMGRAM can detect different errors per sentence at a time.

Considering the correct sentences in the test set, the following results are obtained:

- 70% of grammatically correct sentences in test set are correctly recognized using the rule-based grammar checker. One fifth of the recognized grammatical errors are recognized only by this approach.
- 70% of grammatically correct sentences in test set are correctly recognized using the statistical Amharic grammar checker. 20% of the recognized errors are detected only by the statistical approach.
- 50% of grammatically correct sentences in test set are correctly recognized using both, the rule-based and statistical Amharic grammar checker.

As shown in table 4.1.1, the rule-based grammar checker has higher performance in both the precision and recall values of grammatical error detection. On the other hand, both approaches achieve the same result in identifying grammatically correct sentences.

Test Case Two

The second test case is used to evaluate the STAMGRAM using another test set that contains complex Amharic sentences. The test set has total of 114 sentences. As the test case one, the largest part of the test set has one or more grammatical errors.

Table 4.2 : Performance of STAMGRAM

Approach		Incorrect flags	Correct flags	Total Number of flags	Total number of errors in the test set	Precision	Recall
Statistical	Bigram	62	85	147	130	57.82%	65.38%
	Trigram	50	88	138	130	63.76%	67.69%

Table 4.1.2 describes the performance of STAMGRAM based on the precision and recall values. As shown in this table, the percentage of flagged grammatical errors that are actually grammatical error is 57.82% and 63.76% for bigram and trigram respectively. Both the error coverage and error detection correctness of the trigram shows higher performance than the bigram. The multiple error detection also again tested in this test case, and the result shows that, multiple errors can also be recognized using STAMGRAM.

The evaluation of the STAMGRAM also made in terms of grammatical error types discus in chapter two (See Section 2.1.3). The highest number of grammatical error detection is achieved on adverb-verb disagreement. Form twenty adverb-verb disagreements errors in the test set, only two are not detected (See Table 4.1.3). 80% of the number disagreement grammatical errors are detected by the STAMGRAM and it becomes the second highest grammatical error type. As shown in Table 4.1.3, a large number of false alarms are recorded in person disagreement. As news corpus is used for the training, most of the sentences contain words which refer to third person. For this reason, occurrence of first and second person in the corpus is very small. This issue increases the false positive in the number disagreement checking. Table 4.1.3 summarizes the error detection of the STAMGRAM based on grammatical error types.

Table 4.3 : Evaluation of STAMGRAM based on the grammatical error type

No	Error type	Total number of errors	Flagged errors by STAMGRAM	Precision
1	Incorrect word order	15	11	73.33%
2	Number disagreement	25	20	80%
3	Person disagreement	25	13	52%
4	Gender disagreement	25	15	60%
5	Adjective – noun disagreement	20	11	55%
6	Adverb – verb disagreement	20	18	90%

4.2 Issues Related to the Performance of the System

4.2.1 Corpus

Statistical grammar checker requires corpus that contains grammatically correct (error-free) text for training. However, it's very difficult to find such corpus for Amharic language (See Section 2.2).The corpus that is used in STAMGRAM training is not checked for grammatical mistake and it probably has grammatical errors.

The corpus also has spelling and typographic mistake which lead to incorrect morphological analysis result. The morphological analyzer interprets the incorrect word in a wrong way and this leads to incorrect grammar checking result.

Even though the corpus used in this research is manually POS tagged, a number of words in the corpus are wrongly tagged. Attempts have been made to correct some of the erroneously tagged words. However, the corpus still has some incorrectly tagged words. These errors cause the wrong tag patterns to be written to the repository when training is done. When patterns matched against the erroneous wrong patterns in the repository, the patterns are considered as correct. If error-free corpus is used to train the system, the performance of the system can be maximized.

4.2.2 Morphological Analyzer

The Morphological analyzer (HornMorpho) examines some words in a wrong way during training. This can be caused by the Morphological Analyzer itself or wrong POS tagging made on the corpus. An example is the noun, which is actually singular, analyzed as plural. If the

wrong analysis occurs in the training phase, there are wrong n-grams written to the repository. This generates wrong result in grammar checking.

