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

A TOP-DOWN CHART PARSER FOR AMHARIC SENTENCES

Abdurehman Dawud Mohammed

Advsior: Mulugeta Libsie (PhD)

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDEIS OF THE ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT FOR THE DEGREE OF MASTERS OF SCIENCE IN COMPUTER SCIENCE

April 2015

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

COLLEGE OF NATURAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

A TOP-DOWN CHART PARSER FOR AMHARIC SENTENCES

Abdurehman Dawud Mohammed

Advisor: Mulugeta Libsie (PhD)

APPROVED BY

EXAMINING BOARD:

1. **Dr. Mulugeta Libsie, Advisor** _____
2. _____ _____
3. _____ _____

Dedication

To my aunt Amognesh Mohammed

Acknowledgements

All praises and thanks are due to **Allah (***s.w***)**, the Lord of the Worlds. Without Allah's support nothing would have been possible. I am also very grateful to my advisor, Dr. Mulugeta Libsie for his close supervision and constructive suggestion during my work. He has been devoting his time and providing ideas to carry out the research. He is not only my advisor but he is also my mentor. I thank you for your encouragement starting from the beginning to the end of this thesis work. Next I would like to thank Mr. Abdu Ahmed from Linguistic department in Addis Ababa University for his support in the Context Free Grammar extraction and the manual parsing of testing sentences. My deepest gratitude goes to my beloved wife Semira Idris who was beside me providing moral support. I would also like to thank Seid Muhie, my friend Seid Hassen and other colleagues for providing me with thesis ideas. Finally, I would like to give my gratitude to people who are not mentioned in name but whose effort helped me much all along.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ADJ - Adjective

ADJP - Adjectival Phrase

ADV - Adverb

ADVP- Adverbial Phrase

C- Conjunction

e - Empty

N - Noun

NLP - Natural Language Processing

NP - Noun Phrase

Np, np, and nP – Noun Phrase (within main noun phrase or verb phrase)

Num - Numeric

NumP - Numeric Phrase

POS - Part of Speech

PP - Prepositional Phrase

PREP - Preposition

s – Embedded clause

S – Sentence

V - Verb

VP - Verb Phrase

VREL - Relative Clause

WIC - Walta Information Center

XML- eXtensible Mark-up Language

Abstract

Natural language processing applications have an important role in our daily life, by enabling computers to understand human languages. NLP applications such as machine translation, question answering, knowledge extraction and information retrieval are among the most common applications which we need to accomplish different tasks. For better development of the above mentioned applications the assigning of part of speech of words, the extraction of phrases and sub-phrases, and the extraction of syntactic structure of sentences from natural language texts are important. Amharic is one of the under resourced languages whose natural language tools and applications are not yet built successfully. Therefore, parsing Amharic sentences is a necessary mechanism for many applications. Sentence parsing is one of the tasks of NLP tools which identify the syntactic structure of a specific sentence according to the grammar of a language. For this reason, many natural language applications underlie on sentence parser for better performance.

For foreign languages like English and Arabic, many sentences parsers are developed in different approaches. However in the case of Amharic, there are few works done which still require improvements and additional features. In addition, they are conducted in small dataset on specific types of sentences.

In our study, we have designed a similar system to parse all types of Amharic sentences using a top-down chart parsing algorithm using Context Free Grammar to represent the Amharic grammars. We have developed a lexicon generator to automatically generate the lexicon which is separated from the CFG. In addition, we have integrated a morphological analyzer in the construction of the lexicon. The main purpose of the morphological analyzer is to reduce the number of words required to be stored in the lexicon. The morphological analyzer results the morpheme of the given words so that words which have common root are represented by their morpheme in the lexicon. The parser is tested on test sentences which are extracted from different sources. Experimental results showed the effectiveness of the proposed parser.

Keywords: NLP, Parser, context free grammar, top-down chart parser, lexicon generator, lexicon, morphological analyzer.

Chapter One-Introduction

1.1 Introduction

Amharic language that is mainly spoken in Ethiopia is the second widely spoken Semitic language in the world (after Arabic) [1]. It differs from Arabic and Hebrew languages in terms of syntax, semantics, and morphology. Currently, the number of speakers is estimated to be more than 30 million [2]. The number of speakers of the language is now increasing because of two reasons. The first one is, it is the official language of the federal democratic republic of Ethiopia, a country of more than 85 million people [1]. Second Amharic is a written language with its own alphabet and written materials actively being used every day in newspapers and other media outlets [1].

However, Amharic has been one of the under resourced languages [1, 5] both in terms of electronic resources and processing tools. In recent times, there are attempts to develop processing tools. Two of the outcomes of these attempts are the medium sized-part of speech tagged news corpus [3] which is publicly available and a morphological analyzer [4]. The availability of these resources encourages researchers to involve in Amharic language processing by applying different NLP models.

One basic task in natural language processing is sentence parsing [6, 7, 8, 9]. It is the process of identifying the structure of a specific sentence namely noun phrase (NP), verb phrases (VP), noun (N), verb (V) etc. according to a given grammar [10]. The term parser is used in cases where the sentences are made up of information units of any kind and therefore it also deals with a number of sub problems such as identifying constituents that can fit together, testing the compatibility of number and tense. Even if much works have been done in different languages on different aspects of parsing, it is not possible to apply these parsers to Amharic language context directly. The main reason is Amharic language is highly inflectional, morphologically rich and has its own alphabets and grammatical rules.

The aim of this thesis is to develop a top-down chart parser. Chart parsing is an important method for natural language parsing. It consists of a tabular based, top-down chart parsing algorithm [11]. The basic idea is to obtain a table that contains all substructures generated during the parsing process through different iterations until all possible structures for the sentence are

obtained. The chart parser takes a sentence as an input and grammar rules that are referred to as production rules. Therefore, the proposed system will use Context Free Grammar (CFG) to represent the Amharic grammar rule. CFG grammars consist entirely of rules with a single symbol on the left-hand side. CFGs are very important classes of grammars for two reasons [11].

- The formalism is powerful enough to describe most of the structure in natural languages.
- It is restricted enough so that efficient parsers can be built to analyze sentences.

1.2 Statement of the Problem

Sentence parser is one of important tools in Natural language Processing. It serves as intermediate component for different higher level NLP applications like machine translation. Currently, Internet is one of the main sources of information. These enormous amounts of information could be used to enhance the development of a country by making it accessible to the public. To fully localize and utilize these resources, the translation of documents from one language to another may be necessary. For example the translation of English documents to Amharic may be required. Therefore, the machine translation, which uses Amharic language as an input and sentence parser as a component, plays a great role in solving the translation problem.

In addition to this, since Amharic is the official working language of Ethiopia, more and more materials are being published in Amharic nowadays and this creates a high interest of NLP researches in the language. For instance spell checker [49], grammar checker [27], question answering [17] and word sense disambiguation [48] are among the applications that require sentence parser for successful and full-fledged implementation. Besides to this, it is also helpful for language experts in language teaching for phrase identification and to see what relations words have in a sentence. However, even if there are few works done so far, there is no Amharic sentence parser designed to parse all Amharic sentences. Hence, the development of the parser for all types of Amharic sentences will have a vital role in the implementation of higher level NLP applications with high performance.

1.3 Objectives

General Objective

The general objective of this research work is to design and implement a top-down chart parser for Amharic sentences.

Specific Objectives

The specific objectives of this research are

- ✓ To study the basic word categories, morphological property, phrase structure, and sentences of Amharic Language
- ✓ To collect sample sentences from Amharic documents, for the preparation of the experiment
- ✓ To extract grammar rules using context free grammar formalism to represent the structure of Amharic sentences
- ✓ To prepare lexicon
- ✓ To design a general architecture of the system
- ✓ To develop a prototype for the parser
- ✓ To test the developed prototype

1.4 Methods

To design and develop the parser we will pass through the following activities.

Literature review

Literature review will be done on different areas that are considered to be relevant to our work. Since this research work is on a design and development of Amharic parser, it touches different approaches which can be used to develop a parser. The Amharic language properties will also be studied.

Data collection and Analysis

Data, Amharic Sentences of varies types, will be collected from real world Amharic text sources like Amharic grammar books and newspapers. In addition to this, detail analysis and preprocessing techniques will be made on the corpus or sentences for the experiment to improve the performance of the parser.

Design and development approach

Python programming language will be used to develop the prototype of this work. The implementation of the algorithm and the development of the user interface will be done in Python.

Evaluation

Finally, the evaluation of the developed system will be conducted by comparing the system results with manually parsed sentences (by the help of linguistic expert) and checking how much they are similar.

1.5 Scope

The main focus of this work is the design and prototype development for a top down chart parser for Amharic sentences. The prototype will be designed by studying the word classes of Amharic language, the types of sentences and their construction. However, it is not the scope of this work to incorporate the parser to higher level NLP applications like grammar checker, question answering, etc., as a component.

1.6 Application of Results

The result of this research work can be one of the core components of higher level NLP applications. A system has decisive roles in many areas of NLP for Amharic language. Therefore, researchers who are involved in increasing the capability of computers in processing Amharic language may benefit from the result of this study. Specially, researchers in the area of phrase recognition, conceptual parsing, machine translation, question answering, spell checker, text summarization, etc. are among the main beneficiaries. In addition to this, linguistic students

in the field of Amharic language can also apply the output of this study to parse sentences in the language automatically. It can also be used in language teaching for recognition of phrasal categories and to see the relationship between words in a sentence.

1.7 Organization of the Thesis

The remaining part of the thesis is organized as follows. Chapter 2 covers literature review in which different concepts and approaches related to our thesis are presented. Moreover, the Amharic language, its word classes, phrases structures, and different types of sentences with their respective examples are presented. Chapter 3 is about works related to our study which are done by other researchers in Amharic and other languages. Chapter 4 deals with the design of our system. It presents the general architecture of the system with its basic components and the discussion of the components and their interaction in the system. Chapter 5 focuses on the detail implementation of the system. It discusses the algorithms we used for achieving the goal of every component in the system supported with examples. Chapter 6 is about experiments done in every component and the results obtained together with their explanations. Chapter 7 presents conclusion and future work recommendations for the improvement of the system.

Chapter Two - Literature Review

In this Chapter, we provide a brief overview of grammar formalism and parser strategies. In order to examine how the syntactic structure of a sentence can be computed, we must consider two things: the grammar, which is a formal specification of structures allowable in the language, and the parsing technique, which is the method of analyzing a sentence to determine its structure according to the grammar. There are several types of grammatical formalism and parsing approaches and those that are important for our work are explained in this Chapter.

2.1 Grammar

Grammar can be defined as a formal specification for describing the rules and syntax allowable in the language according to which the parser attempts to analyze and determine the structure of a sentence [18]. In other words, the parser takes a sentence as an input and an abstract description of possible structural relations that may hold between the words as grammar rules to find a combination that generates a tree that could be the structure of the input sentence. The following is an example of the grammar rule for the sentence "**አበበ ወደ ትምህርት ቤት ሄደ**" or "Abebe went to school".

1. **S ➔NP VP**
2. **VP ➔ PP VP**
3. **NP ➔ NAME**
4. **PP ➔ PREP N**
5. **VP ➔ V**
6. **NAME ➔ አበበ**
7. **PREP ➔ ወደ**
8. **N ➔ ትምህርት, ቤት**
9. **V ➔ ሄደ**

Note: S is Sentnece; NP is noun phrase; VP is Verb phrase; PP is prepositional phrase; N is noune; V is verb; PREP is preposition.

Grammar spesifies two things: the set of grammatically correct sentenceses that are contained within the language and the structure to be assigned to each grammartical sentence in the language. There are different types of grammatical formalisms proposed by scholars so far. Among them Context Free Grammar [19], Transformational Grammar [19,20], Transition Network Grammar [21], Unification Based Grammar [22] and Probabilistic Grammar [23] are the most common and most widely used formalisms.

2.1.1 Context Free Grammar

A context free grammar (CFG) can be defined as a finite set of grammar rules which consist of exactly one non terminal on the left hand side but anything on the right hand side. In order to define the grammar rule, there are two kinds of symbols: the terminals, which are the symbols of the alphabet underlying the language under consideration, and the nonterminal, which behave like variables ranging over strings of terminals [24]. A CFG describes a language by specifing how any legal text can be derived from a set of production rules, each of which states that a given symbol can be replaced by a give n sequence of symbols.

A CFG is a four-tuple grammar G=(N, Σ, R, S) [24] where:

- N is a finite set of non-terminal symbols,
- Σ is a finite set of terminal symbols,
- R is a finite set of rules of the form $X \rightarrow Y_1 Y_2 \ldots Y_n$, where $X \in N$, $n \geq 0$, and $Y_i \in (N \cup \Sigma)$ from i=1….n,
- $S \in N$ is a distinguished start symbol.

For example, consider the following context-free grammar

S ➔ NP VP

NP ➔ Adj N

VP ➔ V NP

In the above simple context-free grammar, the set of non-terminals N specifies some basic syntactic categories S, NP, VP, Adj, N, and V. The set Σ contains the set of words in the vocabulary. The start symbol in this grammar is S: this specifies that every parse tree has S as its root. Finally, we have a context free rule

S ➔ NP VP

or

NAME ➔ **አበበ**

The first rule specifies that S (sentence) can be composed of an NP followed by a VP. The second rule specifies that NAME can be composed of the word '**አበበ**'.

Note that the set of allowable rules, as defined above, is quite broad, we can have any rule $X \rightarrow Y_1 \ldots Y_n$ as long as X is a member of N, and each Y_i for i=1….n is a member of either N or Σ. We can also have rules that have a mixture of terminal and non-terminal symbols on the right hand side of the rule , for example

VP➔ ወደ ትምህርት ቤት VP

PP➔ ወደ N

We even have rules where n=0, so that there are no symbols on the right hand side of the rule.

Example VP➔ ϵ

NP ➔ ϵ

Ε refers to the empty string, which means a particular non-terminal (e.g., VP), is allowed to have no words below it in a parse tree.

CFGs are a very important class of grammars for two reasons [6]:

- The formalism is powerful enough to describe most of the structure in a natural lanugage
- It is also restricted enough so that efficient parsers can be built to analyze sentences

2.1.2 Context Sensitive Grammar

These rules are used in a natural langauge to describe subject-verb agrement with respect to number, i.e., singular or plural as refelected in sentences: the student come and the sudent comes. A Context-Sesitive Grammar is a four-tuple, like that of context free grammar, G=(N, Σ, P, S) where N is a set of non-terminal symbols, Σ is a set of terminal symbols, S is the start symbol of the production and P is a finite set of production rules of the form $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$ (where a single non -terminal $A \in N$ and $\alpha_1, \beta, \alpha_2 \in (N \cup \Sigma)^+$) [25].

The production rules of the context sensitive grammar satisfy the following constriants: for the production rule of the form

- A ➔ B where A and B are strings of the aplhabet symbol, the length of (A) should be less than or equal to the length of (B).
- A ➔ y / x_z where A is a non-terminal symbol, y is a sequence of one or more terminal and non-terminal symbols, and x and y are sequence of zero or more terminal and non-terminal symbols.

The meaning of the second production rule is that A can be rewritten as y if it appears in the context ‘x_z’, i.e., immediately preceded by the symbols x and immediately followed by the symbols z [25].

2.1.3 Transition Network Grammar

Transitional Network Grammar (TNG) formalism describes the rules by using nodes and labeled-arcs in a transition network. One of the nodes is specified as initial state, or start state. Starting at the initial state, an arc can be traversed if the current word in the sentence is in the category on the arc. If the arc is followed, the current word is updated to the next word. Simple transiton networks are often called Finite State Machines (FSMs) and have equal expressive power to regular grammars. However, they are not powerful enough to describe all languages that can be described by CFGs [18]. In order for the transition network grammr to get the descriptive power of CFGs, it should allow arc labels to refer to other networks as well as word categories. Thus the grammatical formalism based on such notion is called Recursive transition network [26].

The other commonly used type of TNG fromalism for writing natural language grammars is Augmented Transition Netwrok (ATN), introduced by Woods [21]. This type of formalism represents the grammar in the assumption that if there is a path from the start state to some final state such that the labels on the arcs of the path match the words of the sentence, a sentence is in the language defined by the network.

2.1.4 Unification Based Grammr

Unification-based grammar is the other type of grammar fromalism that makes use of feature structures, like case, gender and tense including their values, represented in the lexical enrtries of words. In order to manipulate these feature structures, unification opertation is required. That is, the entire grammar can be specified as a set of constraints between feature structures. CFGs or any other type of grammar can be the backbone of a unification based grammar [22].

2.1.5 Probalisitic Context Free Grammar

A Probabilistic Context Free Grammar (PCFG) is a simple extension of a context free grammar in which every production rule is associated with a probability [30]. The probability is calculated by counting the number of times each rule is used in a corpus of parsed sentenses [18]. A PCFG is a quintuple G = (N, Σ, R, S, Q) where N is a finite set of non-terminal symbols, Σ is a finite set of terminal symblos, R is a finite set of production rules of the form A→α (where A is an element of N and α is an element of (Σ U N)*), S ∈ N is the start symbol, and Q : R → [0,1] is a function that assigns a probability to each member of R [15,16].

Therefore, the probalility *P(y)* of a parse tree *y* is defined as the product of probabilities of all rule applications in the derivation of y.

$$P(y) = \prod_{i=1}^{|R|} qi\ count(i, y)$$

where:

- *y* is a parse tree gennerated using PCFG.
- *count(i,y)* is the number of times that the ith production rule *ri* ∈ R is used in the derivation of *y*.
- *qi* is the probability of rule *i*.

2.2 Lexicon

Lexicon, which is a linguistic term for dictionary, is a structure that stores lexical rules separately from grammar rules by specifying the possible categories of each word to improve the efficiency of the parsing algorithm [28]. It contains information about all words that can be used in the grammar, including all the relevant value restrictions. When a word is ambiguous, it may be described by multiple entries in the lexicon, one for each different use [18].

A simple context free grammar lexicon indicated by Atelach Alemu and Daniel Gochel [15, 16] is shown below as an example,

N➔ ሰው

V➔መጣ

Adj➔ትልቅ

Adv➔ቶሎ

PREP➔ወደ

Symbols on the left hand side are used to specify the part of speech (POS) categories of words on the right hand side. Therefore, a grammar rule need not contain any lexical rules of the above form; instead it should be specified in a separate lexicon.

2.3 Approaches of Sentence Parsing

Parsing natural language text is challenging because of the problems like ambiguity. It is considered to be an important intermediate stage for semantic analysis in natural language processing applications such as information retrieval (IR) [10], information extraction (IE) [50], and question answering (QA) [17]. It is the process of automatically building syntactic analysis of a sentence in terms of a given grammar. The output of parsing is something like a tree which displays the dominance and precedence relation between constituents of a sentence [18]. This process is accompanied by the algorithm that searches through different ways of combining grammatical rules to find a combination that generates a tree that could be the structure of the input sentences.

By considering the reasoning mechanism of the algorithms, scholars classify the methods of sentence parsing into two types: rule based and stochastic approaches [29].

2.3.1 Stochastic Approaches

This approach, also called corpus-based approach, is based on the use of text corpora. The approach uses the idea of Bayes (Network) theorem, i.e., independent event and the Markov assumption are used to determine the most likely lexical sequence of each word in a given sentence [29]. Based on the type of text corpora used, the corpus based approach can be further categorized into supervised and unsupervised approaches.

Supervised Approach

Supervised approaches use annotated text corpora and systems which are developed using this approach are called supervised parsers. They use probability or statistics in analyzing the syntactic structure. The main source of information for a supervised parser is the lexicon (which lists each word with the entire possible lexical category for each word) and the list of contextual probabilities for each lexical category. The lists of contextual probabilities indicate the particular lexical category that is appropriate for a particular context. However, this approach has two main drawbacks: lack of manually or automatically parsed text (corpora) and the manual parsing is required each time whenever the parser is needed to be applied on a new text.

Unsupervised Approach

Unlike supervised approach, unsupervised approach uses natural corpus as those found in newspapers and books. For this reason, they do not require any pre-tagged text in the training process. Some probabilistic information generated from the corpus is used to develop the syntactic analysis system. These parsers also work based on the assumption of Markov model in that a set (lexical categories in this case) with directed edges labeled with transition probabilities that indicate the probability of moving to the state at the end of the directed edge is utilized.

2.3.2 Rule-based Approaches

The rule-based approach attempts to parse a sentence based on the information from the knowledge base, i.e., grammar rules. Systems which are based on such rules learn a set of rules automatically based on a given list of strings and then parse sentences following these rules. There are three ways in which this approach can be applied: top-down, bottom up, and chart based approach (which integrates the advantages of the two).

Top-down Parsing

Top down parsing is one strategy that builds parse tree from the start symbol (S). Top down parsing is goal oriented. The goal is towards parsing the sentence according to the grammar production. To build a parse, it repeats the following steps until the parse tree matches the input string [18].

1. At the start node S, select a production with S on its left hand side and for each symbol on its right hand side, construct the appropriate child.
2. When a terminal is added to the tree being constructed that doesn't match the input string, then backtrack.
3. Find the next node to be expanded.

If the parse tree does not match the input string then it means that input string is wrong [7]. Top down methods have the advantage of being highly predictive. A word might be ambiguous (i.e., a word may have multiple word categories) in isolation. But if some of those possible categories cannot be used in a legal sentence, then these categories may never even be considered [18]. However, this method has a problem of reduplication effort as large constituents may be rebuilt again and again as they are used in different rules. In addition to this, since the top-down parser predicts the end string from the given grammar, it has to backtrack to where it made the wrong decision at each time when it chooses the wrong path. For example, if the grammar rule contains different alternative production rules to which NP can be extended (like NP ➔ N N P, NP ➔ N P N, and NP ➔ N V N) the parser select one alternative and workout its consequences. If the choice turned out to be wrong, it backtracked. Moreover since there is no recording of each substitution, there will be unnecessary reduplication for the same non-terminal at each time it encountered. The other serious problem is that it sometimes get stuck in a loop, if one or more of the grammar rules are left recursive, i.e., if there is a non-terminal symbol both on the right and left side of the grammar rule. For example a grammar rule NP➔ Adj NP has the same NP in both sides of the rewrite arrow.

Bottom up parsing

Bottom up parsing starts with words in a sentence and uses production rules backward to reduce the sequence of symbols until it consists solely of S [28]. This operation is repeated, at each state, using the sequence of highest level labels as the new string to operate on. The task of the parser is that of attempting to group words into their respective categories together (e.g., a sequence of ART ADJ N can be identified as NP) in a manner permitted by the grammar [18]. Unlike top down parsing, the bottom up parser only checks the input sentence once, and only builds each constituent exactly once. This is because a bottom up parser works from left to right, i.e., it does everything it can with the first item before exploring what it can do with the next

items. But bottom up parser can also get stuck in a loop if the grammar has empty productions. Even if both bottom up and top down parsers have advantages, they are inefficient and have a worst case exponential run-time as the parser would tend to try the same matches again and again, thus duplicating much of its work unnecessarily. To avoid this problem, a data structure called a chart is introduced that allows the parser to store the partial results of the matching it has done so far so that the works need not to be reduplicated [18].

