ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION STUDIES FOR AFRICA

AUTOMATIC SENTENCE PARSING FOR AMHARIC TEXT
AN EXPERIMENT USING PROBABILISTIC CONTEXT FREE GRAMMARS

A THESIS SUBMITED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCINECE IN INFORMATION SCIENCE

BY ATELACH ALEMU ARGAW

JUNE 2002

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION STUDIES FOR AFRICA

AUTOMATIC SENTENCE PARSING FOR AMHARIC TEXT

AN EXPERIMENT USING PROBABILISTIC CONTEXT FREE GRAMMARS

BY
ATELACH ALEMU ARGAW

Signature of the Board of Examiners for Approval

_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Declaration

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

Atelach Alemu Argaw
June 2002

The thesis has been submitted for examination with our approval as university advisors.

Ato Tesfaye Biru	Ato Mesfin Getachew	Dr. Abebe Gebretsadik

Acknowledgment

My foremost gratitude goes to my father, Ato Alemu Argaw, who is my inspiration, and my mother, Wro Tirunesh T/Giorgis, who is my idol. My greatest love and gratefulness goes to them for always being there for me, for loving me so unconditionally, and for having faith in me when I myself couldn't have it.

My special thanks goes to my brothers, Amard and Jad, my sister Kerry, and my friends for making my life prettier, and for making all those hardships I had to go through easier. Words seem so feeble to express my special gratitude for my brother, Amard, for his unwavering support, and above all, for being the greatest brother and my best friend. I couldn't have been able to make it without him.

My deepest gratitude goes also to my advisors: Ato Tesfaye Birru for his critical comments on my work, and for being my driving force throughout this project; Dr. Abebe G/Tsadik, for his enthusiasm, commitment, and constant support; And Ato Mesfin Getachew, for his constructive comments, understanding and eager help.

I would also thank my friends and colleagues at NeuroNet PLC, especially Ato Michael Tesfaye, Mr. Hardeep Sound, Ato Zelalem Bekele, and W/t Ayles Aberra for their understanding and support.

I am also indebted to all SISA Staff for the help they offered to me during my stay in the postgraduate school.

I am grateful to my classmates who also, are now my friends. Their smiles, laughter and unreserved help, and most of all, solidarity through the hard times, made my stay at the school more fun and all the hard work bearable.

Finally, I would like to give my gratitude to people who are not mentioned in name but whose effort helped me much all along.

I thank God for guiding me and helping me all the way through.

Abbreviations and symbols used

Symbols

( ) What is inside is optional except those used to indicate cite (source) of documents used

\ To separate words from their tags

Abbreviations

N	Noun in all forms
NP	A preposition not separated from a noun
NC	A conjunction not separated from a noun
NV	Verbal nouns
NB	Noun prefixed with balä
V	Verb in all forms except auxiliary, compound and all forms of auxiliary and compound verbs
AUX	Auxiliary verbs and all their other forms
VCO	**C**ompound verbs
VP	A preposition not separated from noun
VC	A verb prefixed or suffixed by a conjunction
J , Adj	An adjective
JC	A conjunction not separated from an adjective
JNU	A numeral used as an adjective
JPN	A noun not separated from a preposition and that function as an adjective
JP	An adjective not separated from a preposition
PREP	A preposition
ADV	An adverb
ADVC	An adverb not separated from a conjunction

C	A conjunction
REL	Relative clause
ITJ	Interjections
PP	Prepositional Phrase
NP	Noun Phrase
VP	Verb Phrase
AdjP	Adjectival Phrase
AdvP	Adverbial Phrase
NL	Natural Language
NLP	Natural Language Processing
NLU	Natural Language Understanding