The wrong morphological analysis result on the checking phase can also cause problem in grammar checking. This error is caused in similar way as the previous one (training phase). Here the wrong analysis result takes place on the text to be checked. This causes wrong N-grams in the text to be checked matched against the repository.

Low accuracy Amharic Morphological Analyzer can possibly minimize the grammar checking problem related to this one. Unless there is a morphological analyzer of 100% accuracy, this problem cannot be completely avoided. However, with morphological analyzer of high accuracy, the performance of the grammar checker can be maximized.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

In this thesis, Amharic grammar checker has been designed, developed and tested on real-world grammatical errors. Two approaches of grammar checking are used in this research: rule-based and statistical approach. The rule-based approach is implemented and tested on Amharic simple sentences and it requires manually constructed rules that can handle all grammatical errors. On the other hand, statistical approach is designed and implemented for simple and complex Amharic sentences. The Statistical approach requires a good quality training data set. N-gram of tag and probabilistic method is used to check grammatical errors of a given sentence. As it requires large amount of training data, the n-gram of word is not implemented. Both approaches are capable to handle multiple errors of different type from single sentence.

In the rule-based approach, the rules are constructed manually by considering grammatically incorrect Amharic sentences. The rules in the rule-based grammar checker contain incorrect grammatical structure of Amharic sentences. If the pattern of any sentence matches with any of the rules, the sentence will be considered as grammatically incorrect. However, constructing rules for complex sentences is very difficult. Therefore, a statistical approach is designed to include complex Amharic sentences.

In the statistical approach, the list of unique patterns and corresponding probabilities of occurrence are stored in repository. When a sentence is required to be checked, sentence probability will be calculated for it. The sentence probability can be determined using the product of the sequences probability of the sentence to be checked. Then, the grammatical correctness of the sentence can be determined based on the probability of the sentence and given threshold.

The experiment was conducted in two test cases. The performance is evaluated based on precision and recall results. The first test case is used to test and compare the performance of the rule-based and statistical Amharic grammar checker on Amharic simple sentences. As the rule-based Amharic grammar checker works only for simple sentences, the first test set

contains only simple sentences. In this test case, the rule-based grammar checker show higher performance than the STAMGRAM. The precision and recall is [92.45% and 94.03%] and [67.14% and 90.38%] for the rule-based and STAMGRAM respectively. To test the performance on complex sentence, second test case is performed. The test case is prepared from Amharic complex sentence. In this test case, 63.76% the precision and 67.69% recall is achieved. Based on the two test cases, it is proved that the two approaches can detect multiple errors in a sentence.

In conclusion, both rule-based and statistical approaches work for Amharic text grammar checking. However, the rule-based depends on the manually constructed rules that can cover all grammatical errors whereas the statistical method requires good quality training corpus. The performance of the two approaches is still dependent on external low level tools which are applied on grammar checking as a pre-process. These include POS tagger and morphological Analyzer.

5.2 Recommendation

This work can be extended in many different ways to increase the performance and upgrade the grammar checker. This may also be choosing other method which improves the grammar checking. The following points are suggested as a future work.

Automatic POS Tagger

POS tagger tool is very crucial for both the rule- based and statistical grammar checker. If there is an automatic POS tagger, the grammar checker will not be restricted on some text for training the system like in STAMGRAM. Any text (which is considered as a good) can be selected, tagged with the POS tagger and sent to the next process in grammar checking.

High performance morphological analyzer

Grammar checking in morphologically rich languages like Amharic requires morphological analyzer. In this research, HornMorpho Amharic morphological analyzer is used. If high accuracy morphological analyzer is applied, the grammar checker can be upgraded.

Apply Sentence Parser

The use of a parser could help to apply more complex rules. Sentences are reduced to shorter phrases with the help of rules. Then, grammar checking can be applied between phrases (eg. verb phrase VP or a noun phrase NP). N-gram of phrase can also be constructed and used in grammar checking.