Chart Parser

Chart parser combines some of the advantages of top-down and bottom-up approaches. The combination of the selective behavior of the top-down algorithm in building partial parser based on left context with the bottom-up algorithm behavior building each partial parse only once, form a chart parser [18]. The main objective of chart parsing is to improve parsing efficiency. Therefore, it considers three points for the improvement of the parsing efficiency: it doesn't do twice what can be done once; it doesn't do once what can be avoided altogether and it doesn't represent distinctions if that is not the concern of the study.

In general, a chart parser has three main constituents: a key list, a chart and a set of edges. A chart is a set of chart entries each of which consists of the name of terminal or non-terminal symbols, the starting point of an entry and the entry length. The key list is push down stack of chart entries that are waiting to be added into the chart. The edges are rules that can be applied to chart entries to build them up into large entries [10]. The chart maintains the record of all the constituents derived from the sentence so far in the parse. It also maintains the record of rules that have matched partially but are not complete. Recording of intermediate results is a form of dynamic programming that avoids duplicate work [12].

The basic operation of a chart based parser involves combining an active arc (also called edge) with a completed constituent. The result is either a new completed constituent or a new arc that is an extension of the original active arc. New completed constituents are maintained on a list called the ***agenda*** until they themselves are added to the chart. For example [0, 5, S➔ NP VP •] means that an NP followed by a VP are combined to make an S that spans the string from 0 to 5. The number here indicates the indexing of the grammar rules. The symbol • in an edge separates what has been found so far from what remains to be found. The number before the symbol S

indicates the indexing of the grammatical categories. Edges with • at the end are called complete edges. The edge [0, 2, S ➔NP • VP] says that an NP spans from 0 to 2, and if the parser could find a VP to follow it, then the parser would have an S. Edges like this with the dot before the end are called active arcs [18].

As it is described earlier, chart parser is driven by an agenda of completed constituents and the arc extension, which combines active arcs with constituents when they are added to the chart. The technique of extending arcs with constituents can be applied by using both bottom-up and top-down approach. But the difference is in how new arcs are generated from the grammar. In bottom-up approach, new active arcs are generated whenever a completed constituent is added that could be the first constituent of the right hand side of the rule. In the top-down approach, new active arcs are generated whenever a new active arc is added to the chart [10]. For this reason the number of constituents generated using a top-down chart parser is less than the number of constituents which are generated using bottom-up chart parser. Therefore, the top-down chart parser is considerably more efficient for any reasonable grammar.

2.4 The Amharic Grammar

Parsing is all about discovering a structure in an input based on external information known about the elements of the input and their order. The external information consists of a lexicon, which is a list of input words, and a grammar which describes which structure may be built from sequence of words. In this Chapter, we will discuss the Amharic language in terms of word classes, phrase types and sentence types and their formation due to their importance in the study. The discussion of this chapter is based on the Amharic grammar book "የአማርኛ ሰዋስው" [42] and previous works [15, 16, 34].

2.4.1 Word Classes

According to Baye Yimam [42] word classes are classified into five categories which are noun, verb, adverb, adjective and prepositions. Baye Yimam reduces the number of word classes from eight, according to previous grammar book [43] to five. Pronouns, which have been one of the eight categories, are considered as nouns in [42] because of the fact that pronouns can replace nouns. Conjunctions on the hand have the same property and function as preposition so that they

are included in preposition word class. Interjection word classes, in the previous grammar books, are not considered as grammatical categories in [42] because they are words without syntactic function.

Noun

Noun is a word that is used to identify any of a class of people, place or things. The plural form of nouns in Amharic language take the suffix /-ኦች/, /-ኦች/, and for the accusative case marker it takes /-ን/. For example, the word "ሰው" (man) in the sentence "ትናንት የመጣው ሰው". Amharic nouns can exist either in their original form or as originated from a different word category. Nouns that appear in their original form are said to be primitive whereas when they are created from other categories they are referred to as derived. Nouns can be used as a subject or an object in a sentence.

Verb

Any kind of word that usually comes at the end of the sentence and takes suffixes like/-ሁ/, /-ህ/, /-ሽ/ to mark subjects I, You, and She respectively is said to be a verb. For example, the word "በላ-ሁ" in the sentence "እኔ ምግብ በላ-ሁ". Similar to nouns, verbs can also be derived from other word categories by affixing different morphemes and form compound words of stems with verbs.

Adjective

These are words used to name the attribute of a noun or pronoun. They usually precede the noun or pronoun that they describe or modify. But all words that come before a noun or a pronoun may not be adjectives. For example, the word "ነበዝ" in the sentence "አበበ ነበዝ ተማሪ ነው". Like nouns and verbs, adjectives can be derived from other word categories.

Adverb

Adverb is a word or phrase that modifies the meaning of an adjective, verb, or other adverb of a sentence. It also gives additional information about the action referred by the verb. Adverbs refer to places, time, cause, degree or circumstances, etc., of the action mentioned by the verb However, most of the time, the place, time, cause, degree or circumstance of an action is referred by noun phrases or prepositional phrases which contain an adverb. Therefore, adverbs can be

found either in their primitive form or as their derivative form. The primitive adverbs are very few in number that are: “ገና” (yet), “ክፉኛ” (severely), “ፉሎ” (quickly), “ጅልኛ” (foolish), etc.

Preposition

A type of word that usually precedes a noun or pronoun and expressing a relation to another word or element is said to be preposition. However, sometimes they may come after words. For example “አጠገብ” in “ከተማሪ አጠገብ” and “ማዶ” in “ከወንዙ ማዶ”. In general, words which satisfy the following conditions can be said preposition: they come before a noun and refer time, place, purpose etc.; they do not add any type of affixes and they cannot be a base for the formation of other words. Therefore, words like “እንደ”, “ስለ”, “ወደ”, “ከ”, “ለ”, etc., are some examples of prepositions. Prepositions have no meaning by themselves. But when they are attached with nouns they can form prepositional phrases that are used to refer various adverbial actions. For example “ስለ ትምህርት ብዙ ሰው ይቆረቆራል” and “በስንት ሰዓት ይመጣል”.

Conjunction

A word that is used to connect words, phrases, clauses or sentences is said to be conjunction. “ወይም” (or) and “እና” (and) are examples of Amharic conjunctions. For example in the sentence “አበበ እና ከበደ ወደ አዲስ አበባ ሊመጡ ነው” “Abebe and Kebede are going to come to addis Abeba” the conjunction “እና” connects the two nouns Abebe and Kebede. There are two types of conjunctions: coordinating conjunction and subordinating conjunction. Coordinating conjunctions are used to connect two words, phrases, clauses or sentence which are equal, whereas subordinating conjunctions connect two parts of words that are not equal (i.e., dependent and independent clauses). Subordinating conjunctions introduce dependent clause(s) and indicate the nature of the relationship among the independent clause(s) and the dependent clause(s).

Numeral

Numeral is a figure, word or group of figures which denote a number. Both cardinal and ordinal number types are represented by numerals. Cardinal numbers are used to denote quantity (one, two, three, etc.,) whereas ordinal numbers define a thing’s position in a series, such as first,

second, or third. In Amharic, ordinal numbers are formed from their equivalent cardinal numbers and the suffix /-ኛ/.

2.4.2 Phrase Categories

Phrase is a small group of words standing together. It is a syntactic structure that is wider than a word and smaller than a sentence. Phrase can be constructed only from a head word or a head word combined with other words or phrases. When a phrase is constructed from a head word and other words or phrases, the other words or phrases can be specifiers, modifiers or complements. Specifiers are words used to specify the identity, location, number, possession, etc. of the head word. Specifiers can be either primitive or derived. For example, in the phrase "የካሳ መጽሀፍ" (Kassa's book) the specifier is "የካሳ" (Kassa's) that shows the owner of the book.

Modifiers on the other hand are used to indicate the amount, time, place, type, etc. of the head word or phrase. Adjectival phrase, noun phrase, prepositional phrase or sentences can be considered as modifiers. For example in the phrase "ነጭ መኪና" (white car), the word "ነጭ" (white) indicates what type of color the car has. Unlike specifiers and modifiers complements are words or phrases that are used to make the ideas complete. For example in the sentences "ዳቦ በላሁ" (I ate bread) and "የስንዴ ዳቦ በላሁ" (I ate wheat bread), the first sentence does not give complete information about the bread but in the second sentence "የስንዴ" (wheat) is a complement that indicates from what the bread is made.

The types of the phrases for a language are determined by the categories of words in a language. Therefore, since there are five word categories in Amharic language, there are five phrase classes which are noun phrase (NP), verb phrase (VP), adjectival phrase (AdjP), adverbial phrase (AdvP), and prepositional phrase (PP) [42].

Noun phrase

A noun phrase (NP) is composed of a noun or pronoun as its head and other constituents. NP can be simple (when the head is a single noun or pronoun) or it can be complex (when it is formed by the combination of a noun with other word classes, including noun word category, or phrases). For example "እኔ", "እሷ", "እነሱ", "አንበሳው", "መኪናው" are simple noun phrases and "የሳር ቤት",

“ላሞች በላ” “የወርቅ ቀለበት” are complex noun phrases. Noun phrases are used to refer objects, places, concepts, events or qualities.

Verb phrase

A verb phrase (VP) is made up of a verb, which is the head of the phrase, and other constituents like complements, modifiers or specifiers. Verbs of the verb phrase can be classified as transitive and intransitive based on the word category of the complement they take. Transitive verbs take noun phrases as their complement whereas intransitive verbs do not; instead they take prepositional phrase. For example in the VPs “አስቴር እንቁላሉን ሰበረችው” (aster broke the egg) and “አበበ ልጁን ገረፈው” (Abebe hit the boy), the complements of the head verb are transitive NPs. Therefore, the verbs “ሰበረችው” (broke) and “ገረፈው” (hit) are transitive verbs. The verbs in the VPs “ወደ መርካቶ ሄደ” (he went to merkato) and “ወደ ቤቱ ገባ” (he entered to his home) on the other hand are examples of intransitive verbs because of the prepositional phrase complement they have.

Like noun phrases, verb phrase can also be simple or complex. Simple VPs have single head verb. But when the VP contains more than one verb or embedded sentence, it is said to be complex VP. For example “ብዙ ጊዜ ለሴትዮዋ በስልክ ሚስጥር ነግሯታል” (he told the leady a secret by telephone a lot of time).

Adverbial phrase

Adverbial phrases (AdvP) are constructed from one or more adverbs, as modifier, and other word categories. However, unlike other phrases AdvPs do not take complements as their constituents. Most of the time, the modifiers of adverbial phrases are PPs. For example in “አስቴር እንደ ወንድሟ ቶሎ ተነሳች” (aster stands fast like her brother), the adverbial phrase is “እንደ ወንድሟ ቶሎ” which has the comparative PP modifier “እንደ”. Concerning about the number of adverbs in the adverb phrase, it can be single or it can be more than one. For example in the adverb phrase “ክፉኛ ተጋጩ” (they crashed severely) the head adverb is “ክፉኛ” (severely). In this example the AdvP is composed of a single adverb while in the example “ቶሎ ቶሎ ተራመደ” (he walked briskly) the phrased is made up of two adverbs “ቶሎ” “ቶሎ”.

Adjectival phrases

Adjectival phrases (AdjP), like noun phrases and verb phrases, are composed of an adjective head and other constituents such as complements, modifiers and specifiers. For example, in the sentence "አበበ እንደ እህቱ በጣም መልከኛ ነው" (abebe is very handsome like his sister) the adjectival phrase is "እንደ እህቱ በጣም መልከኛ". In this phrase the modifier is "እንደ እህቱ" (like his sister) and the specifier is "በጣም" (very). AdjP can be simple or complex like that of VPs and NPs. Examples like the above are said to be simple AdjPs whereas AdjPs which contain embedded sentences are complex AdjPs.

Prepositional Phrase

Prepositional phrases (PPs) are made up of a preposition head and other constituents like nouns, noun phrases, verbs and verb phrases. Unlike other types of phrases, preposition cannot be a phrase by itself unless it is combined with other constituents. Noun or noun phrase constituents come after the head preposition whereas when the complement is prepositional phrase the head preposition appears on the right hand side of the phrase.

Example. "ወደ ትልቁ ትምህርት ቤት" (to the large school) ……

"ወንበሩ ከጠረጴዛው አጠገብ" (the chair near to the table).

2.4.3 Amharic Sentences

A sentence is a group of words or phrases that is complete in itself, conveying a statement, question, exclamation, or command and typically containing a subject and predicate. For example when we see the following phrases: "አንድ ትልቅ ልጅ" (One big child) - Noun phrase, "በሁለት ብር" (by two birr) - Prepositional phrase and "አራት እንቁላል ገዛ" (he bought four eggs) - Verb phrase, they do not covey full meaning when they are spoken in separate, instead they raise questions like what did?, who did?, and what has done?. However, when these phrases are combined together they form a sentence "አንድ ትልቅ ልጅ በሁለት ብር አራት እንቁላል ገዛ" (One big child bought four eggs by two birr) and answers all the above questions. From the structural point of view, sentences are basically constructed from noun phrase and verb phrase. Regarding the remaining phrases, they may be included within one of the above main components of a

sentence. Therefore in the above sentence, we have the noun phrase "አንድ ትልቅ ልጅ" (one big child) and the verb phrase "በሁለት ብር አራት እንቁላል ገዛ" (bought four eggs by two birr) which in turn contains the prepositional phrase "በሁለት ብር" (by two birr). The structure of a sentence can be either simple or complex based on the number of verbs it contains.

2.4.3.1 Simple Sentences

In Amharic language, simple sentences are formed from a noun phrase, which may contain other phrases except verb phrase, and a verb phrase that contains a single verb and a predicate. Consider the following sentences.

i. "አስቴር መጽሃፍ ገዛች" (Aster bought a book)
ii. "ካሳ ወደ ጎንደር ሄደ" (Kassa went to Gondar)
iii. "አበበ ለልጁ ኳስ አመጣለት" (Abebe brought a ball for his son)

There is a single verb in each sentence, "ገዛች" (bought), "ሄደ" (went) and "አመጣለት" (brought) respectively. In the first sentence, the verb "ገዛች" is transitive which takes the object "መጽሃፍ". The verb in the second sentence on the other hand does not take any object so that it is intransitive verb. In the third sentence the transitive verb has two objects "ለልጁ" (son) and "ኳስ" (ball). Therefore, simple sentences may contain intransitive verbs or transitive verbs with one or two objects.

According to Baye Yimam [42] simple sentences are classified into four based on the purpose for which they are spoken. These are declarative, interrogative, negative, and imperative sentences. The structure of a sentence is also determined by its type.

Declarative sentences

Declarative sentences have the nature of making a declaration or they are just simple statements. Declaration sentences are used to convey ideas and feelings of the speaker about things, and happenings. In Amharic, declarative sentences always end with the Amharic punctuation mark አራት ነጥብ ("::") which is equivalent to full-stop (.) in the English language. For example, sentences like "ስጋው ጮማ ነው" (the meat is fatty), "አስቴር አስተማሪ ሆነች" (Aster became a teacher) are some types of declarative sentences.

Negative sentences

These are sentences that make declarative sentences negative in meaning. They simply negate a declarative sentence. For example the sentence "ካሳ ወደ ትምህርት ቤት አልሄደም" (Kassa didn't go to school) is negative because the prefix "አል" in the verb "አል-ሄደም" makes the sentence negative declarative sentence.

Interrogative Sentences

A sentence that asks about the subject, complement, or the action specified by the verb is said to be an interrogative sentence. The question can be the one that asks about known things to be sure or the one that asks the unknown thing. In order to ask the unknown thing, the enquirer can use interrogative pronouns whereas in order to make sure the known thing the enquirer can use assurance words or he can change the way of his speech. An interrogative sentence always ends with a question mark (?). For example, in the sentence "ደመወዝክን ወሰድክ?" (did you take your salary?), the inquirer knows that the person will take his salary but he doesn't know whether he took or not. However, if the inquirer asks questions like "የማን መኪና ናት?" (who is the owner of the car?), he is asking for unknown thing (i.e., the owner) so that he uses the pronoun "ማን" (who). Therefore, in order to construct such interrogative sentences, the inquirer usually uses interrogative pronouns like "ማን" (who), "ምን" (what), "የት" (where), "ስንት" (how many), and "መቸ" (when).

Imperative sentences

Imperative sentences are used to convey commands or instructions. Most of the time, the subject (that is second person pronoun) of the sentence is omitted but since Amharic words are highly inflected, subject marker prefixes indicate the specific subject. However, sometimes when the command is passed for a third person, the subject of the sentence can be third person pronoun or noun. For example, in the sentence "ልብስ እጠቢ!" (wash clothes!), the subject is "አንቺ" (you) which is second person feminine singular). Whereas in the sentence "አስቴር ምግብ ታብስል" (let Aster cook food), the command is for the third person that doesn't exist at the time of speech so that the subject is "እሷ" (she) which is the third person feminine singular.

2.4.3.2 Complex Sentences

Complex sentences are formed from either complex noun phrase or complex verb phrase or both. In other words, a complex sentence can have a complex NP and a simple VP, a simple NP and a complex VP or both complex NP and complex VP. Complex NPs contain at least one embedded sentence which can be complement or other type phrase. On the other hand, complex VPs contain at least one sentence or more than one verb. Look at the following examples which are taken from [34] to see the formation of complex sentences from noun phrases and verb phrases.

Example 1: - "ካሳ የገባበት የሳር ቤት በጣም ትልቅ ነው" (the thatched house that Kassa has entered is so big). In this sentences the head of the noun phrase is "ካሳ የገባበት የሳር ቤት" (the thatched house that Kassa has entered). The head with the complement "የሳር" (thatched) forms simple noun phrase "የሳር ቤት" (thatched house) and this noun phrase are combined with the embedded sentence or clause "ካሳ የገባበት" (that Kassa has entered) to form complex noun phrase. But the clause that makes the complex phrase is dependent which is identified by the morpheme "የ" (that).

Example 2: - "ካሳ የገዛው መጽሀፍ ዛሬ ጠፋ" (the book that Kassa has bought is lost today)
Here the NP is "ካሳ የገዛው መጽሀፍ" (the book that Kassa has bought) and the head "መጽሀፍ" (the book) is combined with the dependent clause "ካሳ የገዛው" (that Kassa has bought) to form complex NP. The relativizer "የ" (that) in the dependent clause indicates that the clause is a subordinate clause and it cannot stand alone. Similarly VPs can be complex if they contain at least one sentence or more than one verb. Moreover, complex VPs also contain dependent clauses or sentences like complex NPs. These clauses can be complement or modifier.

Example 3: - "አስቴር ካሳ መኪና እንደገዛ ሰማች" (Aster heard that Kassa has bought a car)
In this sentence, the verb phrase is "ሰማች" (heard) and the embedded sentence or dependent clause is "ካሳ መኪና እንደገዛ" (that Kassa has bought a car). In this example the dependent clause is used as a complement in the VP construction. The prefix እንደ- in the clause indicates that the clause is dependent.
Example 4: - አስቴር [[ካሳ ወደ ጎጃም ሰለሄደ] አለቀሰች] "Aster wept for the reason that Kassa went to Gojam". In this sentence the word "ሰለሄደ" is dependent clause which is identified by the prefix "ሰለ-". This clause is a modifier of the phrase "ካሳ ወደ ጎጃም ሰለሄደ". The above examples show that

embedded sentences or dependent clauses in the VPs can be modifiers or complements. More than one embedded sentences or dependent clauses can also be found in a complex VP.

Example 5: - "ካሳ [ከጎጃም እንደመጣ] [አስቴር ወደ ናዝሬት እንደ ሄደች] ሰማ" (when Kassa came from Gojam he heard that Aster went to Nazret). In this example the VP is "ከጎጃም እንደመጣ አስቴር ወደ ናዝሬት እንደ ሄደች ሰማ" (When came from Gojam he heard that Aster went to Nazret) and there are two dependent sentences in it: "ከጎጃም እንደመጣ" (came from Gojam) and "አስቴር ወደ ናዝሬት እንደ ሄደች" (that Aster went to Nazret). These sentences are used in this phrase construction as a modifier and complement respectively. Unlike the above examples, complex sentences also may contain both complex NP and complex VP.

Example 6: - "ከጎጃም የመጣችው ልጅ ካሳ እንደወደዳት አወቀች" (The girl who came from Gojam knew that Kassa is in love with her). In this sentence both the NP and VP are complex which are "ከጎጃም የመጣችው ልጅ" (The girl who came from Gojam) and "ካሳ እንደወደዳት አወቀች" (knew well that Kassa is in love with her), respectively.

2.5 Summary

In developing a sentence parser, grammar formalism and parsing methods are needed. There are several formalisms and parsing algorithms that have been proposed and used in various researches. We have discussed about some of the grammar formalisms and parsing approaches with their characteristics. Context Free Grammar, Transition Network Grammar, Probabilistic Context Free Grammar, Context Sensitive Grammar, and Unification Based Grammar are discussed. Among the strategies, Stochastic Approaches (which in turn include supervised and unsupervised methods) and Rule-based (that can be applied in either of top-down, bottom-up or chart parsing ways) are presented.

In our research, we propose to use Context Free Grammar for the grammar rules and top-down chart for the parsing. We choose CFG because it is easier to maintain, can add new language features and automatically build efficient parser. The top-down chart parsing, on the other hand is chosen because it does well if there is useful grammar-driven control. It has the advantage of both the traditional top-down and bottom-up parsing. It also avoids problems of left recursive and empty productions by storing partial results of the matching it has already done.

In addition to the grammar formalisms and parsing methods, we discussed the Amharic grammar in detail in this Chapter. We discussed the categories of word classes based on what the language scholars classified. There are noun, verb, adjective, adverb, preposition, conjunction, and numeral word categories. In this study, pronouns are taken as noun as suggested by Baye Yimam [42]. Five types of phrases that are formed from two or more word classes have also been discussed. Such phrases are noun phrases, verb phrases, adjectival phrases, prepositional phrases, and adverbial phrases. Furthermore, sentence types were also the concern of the Chapter. There are simple and complex types of sentences in Amharic language. Simple sentences can be further classified as declarative, negative, interrogative and imperative sentences. Complex sentences, on the other hand, can be formed in different ways. A complex sentence can be constructed from simple noun phrase and complex verb phrase or from complex noun phrase and simple verb phrase or it can be from both complex noun phrase and complex verb phrase.

Chapter Three - Related Works

In recent years, there have been much works done in Natural Language Processing (NLP). Sentence parser is one of the most important NLP tools. Even though there are less works for Amharic language regarding to parsing, much works have been done in different languages on different aspects of parsing based on various approaches.

In this Chapter, we will review previous Amharic and other language works that are more related to our study. We will also identify the gaps between our work and those related works.