Table of Contents

ACKNOWLEDGMENT ..... **I**
ABBREVIATIONS AND SYMBOLS USED ..... **III**
ABBREVIATIONS ..... **III**
TABLE OF CONTENTS ..... **V**
LIST OF FIGURES ..... **VIII**
LIST OF TABLES ..... **IX**
ABSTRACT ..... **X**
CHAPTER ONE **1**
1.1 BACKGROUND 1
1.2 STATEMENT OF THE PROBLEM AND IMPORTANCE 6
1.3 OBJECTIVES OF THE STUDY 11
1.3.1 General objectives 11
1.3.2 Specific objectives 11
1.4 METHOD 12
1.5 APPLICATION OF RESULTS AND BENEFICIARIES 13
1.6 LIMITATIONS OF THE STUDY 14
1.7 SCOPE OF THE STUDY 15
1.8 ORGANIZATION OF THE THESIS 15
CHAPTER TWO **17**
AUTOMATIC SENTENCE PARSING 17
2.1 INTRODUCTION 17
2.2 SENTENCES AND MEANING 17
2.3 GRAMMATICAL FORMALISMS 20
2.3.1 Context Free Grammars 21
2.3.3 Transition Network Grammars 22
2.3.3.1 Recursive Transition Network Grammars 22
2.3.3.2 Augmented Transition Networks 22
2.3.4 Context Sensitive Grammars 23
2.3.5 Unification-based Grammars 24
2.3.6 Probabilistic Context Free Grammars (PCFG) 25
2.3.7 Lexicon 27
2.4 APPROACHES TO PARSING 28
2.4.1 Rule Based Approaches 29
2.4.2 Encoding Structural Ambiguities 32
2.4.3 Statistical Parsing 33
2.4.3.1 Probabilistic Context-Free Grammars (PCFG) Parsing 34
2.4.3.2 Best-First Parsing 36
2.4.2.3 A Simple Context-Dependent Best-First Parser 37
CHAPTER THREE **39**

THE AMHARIC GRAMMAR 39
3.1. INTRODUCTION 39
3.2 THE AMHARIC ALPHABET AND PUNCTUATION MARKS 40
3.3 WORD CATEGORIES IN AMHARIC 40
3.3.1 The Noun Class 41
3.3.2 The verb class 41
3.3.3 The Adjective Class 42
3.3.4 Prepositions 42
3.3.5 The Adverb Class 44
3.3.6 Conjunctions 46
3.3.7 Numerals 47
3.3.8 Interjections 48
3.3.9 Specifiers, Modifiers and Complements 48
3.4 PHRASE STRUCTURE OF AMAHARIC 49
3.4.1 Noun Phrases 50
3.4.2 Verb Phrases 50
3.4.3 Adjectival Phrases 52
3.4.4 Prepositional Phrases 53
3.4.5 Adverbial Phrases 53
3.5 SENTENCES IN AMHARIC 54
3.5.1 Simple Declarative Sentences 57
3.5.2 Simple Interrogative Sentences 57
3.5.3 Simple Negative Sentences 58
3.5.4 Simple Imperative Sentences 59
3.6 CONCLUSION 59

CHAPTER FOUR 61

PROBABILISTIC CONTEXT FREE GRAMMAR EXTRACTION 61
4.1 INTRODUCTION 61
4.2 APPROACH SELECTED TO DESIGN THE SENTENCE PARSER 61
4.3 SAMPLE CORPUS 64
4.4 PART OF SPEECH TAGGER 65
4.5 FEATURES OF THE LANGUAGE CONSIDERED IN THE DESIGN 68
4.6 EXTRACTION OF A PROBABILISTIC CONTEXT FREE GRAMMAR 69
4.7 CONVERSION TO CNF 71
4.8 CONCLUSION 72

CHAPTER FIVE 73

PARSING ALGORITHM AND EXPERIMENTATION 73
5.1 INTRODUCTION 73
5.2 THE INSIDE-OUTSIDE ALGORITHM 73
5.3 PCFG PARSING 76
5.3.1 Parse Tree 76
5.3.2 Parse Chart 78
5.4 THE CONFIGURATION OF THE PARSER 79
5.5 EXPERIMENTATION 87
5.5.1 Findings 87