Large Size and Good Quality Corpus

The sample taken for this study and the grammar rules generated cannot be taken to be a representative of the language. Future researches should be conducted using a larger set of corpus which can show all types of grammatical structure. The training corpus also needs to be checked for its grammatical correctness. With this, the grammar checking can be upgraded.

Hybrid Approach

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.

System Integration

Integrating and testing the approaches, which are used in this study, with any word processing program and make it available for end user will increase the usability of this research.

REFERENCES

[1] Liddy, E. D. "Natural language processing in Encyclopedia of Library and Information Science", 2nd Ed. Marcel Decker, Inc. 2001.

[2] Gelbukh, Alexander. "Special issue: Natural Language Processing and its Applications". Instituto Politécnico Nacional. Centro de Investigación en Computación. México 2010.

[3] "Brief History of Grammar Check Software", available at: http://www.grammarcheck.net/brief-history-of-grammar-check-software/, Accessed On October 28, 2011

[4] Kassa, Markos. "Implementing An Open Source Amharic Resource Grammar In Gf". Chalmers University of Technology. Sweden: November 2010

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

[6] Daniel Naber. "A Rule-Based Style And Grammar Checker". Diplomarbeit. Technische Fakultät Bielefeld, 2003.

[7] Yifiru, Martha; Abate, Solomon; Besacier, Laurent. " Part-of-Speech Tagging for Under-Resourced and Morphologically Rich Languages – The Case of Amharic". Conference on Human Language Technology for Development, Alexandria, Egypt, 2-5 May 2011.

[8] Michael Gasser."HornMorpho". Indiana University, School of Informatics and Computing. 9 January, 2011.

[9] Alemu, Atelach. "Automatic Sentence Parsing For Amharic Text an Experiment Using Probabilistic Context Free Grammars". Addis Ababa University. Ethiopia, 2002.

[10] Fentaw, Mekonnen. "Integration Of Amharic Spell Checker In A Word Processor". Project In Addis Ababa University. Ethiopia, 2009.

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

[12] Steve Richardson. "Microsoft Natuaral language Understanding System and Grammar checker". Microsoft.USA, 1997.

[13] Arppe, Antti. "Developing a Grammar Checker for Swedish". The 12th Nordic conference computational linguistic. 2000. PP. 13 – 27.

[14] Schmidt-Wigger, Antje. “Grammar And Style Checking in German”. Proceedings of CLAW. Vol 98. 1998

[15] Shaalan, Khaled. “Arabic Gramcheck: A Grammar Checker for Arabic”. The British University in Dubai. United Arab Emirate, 2005.

[16] Singh, Mandeep; Singh, Gurpreet; Sharma, Shiv. “A Punjabi Grammar Checker”. Punjabi University. 2nd international conference of computational linguistics: Demonstration paper. 2008. pp. 149 – 132.

[17] Jahangir Md; Uzzaman, Naushad; Khan, Mumit. “N-Gram Based Statistical Grammar Checker For Bangla And English”. Center for Research On Bangla Language Processing. Bangladesh, 2006.

[18] Henrich, Verena; Reute, Timo. “LISGrammarChecker: Language Independent Statistical Grammar Checking”. Hochschule Darmstadt & Reykjavík University: 2009.

[19] S., Philip; M.W., David. “A Statistical Grammar Checker”. Department of Computer Science. Flinders University of South Australia.South Australia, 1996.

[20] 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.

[21] Kinoshita, Jorge; Nascimento, Laнs do; Dantas ,Carlos Eduardo. ”CoGrOO: a Brazilian-Portuguese Grammar Checker based on the CETENFOLHA Corpus”. Universidade da Sro Paulo (USP), Escola Politйcnica. 2003.