3.1 Parsers for Amharic Language

The Amharic sentence parser, Automatic Sentence parsing for Amharic text [15], is an effort towards syntactic analysis of Amharic texts. The study attempted to develop a simple automatic parser for Amharic sentences to address the problem of developing systems that can automatically process Amharic text. The research has been developed using the Inside Outside algorithm with bottom up chart parsing strategy. The Probabilistic Context Free Grammar (PCFG) has been used as a grammatical formalism to represent the phrase structural rules of the Amharic language. The study, in general, tried to combine probabilistic formalism and rule based reasoning for the purpose of developing automatic sentence parser.

In this research, a small sample corpus (i.e., 100 sentences) was selected for the experiment from an Amharic grammar book titled "የአማርኛ ሰዋስው" [31]. The sentences were only from similar type of sentences, i.e., simple declarative type of sentences which are composed of four words. The selections of the sentences were carried out randomly; however, they were composed of two or more of the phrase classes of the language. The sample corpus then needs to pass through sentence pre-processing phase to make them have a suitable format. As a result, the sentences were automatically tagged using the Part of Speech (POS) tagger developed by Mesfin Getachew [32] to assign a part-of-speech tag like noun, verb, pronoun, preposition, adverb, adjective or other lexical class marker to each word in the sentence. Manual hand parsing was also the other pre-processing phase done by the researcher after the corpus has passed through the POS tagger. The researcher picked 80 sentences randomly from the sample to conduct the probability calculations for the words in the sentences, grammar rule inductions and the assignment of the

probability to the grammar rules. The remaining 20 sentences were used as a test set. However, the number of sample sentences was small and only from one type of sentences which is declarative sentences and limited only to four words length.

The pre-processed texts were then used for extracting the probabilistic context free grammar. After this, the number of times each rule was used in the corpus which contains parsed sentences was used to estimate the probability of each rule being used. Finally, the probability values were assigned to each rule extracted from the parsed sample that divides the number of rules used by the total word category occurrences on the left hand side of rules used (i.e., in the PCFG).

The second important work in Amharic sentence parser is done by Daniel Gochel [16] which has integrated ideas and outputs of previously attempted Amharic NLP prototypes [15] to develop automatic sentence parser particularly for complex Amharic sentences. The study is just an extension of Atelache Alemu's [15] work specifically focused on complex Amharic sentences. The parsing algorithm and the grammar formalism adopted in this research is similar with the Automatic sentence parser for Amharic sentences [15] which are Input Output Algorithm with bottom-up strategy and PCFG rules respectively.

The researcher has collected 350 Amharic complex sentences from two commonly known grammar books in the language, named as "የአማርኛ ሰዋስው" [31] and "Amharic for beginners" [33]. The sentences were selected in a way that each sentence contains two or more phrase classes and one of the clause types (i.e., relative clause, reason clause, and time clause) in the language. Then after, sentences were passed through morphological analyzer and POS tagger for the purpose of pre-processing. Morphological analyzer is one of the most important NLP systems in the development of sentence parser. In representing the lexicon for sentence parser, it would only need to store one entry for a word that can have a number of derivations and inflectional words.

However, if we do not have one morpheme in the lexicon for those related words, there would be a large number of words collected in the lexicon which is a serious problem to manipulate and access a huge dictionary. The POS tagger takes a string of stems (which is a sentence containing only stems of words it was built from) that the morphological analyzer outputs. Inputting of a

string of stems into the POS tagger has a potential advantage in making a considerable reduction on the occurrence of getting words whose category is not known by the POS tagger [1].

Among the sample corpus, 280 sentences were selected as training set to calculate the probability of words in the sentences, grammar rule formation and probability assignment to the grammar rules. The remaining 70 sentences were taken as a test set. The extraction of the probabilistic context free grammar from POS tagged and hand parsed sentences was performed in similar way to what Atelach Alemu [15] did. The PCFG is then converted to Chomsky Normal Form (CNF) for the simplicity and its easiness to manipulate because of the binary structure it has. However, the conversion is done manually by replacing rules of the form **A➔BCDE** (p), where p is the probability, by a set of rules [16]: **A➔BC'** (p), **C'➔CD'** (1) and **D'➔ DE** (1).

Since both Atelach Alemu [15] and Daneil Alemu [16] have done the implementation in the same way, they have used Input Output algorithm with the bottom-up chart strategy. As a result, when the parser is given a sentence which is already pre-processed, the parser gives the probability of the selected parse structure, the parse result, and the grammar rules used in the parsing.

Abeba Ibrahim [34] developed a phrase chunker that divides a text in syntactically correlated words from a stream of text for Amharic language. Text chunking is an intermediate step for full parsing. The main objective of the research was to extract different types of Amharic phrases by grouping words which are found at different level of the parser and transform the chunker to full parser. The researcher used Hidden Markov Model (HMM) to develop the chunker and bottom-up approach with transformation algorithm to transform the chunker to the parser. For the identification of the boundary of the phrases, the author used Inside Outside Beginning (IOB) chunk specification method.

In this research, 320 different types of sample sentences were collected from Amharic grammar books and news of Walta Information Center (WIC) for the training and testing dataset. Among 320 sentences, 288 sentences were used to train the system while 32 sentences were used to test the system. Since the sentences collected from Amharic grammar books are not tagged, they were analyzed and tagged manually. The entire data set were also chunk tagged manually for the training set. However, all kinds of Amharic sentences were not included in the study.

Interrogative and imperative were not part of the study. In addition to this, the size of the corpus is small.

3.2 Parser for Oromo Language

The automatic Oromo sentence parser which has been developed by Diriba Megersa [35] was aimed to parse simple sentences for Oromo language. The study has been conducted using the chart algorithm with the grammar formalism Head-driven Phrase Structure Grammar (HPSG) compiled into left to right table. It is a representation that allows to minimize the number of syntactic rules and to provide rich and well structured lexical representation. The grammar table contains the LHS (left hand side) rules that represent the non-terminal nodes in tree construction and RHS (right hand side) rules that contain constituents of the phrase rule on the LHS. In addition to grammar rules, the system also contained mapped rules that would be created automatically while the parser reads the grammar rules. The main purpose of the mapped rule was to control the problem of recursive rules in the main grammar rule (i.e., it protects the parser from entering into an infinite loop as a result of recursive rules). Along with the chart algorithm, the system has also used the supervised learning algorithm to enable the parser predict unknown and ambiguous words and a morphological analyzer to split words into root form and their corresponding morpheme.

For the purpose of the experiment, the researcher collected a sample text which consists of 352 sentences from the handout "Seerluga Afaan Oromoo" that was used as a reference for the course Oromo syntax in the department of language and literature, Oromo unit at the time. The sample data was divided into two sets: the training set which contains 300 sentences and the testing set with the remaining 52 sentences. However, in addition to the small number and similar sentence type of the text, the part of speech tagger that preprocesses the text to improve the performance of the parser was not used.

3.3 Parsers for Arabic Language

The top-down chart parser for Arabic sentences [10] is designed for parsing simple Arabic sentences, which includes nominal and verbal sentences within specific domain Arabic grammar. The grammar formalism CFG was used to represent the Arabic grammar. The researchers

developed the grammar rules which give the precise description of grammatical sentences first. Then, they implemented the parser which assigns grammatical structure of the input sentences. The system was tested on 70 sentences collected from Arabic documents with different sizes ranging from 2 to 6 words.

The Arabic chart parser was designed to follow three main steps to parse the Arabic sentences which are: word classification, grammar identification and finally parsing the sentences. In Arabic language, words are classified in to three: nouns, verbs, and particles. The particles are again classified as prepositions, adverbs, conjunctions, interrogative particles, exceptions and interjections. Therefore, the system enables the user to categorize words of the sentence automatically or manually based on the extent to which the automatic categorization succeeds. That is, the user might initially use the automatic categorization to break the sentence into its main components. However, if the automatic categorization failed to find the correct word categorization, then the user can perform the manual classification by inserting the component type (i.e. PRO, N, ART, etc.). After that the system identifies those grammars which are used to analyze and determine the structure of the given grammar. In order to parse the sentence, since the system used the top-down chart approach, matches will be made by taking a sequence of symbols and match it to the right hand side of the rules. The state would simply consist of a symbol list, starting with the words in the sentence. Successor states could be generated by rewriting a word by its possible lexical categories and by replacing a sequence of symbols that matches the right hand side of the grammar rule by its left-hand side symbol. Besides to this the system stores the partial results of the matching that have been done so far as a result works need not to be reduplicated [18].

The parser in [36] has been developed with the aim of analyzing and extracting the attributes of Arabic words. In the implementation of the parser, top-down parsing technique with recursive transition network grammar has been used. As it is stated in Chapter two transition network grammars represent grammars based on the notion of transition network consisting of nodes and labeled arcs. When the arcs of the grammar call other levels of the network to recognize subordinate constituents that can in turn call other levels, it is said to be Recursive Transition Network (RTN). Morphological analyzer is also used to consider words that are not found in lexicon directly.

The researchers collected a sample of 90 sentences that are 6 words length from grade 6 Arabic text book. The sentences are made to have gender and number agreement to ensure the correction of syntax structure of the Arabic sentences. After the evaluation of the parser, it is found that some sentences are unparsed totally and some other sentences are parsed incorrectly. Sentences are not parsed because of the following reasons; first when the parse does not found the word in the lexicon, second because of the incorrect input sentence and third when the parser is unable to produce a rule for the input sentences because the syntactic form of the sentences is not included in the grammar.

The ARSYPAR (ARabic SYntacic PARser) [37] has been developed as a tool for parsing the Arabic language based on the supervised machine learning. The system used the Support Vector Machine (SVM) algorithm for the learning phase and the Penn Arabic Tree Bank (ATB), which was developed in the laboratory of Linguistic Data Consortium (LDC) at the University of Pennsylvania, as a learning corpus.

The system works in two phases: the learning phase and the analysis phase. The learning phase involves the use of a training corpus in order to extract a set of features and rules which are used to train the SVM. The extracted features are used to specify the morphological category (POS) of the word being processed and the POS of the words in the left vicinity of the word being analyzed with a maximum depth equal to four. On the other hand, extracted rules are used to train the system in grouping of the sequence of labels that may belong to the same syntactic grouping and thus define their border. Therefore, the system will be trained to be able to allocate each word in the sentence to its most probable syntactic group and classify them automatically. In the analysis phase, the results of the learning phase are used to parse a sentence. But the user must provide segmented and morphologically annotated text as input to the system. The evaluation of the system was made by the cross validation method using the Weka tool by dividing the corpus into two parts, one which contains 80% of ATB for learning and the other which contains 20% for testing. When the system is evaluated on 100 sentences, the result had 89.01 precision, 80.24 recall and 84.37 F-score.

Generally, unlike the previous Arabic parsers [10, 36], ARSYPAR used Part-of-speech features, which is important to improve the efficiency of the parser. However, ARSYPAR didn't incorporate lexical data (external dictionary) to identify multi-word expressions.

3.4 Parsers for English Language

The researchers in [38] developed a new grammar formalism called a link grammar that has equivalent expressive power to that of CFG to parse English sentences. A link grammar consists of a set of words (the terminal symbol of the grammar), each of which has a liking requirement that is contained in the dictionary. A sentence of a language is defined by the link grammar if there exist a sequence of words and there exist a way to draw arcs among the words so as to satisfy the following conditions. Planarity: the links do not cross when drawn above words; connectivity: the link suffices to connect all words of the sequence together; and satisfaction: the linking requirement of each word in the sequence.

The researchers have written a link grammar of seven hundred definitions that capture many phenomena of English grammar. The grammar handles noun-verb agreement, questions, imperatives, complex and irregular verbs, different types of nouns, past or present participles in noun-phrases, commas adjective types, prepositions, adverbs, relative clauses, possessives and other parts of the language. Moreover, the researchers developed an algorithm based on dynamic programming which tries to build a linkage in a top down fashion.

The system was tasted by applying it to articles taken from newspapers and the result indicated that the performance of the system is good. However, there are a number of English phenomena that are not handled by the system. For example, the system accepts sentences and clauses that end with preposition. There are also problems on the placement of the adverbs and prepositional phrases modifying verbs.

Charniak developed statistical based parser for English language [39]. The statistical parser produces its grammar and probabilities from a hand-parsed corpus (i.e., a tree bank). This parsing system is based on a language model which in turn is based upon assigning probabilities to possible parses of a sentence. The model is used in the parsing system by finding the parse for the sentence with the highest probability. Therefore, the parser operates by assigning probabilities to the sentence under all its possible parses and then choosing the parse for which the probability is highest. In line with this, rules of the context free grammar specifies how each phrase constituent will be expanded.

The researcher evaluated the performance of the system by training the parser on about one million words of the Peen Wall Street Journal Tree bank and testing on 50,000 words and claimed that its performance is superior to previous parsers in the area. However, creating the corpus or tree-bank is a difficult task that requires great strength or effort.

3.5 Parser for Indian Language

The Hindi parser [40] was developed to parse sentences of Hindi language, which is one of the official languages of India and the fourth most widely spoken language in the world. The parser was developed using the Cocke Kasami Younger (CKY) algorithm which can recognize languages defined by context free grammar in Chomsky normal form. In this study, the researchers developed a set of grammars that has 14 non-terminals and 13 terminals to represent sentences of the language.

The parser is developed to have three components: interface, database for Hindi words and the parse. The interface allows the user to enter sentences and tokenize the input sentence and assign tag to each token. The database on the other hand stores the tag of Hindi language words. The parser will then take string of tag as input and states whether or not input string is correct. Concerning about the performance of the parser, unlike other researches the paper didn't mention anything. Moreover, the number of sentences used, the types of sentences, and the amount of words the database contains for evaluation is not indicated. Hence it is difficult to say anything about how the parser performs compared with other parsers.

3.6 Parser for Myanmar Language

Context Free Grammar Based Top-down Parsing of Myanmar sentences [7] has been developed to parse simple and complex Myanmar sentences of 5 to 50 words length. The simple sentences were declarative, negative and interrogative types. Three types of complex sentences which joined particles, adjectives and adverbs respectively were also contained in the training and test sets. The study used CFG formalism to represent production rules whereas top-down parsing is used to build the parse tree. The researchers collected nearly 3000 training sentences and 530 testing sentences from Myanmar religious books, short stories and newspapers. The corpus was pre-processed before it is passed to the actual parsing process. In the sentence level, the

researchers annotated the corpus for part of speech, chunk, and function tags relationship between the words in the sentences. The function tags are combined to get the phrases. However, top-down is not efficient compared to top-down chart parser.

3.7 Parser for Chinese Language

In the research in [41] the Maximum-Entropy-Inspired parser was applied on the peen Chinese Tree Bank (CTB). The major idea in the Maximum-Entropy is the generative model $P(\pi, s)$, where π is the parse tree for a sentence s. The model assigns a probability to a parse by a top-down process. A parse tree will be generated by starting from the tree root S (sentence), and use the context-free grammar for branching. Each expansion is assigned a probability, and the probability of a tree would be the product of the probabilities of all expansions that generate the given sentence. Then the parse that has maximum probability $P(\pi, s)$ for a given sentence s will be selected.

The evaluation of the system was conducted by first transforming the tree bank. Since words in Chinese are not delimited by white-spaces, the original tree bank was converted into a tree in which the terminal consists of a single character instead of words. Moreover, a MaxEnt reranker which assigns a new probability to each one of the parse of a sentence was used to improve the performance of the parser. The system was tested on the two versions of the Chinese tree bank CTB1.0 and CTB4.0 with 3485 and 12334 sentences respectively. The paper concluded that the performance of the parser is better than previously obtained results.

3.8 Summary

Sentence parsing is important in linguistic and natural language processing to understand the syntax and semantics of natural language grammar. Sentence parsers are developed in different approaches for a number of languages around the world. In this chapter, we have tried to present some of the works which are related to our work. In the first section, we have seen parsers developed for Amharic language. The parsers were designed by employing rule based and hybrid approaches. Atelach Alemu [15] and Daneil Gochel [16] developed bottom-up based parsers for simple and complex sentences respectively. However, the parsers are limited in terms of the sentences types, word length of the sentences and the size of the corpus used for evaluation.

Atelach's parser accepts only 4 word length simple declarative sentences and it was tested on a corpus that contains 100 sentences. The same is true for [35] which were designed to parse only simple Oromo sentences. In addition, the number of sentences is small and they are not annotated or POS tagged. Daneil's [16] parser parses only sentences that have simple noun phrase and complex verb phrase. The other parser was developed by Abeba Ibrahim [34] based on a hybrid approach. Abeba Ibrahim developed a text chunker (shallow or partial parser) and then transforms this chunker to a parser. The study considered simple declarative sentences and all types of complex sentences. However, still all types of sentences are not included in the study. Therefore, in our research, we are aiming to use more sample corpus from different types of sentences and various sources like newspapers, grammar books, and others. Moreover, the parser will be used to parse all types of sentences.

The researches [37, 39, 41] used tree-banks to develop a parser for Arabic, English, and Chinese texts respectively. The tree bank allows the parser to produce its grammar and probabilities to possible parses of a sentence. In this approach the machine learning algorithm like SVM will learn from the training corpus and enable the parser to choose the parse that has maximum probability. Parsers made from tree bank are often the best parsers because of the reason that they can simply exercise in machine learning and best parse of a sentence can be obtained. However, creating the requisite training corpus, or tree bank, is a difficult task. Hence, it is difficult to apply this approach for Amharic texts because of the absence of such tree bank.

The authors of [7, 10, 36] used a rule based approach (i.e., CFG with top-down chart parsing, transition network with top-down and CFG based top-down technique respectively). However, top-down parsing has problems of reduplicating; backtracking to where it made the wrong decision at each time it chooses the wrong path, and getting stuck when there are grammar rules which are left recursive. Chart paring on the other hand avoids problems of top-down and bottom-up approaches. So that top-down charting will be used in our research.

Chapter Four – Design of the Parser

In this Chapter we will discuss the architectural design of the parser, the main components and the interaction between them. The design of the proposed architecture has six major components; sentence tokenizer, morphological analyzer, pre-processor, lexicon generator, Context Free Grammar rules and the parsing algorithm. We begin by discussing the main components of the parser along with the resources needed for each module and in what way they will interact with each other. Finally the architectural design of the parser will be shown.

4.1 Components of the Parser

Every sentence parser which is implemented using rule-based approach will have the basic components: grammar rule, lexicon and the parsing algorithm. The grammar component will be responsible for storing grammar rules written in one of the grammatical formalisms. Grammar rules are used to enable the parser to learn a set of rules automatically based on a given list of strings and then parse sentences following those rules. The lexicon component will be used as a dictionary for the parser by storing lexical rules which are separated from grammar rules. Lexical rules specify the possible categories of each word so that the efficiency of the parser will be improved.

Even though these are the basic components that the rule based parser comprises of, the structure of the grammar rules, the lexicon and the parsing algorithm differs from system to system and from one language to the other. Besides the basic components, our system has components that are used to pre-process the corpus, tokenize the input sentence and prepare the lexicon automatically. The pre-processor component is responsible to remove unwanted and unnecessary texts from the corpus and give only POS tagged sentences. The tokenizer component is used to break the input sentence into individual words so that the morphological analyzer can get words of the given sentences one-by-one. The lexicon generator is developed to avoid the manual preparation of the lexicon. As a result, the lexicon can be generated automatically from the tagged sentences. In addition to this, one can add other components like part-of-speech tagger and morphological analyzer to improve the performance of the parser. Morphological analysis is important for morphologically complex languages like Amharic because it is difficult to store all possible words in lexicon and many words have close to zero probability of occurrence in any

given corpus [44]. Since we will use the corpus which is manually annotated or POS tagged whose names of POS tag are shown in Appendix B, the POS tagger is not required for our system. However, the morphological analyzer developed by Gasser [44] is integrated with our system to develop a parser with a better performance.

Therefore, in this study we have six components: the parsing algorithm, the grammar rules, lexicon generator, morphological analyzer, corpus pre-processor and sentence toknizer. Figure 4.1 shows the general architecture of the parser. The architecture has two parts that contain components from which the parser learns grammar and lexical rules to parse a given sentence in one section and components which perform the actual parsing in the other section. Corpus pre-processor, lexicon generator, morphological analyzer, the lexicon and the Context Free Grammar are grouped in one part. The other section contains sentence tokenizer, morphological analyzer and the parser which will be invoked whenever there is parsing. The Amharic grammar rule component is required to store Amharic Context Free Grammar rules. The grammar rules are identified in a way that they can represent the structure of all types of Amharic sentences in terms of what phrases and word categories are subparts of others. Unlike the grammar rule component, the lexicon module stores a list of lexical rules, which specify the possible categories of each word. The morphological analyzer component analyzes words, which are not directly found in the lexicon, into their constituent morphemes (meaningful parts) and generates words, given a root or stem. The parser component will accept input sentence, consider the grammar rules, retrieves the POS tag of each word in the sentence from the lexicon or send words for the morphological analyzer (when the word is not found directly in the lexicon) and finally returns parsed sentences as an output. The details of each component will be presented in the following sections.