5.5.2 Solution to Identified Problems 91
CHAPTER SIX 94
CONCLUSION AND RECOMMENDATIONS 94
6.1 CONCLUSION 94
6.2 RECOMMENDATION 99
REFERENCES 101
APPENDICES 106
APPENDIX 1. THE AMHARIC ALPHABET, EXTRACTED FROM LESLAU (1976) 106
APPENDIX 2. LIST OF PUNCTUATION MARKS IN AMHARIC 108
APPENDIX 3. TRANSCRIBED SAMPLE SENTENCES 109
APPENDIX 4. SPECIAL TAGS IDENTIFIED BY MESFIN (2001) 115
APPENDIX 5. TAGGED CORPUS 117
APPENDIX 6. EXTRACTED PROBABILISTIC CONTEXT FREE GRAMMAR 123
APPENDIX 7. PARSES 124
APPENDIX 8. THE CODE WRITTEN IN VB 6.0. 134

List of Figures

Figure 2.1. A probable structure of the sentence “Kassa wädä 35
Figure 3.1 A parse structure for the sentence. “And t*ə*l*ə*k l*ə*j bä-hulät b*ə*r arat 56
Figure 5.1 The words outside and inside $N^{j}_{k,l}$ 74
Figure 5.2 Algorithm to Calculate the Inside Probabilities 75
Figure 5.3 Parse tree space of 4-word sentences 77
Figure 5.4 4-level-chart 79
Figure 5.5 A Tree Structure for the parse of the sentence 81
Figure 5.6 PCFG representation 82
Figure 5.7 Bottom Up Chart Parsing Algorithm 83
Figure 5.8 The Arc Extension Algorithm 83
Figure 5.9 Modified Algorithm 84
Figure 5.10 Chart Structure During Parsing 84
Figure 5.11 A complete POS Matrix 85
Figure 5.12 Diagrammatic Representation of the Parser 86
Figure 5.13 Algorithm to Guess Unknown Word Categories 89

List of Tables

Table 2.1 Probabilistic Context Free Grammar 26
Table 2.2 Some rule estimates based on the first word..... 37
Table 4.1 Probabilistic Context Free Grammars..... 70
Table 4.2 Probabilistic Context Free Grammars in CNF 72

Abstract

Natural Language processing, as a field of scientific inquiry, plays an important role in increasing computers capability to understand natural languages, the language by which most human knowledge is recorded. Works in the area of Natural Language Processing try to design and implement computer programs that can understand natural language and act appropriately on the information contained in the text or utterance. Enabling computers to understand natural language involves extraction of meaning from natural language sentences. And one of the steps in this process is sentence parsing.

Sentence parsing, which is also called syntactic parsing, is the process of identifying how words can be put together to form correct sentences and determining what structural role each word plays in the sentence and what phrases are subparts of what other phrases. A sentence parser outputs a parse structure that could be used as a component in many applications including semantic analysis, machine translation, information storage and retrieval of textual data etc.

Today, parsers of different kinds (e.g. probabilistic, rule based) have been developed for languages, which have relatively wider use nationally and/or internationally (e .g. English, German, Chinese, etc). The same story is not true for Amharic, the working language of the Federal Government of Ethiopia, and one of the major languages of Ethiopia (Bender et al, 1976) since to the best of my knowledge, there are no sentence parsers of any sort that process this language.

This study, thus, attempted to develop a simple automatic parser for Amharic texts/sentences to address the need for developing systems that automatically process the Amharic language.

In the study, the Inside Outside algorithm with a bottom up chart parsing strategy has been used. The probabilistic context free grammar has been used as a grammatical formalism to represent the phrase structure rules of the language. A small sample corpus was selected from sentences in the language, and has been used to serve as a training and test set. The sample was then hand parsed, automatically tagged, and was used as a corpus to extract the grammar rules and assign probabilities.

The thesis, in short, describes processes of automatic sentence parsing using a combination of probabilistic and rule-based reasoning. It describes the whole process from manually parsing simple sentences to developing a prototype and conducting an experiment with it. The results obtained using the small manually parsed corpus seems to encourage further research to be launched, especially with the aim of developing a full-fledged Amharic sentence parser.