[22] Gamback, Bjorn; Olsson, Fredrik; Alemu, Atelach, Lars Asker. “Methods for Amharic Part-of-Speech Tagging”. Proceedings of the EACL 2009 Workshop on Language Technologies for African Languages – AfLaT 2009, Athens. Greece, 31 March 2009. PP 104–111.

[23] Getachew , Mesfin. “Automatic Part of Speech Tagging for Amharic Language: An Experiment Using Stochastic Hidden Markov (HMM) Approach”. Master Thesis at School of Information Studies for Africa. Addis Ababa. Ethiopia, 2001.

[24] “Support Materials and Exercises for GRAMMAR: PART I. Parts of Speech”. English Curriculum Content Expert, New Brunswick Community College: 1998.

[25] Gochel, Daniel. “An Integrated Approach to Automatic Complex Sentence Parsing For Amharic Text”. Master thesis in Addis Ababa University: Ethiopia, 2003.

[26] Fissaha, Sisay.“Part of Speech tagging for Amharic using Conditional Random Fields”.

Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, 47–54: 2005.

[27] Gebrekidan , Binyam. “Part of speech tagging for Amharic”. Centre Tesnière. Université de Franche-Comté, France .Research Institute in Information and Language Processing. Wolverhampton University, UK. 2010.

[28] Aronoff ,Mark; Fudeman, Kirsten. “What is Morphology?”. Blackwell publishing. Vol 8. 2001.

[29] Bayu, Tesfaye. “automatic morphological analyzer for Amharic an experiment employing unsupervised learning and autosegmental analysis approaches”. Addis Ababa University. thesis for Master of Science in information science, June 2002

[30] Jurafsky Daniel, H., James. Speech and Language Processing: “An introduction to natural language processing, computational linguistics and speech recognition” June 25, 2007.

[31] Deksne, Daiga; Skadinš, Raivis. “CFG Based Grammar Checker for Latvian”. Vienibas gatve 75a, Riga, Latvia. Bolette Sandford Pedersen, Gunta Ne¡spore and Inguna Skadin¸a(Eds.) NODALIDA 2011 Conference Proceedings, pp. 275–278

[32] Mackevièiute, J.Lithuanian morphological analysis system and grammar checker: Tilde‘s technologies in practice. In Proc. HLT.Riga,Latvia. 2004.

[33] Amare, Getahun. “ዘመናዊ የአማርኛ ሰዋስው በቀላል አቀራረብ”: (“Modern Amharic Grammar in a simple approach”). Addis Ababa, Ethiopia. 2010.

[34] Yimam, Baye. “የአማርኛ ሰዋስው”: (“Amharic Grammar”). Addis Ababa, Ethiopia. 2000.

[35] Awgichew, Girma; Meyer, Ronny. “A reexamination of tense in Amharic” .University of Tromsl!J and Johannes-Gutenberg University Mainz. MP65: 143-155. 2001.

[36] Fantaya ,Tlikla-Maryam. “hohete t'abeb zesanes'ahuf” Addis Ababa: Central Printing Press. 1964 E.C

[37] Amare, Getahun . “Issues of time and place adverbs in Amharic”. African languages and cultures 8,123-136. 1995.

APPENDICES

APPENDIX 1: List of part of speech tags and their Description

	POS name	Description
1	<ADJ>	Adjective
2	<N>	Noun
3	<VREL>	relativized verb
4	<NUMCR>	Number
5	<V>	Verb
6	<ENDPUNC>	sentence end punc (Aratnetb)
7	<NP>	noun phrase
8	<VP>	verb phrase
9	<NUMP>	number phrase with conjuction
10	<PREP>	Preposition
11	<VN>	Verb
12	<ADJP>	Adjective phrase
13	<NC>	noun with conjuction
14	<ADV>	Adverb
15	<PUNC>	Punctuation
16	<NPC>	noun phrase with conjuction
17	<AUX>	auxilary verbs
18	<PRONP>	pronoun phrase
19	<CONJ>	Conjuction
20	<NUMOR>	Number
21	<VPC>	verb phrase with conjuction
22	<PRON>	Pronoun
23	<PRONPC>	pronoun phrase with conjuction
24	<ADJC>	adjective with conjuction
25	<VC>	verbs with conjuction
26	<PRONC>	pronoun with conjuction
27	<UNC>	
28	<ADJPC>	pronoun with conjunction
29	<INT>	
30	<NUMC>	number with conjunction
31	<NUMPC>	number phrase with conjunction