4.2 Corpus pre-processor

Sentences for this research are collected from the Walta Information Center (WIC) corpus, previous research works and Amharic grammar books. Sentences in the WIC corpus are tagged and presented using the Extensible Markup Language (XML format) as shown in Appendix C. However, we cannot use it directly due to the existence of unnecessary texts and part of speech tags. POS tags like punctuation marks and texts that are not either POS of the word or normal

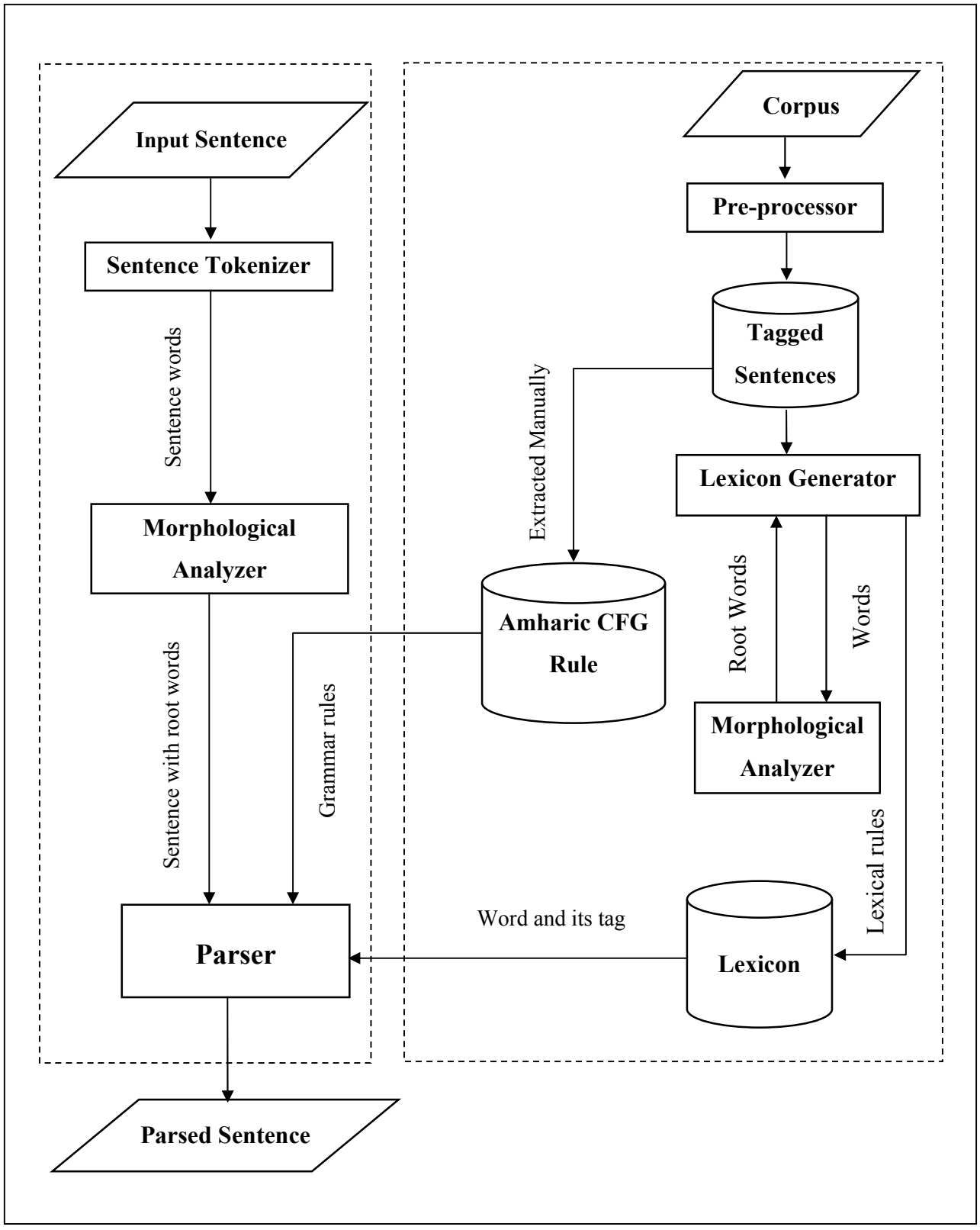

Figure 4.1: The Architecture of the Parser

text like <title>, <body>, <document>, etc. needs to be removed. In addition to this, the POS tag names are enclosed in "<" and ">" which is also unnecessary. Therefore, removing all these things is the first task. However, doing this task manually is a tedious and time consuming. For this reason, we have developed a simple pre-processer that removes all unnecessary words and characters from the text. The pre-processor scans the text and whenever it encounters the above mentioned texts it jumps and output the text that is appropriate for use and free from any unwanted things. Finally, the processed text and sentences which are collected from other sources are stored in a storage named as POS tagged sentences. These sentences are then used for preparing the lexicon and to extract Context Free grammar rules.

4.3 Lexicon Generator

Again once we have the corpus that is pre-processed, the next task will be preparing the lexicon. As we know the lexicon is a list of words with their POS tag name. Preparing the lexicon manually by typing the word and its word category, especially when there is a large size corpus, is error prone and time taking. Therefore, it is better to develop an automatic lexicon generator that outputs the result correctly and within a short time. As a result we have developed a lexicon generator that output lexical rules from POS tagged sentences automatically. The lexicon generator reads the POS tagged sentence, on which the POS tag name is written immediately after the word, line by line and presents the result as POS tag name followed by an arrow, " →", and then the root words, which are returned from the morphological analyzer, enclosed in double quotation. The result is produced in this format because this is the proper format of the lexical rules in the lexicon.

4.4 The Morphological Analyzer

In our system, the morphological analyzer is used in two places; in the construction of the lexicon when the lexicon generator produces lexical rules it is used to find the root words that will appear after an arrow in the rules and in the parsing process, before the input sentence is given to the parser. At the time of parsing, the parser requires the POS tag of the input sentence words from the lexicon. For this reason, the lexicon must have lexical rules within which words of the input sentence are hold. However, the lexicon doesn't need to contain all words of the language; instead it stores only the root (the minimal unit of the morphology) of words. Hence,

the morphological analyzer is required during the preparation of the lexicon to find out the root of the words. As a result a number of words can be represented by a single root word. At the same time we need the morphological analyzer before the input sentence is given to the parser. This is because of the fact that words of the input sentence should be in their root forms. As a result, the sentence has to pass through the morphological analyzer. In our study we integrated the morphological analyzer known as HornMorpho 2.5 developed by Gasser [44].

The Lexicon

This component is a repository that is used to store a collection of information about the words of a language and the word categories to which they belong. Lexicon is required to contain lexical rules that have the part of speech category, on the left hand side, and the word, on the right hand side. Therefore, the grammar rules need not contain any lexical rules of such form.

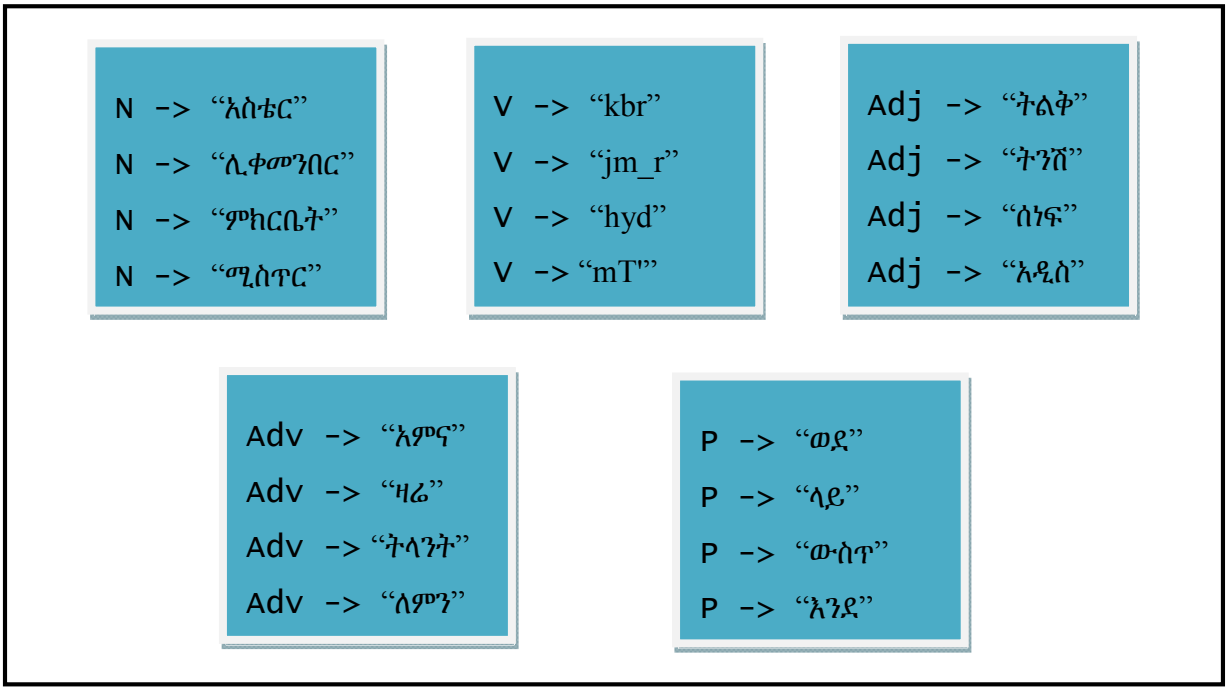

Figure 4.2: An Example of Lexical Rules in the Lexicon

The syntactic properties specify what role the word plays, such as feature information (whether the word is transitive, intransitive or bi-transitive), what form the verb takes such as present participle, past participle or the gender of the noun. However, these features are not included in our study. Figure 4.2 shows an example of the lexicon for different word classes

Amharic Context Free Grammars

A grammar in human language represents understandable specification of language syntax. It is not concerned with semantics. In other words, the grammar is a collection of words that describes well-informed sentences in a language. Furthermore, Context Free Grammar in natural languages represents a formal system which describes a language by specifying how any legal text can be derived from a distinguished symbol called the syntactic symbol [45]. CFG rules for this thesis work are extracted from sentences collected from Amharic grammar books, previous research papers and the Amharic news of WIC which are already tagged manually by researchers and linguistic experts [3]. However, since sentences collected from grammar books and previous research works are manually tagged with slightly different POS tag name, they will be made consistent to each other. Context Free Grammar rules are extracted to represent the grammatical structure of many valid sentences as much as possible. As we have discussed in Chapter two, there are four types of simple sentences and complex sentences. Hence, the grammar rule should represent all of these sentences. But the structure of negative sentences can be represented by the grammar rule of declarative sentences. Figure 4.3 shows what types of sentences the grammar rules represent.

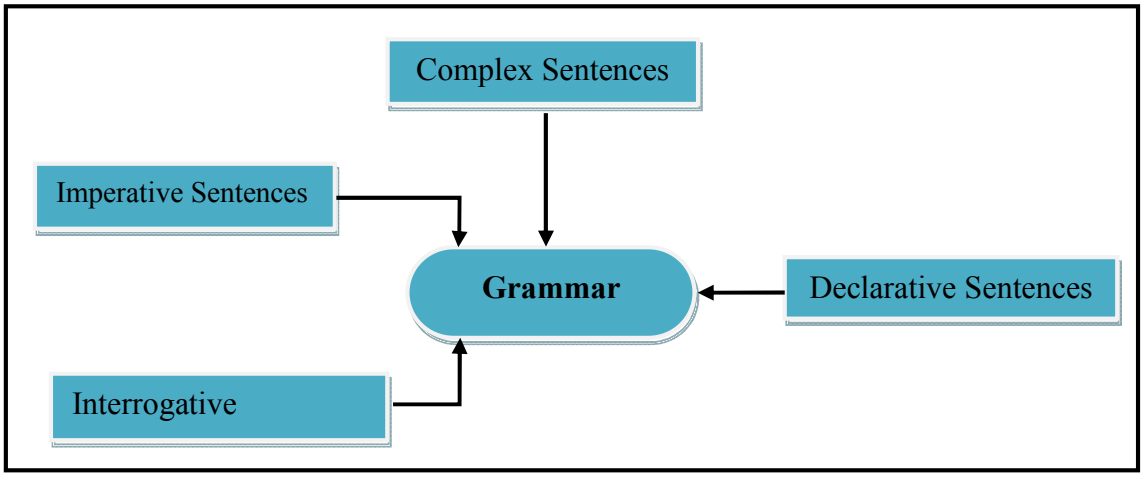

Figure 4.3: Types of Sentences Represented by the Grammar Rules

Table 4.1 shows some of the grammar rules contained in the Amharic CFG component for each sentence type.

E.g. Table 4.1: An Example of Grammar Rules in the Amharic CFG

Context Free Grammar rules in the Grammar rules		
Declarative Sentences	**Negative Sentences**	**Interrogative Sentences**
S➔NP VP NP➔N P NP ➔ ADJ N NP➔ N N P VP➔ ADV V VP ➔ V V	S➔NP VP NP➔N P N NP➔ N ADJ N NP ➔ ADV N VP➔ V VP ➔N V	S➔ NP VP NP➔ADV N NP➔N V N VP➔ ADV ADJ V VP➔V VP ➔ P V
Imperative Sentences	**Complex Sentences**	
S➔NP VP NP➔N NP➔ N P VP➔V VP➔N V	S➔NP VP NP➔s N np s➔PP VREL np ➔PP N PP➔PREP N VP ➔V	

4.5 Sentence Tokenizer

Since the morphological analyzer analyzes one word at a time, words of the input sentence should be separated and given to the morphological analyzer one by one. For this reason, we need to have a tokenizer that is responsible to detach words of the sentence. The tokenizer spans through the sentence from the beginning to the end and whenever it gets a space gap it considers the text before the space as one word. Each word will be identified in this way and given to the morphological analyzer whenever it is obtained. However, compound words are not treated as one word because of the white space and this is the limitation of the work. When the morphemes of the words are returned from the analyzer they will be rejoined to form a sentence that has root words of the input sentence.

4.6 The Parser

This module is used to apply the charting algorithm in combination with other components. Let us see how it works. Unlike the traditional parsing, chart parsing is based on the basic idea "don't throw away information. Keep a record (a chart) of all the structure you have found so far". Chart parsing have two forms: passive chart parsing and active chart parsing. In case of passive chart parsing, the chart is simply a record of all constituents that have been recognized. However, in case of active chart parsing, in addition to keeping track of a record of complete constituents that we have found, we record hypothesis (i.e., we record what we are actually looking for and how much of it we have found so far). Such information is recorded in active edges or active arcs. An example is shown in Figure 4.4.

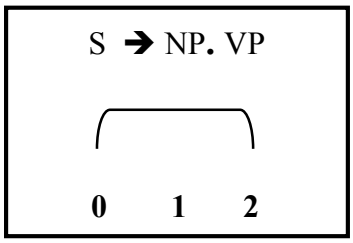

Figure 4.4: An Example of Active Arc

The active arc is between the nodes 0 and 2. This arc, S ➔ NP**.**VP, is interpreted as S is going to be built from an NP followed by a VP. So far an arc NP is found and we are still looking for the VP. Here, we have a dot '**.**' between NP and VP to mark the boundary between what we have found so far and what we are still looking for. Therefore, since constituents before the dot are passive edges, active edges can be combined with these passive edges to create new edges. The new edges created may be either passive (that is already recorded) or active. The combination of the passive edges and an active edge can be performed as follows: suppose the active edge goes from node n1 to node n2 and has category 'c' immediately to the right of the dot as shown in Figure 4.5.

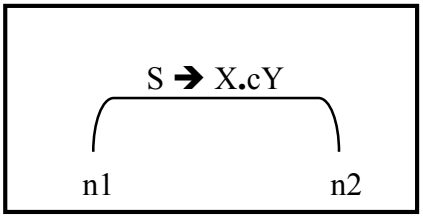

Figure 4.5: An Example of an Active Arc between two Nodes

Suppose that there is the passive edge going from node n2 to node n3 (it starts from where the active edge ends) and has category c on the left side as shown in Figure 4.6.

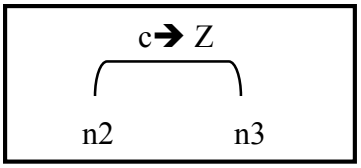

Figure 4.6: An Example of Passive Edge that Starts from the End of Active Edge

Therefore, these two edges can be combined to build a new edge that starts in node n1 and ends in n3. The dot in the active edge is now moved one category forward (i.e., S➔Xc**.**Y). For example as shown in Figure 4.7, while we are working with active chart parser we make use of an agenda. An agenda is a data structure that keeps track of the things that we still have to do. When new edges are created, we have to remember that we have to look at them to see whether they can be combined with other edges in any way. To not forget this, we store them in the agenda. We will then take one edge at a time from the agenda, add it to the chart and use it to build new edges.

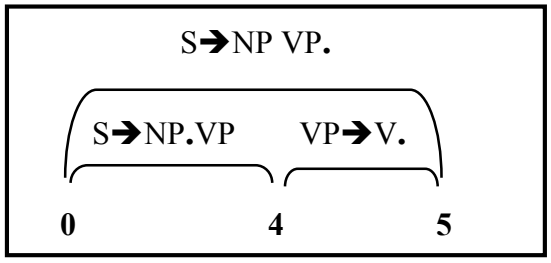

Figure 4.7: An Example of a New Edge Combined from Two Edges

Top-down searching will be used to use rules to make active edges. In addition to this, the agenda will have at least one active edge to start the parsing process. The active edge starts at the position zero from the sentence S. Hence, the active edge will be taken from the input sentence. The chart will remain empty until an active edge is added to it.

4.7 Summary

In this Chapter the architecture of the parser and its main components are discussed. The architecture of the system contains six main components. We have the pre-processor to remove unwanted texts from the WIC corpus and produce POS tagged sentences. The lexicon generator used sentences processed by the pre-processor and sentences gathered from Amharic grammar

books and previous research works to produce the lexicon that is passed through the morphological analyzer to store only root words in the lexicon. The parser begins by accepting the input sentence from the user, which is tokenized, morphologically analyzed and again rejoined sentence, and identifies each word and look at the lexicon to determine the word category. The morphological analyzer module in the lexicon side will find the morpheme of the words so that inflected and derived words of the root word will not be found in the lexicon and it also prevents the storage of possibly large number of words. Hence, the lexicon is not required to contain all words of the language. The grammar rule module allows the parser to determine the structure of the parsed sentence by considering all possible grammar rules to create the output. Finally the result, which is the parsed sentence, will be presented to the user.

Chapter Five – Implementation of the Parser

In this Chapter we will present the detail implementation of the parser. First, we discuss the used development environment. Second, the corpus collection, the techniques used and the modifications made to make it appropriate for use will be explained. Third we will explain the detail of the algorithm implemented for pre-processing sentences particularly collected from WIC. The fourth section explains the integration of the HornMorpho with our system. The last sections, which are the main parts of this work, focus on the extraction of the context free grammar and the preparation of the lexicon. Finally the main components in the implemented Top-down Chart parser will be discussed.

5.1 Development Environment

We have prepared a prototype which takes an input sentence from the user, parses the sentence according to the CFG and lexicon based on the above mentioned parsing method and finally delivers a parsed sentence for the user. The prototype also includes the pre-processor and the automatic lexicon generator as its components. The pre-processor takes the WIC corpus and makes it appropriate to be used by the lexicon generator. The lexicon generator, on the other hand, takes POS tagged sentences and produces lexical rules automatically. The prototype is developed and tested on a system with Intel Core I3 CPU of 2.4 GHZ speed, a 2 GB RAM, and Windows 7 Operating system. The system is developed using Python 3.2 programming language. The language is chosen because of two reasons: first the language is easy to use and understand with its capability to enable to use Unicode representations easily and it takes much less time to develop a program in Python than other programming languages [46], second since we used a morphological analyzer which is developed in Python, we need to use the language for ease of integration of the analyzer with our system.

5.2 Collecting the Corpus

In this study, we have collected the sample sentences from three main sources. Some of the sentences are taken from previous research works [15, 16, 34] in the area, some others are taken from the Amharic grammar book [42] and most of the sentences, more than half of the total sentences, are taken from the Walta Information Center corpus. The total size of the corpus is

480 sentences. Sentences taken from the WIC and research works are already tagged. But sentences of the grammar book are tagged manually by the researcher based on tagged sentences. Moreover, sentences collected other than WIC and the research work in [34, 47] are not written in Amharic alphabet 'ፊደል' so that we transcribed these sentences according to the Amharic alphabet Unicode standard. Some of the sentences collected from sources other than WIC are attached in Appendix F. The Unicode representation of Ethiopic characters is shown in Appendix A. The selections of the sentences were done in random fashion with proportion number from each type.

The name of the POS tag in the collected corpus is different from one source to the other. For example, Atelach and Daniel used the same tag name which is different from WIC's corpus. In [34] the researcher tried to use an own tag name by converting the previous by a new one. In our study, we have POS tag name that is omitted, replaced by a new and newly added. Part of speech tag names NUMPREP (Numeric Preposition), NPREP (Noun Preposition), VPREP (Verb Preposition), ADVPREP (adverb Preposition), and ADJPREP (Adjective Preposition), which have been used in [34], are omitted becaus we used morphological analyzer, such types of words will not have place in the lexicon. The tag names PRON (pronoun) and AUX (auxiliary) are replaced by N (noun) and V (verb), respectively. In addition to this, the tag names of the Amharic phrases are specified. In Amharic language, the number of phrases is equal to the number of word classes in the language [42]. Therefore, we have five phrases as it was discussed in Chapter two. The tag name of each phrase type in this work is shown in Table 5.1.

Table 5.1: The Tag Name of Amharic Phrases

Name of the Phrase	***Tag Name***
Noun Phrase	NP
Verb Phrase	VP
Adjectival Phrase	ADJP
Adverb Phrase	ADVP
Prepositional Phrase	PP

5.3 Text pre-processing

Sentences which are not pre-processed before they are being used by the system components need efforts for the removal of unnecessary texts. The main reason we need to process the text is that it contains some texts which are not known and needed by the morphological analyzer and the lexicon. Furthermore, the performance and the effectiveness of the system can be improved because of the reduced size corpus. Hence, we added the corpus pre-processor component in our system. The algorithm that we developed to pre-process the text is shown in Algorithm 5.1.

```
Input: Corpus
Read the text from the local disc
For each word in the text
        Check whether the word is <title>, <body>, <PUNC>, or <dateline place>
        If the word is one of the above
              Leave or omit the word
        Else
            For each character in the word
                 Check the presence of '<', '>', '፣', '።', '፤'
                 If one of the above is detected
                       Leave or omit the character
                 Write the characters of the word in the file(write the word)
            End For
        End If
End For
Output: pre-processed text
```

Algorithm 5.1: WIC Text Pre-processor Algorithm

5.4 Integration of the Morphological Analyzer

As we have discussed in Chapter four, we need the morphological analyzer for lexicon generation and before the sentence go to the parsing process. The main purpose of the analyzer is

to find the root of a given word (if it has). During the preparation of the lexicon, we have to consider that it is practically difficult to store all words of a language especially for morphologically rich languages like Amharic, where a single Amharic word can have a number of inflections and derivations. Therefore, the analyzer has a crucial role in reducing the number of words in the lexicon significantly. At the time of parsing, the analyzer will be needed before the input sentence is given to the parser because of the lexicon we have. The prepared lexicon consists of words that are morphologically analyzed. For this reason, we must give a sentence for the parser whose words are morphologically analyzed unless it is impossible to get the word classes of the input sentence words in the lexicon. Some of the lexical rules used in the study are shown in Appendix E. Algorithm 5.2 shows an algorithm on how the morphological analyzer was integrated in our system.

```
Input : word
Take a word from the user input sentence or from the corpus sentences
Call HornMorpho
Find out the root word
    Check whether the given word has root or not
    If  the input word has root
        Select the root word from the output of the morphological analyzer
        Return the root word
    Else
       Return the original word
    End If
Output: either root word or the original input word
```

Algorithm 5.2: An Algorithm for the Integration of the Morphological Analyzer

The above algorithm is used for both cases; during the lexicon generation and at the time of parsing. However, the analyzer may result in wrong output like words that are the name of a person or place may be analyzed even if this is not correct. Therefore, such errors, especially encountered when the lexicon is prepared, are fixed manually.

5.5 Context Free Grammar Extraction

Lexicon and Context Free Grammar rules are important and necessary components in our system. CFG rules are used to train the parser with a set of grammar rules and enable it to parse sentences based on those rules. In our study, CFGs are extracted manually from the collected corpus with the help of linguistic expert. Sample CFG rules are attached in Appendix D. The corpus contains POS tagged sentences in which the word category of each word in the sentence is identified. We have used the Amharic grammar we discussed in Chapter two in the preparation of the CFG. The Amharic grammar enabled us to know how any legal text can be formed from different types of words with their correct order. It also describes which structure of a sentence can be built from which sequence of words. In other words the grammar specifies how we are able to determine whether a given sentence is valid or not according to the rule. Therefore, we identified five types of phrases, which are noun phrase, verb phrase, prepositional phrase, adjectival phrase, and adverbial phrase and the possible combination of words from which the above mentioned phrases can be formed. When we construct the CFG, we looked at the sentences and we identified which one is the noun phrase and which one is the verb phrase. A number of sentences have similar phrase structure and some others share common sub-phrases so that a single CFG rule can represent many sentences. For this reason we prepared around 175 rules from more than 480 sentences.