Chapter One

1.1 Background

Natural language is one of the fundamental aspects of human behavior and is a crucial component in our lives. It is a tool for communicating about the world. Natural language processing can be described as the ability of computers to generate and interpret natural language (Lenat et al, 1990). Natural language processing (NLP), is widely regarded as a promising and critically important endeavor in the field of computer research. According to Lenat et al (1990), the goal of computational natural language processing is two fold: to create computational representations of the relationships that hold between language and some computational model (a knowledge base or a database schema) of the world; and to exploit those relationships to understand and generate language as appropriate to some set of tasks. The general goal for most computational linguists is to imbue the computer with the ability to understand and generate natural language so that eventually people can address their computers through text and speech as though they were addressing another person.

One of the reasons that have created a great interest in NLP is the fact that most human knowledge is recorded using natural language. Only computers that have the capability to understand natural language can access all the information contained in the natural language efficiently. The applications that will be possible when NLP capabilities are fully realized are impressive--computers would be able to understand and process natural language, translate languages accurately and in real time, or extract and

summarize information from a variety of data sources, depending on the users' requests. (Grishman, 1994)

Natural language has sentences as constituents. A sentence expresses a complete thought and is made up of words, which individually or in combination with other words represent ideas. Sentences can in general be broken down into discrete prepositional segments that through their very form and organization contribute to the natural language understanding process (Harris, 1984).

There are several different levels of language processing: phonological, morphological, syntactic, semantic, discourse integration, and pragmatic levels. The syntactic level is designed to group the words of a sentence into structural units such as prepositional phrases, and subject-object-verb groupings that collectively represent the grammatical structure of the sentence. A syntactic analysis is usually based on the surrounding structure in which the individual words are embedded in a sentence and on the use of syntactic features characterizing the individual words.

The overall objective of sentence analysis, which is one of the foremost tasks to be accomplished in natural language processing, is to determine the meaning of a sentence (Grishman,1994). In practice, this involves translating the natural language input into a language with a simple semantics, which seeks to determine the conditions under which a sentence is true (e.g. a formal logic) or into a language, which can be interpreted by an existing computer system. In most systems, the first stage of this

translation is syntax analysis – the determination (and possible regularization) of a sentence structure (Grishman, 1994).

The syntactic structure of a sentence indicates the way that words in the sentence are related to each other. It also indicates how the words are grouped together into phrases, what words modify other words, and what words are of central importance in the sentence (Allen, 1995).

These syntactic structures are commonly represented by means of sentence diagrams, where the syntaxes are encoded as a list structure or a parse tree (Shutzer, 1987). Parse trees are trees in which the nodes correspond to sentence constituents such as phrases (Rauch-Hindin, 1986). Parsing is the process of going from words to syntactic structures. Therefore, it can also be described as a method of analyzing the various parts of a string to determine whether or not the string is a sentence in the language (Trembley and Sorenson, 1976). In general, sentence parsing is a task of NLP. It is a method that deals with the decomposition of a sentence into its major subparts, namely noun phrase (NP), verb phrase (VP), noun (N), verb (V), etc (Mesfin, 2001). In addition to this, parsing deals with a number of sub problems such as identifying constituents that can fit together, testing the compatibility of number and tense etc (Metzer et al, 1989).

In general, a sentence parsed with a sentence parser:

- Indicates how words in a sentence are related to each other
- Can be checked for well-formed-ness

- Helps to understand how words are put together to form correct sentences
- Helps us to determine what structural role each word plays in a sentence
- Helps to see what phrases are sub parts of other phrases, etc.

Syntactic processing (i.e. parsing) plays an important role in many natural language understanding systems for two main reasons.

1. Semantic processing must operate on sentence constituents. If there is no syntactic parser, then the semantic system must decide on its own constituents. On the other hand, if parsing is done, i.e. if a parser is used as a component, it constrains the number of constituents that a semantic parser can consider. Syntactic parsing is computationally less expensive than is semantic processing (which may require substantial inference). Thus it can play a significant role in reducing overall system complexity.
2. Although it is often possible to extract the meaning of a sentence without using syntactic analysis and grammatical facts, it is not always possible to do so. (Rich and Knight, 1991).

In the actual development of parsers for syntactic analysis, it is standard practice to posit two working levels: the grammar, on one hand, and the algorithms which produce the analysis of the sentence by using the grammar as the source of syntactic knowledge on the other hand (Merlo, 1996). Information can be stored in a grammar in many different ways. Grammars are represented by different formalisms such as the context

Deleted: .