APPENDIX 2: DTD for the rules XML file

```
<!-- definition for the root element -->
<!ELEMENT rules (rule|rulegroup)*>
<!-- definition for the rulegroup element -->
<!ELEMENT rulegroup (message?,rule+)>
<!-- definition for attribute list for the rulegroup element -->
<!ATTLIST rulegroup id ID #REQUIRED>
<!ATTLIST rulegroup name CDATA #IMPLIED>
<!-- definition for the rule element -->
<!ELEMENT rule ( pattern,message?,example,example+)>
<!-- definition for attribute list for the rule element -->
<!ATTLIST rule id ID #IMPLIED>
<!ATTLIST rule name CDATA #IMPLIED>
<!-- definition for the pattern element -->
<!ELEMENT pattern (#PCDATA)>
<!-- definition for attribute list for the pattern element -->
<!ATTLIST pattern case_sensitive (yes|no) #IMPLIED>
<!-- message shown to the user if a rule matches: -->
<!ELEMENT message (#PCDATA)*>
<!-- definition for the example element -->
<!ELEMENT example (#PCDATA)*>
<!-- definition for attribute list for the example element -->
<!ATTLIST example type (correct|incorrect) #REQUIRED>
```

APPENDIX 3: Adverb list with its probability for tenses

Adverb Name	Perfective	Imperfective	Jussive	Gerundive
ቅርብ	1.0	0.0	0.0	0.0
ቅድሚያ	0.5	0.5	0.0	0.0
በሚቀጥለው	0.0	1.0	0.0	0.0
ቀጣይ	0.0	1.0	0.0	0.0
በስቲያ	0.7272727272727273	0.2727272727272727	0.0	0.0
ገና	1.0	0.0	0.0	0.0
ዘንድሮ	0.42857142857142855	0.4489795918367347	0.0	0.12244897959183673
ቀደም	0.8	0.2	0.0	0.0
ባለፉት	0.6470588235294118	0.17647058823529413	0.0	0.17647058823529413
በትናንትናውእለት	0.6666666666666666	0.3333333333333333	0.0	0.0
በቅድሚያ	0.0	0.5	0.0	0.5
አስቀድሞ	0.0	0.6666666666666666	0.0	0.3333333333333333
እንደገና	0.18181818181818182	0.6363636363636364	0.0	0.18181818181818182
ትናንት	0.8484848484848485	0.08484848484848485	0.0	0.06666666666666667
በየአመቱ	0.5	0.5	0.0	0.0
አምና	0.7619047619047619	0.09523809523809523	0.0	0.14285714285714285
ዛሬም	1.0	0.0	0.0	0.0
ባሁኑ	0.0	1.0	0.0	0.0
በቅርቡም	0.0	1.0	0.0	0.0
ባለፈው	0.75	0.05	0.0	0.2
ከነገ	0.0	0.0	0.0	0.6
በየጊዜው	1.0	0.0	0.0	0.0
ቀደምሲል	0.75	0.0	0.0	0.25
አልፎ	0.5	0.5	0.0	0.0
ጊዜ	0.42105263157894735	0.42105263157894735	0.0	0.15789473684210525
በቀጣይ	0.36363636363636365	0.5454545454545454	0.0	0.09090909090909091
ቀድሞ	0.0	1.0	0.0	0.0
አሁኑኑ	0.0	1.0	0.0	0.0
በፊት	0.3333333333333333	0.2222222222222222	0.0	0.4444444444444444
አንስቶ	0.5	0.5	0.0	0.0
ትላንት	0.75	0.0	0.0	0.25
ከአምናው	0.6666666666666666	0.3333333333333333	0.0	0.0
በአሁኑ	0.4	0.3333333333333333	0.0	0.26666666666666666
በመጪዎቹ	0.0	1.0	0.0	0.0
ከትናንትበስቲያ	0.5	0.0	0.0	0.5
በየቀኑ	0.6666666666666666	0.0	0.0	0.3333333333333333
በዛሬው	1.0	0.0	0.0	0.0
ማምሻውን	0.875	0.125	0.0	0.0
ትናንትና	0.0	0.0	0.0	1.0
ላለፉት	0.75	0.25	0.0	0.0
እስከአሁንድረስ	0.0	1.0	0.0	0.0
ወደፊት	0.0	1.0	0.0	0.0
ካለፈው	0.25	0.5	0.0	0.25
ወደፊትም	0.0	1.0	0.0	0.0
በቀጣይም	0.5	0.16666666666666666	0.0	0.3333333333333333
አሁን	0.2857142857142857	0.5952380952380952	0.0	0.11904761904761904
እስከአሁን	1.0	0.0	0.0	0.0
አሁንም	0.36363636363636365	0.6363636363636364	0.0	0.0
ዛሬ	0.8590021691973969	0.10629067245119306	0.0	0.03470715835140998