Once we have the structure (phrases, sub-phrases and words in the phrases) of the sentence we transformed it into the proper format of the CFG, which is a non terminal followed by an arrow and then terminal or other non terminals which can replace the non terminal before the arrow. The CFG begins from the non-terminal S, which represents sentences, and then phrases which can replace S (most of the time noun phrase and verb phrase). Each phrase which is on the right side of S will be expanded or expressed by other non-terminals in next rules. We constructed a separate CFG for each sentence type for the purpose of simplicity. Meaning we have a CFG for complex sentences and we have another CFG for each types of simple sentences. Hence, the parser can be easily trained. Extracting the grammar rules is a onetime process that will be done after the text pre-processing. However, the system will use these rules every time the parser is initiated. Moreover, the correctness of the rules with respect to the grammar rules of the

language should be maintained unless the parser may result in a parse structure which is not supported in the language.

5.6 Lexicon Generation

The main objective of this module is to produce lexical rules, which are separated from the CFG rules, automatically. We used the same corpus that is used in CFG extraction for the generation of the lexical rules. The construction of the lexicon is a one-time process that is done at the very beginning of the parser implementation. However, the result will be needed whenever there is parsing. During the lexicon construction, the lexicon generator reads the corpus from local disk and goes through each sentence. While scanning each sentence, the generator identifies the POS tag name and the word which will be associated with it in lexical rules. Then after, the identified words have been given to the morphological analyzer. The result of the analyzer will be checked. Because of that all words which are given to the analyzer may not have root word or the analyzer may not support them at all, i.e., the analyzer is not 100 percent perfect. If the analyzer returns a non empty result, the generator selects the root word and connects it with the POS tag name identified earlier. The output of the lexicon generator is expected to be formal lexical rules that will be stored in the lexicon. For this reason, the output is displayed in the following format; in the left hand side the POS tag name followed by an arrow and the root word, which is enclosed in double quotations in the right hand side.

Moreover, we have made numeric values to have one common word representation, i.e., "ቁጥር". The common representation is required because of the fact that numeric values are infinite in number and it is impossible to store all numbers that the user input sentences might contain. As a result, we put one lexical rule Num➔"ቁጥር" to denote the POS tag of any given numeric values in the user input. However, whenever there is numeric value in the input sentence, the number should be replaced by the word "ቁጥር" before it is given to the parser. But after parsing, the actual number will appear within the parsed sentence. The algorithm on how the numeric value in the input sentence can be replaced by the word "ቁጥር" will be discussed in the parser section. The algorithm on how the lexical rules can be generated is shown in Algorithm 5.3.

In the preparation of the lexicon, the corpus contains more than 2,400 words. But after it is passed through the lexicon generator, we removed replicated words (or similar lexical rules are

made to be represented only by one of them). This reduces the number of lexical rules significantly. Moreover, the representation of numeric values by a common word prevents the existence of exhaustive numbers in the lexicon. Therefore, using the morphological analyzer in the preparation of the lexicon enabled us to have a lexicon with few words, which can represent a large number of words. At the same time, the denotation of all numeric values by one single lexical rule has an important role in the reduction of the lexicon size. As a result the performance of the parser has been improved.

```
Input: corpus
Read the corpus from local disk
Scan each sentence of the corpus
Identify the POS tag name and the word that will be associated with it
Give the words to the morphological analyzer
For each word in each sentence
    Call the Morphological Analyzer (HornMorpho)
    Check the result of the analyzer
    If the result is non empty
       Select the root word from the result
       Return the result
    Else
        Return the actual word
    Write the results with their proper format in the file
End For
Output: lexical rules
```

Algorithm 5.3: Lexicon Generator Algorithm

5.7 Sentence Tokenizing

Breaking the input sentence into individual words is required during the parsing process. The main importance of the breaking is to allow the morphological analyzer to get individual words of the sentence one by one, since the analyzer takes one word at a time. Therefore, the tokenizer

identifies a separated word whenever it encountered a white space while it scans the sentence from the beginning to end. The algorithm of the tokenizer *that* we have used is shown in Algorithm 5.4.

```
Input: the user sentence
Scan the sentence and check the presence of a whitespace
For each character in the sentence
    Check whether the character is white space or not
    If the character is white space
        Print the characters that are before the white space (the word)
    Else
        Append the character to a string variable (the variable that will be printed when
        the if condition is satisfied)
    End If
End For
Output: separated words of the sentence
```

Algorithm 5.4: Sentence Tokenizer Algorithm

5.8 Sentence Parsing

Parsing the user sentence is the main objective of this study. For the success of this task, the parser interacts with other components which are discussed in the above sections. The parser uses two basic references at the time of parsing; Context Free Grammar rules and lexical rules. Initially, the parser accepts the input sentence resulted from the tokenizer and the morphological analyzer. Before the sentence undergoes through the parsing process, the parser checks the existence of numeric values in the sentence. Each character of the sentence will be checked whether it is a digit or not. If a digit character is encountered, the word within which the digit character is obtained will be replaced by the word "ቁጥር" (number). For example if the input sentence is "ለሆሊውድ ፊልም ስራ **255** ኢትዮጵያዊያን ተዋንያን ወደ ናሚቢያ ሊጓዙ ነው" (255 Ethiopian actors will go to Namibia for playing movie), the parser will start the parsing process after it changes the sentence to "ለሆሊውድ ፊልም ስራ **ቁጥር** ኢትዮጵያዊ ተዋንያን ወደ ናሚቢያ ሊጓዙ ነው". This is

needed due to the lexical rule, Num ➔ "ቁጥር", in the lexicon. Unless the replacement takes place, the part of speech tag name of the number will not be returned from the lexicon. In other words, the parsing process will not succeed if the POS tag name of the word in the sentence is not in the lexical rules. However, the replacement will be done internally, the user does not know anything about this, and in the final output "ቁጥር" will be replaced back by the actual number, in the above case 255.

Now the input sentence is ready for parsing. The parser scans through the sentence and asks for the POS tag of each word from the lexicon. Then after, the chart will be initialized by an active edge, which is a grammar rule that has S symbol at the left hand side. The agenda will also be initialized by a grammar rule that has a non-terminal at its left hand side. The left hand side terminal is similar with that of the non-terminal which is immediately after the arrow in a grammar rule in the chart. The grammar rule in the agenda will move to the chart to replace the first non-terminal, in the right hand side of S, if it can replace unless another grammar rule will be added to the agenda till the grammar rule that can replace the non-terminal is found. If there is a terminal that can replace the non-terminal, the parser will replace it and continues to the next non-terminal. However, if the non-terminal in the chart can't be replaced by the terminal, it looks for other non-terminals which can replace it and which can be replaced by terminals later on. This process will continue until all non-terminals in S are replaced by the terminals and the structure of the sentence S is recognized. This means when new edges are created, the parser has to look at them to see whether they can be combined with other edges to create another new edges in any way. To not forget this, it stores them in the agenda. The agenda contains edges (grammar rules that can replace non-terminals in the chart and creates new active edge or new grammar rule). The parser will then take one edge at a time from the agenda, add it to the chart and then use it to build new edges. Finally the parser results in which word has what POS tag and which phrases have what sub-phrases and words. The algorithm shown in Algorithm 5.5 is taken from [18] with some modification shown in bold face.

Input: User Sentence

Scan the input sentence

Check the existence of number in the sentence

If there is numeric value

Replace it by "ቁጥር"

Take the rearranged sentence

Make initial the chart and the agenda

Repeat the following until agenda becomes empty:

a. *Take the first arc (grammar rule) from the agenda*
b. *Add the arc to chart (if the edge is not already on the chart)*
c. *Combine this arc with arcs from the chart and add the obtained edges to the agenda*
d. *Make hypothesis about new constituents based on the arc and the rule of the grammar. Add these new arcs to the agenda.*

End repeat

See if the chart contains passive edges from the first node to the last node that has label S.

if the chart contains the passive edges that represent all nodes of the sentence then the parsing process succeeds, if not the input sentence has a syntax error with respect to the grammatical production rules in the CFG.

Return the parsed sentence; if the parsing process succeeded.

Output: parsed sentence

Algorithm 5.5: Top-down Chart Parsing Algorithm

5.9 Summary

The implementation of the parser used six components. The pre-processor component is used to remove unwanted text from the corpus and makes it ready for the next components. In the pre-processing, texts (like <PUNC>, <title>), characters (like < and >), and Amharic punctuation marks (like ፣, ፡, and ፤) are removed. The processed text and text which is already processed are

stored in one repository named as tagged sentences. The repository stores POS tagged sentences from which the lexicon is constructed and the CFG is extracted.

The lexicon generator component is used to produce lexical rules, which will be stored in the lexicon, automatically. The grammar works in collaboration with the morphological analyzer. The morphological analyzer is responsible to find the root word in the preparation of lexical rules. The root words are needed because of the difficulty to store all words of the language in the lexicon. At the same time, using the analyzer will improve the performance of the entire system. Finally, the lexicon generator produces lexical rules and stores them in the lexicon.

The sentence tokenizer component is responsible to break the user input sentence into words that will be given to the morphological analyzer one by one. The analyzer will also be used during parsing. Then after, the input sentences will be rejoined and given to the parser.

The parser component is responsible to parse the user sentence and give the final output. Initially, the parser makes replacements to numeric values in the sentence. After that, the parsing will be carried out by using the algorithm and the parsed sentence will be displayed to the user at the end.

Chapter Six – Experiment

In this Chapter we focus on the evaluation of the system. The evaluation is conducted towards the performance, effectiveness and correctness of the system. In the first section we focus on the evaluation of each modules of the system. Most of the evaluations are performed manually. One hundred sentences that are selected randomly from all types of sentences have been used for testing the parser. Then, comparison has been made between the output of the system and the result of manual parsing. The lexicon generator and the pre-processor are evaluated by checking whether they output correct and expected result or not. In the final section we present discussion that compares and contrasts our result with results of previous works.

6.1 Pre-processing Evaluation

We have written a Python code for removing unwanted texts and characters in the corpus before the corpus is provided for lexicon generation and context free grammar extraction. We evaluated the effectiveness of the module by checking whether all unwanted texts are eliminated or not. So that, we have seen that all texts and characters that were supposed to be removed, doesn't exist in the corpus. In addition to this, the performance of the pre-processer is excellent. It takes about half of a minute to process one hundred sentences, which contains five up to nine words in a sentence.

6.2 Lexicon Generating Evaluation

We have developed lexicon generator that constructs lexical rules automatically, which is later used by the parser. The generator used morphological analyzer in the construction of lexical rules. For this reason when we see the performance of the lexicon generator, it is somehow delayed due to the time that the morphological analyzer takes in loading morphological data for the analyzing process. The morphological analyzer takes one and half up to two minutes to load the data. The morphological analyzer and the lexicon generator were running in an ordinary laptop with 2.4 GHz speed and 2 GB RAM.

The correctness of the lexical rules was inspected manually by checking whether words are categorized in their proper word class and whether the morphemes of the analyzed words are correct. We have encountered some errors in the lexical rules. However, the errors come from

the incorrect part of speech tag in the input sentences and incorrect morpheme output from the analyzer. For example, some adverbs are classified as verbs, prepositions as conjunctions, and some verbs as nouns. Moreover, a word like "ተባይ" (insect) has indicated that its morpheme is "ብል" (b'l), which is not correct. As a result, we tried to fix those errors manually. Therefore, we have lexical rules which are free from errors.

```
N -> " ኮሜሳ "
N -> " sebl "
PREP -> " ከ "
N -> " b'l "
C -> " ና "
N -> " bex_ta "
PREP -> " ለ "
N -> " klkl "
N -> " ፕሮጀክት "
V -> " ndf "
PREP -> " የ "
N -> " kasa "
N -> " gWad_eN_a "
PREP -> " nde "
N -> " kasa "
ADJ -> " gobez "
N -> " m'r "
PREP -> " ለ "
V -> " hwn "
V -> " mWk_r "
PREP -> " የ "
N -> " አዲስአበባ "
N -> " ምክርቤት "
PREP -> " የ "
N -> " 'aqm "
N -> " gnbata "
N -> " slTe "
V -> " sT* "
V -> " jm_r "
```

Figure 6.1: Screenshot of Lexical Rules Produced by the Lexicon Generator

Figure 6.1 shows the result of the lexicon generator for three randomly selected sentences from the corpus. The sentences are "ኮሜሳ ሰብልን ከተባይና በሽታ ለመከላከል ፕሮጀክት ነደፈ" (COMESA proposed a project to protect crops from disease and insects), "የካሳ ጓደኛ እንደ ካሳ ጎበዝ ተማሪ ለመሆን ሞከረ" (Kassa's friend tried to be as clever as Kassa) and "የአዲስ አበባ ምክር ቤት የአቅም ግንባታ ስልጠና

መስጠት ጀመረ" (The administration of Addis Ababa has started giving capacity building training). Here in the lexical rules, the morpheme words are displayed in an English font since the morphological analyzer gives output in such a way. However, they will replaced by the words of the sentence entered by the user when the final result is displayed.

6.3 Parsing Evaluation

The parser uses the extracted grammar rules and the lexicon that is produced by the lexicon generator to parse the input sentences. The input sentence will undergo to the parsing process after it is tokenized word by word and morphologically analyzed. For this reason, the sentence tokenizer has an impact on the output of the parser. Meaning if words are not tokenized perfectly, the result of the parser will not be correct as expected. Therefore, we can evaluate the effectiveness of the sentence tokenizer by looking the output of the parser. In our study, we didn't encounter sentences which are tokenized wrongly. Hence, we can say that the sentence tokenizer is hundred percent perfect. However, some errors like miss categorization of words in the parser output are not caused by the tokenizer. On the other hand, the morphological analyzer affects the performance of the parser quietly in terms of speed. As a result, the output of the parser can't be displayed before two minutes.

In order to test the effectiveness of the parser, we used one hundred sentences as shown in Appendix G that are selected randomly from all types of sentences, on average 20 sentences ranging from four to nine word length from each sentence types. The correctness of the parser is evaluated or examined by inspecting its result manually. The output can be checked with respect to the right categorization of words in their proper word class, the right identification of sub phrases and main phrases, the right order of sub phrases in building main phrases, and whether all words and phrases are involved in the construction of the sentence S. Therefore, before we come to testing, we parse the above sentences manually on paper and then we compare the results of the parser for the same sentences with what we have on the paper. Any one of the result which doesn't satisfy one of the above criteria is considered as wrong output or if the result doesn't display at all. Hence, we have the following result (shown in Figures 6.2 - 6.8 and Tables 6.1 – 6.5) from each sentence type with their respective number of correctly parsed and

wrongly parsed sentences. In Figure 6.2 the noun phrase and verb phrase are “ኮከቦች” and “በንጋት በጣም ያበራሉ” respectively.

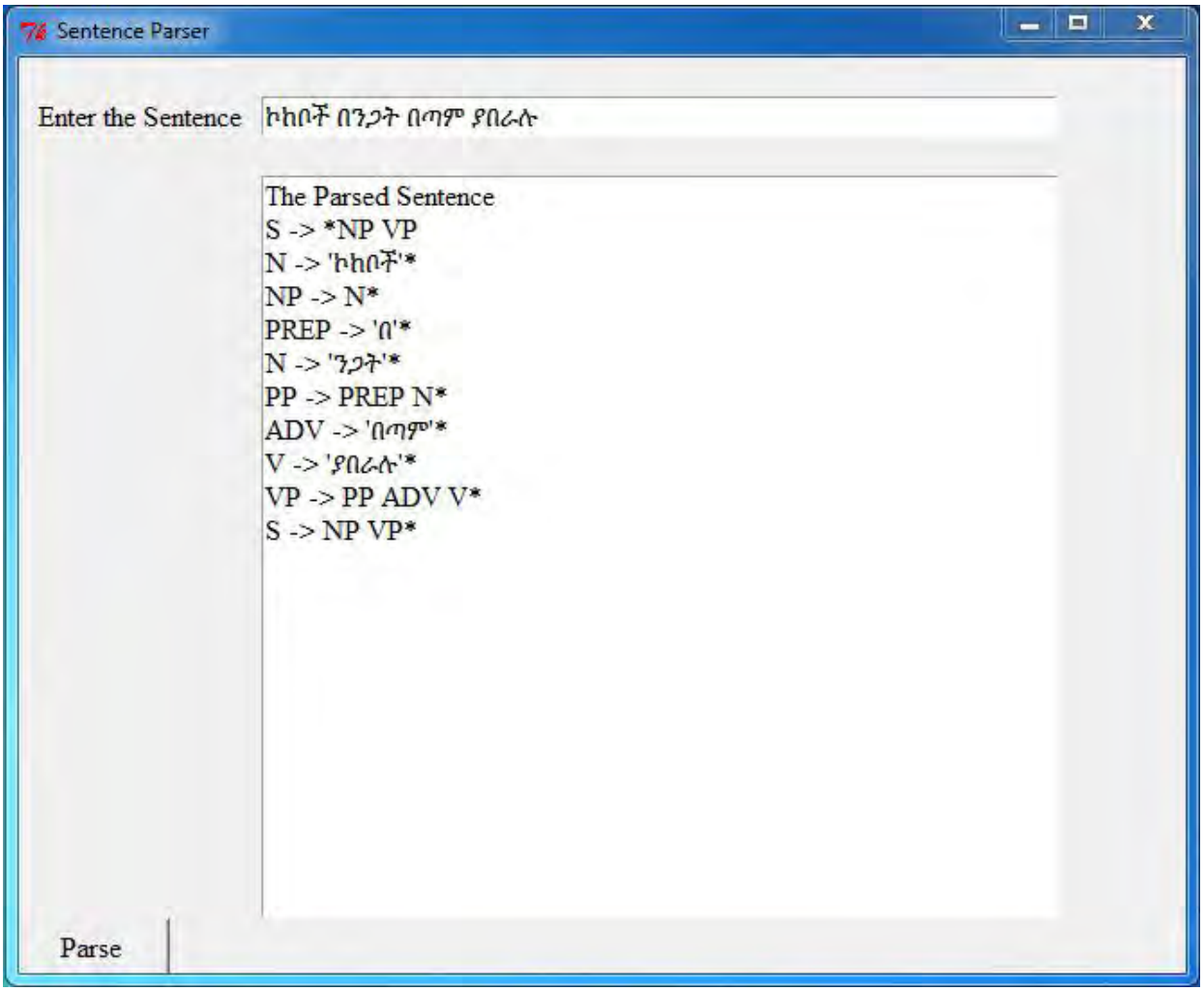

Figure 6.2: Screenshot of Parsed Declarative Sentence

The noun phrase is constructed from the noun “ኮከቦች “(i.e. NP➔ N) and the verb phrase from prepositional phrase, adverb, and verb(i.e. VP➔ PP ADV V). The prepositional phrase is made up of preposition and noun (i.e. PP➔ PREP N). However, the sentence “እህል ንተራው ውስጥ አለ” is parsed wrongly with the noun phrase “እህል ንተራው ውስጥ” and verb phrase “አለ”. But the correct noun phrase is “እህል” and the verb phrase is “ንተራው ውስጥ አለ”.

Table 6.1: Number of Correctly and Incorrectly Parsed Declarative Sentences

No of Declarative Sentences	Correctly Parsed	Incorrectly Parsed	Percentage
20	**17**	**3**	**85%**

For the negative sentence “ካሳ ከእናቱ ጋር ምሳ አልበላም” we have the following result after parsing.

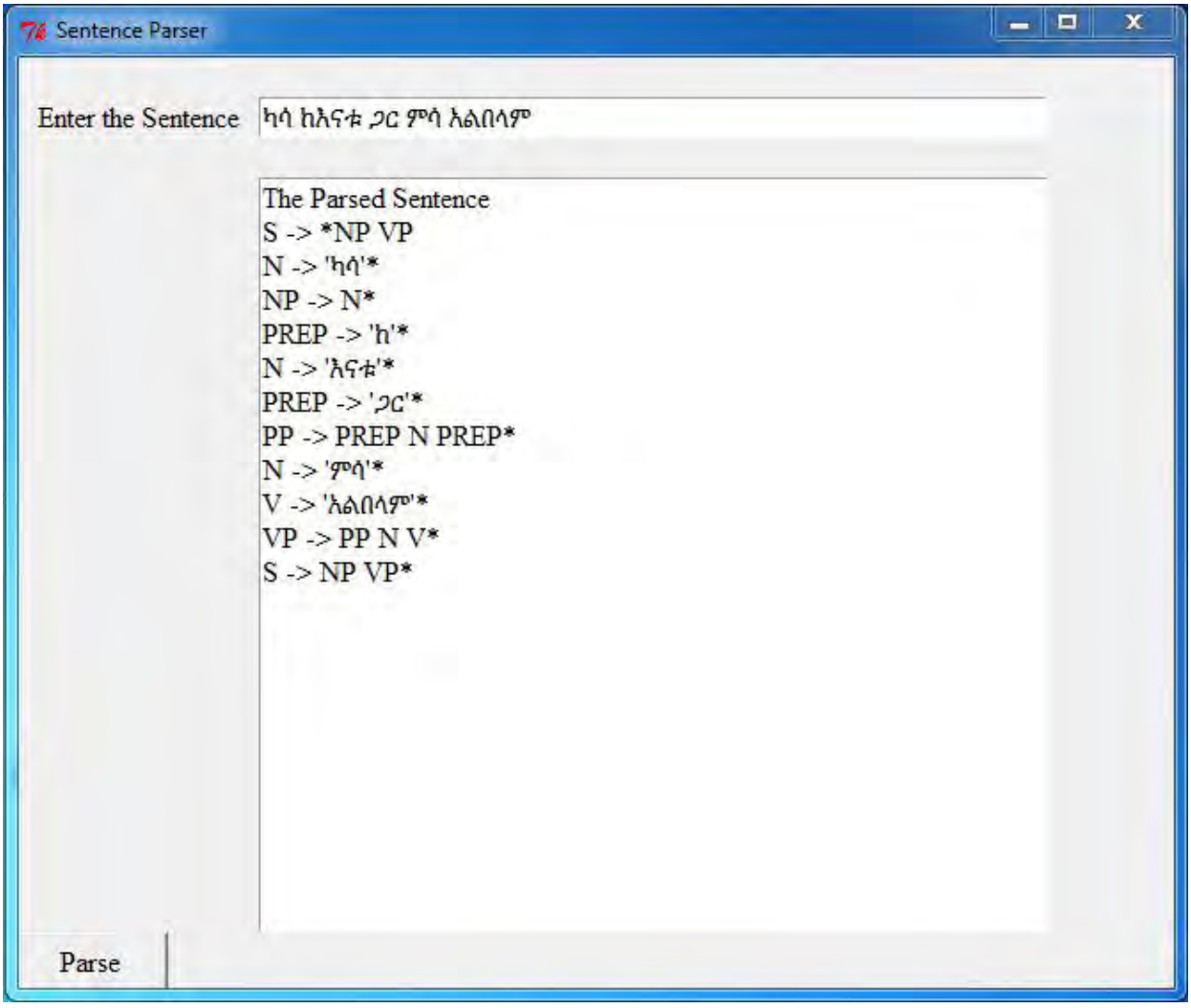

Figure 6.3: Screenshot of Parsed Negative Sentence

In figure 6.3 the noun phrase is “ካሳ” and the verb phrase is “ከእናቱ ጋር ምሳ አልበላም”. The verb phrase is constructed from “PP”, “N”, and “V” where “PP” is extended to PP ➔PREP N PREP.