Deleted: (Merlo, 1996)

free grammar, transition network grammars, context sensitive grammar, unification-based grammar etc.

A parsing algorithm/technique can be described as a procedure that searches through the various ways of combining grammatical rules to find a combination that generates a tree that could be the structure of the input sentence. The process of parsing can also be considered as a special case of a search problem as defined in Artificial Intelligence (Allen, 1995). Basically there are three parsing techniques that are very much related to the search algorithms in AI: Top down parsing (depth first search), bottom up parsing (breadth first search), and chart parsing (somewhat like the A* search). Details on these search techniques could be found in Rich and Knight, (1991).

A major problem in parsing is that of “syntactic ambiguity”, which arises when the parser encounters sentences with two or more parses. In such cases, it is necessary for the parser (or the understanding system in which the parser is embedded) to choose the correct one among the possible parses (Charniak, 1997). Based on reasoning techniques, parsers can be divided into two, rule-based and statistical. In parsing, rule based parsers employ pure rule reasoning driven by grammar rules, while statistical parsers employ probabilistic reasoning also driven by grammar rules but associated with statistical probabilities. In order to address the problem of structural ambiguities, rule based parsers use heuristics, while statistical parsers use probabilistic reasoning to select the best parse.

Statistical parsers work by assigning probabilities to possible parses of a sentence, locating the most probable parse, and then presenting that parse as the answer. Thus, to construct a statistical parser, three major tasks are essential:

- Find all possible parses to an input sentence,
- Assign probabilities to the possible parses, and
- Pull out the most probable one (Charniak, 1997).

In this study, a statistical parsing technique was employed, and more emphasis is given on these techniques throughout the report.

1.2 Statement of the problem and Importance

A great deal of research has been conducted on NLP, and currently, there are systems that extract syntactic and semantic information from natural language input texts for languages such as English, Chinese, German, Finish, etc. This is not the case for Amharic, the official language of Ethiopia.

Ethiopia is a multi lingual country with over 80 distinct languages (Bender et. al, 1976), and with a population of more than 59.9 million as authorities estimated on the basis of the 1994 census. Amharic being the official language of Ethiopia is spoken by a substantial segment of the population. In the 1994 census, 17.4 million people claimed Amharic as their first language and 5.1 as their second language. Owing to political and social conditions and the multiplicity of the languages, Amharic has gained ground through out the country. Amharic is used in business, government, and education. Newspapers are printed in Amharic as are numerous books on all subjects (Leslau,

1968). Amharic is the official government language and more and more materials are being published in Amharic currently.

There are a number of Amharic word processing software available in the market. From my observation, the most widely used are the Power Geez, Visual Geez and Samawerfa. None of these, or any of the other Amharic software, supports language specific utilities like spell check, grammar, thesaurus, etc. It is my observation following discussions with colleagues that the absence of these word processing tools for Amharic language has made word processing activities using the language incomplete.

The enormous amount of information on the Internet could be used to enhance development by making it accessible to the public. To fully localize and utilize these resources which are available on the Internet, translation of documents from one language to another may be necessary. For example, many documents on the Internet are written in English and English to Amharic translation and vice versa may be required. Machine translation, which uses natural language as an input, and sentence parsers as a component, plays a great role in solving the translation problem. And hence, the need to develop an Amharic sentence parser.

Currently, a research is being conducted on Machine Translation of the Amharic language, by Sisay Fisseha. He is working on a prototype system which is rule-based. The basic underlining computational mechanism is unification and all linguistic information should be hand-coded. It does not make use of any statistical methods. That means in order to parse a sentence, he must write detailed rules using the formalism

provided by the system. The system then compiles the rules and analyses sentences based on the rules. One of the shortcomings of such system is that of resolving ambiguity. It is not possible to anticipate in advance all possible analysis of a sentence and write disambiguation rules. The absence of a parser for the language, which can resolve ambiguities, has been a difficulty for the research on Machine Translation[1].