APPENDIX 4: Transliteration of Amharic Alphabets using HornMorpho morphological Analyzer

ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
ha	hu	hi	ha	hE	h	ho
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
le	lu	li	la	lE	l	lo
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
Ha	Hu	Hi	Ha	HE	H	Ho
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
me	mu	mi	ma	mE	m	mo
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
^se	^su	^si	^sa	^sE	^s	^so
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
re	ru	ri	ra	rE	r	ro
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
se	su	si	sa	sE	s	so
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
xe	xu	xi	xa	xE	x	xo
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
qe	qu	qi	qa	qE	q	Qo
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
be	bu	bi	ba	bE	b	bo
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
te	tu	ti	ta	tE	t	to
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
ce	cu	ci	ca	cE	c	co
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
^ha	^hu	^hi	^ha	^hE	^h	^ho
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
ne	nu	ni	na	nE	n	no
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
Ne	Nu	Ni	Na	NE	N	No
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
‘a	‘u	‘i	‘a	‘E	‘	‘o
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
ke	ku	ki	ka	kE	k	ko
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
He	Hu	Hi	Ha	HE	H	Ho
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
we	wu	wi	wa	wE	w	wo
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
`a	`u	`i	`a	`E	`	`o
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
ze	zu	zi	za	ze	z	zo
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
Ze	Zu	Zi	Za	ZE	Z	Zo
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
ye	yu	yi	ya	yE	y	yo
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
de	du	di	da	dE	d	do
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ

je	ju	ji	ja	jE	j	jo
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
ge	gu	gi	ga	gE	g	go
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
Te	Tu	Ti	Ta	TE	T	To
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
Ce	Cu	Ci	Ca	CE	C	Co
ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
Pe	Pu	Pi	Pa	PE	P	Po
ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
Se	Su	Si	Sa	SE	S	So
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
^Se	^Su	^Si	^Sa	^SE	^S	^So
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
fe	fu	fi	fa	fE	F	fo
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
pe	pu	pi	pa	pE	P	po

ሏ	ሗ	ሟ	ሧ	ሯ	ሷ	ቧ
lWa	HWa	mWa	sWa	rWa	sWa	bWa
ቷ	ቿ	ኋ	ኗ	ኟ	ኳ	ኽ
tWa	cWa	hWa	nWa	NWa	kWa	KWa
ዟ	ዧ	ዷ	ጇ	ጓ	ጧ	ቈ
zWa	**ZWa**	**dWa**	**jWa**	**gWa**	**TWa**	**CWa**
ፏ	ፗ	ቈ	ቋ	ቌ		
fWa	**pWa**	qWa	**PWa**	**SWa**		