Table 6.2: Number of Correctly and Incorrectly Parsed Negative Sentences

No of Negative Sentences	Correctly Parsed	Incorrectly Parsed	Percentage
20	**18**	**2**	**90 %**

For the question type sentence "አበበ ና ካሳ ለምን ተጣሉ", the noun phrase and the verb phrase are "አበበ ና ካሳ" and "ለምን ተጣሉ" respectively as shown in figure 6.4.

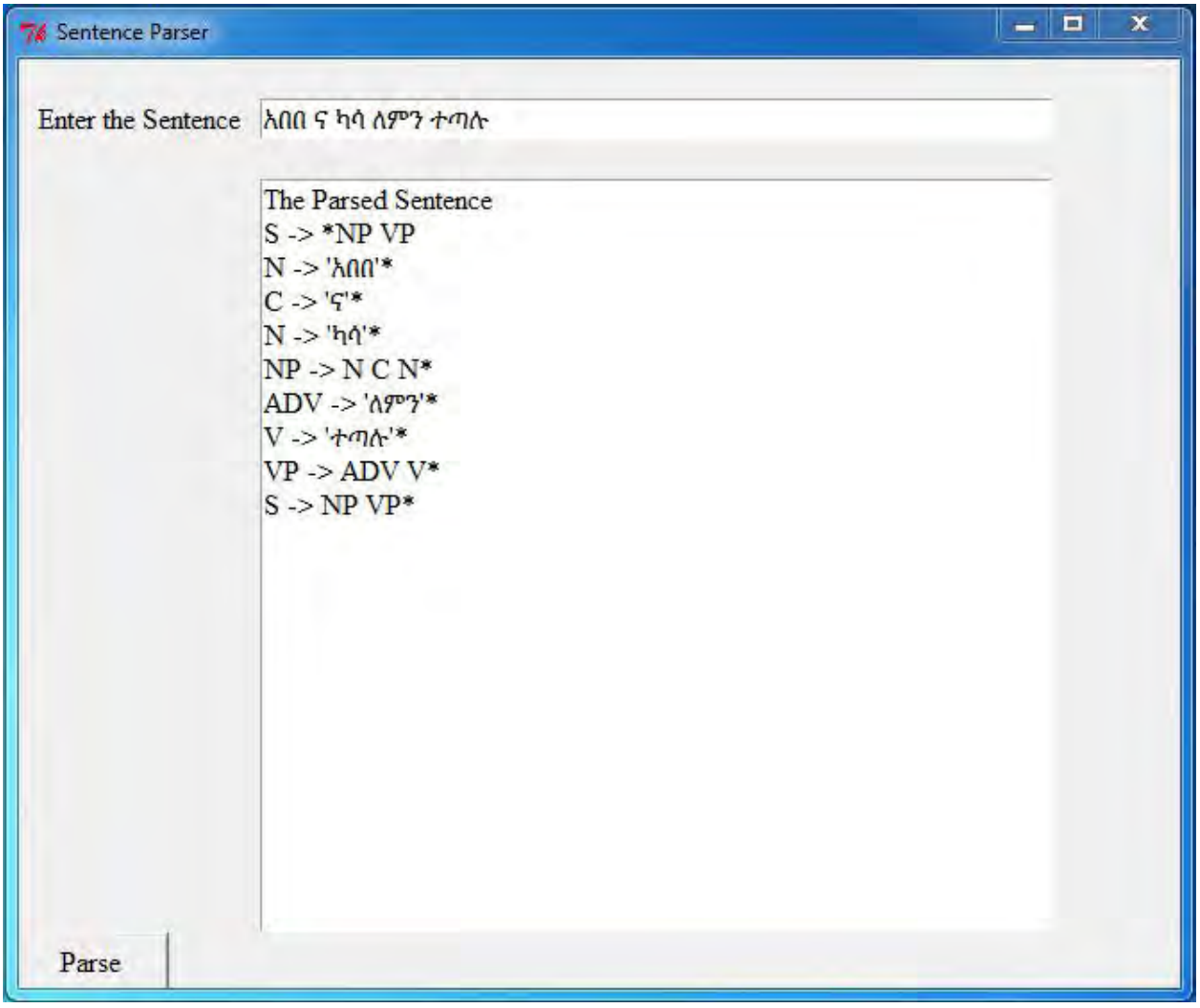

Figure 6.4: Screenshot of Parsed Interrogative Sentence

The noun phrase is constructed from N, C, and N (i.e. NP ➔ N C N) where the verb phrase is from ADV and V (i.e. VP ➔ ADV V).

Table 6.3: Number of Correctly and Incorrectly Parsed Interrogative Sentences

No of Interrogative Sentences	Correctly Parsed	Incorrectly Parsed	Percentage
20	**17**	**3**	**85 %**

For the imperative sentence “ፈተናዉን በትእዛዙ መሰረት ብቻ ስሩ”, the noun phrase is empty which is

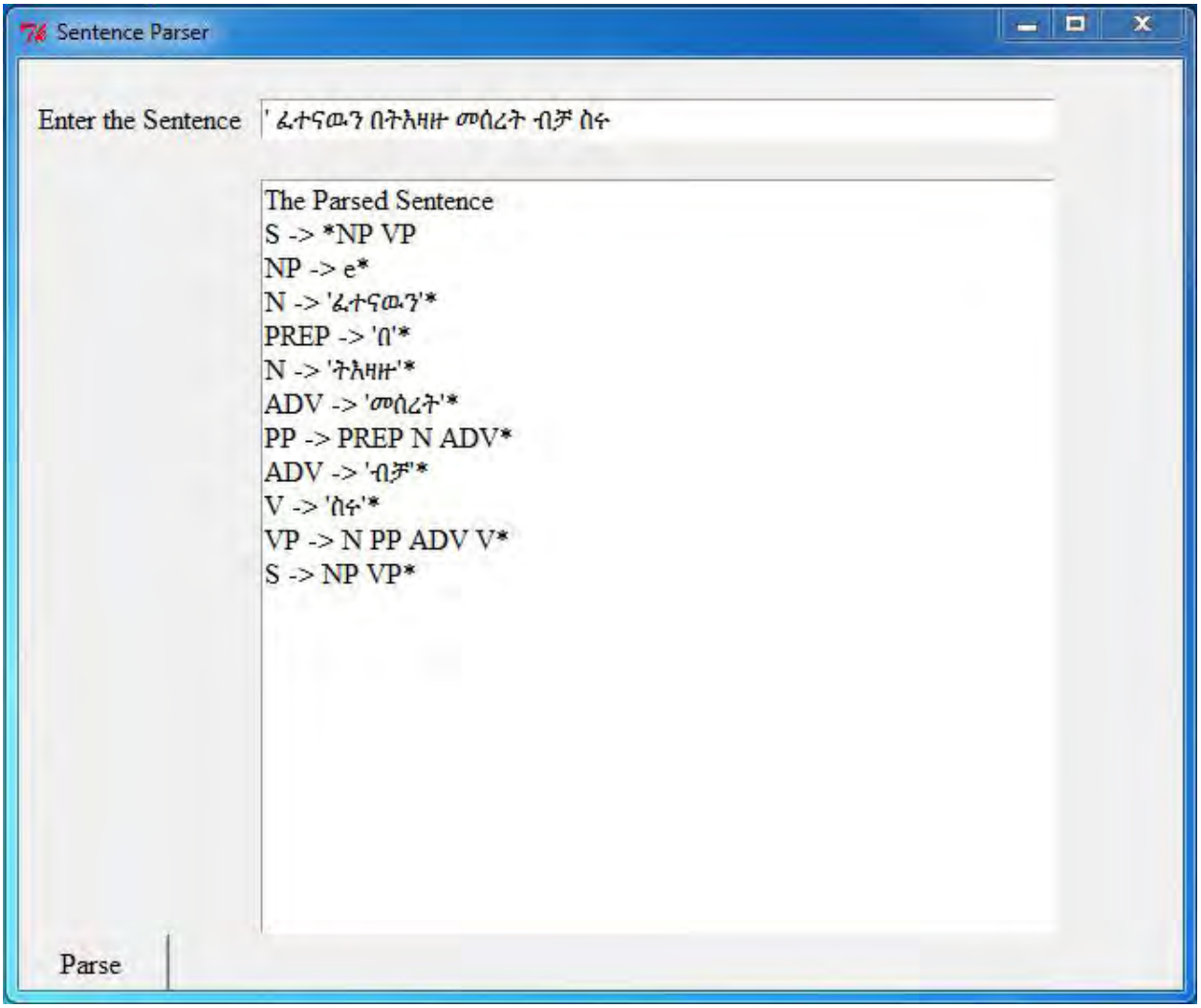

Figure 6.5: Screenshot of Parsed Imperative Sentence

NP➔e. This is because the noun phrase is “እናንተ” (you). However, it does not appear in the sentence. As a result the noun phrase is represented as empty. Therefore, the entire sentence will be considered as verb phrase (which is “ፈተናዉን በትእዛዙ መሰረት ብቻ ስሩ”).

Table 6.4: Number of Correctly and Incorrectly Parsed Imperative Sentences

No of Imperative Sentences	Correctly Parsed	Incorrectly Parsed	Percentage
20	**16**	**4**	**80%**

In figure 6.6, the complex sentence "ዘበኛው አሰሪው ስለ ፈቀደለት የማታ ትምህርት ተመዘገበ" is parsed as

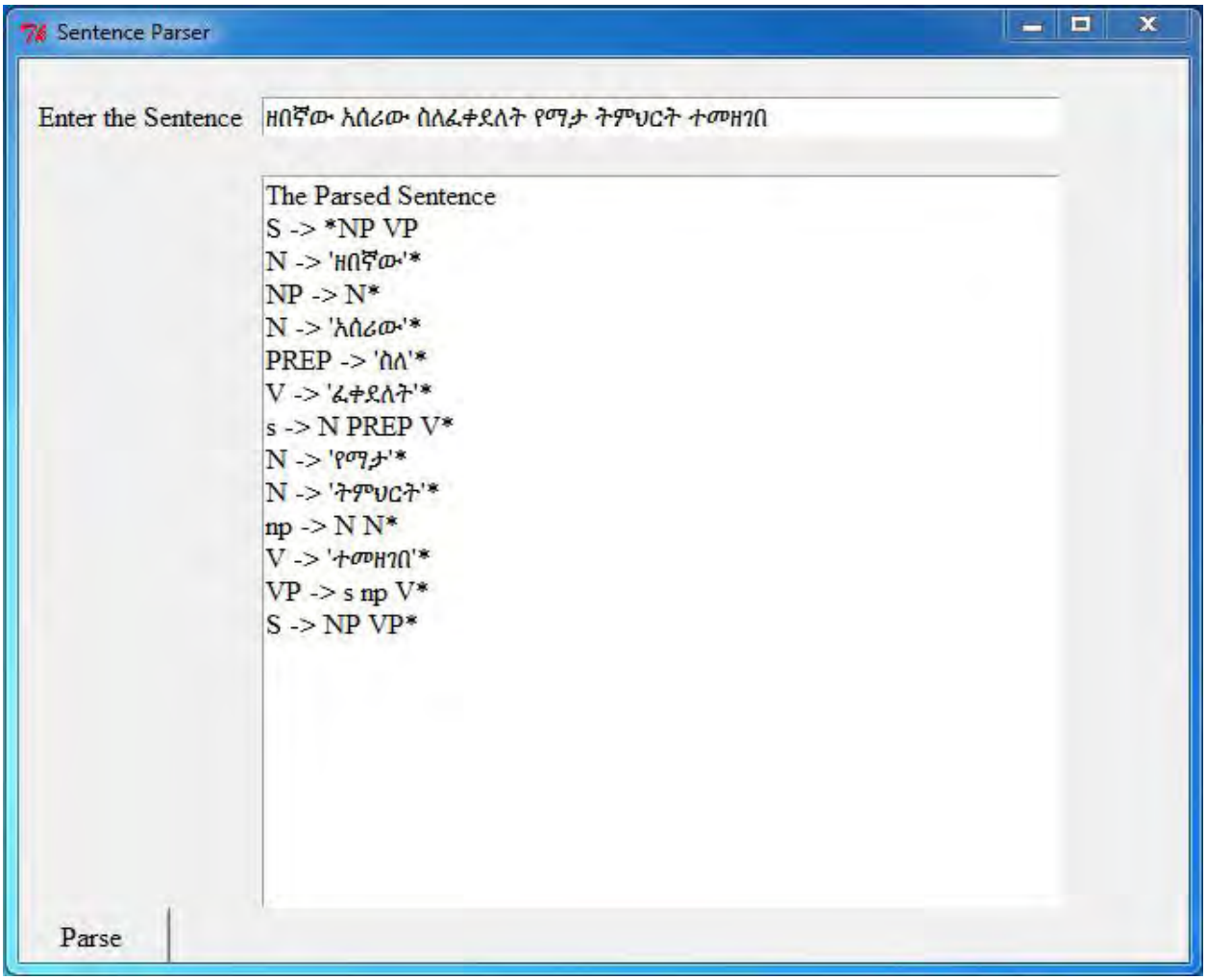

Figure 6.6: Screenshot of Parsed Complex Sentence

the noun phrase is "ዘበኛው" which is noun and the remaining part of the sentence (i.e. አሰሪው ስለፈቀደለት የማታ ትምህርት ተመዘገበ) is the verb phrase. The verb phrase is constructed from embedded clause "አሰሪው ስለፈቀደለት" represented as 's', noun phrase "የማታ ትምህርት" as denoted as 'np' and verb "ተመዘገበ " as 'V'.

Table 6.5: Number of Correctly and Incorrectly Parsed Complex Sentences

No of Complex Sentences	Correctly Parsed	Incorrectly Parsed	Percentage
20	**15**	**5**	**75%**

Figure 6.7 shows the parse structure of the complex sentences "ለፓርክ የሚሆኑ ቦታዎች በቦረና ዞን ተከለሉ". However, the parsing stops after it parsed up to the main noun phrase. The verb phrase is

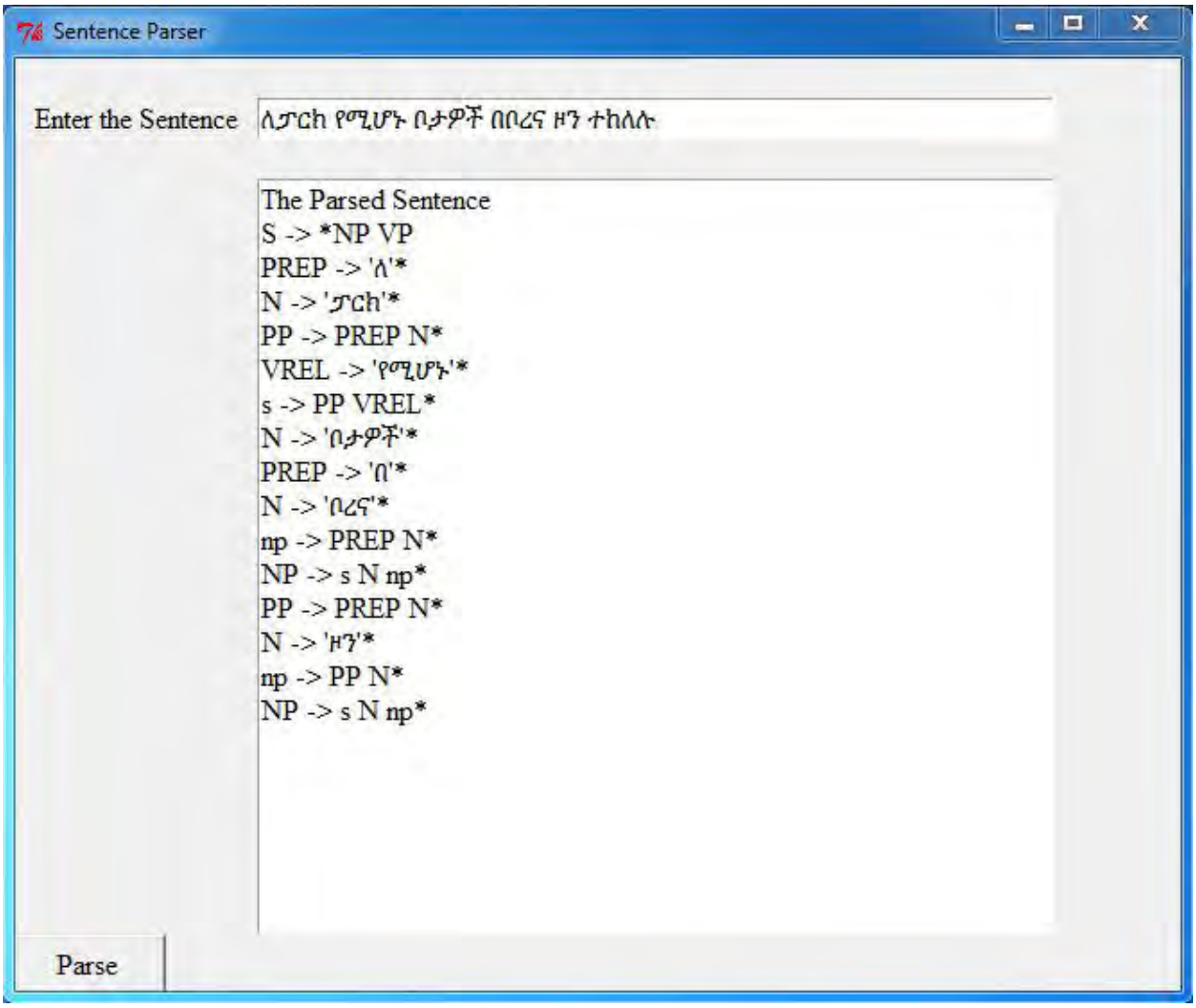

Figure 6.7: Screenshot of Wrongly Parsed Complex Sentence

not indicated in the output. For this reason the output is not treated as correctly parsed. The noun phrase and the verb phrase of the sentences are "ለፓርክ የሚሆኑ ቦታዎች በቦረና ዞን" and "ተከለሉ". The noun phrase on the other hand contains embedded clause 's', noun 'N' and another noun phrase 'np'. The embedded clause ('s') and the noun phrase ('np') are further extended to "s➔PP VREL" and "np➔ PP N" respectively, where PP➔ PREP N. This is correctly obtained from the

output. But the verb phrase “VP➔V” is not shown in the output even if the morpheme of the verb “ተከለሉ” is in the lexicon.

In figure 6.8, the imperative sentence “የቤት ስራውን ለነገ ሰርታችሁ እንድትምጡ” is shown with incorrect parse structure. The main noun phrase (i.e. “እናንተ”) of the sentence is empty (NP➔ e)

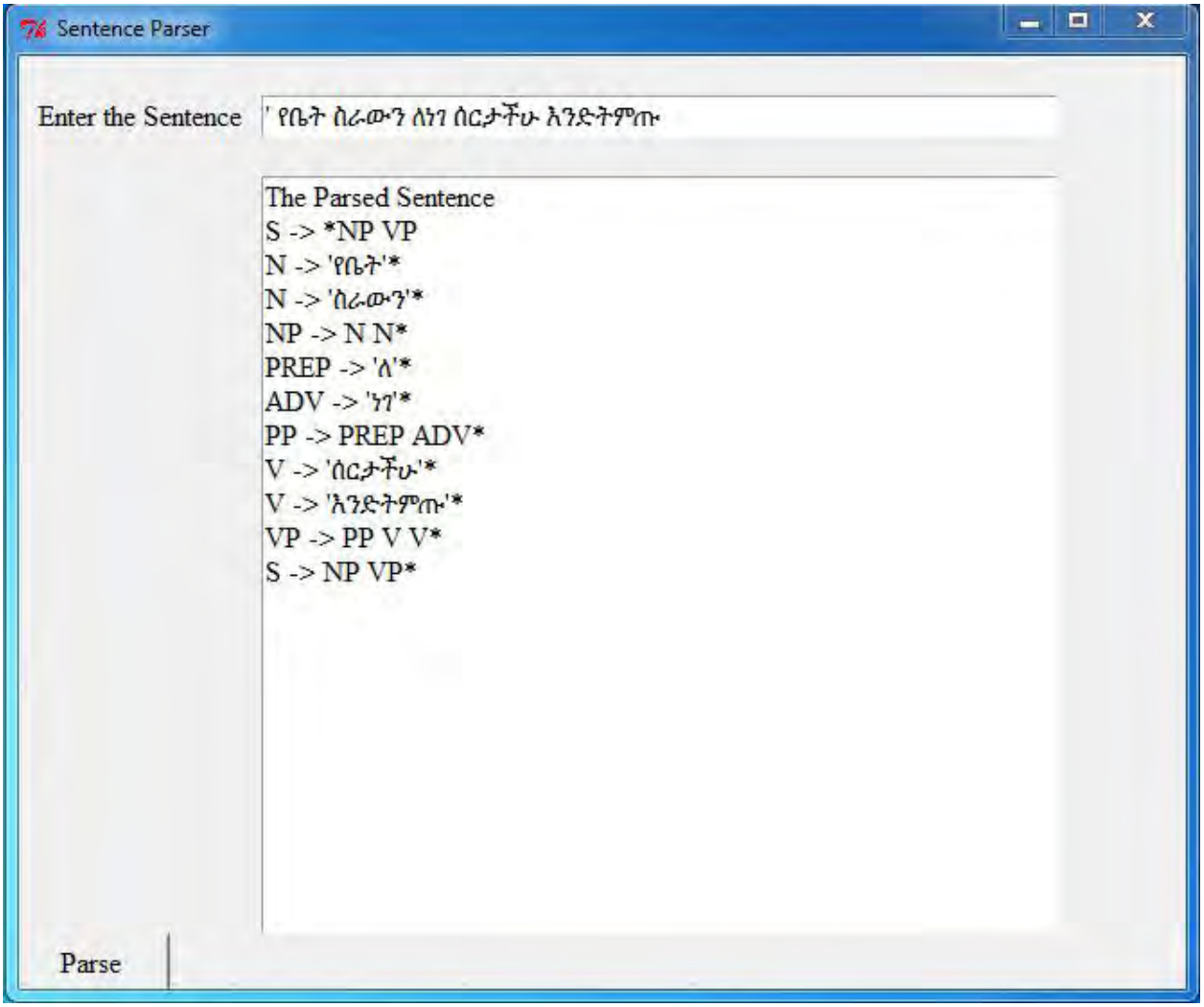

Figure 6.8: Screenshot of Wrongly Parsed Imperative Sentence

because it does not appear in the sentence and the rest of the whole sentence is the verb phrase which is ““የቤት ስራውን ለነገ ሰርታችሁ እንድትምጡ” with a representation of “VP ➔ s V” where “s➔ np PP” . However, the output indicates the noun phrase is “የቤት ስራውን” (NP➔ N N) and the verb phrase is “ለነገ ሰርታችሁ እንድትምጡ” (VP ➔ PP V V) which is wrong.