The development of a sentence parser by itself would not solve the need for NLP systems for Amharic. A lot of research in the area such as POS tagger, morphological analyzer, semantic parser, etc, is crucial. So far, a limited number of researches have been conducted in the area of natural language processing for Amharic. Notable among these are the following: an Amharic word parser by Abiyot (2000), and a part of speech tagger by Mesfin (2001). As these researches are initial attempts, both tried to address the issues that concern the need to develop natural language processing systems for the language. According to my review, both have done an encouraging and promising work in NLP of Amharic. Abiyot has tried to design and implement a word parser for Amharic verbs and their derivation. He designed a knowledge-based system that parses verbs, and nouns derived from verbs. He used root pattern and affixes to determine the lexical and inflectional category of the words. The study did not include the property of words at syntactic level. He experimented on a limited number of words (200 verbs and 200 nouns). The result showed that 86% of the verbs and 84% of the nouns were recognized correctly. Mesfin, on the other hand, tried to develop a part of speech tagger using the stochastic Hidden Markov Model approach. Part of Speech (POS) tagging is

[1] This information is obtained from the researcher, Sisay Fisseha, through personal communication.

the task of assigning an appropriate part of speech word category to each word in a sentence and is usually taken to be the first step in automatically processing language at the sentence level. Its output is crucial for tasks such as sentence parsing and word sense disambiguation (distinguishing words with the same spelling and/or pronunciation) (Mao, 1997). Mesfin's tagger extracts major word classes (Noun, Verb, Adjective, auxiliary, …etc) and classes that are unique to the language. Mesfin's tagger does not support subcategory acquisition or constraints such as number, gender, polarity, tense case, and definiteness. And it does not have a mechanism to estimate or guess POS tags for unknown words, it assigns "UNC" for such words. UNC here stands for unknown category. The sample corpus he used is a one page long Amharic text.

Another important research has been conducted by Tesfaye Bayu (2002) on Morphological analysis of the language. In his study, he used two separate systems in order to develop the morphological system. The first system applies an unsupervised learning approach based on probabilistic models to extract morphemic components (prefix, stem and suffix) and construct a morphological dictionary. The second system, developed applying the principle of Auto segmental Phonology, was used to identify morphemic component of a stem such as consonantal root, vocalic melodies and CV-templates. And the test result showed that the first system was able to parse successfully 87% of words of the test data (433 of 500 words). This result corresponds to a precision of 95% and a recall of 90%. Tested with 255 stems, the second system has also identified the morphemic compotes of 241 (or 94% of the) stems correctly.

To the best of my knowledge, there is no Amharic sentence parser designed or developed so far. The absence of a syntactic parser for Amharic limits higher forms of NLP for this language. This includes semantic analysis, discourse integration, machine translation, spell checking and grammar, etc among many others.

Taking the above-mentioned issues into consideration, it is considered necessary to conduct a research and develop an automatic sentence parser for Amharic language based on the phrase structure and property of the Amharic word classes. In doing this, it is believed that the work would contribute to the research and the attempt already made in the NLP of the Amharic language.

Deleted: research

Deleted: ¶

Deleted: T

The major concern of this study is to contribute to the research in NLP of Amharic, by developing a syntactic analyzer (i.e. sentence parser). The approach followed in the design and development of the parser is one that combines rule based and statistical techniques. This sort of statistical NLP applications require a large volume of data such as hand tagged and hand parsed corpus. Such corpus is currently made available for many natural languages (for instance, for English). But there is no such corpus available for the Amharic language and studies of this kind are believed to contribute to the initiation of compiling and producing the corpus mentioned above.

Deleted: parser

1.3 Objectives of the Study

1.3.1 General objectives

The general objective of the research is to design and develop a prototype automatic sentence parser for Amharic texts (sentences).

1.3.2 Specific objectives

In line with achieving the general objective stated above, the research would attempt to address the following specific objectives.

1. Review the basic word categories, morphological property, phrase structure, and sentences of the Amharic language to identify properties useful for automatic sentence parsing.
2. Select sample sentences that would potentially serve for the experiment, for the preparation of the required corpus.
3. Generate the grammar rules appropriate for the language.
4. Generate lexical probabilities for the words in the sample text, and estimate the probabilities of the grammar rules extracted.
5. Enhance the POS tagger developed by Mesfin (2001) using a larger corpus with the aim of obtaining a more accurate POS and by adding a feature that can handle unknown words by using bi-gram analysis.
6. Prepare the lexicon, grammar and probability values database using Access 2000, for use in the development of the parser.
7. Select and customize, or develop a parsing algorithm for use with the Amharic language.