To sum up, the above results show that the parser was able to correctly parse 83 (which is 83% in average) sentences out of the given hundred testing sentences which is a promising result.

6.4 Discussion

Evaluation of the system was somehow challenging because of the absence of a standard criteria which says if such and such conditions are satisfied, the parsing is correct and if not the parsing is wrong. However, we have formulated our own criteria depending on the expected output of the system, to determine the effectiveness of the system. In addition to this, it was difficult to get Amharic experts who can evaluate the correctness of manually parsed sentences which we prepare for testing and in the extraction of the Context Free Grammar from the corpus. But, after repeated trials we can able to get an expert.

Performance evaluation of the proposed approach, which is the top-down chart parser, compared with the traditional methods (top-down and bottom-up approaches) is not shown due to time constraint. Meaning we were not able to evaluate parsing methods other than top-down chart parsing with the same testing sentences. However, by investigating their respective algorithms (how they are carry-out sentence parsing) we can conclude that the proposed approach outperformed traditional methods.

The other problem that we face during the evaluation is the efficiency of the morphological analyzer. Even though the morphological analyzer plays an important role in reducing the number of lexical rules in the lexicon, it has some problems in terms of producing morphemes for words which really do not have and producing morphemes in Amharic fonts in one time and in English fonts in another time. For example, names of persons, places and organizations are analyzed in wrong way by the morphological analyzer. This causes reduction in the accuracy of the result. Moreover, it takes long time to analyze each word of sentences. For this reason it requires high patience to test the whole sentence especially when slight errors are encountered. However, the presence of the morphological analyzer gives a great contribution in the construction of the lexicon. As a result, we did not store all words of the language, which is required in the absence of the morphological analyzer, in the lexicon. Moreover, it minimizes the time which is required to search the part of speech tag of words on a given sentence from the lexical rules.

In general, our proposed approach has shown promising results, in terms of handling prepositional phrase (which was treated as one word category in previous works), covering all Amharic sentences, integrating morphological analyzer for preventing the storage words which have common morpheme and reducing the number of words in the lexicon significantly and the automatic construction of the lexical rules compared to previous approaches which do not have morphological analyzer and use only traditional parsing methods like top-down and bottom-up approaches.

Chapter Seven – Conclusion and Future Work

Nowadays natural language processing is an active area of research. However, for under resourced languages like Amharic, there are limitations in electronic resources and processing tools. Sentence parser is one of natural language processing tools which is used for identifying the structure of a sentence in terms of noun phrase, verb phrase, noun, verb, adjective, etc. according to the grammar of the language. The results of sentence parser as a component plays a great role in solving machine translation, spelling checking, grammar checking, question answering and word sense disambiguation problems. Therefore, with the help of sentence parser the efficiency of the above mentioned applications can be improved.

7.1 Conclusion

The top-down chart parser has two major parts. The first part enables the system to learn rules and parse a given sentence according to them, and the second part accepts a sentence from the user, parse the sentence based on top-down chart algorithm and finally display the output for the user.

As the proposed approach is rule-based, the parser needs to have components which are used to enable the system learn how to parse from grammar rules. This part is composed of pre-processor, lexicon generator, morphological analyzer, context free grammar rules and lexicon. The corpus (collected mostly from the Walta Information Center corpus, Amharic grammar books and previous research works) required in the construction of grammar rules and lexicon would have to be free from unwanted texts and characters. For this reason, we have used a pre-processor to remove those texts before the corpus is used by other components. Lexicon generator is the other component used for the automatic construction of the lexical rules. It uses the same POS tagged corpus which is used for the extraction of context free grammar rules. In collaboration with the lexicon generator, a morphological analyzer has been used in the production of the lexicon. The main purpose of the morphological analyzer is to enable words which have common root word be represented by their morpheme in the lexicon, so that the number of words in the lexicon is reduced and the time required for searching word class of words for the given sentence is minimized.

In the testing part we have sentence tokenizer, morphological analyzer and chart parser. Sentence tokenizer is responsible for breaking the input sentence into words so that individual words can be given to the morphological analyzer. Morphological analyzer is used again in the testing part. This is because of the fact that words in the lexicon are morphologically analyzed. In other words, words of the input sentence may not be found directly in the lexicon as most of the words in the lexicon are root words. For this reason, words of the input sentence have to be morphologically analyzed before they are processed by the parser. Therefore, once the input sentence is passed through the tokenizer and the morphological analyzer, words will be rejoined to construct the sentence again with their respective morpheme. Then, the chart parser parses the sentence according to the algorithm and output the result for the user. The basic concept of chart parser is to obtain a table that contains all substructures generated during the parsing process in different iterations until all possible structures for the sentence are obtained. The parser learns how the parse structures of the given sentence can be constructed from the grammar rules and the lexicon. With this regard the parser tested on 100 sentences collected from all types of sentence (20 sentences from each type) and able to parser 92 sentences correctly.

7.2 Contributions of the work

The main contributions of the work are listed below.

- The general architecture of top-down chart parser for analyzing Amharic sentences is proposed
- The study has implemented chart parser algorithm used in other languages.
- The system has integrated morphological analyzer which was not used in previous works to prevent the storage of all language words in the lexicon and avoid word classes like noun preposition, verb preposition, numeric preposition and other such word classes in the lexicon. For example words like የኢህአዴግ, ለመከላከል and በ2ኛው have word classes' noun preposition, verb preposition and numeric preposition respectively.
- The study is designed to be a desktop application which gives end users with a user interface for interaction with the system.
- The system has implemented a rule-based approach which does not require a tree bank from which the parser learns how to parse through iterative trainings.

- The system has shown how to construct lexical rules automatically from tagged corpus.

7.3 Recommendations

The study has shown that sentence parsing can be done automatically using top-down chart parsing algorithm for Amharic sentences. However, sentence parsing is not an easy task, which requires more time and needs more features to make it full-fledged. Hence, further improvements and modifications are required. Below, additional features that can be added to increase the performance of the system and future research directions are listed.

- Integrating the parser with higher natural language processing applications like machine translation, information retrieval and see how their performance is improved.
- Improving the performance of the parser by adding automatic part of speech tagger. POS tagger has an important role in increasing the effectiveness of the parser and other applications.
- Improving the performance of the morphological analyzer. The performance of the morphological analyzer has an impact on the effectiveness of the sentence parser. Therefore, having a morphological analyzer with high efficiency plays an important role in the increase of the performance of the parser.
- Improving the performance of the system by using large dataset. The corpus that we used in the research is not quite enough to judge the effectiveness of the parser. Therefore, experimenting the system with large size corpus with various approaches is recommended.
- Experimenting how the sentence parser could perform using stochastic approaches other than rule-based approaches. Stochastic approaches use the idea of Bayer (Network theorem) and the Markov assumption to determine the most likely lexical sequence of each word in a given sentence. Though we didn't make experiments, better results might be obtained.
- Replicating the work in other local languages. In our country, there are a number of languages which are under resourced than Amharic language. Hence, implementing the sentence parser for those languages is important.

References

[1] Binyam Gebrekidan Gebre., "Part of Speech Tagging for Amharic", *in Natural Language Processing and Human Language Technology*, BULAG n 34, PUFC, ISSN 0758 6787, ISBN 978-2-84867-312-7, pp. 53-71, 2010.

[2] Gamback B., Ollson F., Argaw A. A., and Asker L.,"Methods for Amharic part of speech tagging ", *in 'AfLaT' 09: Proceedings of the first Workshop on Language Technologies for African Languages'*, Association for Computational Linguistics, Morristown, NJ, USA, pp.104-111.2009.

[3] Girma Awgichew Demeke and Mesfin Getachew, "Manual Annotation of Amharic news items with part of speech tags and its challenges". *Ethiopian Languages Research Center Working Papers 2, pp. 1-16, 2006.*

[4] Gasser M., "Semitic morphological analysis and Generation using finite state transducers with features, in 'EACL' 09: *Proceedings of the 12th Conference of the European Chapter of the Association for computational Linguistics'*, Association for Computational Linguistics, Morristown, NJ, USA, pp. 309-317.

[5] Atelach Alemu Argaw, "An Amharic stemmer: Reducing word to their citation form", *Proceedings of the 5th workshop on important unresolved matters,* pp. 104-110, June 2007.

[6] Reut Tsarfaty, "The Interplay of Syntax and Morphology in Building Parsing Models for Modern Hebrew", *Proceedings of the Eleventh EssLLI Student Session,* pp. *263-274,* 2006.

[7] Win Win Thant, Tin Myat Htwe, and Ni Lar Thein, "Context free Grammar Based top down Parsing of Myanmar sentences", *International Conference on Computer Science and Information technology (ICCSIT'2011)* Pattaya Dec. 2011.

[8] Björn Gambäck and Gunnar Eriksson, "Natural Language Processing at the School of Information Studies for Africa", *Swedish Program for ICT in Developing Regions Royal Institute of Technology/KTH Forum 100*, SE–164 40 Kista, Sweden.

[9] Wei Lu, Hwee Tou Ng, See Sun Lee, and Luke S. Zettlemoyer, "A Generative Model for Parsing Natural Language to Meaning Representations", In *Empirical Methods in Natural Language Processing (EMNLP),* 2008.

[10] Ahmad Al-Taani, Mohammad Msallam, and Sana Wedian, "A Top-Down Chart parser for Analyzing Arabic Sentences", *The international Arab Journal of Information Technology, Vol. 9, No. 2,* pp 109-115, March 2012.

[11] Dick G. and Ceriel H., "Parsing Techniques, a Practical Guide," *Technical Report, England, 1990.*

[12] Othman E., Shallan K., Rafea A., "A Chart Parser for Analyzing Modern Standard Arabic Sentences", *in Proceedings of the MT Summit IX Workshop on Machine Translation for Semitic Languages: Issues and Approaches, USA, pp.33-39, 2003.*

[13] Essam Al Daoud and Abdullah Basata, "A Framework to Automate the Parsing of Arabic Language Sentences", *The International Arab Journal of Information Technology, Vol. 6, No. 2,* pp.191-195, April 2009.

[14] Abiyot Bayou, "Developing Automatic Word Parser for Amharic Verbs and Their Derivation", *Unpublished, Master Thesis, School of Information Studies for Africa, Addis Ababa, 2000*.

[15] Atelach Alemu., "Automatic Sentence Parsing for Amharic Text: An Experiment Using Probabilistic Context Free Grammars", unpublished, *Master's Thesis School of Information Studies for Africa, Addis Ababa University*, 2002.

[16] Daniel Gochel, "An Integrated Approach to Automatic Complex Sentence Parsing for Amharic Text", Unpublished, *Master Thesis at School of Information Studies for Africa, Addis Ababa, 2003.*

[17] Seid Muhie Yimam, "TETEYEQ: Amharic Question Answering System for Factoid Questions", Unpublished, *Master Thesis, Department of Computer Science, Addis Ababa University*, 2009.

[18] Allen, J., "Natural Language Understanding, 2nd Ed.", *the Benjamin/Cummings*

Publishing Company, Inc., California, 1995.

[19] Chomsky Naom, "Syntactic Structures" Netherlands: Mounton & Co., 1995.

[20] Radford A., "Transformational Syntax", Cambridge University Press, 1981.

[21] Woods William A., "Transition Network Grammars of Natural language Analysis", Communication of the ACM Reprinted in Barbara J. Grosz, Karen, 1970.

[22] Key M., "Functional Grammars", in proceedings of the fifth Annual Meeting of the Barclay linguistic Society, 1979.

[23] Charniak, Eugen "Statistical Language Learning", MIT Press, Cambridge, 1993.

[24] Bala Sundara Raman L, Ishwar S, Sanjeeth Kumar, and Ravin Dranath, "Context free Grammar for Natural Language Constructs – *An Implementation for Venpa Class of Tamil Poetry", Tamil Internet, Chennai, India, 2003.*

[25] Grishman, Ralphe, "Natural Language Processing", *Journal of the American Society for Information, vol 1 35(5): 291-296, 1984.*

[26] Bilal M. Bataineh and Emad A. Bataineh, "An Efficient Recursive Transition Network Parser for Arabic languages", *Proceedings of the World Congress on Engineering, Vol II, London, U.K, 2009.*

[27] Aynadis Temesgen, "Development of Amharic Grammar Checker Using Morphological Features of Words and N-Gram Based Probabilistic Methods", *Master's thesis, Addis Ababa University, Unpublished, 2013.*

[28] Christopher D. Manning, "Foundations of Statistical Natural Language Processing", *The MIT Press Cambridge, Massachusetts London, England, 2000.*

[29] Nabil Khoufi, Chafik Aloulou and Lamia Hadrich Belguith, "ARSYPAR: A Tool for Parsing the Arabic language based on Supervised Learning", *The International Arab Conference on Information Technology (ACIT), 2013.*

[30] T. L. Booth and R. A. Thompson, "Applying Probability Measures to Abstract Language", *IEEE Transactions on Computers C-22, pp 442-450, 1973.*

[31] Baye Yimam, "የአማርኛ ሰዋስው", EMPDA, Addis Ababa, 1987 (EC).

[32] Mesfin Getachew, "Automatic Part of Speech Tagging for Amharic: An experiment using *Hidden Markov Model (HMM) Approach", Unpublished Master's Thesis, Addis Ababa University, 2001.*

[33] Frydenlund, M. and Sevensene, K., "Amharic for Beginners", Norwegian Lutheran Mission Language School, Addis Ababa, 1998.

[34] Abeba Ibrahim, "A Hybrid Approach to Amharic Base Phrase Chunking and Parsing", *Master's thesis, Addis Ababa University, Unpublished 2013.*

[35] Diriba Megersa, "Automatic Sentence Parser for Oromo Language Using Supervised Learning Technique", *Master's thesis, Addis Ababa University, Unpublished 2002.*

[36] Bilal M. Bataineh and Emad A. Bataineh, "An Efficient Recursive Transition Network Parser for Arabic Language", *Proceedings of the World Congress on Engineering Vol II, London, 2009.*

[37] Nabil Khoufi, Chafik Aloulou and Lamia Hadrich Belguith, "ARSYPAR: A tool for Parsing the Arabic language based on supervised learning", *The International Arab Conference on Information Technology, ACIT 2013.*

[38] Daniel D. K. Sleator and Davy Temperley, "Parsing English with a Link Grammar", School of Computer Science, Carnegie Mellon University, 1991.

[39] Eugene Charniak, "Statistical Parsing with a Context-free Grammar and Word Statistics", Department of Computer Science, Brown University, 1997.

[40] Nitin Hambir and Ambrish Srivastav, "Hindi Parser-based on CKY algorithm", *Nitin Hambir et al, Int.J. Computer Technology & Applications,Vol 3 (2), pp 851-85,2012.*

[41] Heng Lian, "Chinese Language Parsing with Maximum-Entropy-Inspired Parser", Brown University, 2005.

[42] Baye Yimam, "የአማርኛ ሰዋስው", EMPDA, Addis Ababa, 2002 (EC).

[43] M. Woldeqirqos, "የአማርኛ ሰዋስው", Birhanena Selam Printing Press, Addis Ababa, 1934.

[44] Gasser M., "HornMorpho 2.5", School of Informatics and Computing, Indiana University, 2012.

[45] Shihadeh A., Hasan M., and Mahmud S., "Context-Free Grammar Analysis for Arabic Sentences", *International Journal of Computer Applications (0975-8887), Volume 53-No.3, 2012.*

[46] https://www.python.org/doc/essays/comparisons, Last accessed on Dec 4, 2014.

[47] Desalegn Abebaw, "LETEYEQ (ልጠየቅ) - A Web Based Amharic Question Answering System for Factoid Questions Using Machine Learning Approach", *Master's thesis, Addis Ababa University, Unpublished 2013.*

[48] Wenpeng LU, Heyan HUANG, and Chaoyong ZHU, "Feature Words Selection for Knowledge-based Word Sense Disambiguation with Syntactic Parsing", *Electrical Review, ISSN 0033-2097, R. 88 NR 1b, 2012.*

[49] Jaspreet Kaur and Kamaldeep, "Hybrid Approach for Spell Checker and Grammar Checker for Punjabi", *international Journal of Advanced Research in Computer Science and Software Enginnering, ISSN: 2277 128X, Vol 4, issue 6, 2014.*

[50] Rajula Srilatha and K. Murali, "A Novel Incremental Information Extraction Using Parse Tree Query Language and Parse Tree Databases", *International Journal of Computer Trends and Technology (IJCTT), Vol 4, 2013.*

Appendices

Appendix A: Ethiopic Unicode representations (1200-137F)

	120	121	122	123	124	125	126	127	128	129	12A	12B
0	ሀ 1200	ሐ 1210	ሠ 1220	ሰ 1230	ቀ 1240	ቐ 1250	በ 1260	ተ 1270	ኀ 1280	ነ 1290	አ 12A0	ኰ 12B0
1	ሁ 1201	ሑ 1211	ሡ 1221	ሱ 1231	ቁ 1241	ቑ 1251	ቡ 1261	ቱ 1271	ኁ 1281	ኑ 1281	ኡ 12A1	
2	ሂ 1202	ሒ 1212	ሢ 1222	ሲ 1232	ቂ 1242	ቒ 1252	ቢ 1262	ቲ 1272	ኂ 1282	ኒ 1292	ኢ 12A2	ኲ 12B2
3	ሃ 1203	ሓ 1213	ሣ 1223	ሳ 1233	ቃ 1243	ቓ 1253	ባ 1263	ታ 1273	ኃ 1283	ና 1293	ኣ 12A3	ኳ 12B3
4	ሄ 1204	ሔ 1214	ሤ 1224	ሴ 1234	ቄ 1244	ቔ 1254	ቤ 1264	ቴ 1274	ኄ 1284	ኔ 1294	ኤ 12A4	ኴ 12B4
5	ህ 1205	ሕ 1215	ሥ 1225	ስ 1235	ቅ 1245	ቕ 1255	ብ 1265	ት 1275	ኅ 1285	ን 1295	እ 12A5	ኵ 12B5
6	ሆ 1206	ሖ 1216	ሦ 1226	ሶ 1236	ቆ 1246	ቖ 1256	ቦ 1266	ቶ 1276	ኆ 1286	ኖ 1296	ኦ 12A6	
7	ሇ 1207	ሗ 1217	ሧ 1227	ሷ 1237	ቇ 1247		ቧ 1267	ቷ 1277	ኇ 1287	ኗ 1297	ኧ 12A7	
8	ለ 1208	መ 1218	ረ 1228	ሸ 1238	ቈ 1248	ቘ 1258	ቨ 1268	ቸ 1278	ኈ 1288	ኘ 1298	ከ 12A8	ኸ 12B8
9	ሉ 1209	ሙ 1219	ሩ 1229	ሹ 1239			ቩ 1269	ቹ 1279		ኙ 1299	ኩ 12A9	ኹ 12B9
A	ሊ 120A	ሚ 121A	ሪ 122A	ሺ 123A	ቊ 124A	ቚ 125A	ቪ 126A	ቺ 127A	ኊ 128A	ኚ 129A	ኪ 12AA	ኺ 12BA
B	ላ 120B	ማ 121B	ራ 122B	ሻ 123B	ቋ 124B	ቛ 125B	ቫ 126B	ቻ 127B	ኋ 128B	ኛ 129B	ካ 12AB	ኻ 12BB
C	ሌ 120C	ሜ 121C	ሬ 122C	ሼ 123C	ቌ 124C	ቜ 125C	ቬ 126C	ቼ 127C	ኌ 128C	ኜ 129C	ኬ 12AC	ኼ 12BC
D	ል 120D	ም 121D	ር 122D	ሽ 123D	ቍ 124D	ቝ 125D	ቭ 126D	ች 127D	ኍ 128D	ኝ 129D	ክ 12AD	ኽ 12BD
E	ሎ 120E	ሞ 121E	ሮ 122E	ሾ 123E			ቮ 126E	ቾ 127E		ኞ 129E	ኮ 12AE	ኾ 12BE
F	ሏ 120F	ሟ 121F	ሯ 120F	ሿ 123F			ቯ 126F	ቿ 127F		ኟ 129F	ኯ 12AF	

	12C	12D	12E	12F	130	131	132	133	134	135	136	137
0	ኸ 12C0	ዐ 12D0	ዠ 12E0	ደ 12F0	ጀ 1300	ገ 1310	ጠ 1320	ጰ 1330	ፀ 1330	ፐ 1350	፠ 1360	፩ 1370
1		ዑ 12D1	ዡ 12E1	ዱ 12F1	ጁ 1301		ጡ 1321	ጱ 1331	ፁ 1341	ፑ 1351	፡ 1361	፪ 1371
2	ኹ 12C2	ዒ 12D2	ዢ 12E2	ዲ 12F2	ጂ 1302	ጊ 1312	ጢ 1322	ጲ 1332	ፂ 1342	ፒ 1352	። 1362	፫ 1372
3	ኺ 12C3	ዓ 12D3	ዣ 12E3	ዳ 12F3	ጃ 1303	ጋ 1313	ጣ 1323	ጳ 1333	ፃ 1343	ፓ 1353	፣ 1363	፬ 1373
4	ኻ 12C4	ዔ 12D4	ዤ 12E4	ዴ 12F4	ጄ 1304	ጌ 1314	ጤ 1324	ጴ 1334	ፄ 1344	ፔ 1344	፤ 1364	፭ 1374
5	ኼ 12C5	ዕ 12D5	ዥ 12E5	ድ 12F5	ጅ 1305	ግ 1315	ጥ 1325	ጵ 1335	ፅ 1345	ፕ 1355	፥ 1365	፮ 1375
6		ዖ 12D6	ዦ 12E6	ዶ 12F6	ጆ 1306		ጦ 1326	ጶ 1336	ፆ 1346	ፖ 1356	፦ 1366	፯ 1376
7			ዧ 12E7	ዷ 12F7	ጇ 1307		ጧ 1327	ጷ 1337	ፇ 1347	ፗ 1357	፧ 1367	፰ 1377
8	ወ 12C8	ዘ 12D8	የ 12E8	ደ 12F8	ገ 1308	ጘ 1318	ጨ 1328	ጸ 1338	ፈ 1348	ፘ 1358	፨ 1368	፱ 1378
9	ዉ 12C9	ዙ 12D9	ዩ 12E9	ዹ 12F9	ጉ 1309	ጙ 1319	ጩ 1329	ጹ 1339	ፉ 1349	ፙ 1359	፩ 1369	፲ 1379
A	ዊ 12CA	ዚ 12DA	ዪ 12EA	ዺ 12FA	ጊ 130A	ጚ 131A	ጪ 132A	ጺ 133A	ፊ 134A	ፚ 135A	፪ 136A	፳ 137A
B	ዋ 12CB	ዛ 12DB	ያ 12EB	ዻ 12FB	ጋ 130B	ጛ 131B	ጫ 132B	ጻ 133B	ፋ 134B		፫ 136B	፴ 137B
C	ዌ 12CC	ዜ 12DC	ዬ 12EC	ዼ 12FC	ጌ 130C	ጜ 131C	ጬ 132C	ጼ 133C	ፌ 134C		፬ 136C	፵ 137C
D	ው 12CD	ዝ 12DD	ይ 12ED	ዽ 12FD	ግ 130D	ጝ 131D	ጭ 132D	ጽ 133D	ፍ 134D		፭ 136D	
E	ዎ 12CE	ዞ 12DE	ዮ 12EE	ዾ 12FE	ጎ 130E	ጞ 131E	ጮ 132E	ጾ 133E	ፎ 134E		፮ 136E	
F	ዏ 12CF	ዟ 12DF	ዯ 12EF	ዿ 12FF	ጏ 130F	ጟ 131F	ጯ 132F	ጿ 133F	ፏ 134F		፯ 136F	

Appendix B: POS tag by WIC (whose corpus is used in this study)

No.	Name of POS tag	Description
1	ADJ	Adjective
2	N	Noun
3	VREL	Relative Verb
4	NUMCR	Cardinal Number
5	V	Verb
6	ENDPUNC	Sentence end punctuation
7	NPrep	Noun with Preposition
8	VPrep	Verb with Preposition
9	NUMPrep	Number with preposition
10	PREP	Preposition
11	VN	Verbal Noun
12	ADJPrep	Adjective with Preposition
13	NC	Noun with conjunction
14	ADV	Adverb
15	PUNC	Punctuation
16	NPC	Noun with Preposition and Conjunction
17	AUX	Auxiliary verbs
18	PRONP	Pronoun with Preposition
19	CONJ	Conjunction
20	NUMOR	Ordinal Number
21	VPC	Verb with Preposition and conjunction
22	PRON	Pronoun
23	PRONPC	Pronoun with Preposition and conjunction
24	ADJC	Adjective with Conjunction
25	VC	Verb with Conjunction
26	PRONC	Pronoun with Conjunction
27	UNC	Unclear

28	ADJPC	Adjective with Preposition and Conjunction
29	INT	Interjection
30	NUMC	Number with Conjunction
31	NUMPC	Number with Preposition and Conjunction

Appendix C: Sample tagged WIC text

<title><dateline place=አይሳኢታ> <body>በአፋር <NP> ክልል <N> በአጭር <ADJP> ጊዜ <N> ከሳምባ <NP> በሽታ <N> ሊፈወስ <VP> የሚያስችል <VN> " <PUNC> ዶትስ <N> " <PUNC> የተባለ <VP> ዘመናዊ <ADJ> የህክምና <NP> አሰጣጥ <V> መጀመሩን <VN> የክልሉ <NP> ጤና <N> ቢሮ <N> አስታወቀ <V> ። <PUNC> በቢሮው <NP> የተላላፊ <ADJP> በሽታዎች <N> መከላከያና <NPC> መቆጣጠሪያ <NPC> መምሪያ <N> ሃላፊ <N> አቶ <ADJ> አህመዲን <N> ሃጂዋሴ <N> ዛሬ <N> እንዳስታወቁት <VP> ከያዝነው <VP> ወር <N> አጋማሽ <ADJ> ጀምሮ <V> የተጀመረው <VP> ይሽው <PRON> ህክምና <N> ቀደም ሲል <VP> ለአንድ <NUMP> በሽተኛ <N> ከ1 <NUMP> አመት <NUMCR> እስከ <PREP> 1 <NUMCR> አመት <N> ተኩል <ADJ> ይሰጥ <V> የነበረውን <VP> ህክምና <N> ወደ <PREP> 6 <NUMCR> ወር <N> ዝቅ <N> ያደርገዋል <V> ። <PUNC> ፕሮግራሙ <N> ለጊዜው <NP> በዞን <NP> 1 <NUMCR> እና <CONJ> ዞን <N> 3 <NUMCR> የዱብቲ <NP> እና <CONJ> የናሽናል <NP> ሆስፒታሎችን <N> ጨምሮ <V> በአይሳኢታ <NP> ፣ <PUNC> ገዋኔ <N> ፣ <PUNC> መልካወረርና <N> አዋሽ <N> 7 <NUMCR> ኪሎ <N> ጤና <N> ታቢያዎች <N> መጀመሩን <VN> ገልጸው <V> ፤ <PUNC> ህክምናውን <N> የሚሰጡ <VP> እና <CONJ> በመስኩ <NP> የሰለጠኑ <ADJP> ከ100 <NUMP> በላይ <PREP> የጤና <NP> ባለሙያዎች <N> መሰማራታቸውን <VN> አስረድተዋል <V> ። <PUNC> በቀጣይ <ADJP> በቂ <ADJ> ባለሙያዎች <ADJ> በማሰልጠን <V> እና <CONJ> ተጨማሪ <V> መድሃኒት <N> በማስመጣት <VP> በሁሉም <PRONP> ዞኖች <N> ፕሮግራም <N> እንደሚስፋፉ <NP> አስታውቀዋል <V> ። <PUNC> ፕሮግራሙን <N> በክልሉ <NP> ለማስጀመር <NP> ታቅዶ <V> የነበረው <VREL> ባለፈው <ADJ> አመት <N> ቢሆንም <VP> በወቅቱ <NP> መድሃኒት <N> ባለመኖሩ <VN> ሊጀመር <VP> ያለመቻሉን <VN> አስታውሰው <V> ፤ <PUNC> ባሁኑ <NP> ጊዜ <N> ከጤና <N> ጥበቃ <N> ሚኒስቴር <N> ፕሮግራሙን <N> ለማስጀመር <VN> የሚያስችል <VN> መድሃኒት <N> መላኩን <VN> ተናግረዋል <V> ። <PUNC> በክልሉ <NP> ያሉትን <ADJ> የሳምባ <NP> ነቀርሳ <N> በሽተኞች <N> ቁጥር <N> ለማወቅ <VP> የሚያስችል <VN> ጥናት <N> ባይካሄድም <V> በሽታው <N> በክልሉ <NP> ከወባ <NP> ቀጥሎ <ADJ> በገዳይነቱ <VN> በ2ኛ <NUMP> ደረጃ <N> ላይ <PREP> እንደሚገኝ <VP> ገልጸው <V> ፤ <PUNC> ይሄ <PRON> ህክምና <N> የበሽታውን <NP> ስርጭት <N> በከፍተኛ <ADJP> ደረጃ <N> እንደሚቀንሰው <VN> አመልክተዋል <V> ።

Appendix D: Sample Context Free Grammar Rules Extracted from the Corpus

1. S ➔ NP VP
2. NP ➔ N
3. NP ➔e
4. NP➔s V
5. s➔PP N
6. NP ➔ ADJ N
7. NP➔ N N PREP
8. NP➔N PREP N
9. NP➔ N ADJ N
10. NP ➔ N N PREP
11. NP ➔ ADV N
12. NP➔N V N
13. NP ➔ N C C
14. NP ➔ N N V PREP
15. NP ➔Num PREP ADJ N
16. VP ➔s V
17. s➔NP PP
18. s➔NP PP V
19. VP➔ ADV ADJ V
20. VP ➔ PREP V
21. NP➔N V N
22. NP➔N NP
23. NP ➔NP N
24. NP ➔
25. VP➔ADV V
26. VP ➔ N PP
27. VP ➔ s NP V
28. s➔ PREP V
29. s➔ NP VREL
30. s➔VREL
31. NP ➔Num N
32. VP ➔ ADV VP
33. PP ➔ N PREP
34. PP ➔ PREP N
35. PP ➔ PREP N PREP
36. ADJP ➔ ADV ADJ
37. ADJP ➔ ADJ
38. ADVP ➔ ADV
39. VP ➔ PP V
40. VP ➔ N VP
41. VP ➔ N V
42. VP ➔ NP V
43. VP ➔ADV V
44. VP ➔ ADJP V
45. VP ➔ ADV VP
46. VP ➔ V V
47. NP ➔ N N
48. NP ➔ N NP
49. NP ➔ NP N
50. NP ➔ ADJ NP
51. NP ➔ N C N
52. NP ➔ N PREP
53. PP ➔ N PREP
54. PP ➔ PREP N
55. VP ➔ ADJ V
56. VP ➔ ADV N V
57. VP ➔PP N V
58. VP ➔ NP PP ADV V

Appendix E: Sample Lexical rules Generated by the Lexicon Generator

1. PREP -> " የ "
2. N -> " አዲስአበባ "
3. N -> " ዩኒቨርሲቲ "
4. PREP -> " ለ "
5. ADJ -> " dokter "
6. N -> " ብርሃነ "
7. N -> " ፕሮፌሰርነት "
8. N -> " ma'reg "
9. V -> " sT* "
10. N -> " 'y_l "
11. ADJ -> " tn_x "
12. N -> " 'njera "
13. V -> " bl' "
14. ADV -> " 'mn "
15. N -> " ministr "
16. N -> " wetad_er "
17. N -> " hager "
18. PREP -> " ስለ "
19. ADJ -> " Tam "
20. V -> " msgn "
21. N -> " 'amErika "
22. N -> " dWlarr "
23. PREP -> " በ "
24. V -> " xyT "
25. ADV -> " hulgiz "
26. N -> " ljt "
27. ADJ -> " l'q "
28. N -> " wendm_ "
29. V -> " Tn' "
30. C -> "ወይም"
31. N -> " kfl "
32. PREP -> " wsT "
33. V -> " ngr "
34. N -> " 'astEr "
35. ADV -> " zarE "
36. V -> " 'mm "
37. PREP -> " ወደ "
38. N -> " sra "
39. V -> " hyd "
40. N -> " mEtr "
41. N -> " gemed "
42. PREP -> " ከ "
43. N -> " gebeya "
44. V -> " gz' "
45. C -> " ና "
46. N -> " 'abatye "
47. V -> " wT' "
48. N -> " werq "
49. N -> " se'at "
50. V -> " xl_m "
51. C -> " wey "
52. N -> " 'awrop_a "
53. N -> " ሀብረት "
54. ADV -> " tnant "
55. N -> " 'Ertra "
56. N -> " 'ambasader "
57. N -> " halafin_et "
58. V -> " lqq "
59. V -> " ne "
60. N -> " borsa "
61. ADJ -> " bzu "
62. N -> " 'qa "
63. V -> " y'z "
64. ADJ -> " gobez "
65. N -> " m'r "
66. ADV -> " tlant "
67. N -> " bEt "
68. V -> " mT' "
69. N -> " denqoro "
70. N -> " den_ "
71. ADJ -> " qey_ "
72. N -> " qebero "
73. N -> " quTr "
74. V -> " Cm_r "
75. C -> " እና "
76. N -> " qWnjt "
77. N -> " serg "
78. V -> " drs "
79. N -> " gWad_eN_a "
80. N -> " ሽርጉድ "
81. V -> " b'l "
82. N -> " ኦሀዬድ "
83. N -> " ተሃድሶ "
84. N -> " wyy_t "
85. V -> " jm_r "
86. ADV -> " መቸ "
87. V -> " dngT "
88. N -> " xfan "
89. Num -> " ቁጥር "
90. V->"'dg"

Appendix F: Sample Sentences collected from sources other than WIC

1. ኮከቦች በንጋት በጣም ያበራሉ
2. ፕሬዘዳንቱ አርቲስቱ በፊልም የገለፀውን ብልህ ንጉስ አደነቁ
3. ከንግዲህ እኔ ቤት ድርሽ እንዳትል
4. ቢሮው የንግድ ፈቃድ የተሰረዘባቸውን ነጋዴዎች ዝርዝር አወጣ
5. ወንበዴዎች በጉ ፈቃደኞች የነበትን ተቋም ከጥቅም ውጭ አደረጉ
6. ዛሬ ወደ ስራ አልሄድም
7. ንፋሱ ዘላኖች አሸዋ ላይ የሰሩትን ቤት ጨርሶ አፈረሰ
8. አስቴር እህቷ በጣም ስለወፈረች ከመጠን በላይ ተናደደች
9. ትምህርት እንዴት ይዞሃል
10. ቢሮው ፈቃድ የተሰረዘባቸውን ነጋዴዎች ዝርዝር አወጣ
11. ሰውየው በጣም ረጅም ነዉ
12. እህል ጎተራው ውስጥ አለ
13. ሰራተኛዋ ዝናብ ያረጠበውን ልብስ ቤት ውስጥ አሰጣች
14. ከበደ ስለታመመ ትምህርት ቤት አልመጣም
15. ዘበኛው አሰሪው ስለፈቀደለት የማታ ትምህርት ተመዘገበ
16. ልጅቱ ቁርጥ እናቷን ትመስላለች
17. ወደ አዲስአበባ መቸ መጣህ
18. የትምህርት ሚኒስቴር ለኤችአይቪኤድስ መከላከያ ነዐ.8ሚሊየን ብር መደበ
19. እመጣለሁ ብለህ ለምን ቀረህ
20. ያዘዝኩህን ብቻ ሰርተህ ጠብቀኝ
21. ህጻኑ ስለታመመ ምግብ አልተመገበም
22. አበበ ና ካሳ ለምን ተጣሉ
23. የቤት ስራዉን ለነገ ሰርታችሁ እንድትመጡ
24. ፈተናዉን በትእዛዙ መሰረት ብቻ ስሩ
25. የካሳ ጓደኛ እንደ ካሳ ጎበዝ ተማሪ ለመሆን ሞከረ
26. ወፎቹ በቀለ ያፈሰሰውን ስንዴ በችኮላ ለቀሙ
27. አሮጊቱ ሌቦች አምና የገደሉትን ብቸኛ ልጃቸውን አስታወሱ
28. ተማሪው ጓደኛው ስለዘገየ ወደ ክፍል ብቻውን ገባ
29. እኛ አስተማሪው የሰጠንን የቤት ስራ ክፍል ውስጥ ሰራን
30. ኤርትራ ከስንት ሀገሮች ጋር ትዋሰናለች

Appendix G: Sentences used for testing the parser

1. ነጋዴው ገበሬው የሸጠለትን እህል ወደ ከተማ አመጣ
2. ሚኒስተሩ ወታደሮቹ ሀገራቸውን ከወራሪ ስለታደጉ በጣም አመሰገኑ
3. አበራ ወንድሙ የጋበዘውን ጎበዝ የኮሌጅ ተማሪ ተዋወቀ
4. ፕሬዘዳንቱ አርቲስቱ በፊልም የገለፀውን ብልህ ንጉስ አደነቁ
5. አባቴ ከስራ ሲመጣ ወንድሜ ማንበቢያ ከፍል ገባ
6. የኢትዮጵያ መንግስት በአሜሪካ የደረሰውን አደጋ አወገዘ
7. አባላቱ ያካሄዱት ውይይት ዴሞክራሲያዊ ነበር
8. በአዳማ በ6.8 ሚሊየን ብር ፕሮጀክቶች ተሰሩ
9. በትግራይ ክልል 2047 ተማሪዎች የ12ኛ ክፍል መልቀቂያ ፈተና ወሰዱ
10. የኢትዮጵያ የትምህርት ሽፋን በ6.4 በመቶ ማደጉ ተገለጸ
11. የትምህርት ግባትና ቁሳቁስ ተሟላ
12. ከ11ሚሊየን ብር በላይ ለተቀናሽ ሰራዊቶች ተከፋፈለ
13. ሀኪሙ ልጇ አስቸጋሪ የሆነውን ሴትዮ በቅድሚያ አስተናገደ
14. እኔ ካሳ የገዛውን ነጭ በግ ትላንት አየሁ
15. ቦርሳው ብዙ እቃ በውስጡ ስለያዘ በጣም ከበደኝ
16. በውጪ የሚኖሩ ኢትዮጵያዊያን 260ሺ ዶላር ለገሱ
17. ሻእቢያ ተጨማሪ ከፍተኛ ባለስልጣናትን አሰረ
18. ምክርቤቱ ከጣሊያን ተቋም ጋር በመተባበር የአቅም ግንባታ ስልጠና አዘጋጀ
19. እንግሊዝ በኤርትራ የሚፈጸመውን እስራት ተቃወመች
20. ለፖርክ የሚሆኑ ቦታዎች በቦረና ዞን ተከለሉ
21. ዘበኛው አሰሪው ስለፈቀደለት የማታ ትምህርት ተመዘገበ
22. የእርዳታ እህል ተከፋፈለ
23. ፖስታ ቤቱ ድንገተኛ ጭማሪ አደረገ
24. የድርጅቱ አባላት በጉባኤው ተደሰቱ
25. የኢፌዴሪ የካቢኔ አባላት ሹመት ጸደቀ
26. የኢትዮጵያ አየርመንገድ ወደ አሜሪካ ለመብረር ተፈቀደለት
27. ምክርቤቱ ሁለት አዋጆችን መረመረ
28. አልማ የህክምና መሳሪያዎችን አሰራጨ
29. የአለም የምግብ ቀን ነገ ይከበራል
30. ጤናጥበቃ የወባ በሽታን ለመከላከል እንቅስቃሴ ጀመረ

31. ጉሙሩክ 9.8ሚሊየን ብር ገቢ አደረገ
32. የኢትዮጵያ ምሽት በቶኪዮ በደማቅ ሁኔታ ተካሄደ
33. ኢትዮጵያ ና ጅቡቲ የንግድ ስምምነት ተፈራረሙ
34. ነብር በኢትዮጵያ የለም
35. የቴክኖሎጂ ሽግግር በኢትዮጵያ እየተጠናከረ መጣ
36. አልማ የህክምና መሳሪያዎችን አሰራጨ
37. እንስሶቹ እንደ አሳ ይዋኛሉ
38. የኢትዮጵያ ህዝብ መዝሙር ተዘመረ
39. ሰውየው በጣም ረጅም ነዉ
40. ልጁ ወደ ቤቱ ሮጠ
41. ኮከቦች በንጋት በጣም ያበራሉ
42. ሰውየው በጣም ረጅም ነዉ
43. የኢሃዲግ 4ኛ ድርጅታዊ ጉባኤ ማምሻውን ተከፈተ
44. የጤናጥበቃ ሚኒስቴር ሚኒስትር ማን ይባላሉ
45. የመጀመሪያው የስልክ አገልግሎት በኢትዮጵያ መቼ ተጀመረ
46. አክሊሉ ለማ መቼ ተወለዱ
47. አንድ ጋሎን ስንት ሊትር ነው
48. በአዲስ አበባ ከተማ ስንት ሙዚየሞች ይገኛሉ
49. የጢስ አባይ ፏፏቴ ርዝመቱ ምን ያህል ነው
50. የግብጽ ዋና ከተማ ማን ይባላል
51. የካልሃሪ ደሴት በየት አህጉር ይገኛል
52. የኢትዮጵያ ንግድ ባንክ መቼ ተመሰረተ
53. የምድር ባቡር ድርጅት መቼ ተቋቋመ
54. ሀረር የተመሰረተችው በማን ነው
55. የመጀመሪያው ፓኪስታናዊ የኖቤል ተሸላሚ ማን ይባላል
56. ሸንብራ ኩሬ ላይ የተካሄደው ጦርነት በማንና በማን መካከል ነበር
57. የዓለም ዋንጫ መቼ ተጀመረ
58. ልጆቹ ምሳ አልበሉም እንዴ
59. ኮለኔል መንግስቱ ሀይለ ማርያም መቼ ኮበለሉ
60. የአልጀሪያ ዋና ከተማ ማን ይባላል
61. በኢትዮጵያ የመጀመሪያው ትምህርት ቤት የት ተገነባ
62. በአለም ላይ ስንት ቋንቋዎች ይነገራሉ

63. ጨረቃ ከምድር በምን ያህል ርቀት ላይ ትገኛለች
64. የራስ ዳሽን ተራራ ርዝመቱ ምን ያህል ነው
65. አበበ ና ካሳ ለምን ተጣሉ
66. ሰራተኛዋ ዝናብ ያረጠበውን ልብስ ቤት ውስጥ አሰጣች
67. ከበደ ስለታመመ ትምህርት ቤት አልመጣም
68. ወታደሮቹ N ግብጣ N ተጠርተው V አልመጡም V
69. ካሳ ከ እናቱ ጋር ምሳ አልበላም
70. ከእናቱ ጋር ምሳ የበላው ካሳ አይደለም
71. ብዙ ህፃናት ያለረዳት ያድጋሉ
72. ብዙ ሰዎች ያለትዳር ይኖራሉ
73. እነዚያ ልጆች ትምህርትቤት አልሄዱም
74. ካሳ ያለገብስ ዳቦ አይበላም
75. ማንም ለማንም ምንም አይሰጥም
76. አለሙ ማንንም ሰው አያምንም
77. ልጆቹ እንደወጡ አልተመለሱም
78. ካሳ ቤት ውስጥ የለም
79. ነጋዴው ጨረታዉን አላሸነፈም
80. ማንም ሰው ወደ እኔ ሊመጣ አይችልም
81. ካሳ ለበዓል በግ አልገዛም
82. ተማሪዎች የቤትስራቸውን ሳይሰሩ መጡ
83. የቤት ስራዉን ለነገ ሰርታችሁ እንድትመጡ
84. ፈተናዉን በትእዛዙ መሰረት ብቻ ስሩ
85. ያዘዝኩህን ብቻ ሰርተህ ጠብቀኝ
86. አስተናጋጅ ሻይና ማኪያቶ አምጣ
87. ሰራተኛዋ አጽድታ ስትጨርስ ልብስ ትጠብ
88. ስራህን እንደጨረስክ በቶሎ ና
89. አያልነሽ እቃውን ታምጣ
90. አስቴር ወደ ትምህርት ቤት ትሂድ
91. ነገ ትምህርት እንደሌለ ለተማሪዎች ንገራቸው
92. መኪናዋን በቶሎ ጋራጅ ውሰድልኝ
93. ሃላፊውን አስፈርመሽ አምጭልኝ
94. የነገው ስብሰባ እንደተሰረዘ የሚገልጽ ማስታወቂያ ለጥፍ

95. አለሙን ለአስቸኳይ ስራ እንደምንፈልገው ንገረው
96. አሳይመንቱን በሰዓቱ አስገቡ
97. የሂሳብ ባለሙያዉን ባስቸኳይ ጥሪልኝ
98. የመስሪያቤቱ ሹፌር ትላንት ለምን እንደቀረ ጠይቁት
99. ያጋጠመህን ችግር ለያላፊው በደንብ አስረዳ
100. ከስራ እንደተመለስክ ከተለመደው ቦታ ጠብቀኝ

Declaration

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

Declared by:

Name: _____

Signature: _____

Date: _____

Confirmed by advisor:

Name: _____

Signature: _____

Date: _____