8. Develop a prototype Amharic "Simple-Sentence" Parser and design an interface for it, using Visual Basic 6.0.
9. Test the prototype parser developed.

Deleted: ¶
¶

1.4 Method

Developing an automatic sentence parser for Amharic language requires one to investigate and identify the properties of the language. For this purpose, a **review** of related literature was made in the area of Amharic sentences, phrases, and word classes. Literatures in the area of parsing were also reviewed for this study.

Discussion with linguists and experts in the area of Amharic language were made to better understand the phrase structure of the language and get suggestions that are invaluable for the study.

Deleted: is

A sample data was selected from texts in the language, from which, the phrase structure rules were extracted and the training is conducted. The sampling technique used is **Judgmental Sampling technique**, i.e. samples are selected on the basis of the researcher's knowledge about the language, and the nature of the research aims.

Deleted: Based on the analysis of the Amharic language,

Deleted: was

The Inside outside algorithm and the bottom up chart-parsing algorithm were selected and used with slight modification. A **lexicon, probabilistic grammar rules** and **statistical databases** were designed and implemented using Access 2000 and a

prototype was developed to parse simple sentences appropriately. To implement the parsing algorithm, Visual Basic 6.0 was used.

The prototype developed was **tested** using a small sample text selected for the test purpose. The evaluation of the performance of the parser was made based on only the percentage of correct parses of input sentences, by comparing with the hand parsed counterpart and by counting manually. The figures obtained were summarized and analyzed by using statistical techniques to show the level of accuracy. The experiment was conducted in two phases, on training set and test set, and was repeated several times until the parser performance was found to be satisfactory. Test results achieved iteratively are finally reported together with the discussions made.

1.5 Application of Results And Beneficiaries

As outlined in the statement of the problem, parsing systems are useful in many areas of NLP for Amharic language. Thus, the beneficiaries of the results of this study include researchers involved (or want to be involved) in increasing computers capability of processing Amharic Language.

In general, potential applications of the Amharic sentence parser include the following.

Deleted: to be developed

- It can be used as a component, if made full fledged, in higher forms of NLP of the language such as semantic parser, machine translation, spell checking and grammar, friendly and flexible user interfaces, information storage and retrieval of textual data, automatic abstracting, etc.

- To check a sentence's well-formed-ness (i.e. how grammatical a sentence is in a language)
- In language teaching
- To further explore the syntactic structure of the language
- To automatically produce syntactically annotated texts for use in statistical natural language processing of Amharic
- To provide a more user-friendly system for end users, etc.

1.6 Limitations of The Study

1. This study uses a small sample prepared for the purpose of the work due to a lack of large corpora in the language annotated with POS and parsed (structurally analyzed). Generating such large corpora is very expensive and time consuming. Although it is essential to produce such corpus, it was not generated for the purpose of this study due to time limitations and financial constraints. It is also beyond the scope of the work.
2. The prototype developed in the study parses only 4-word Amharic sentences, again due mainly to time constraint and unavailability of processed data needed.
3. This study does not incorporate lexical and morphological properties of the language (such as inflections) in structural ambiguity resolution. It uses purely statistical techniques. This is a preference made by the researcher.

4. The texts used are transcribed (i.e. the Amharic “fidel” is not used). There is no standard coding scheme yet for the Amharic writing system yet. This unavailability has made the direct implementation of the Amharic fidel difficult.

1.7 Scope of the Study

The scope of the thesis is limited to demonstrate the potential of using grammatical constructions and probabilistic reasoning in developing a simple Automatic sentence parser for Amharic texts. Given the time available, this scope seems reasonable.

1.8 Organization of the Thesis

This section describes the organization of the rest of the thesis. Chapter two discusses different techniques and approaches to sentence parsing, with a particular emphasis on the selected approach in this study. Linguistic features of natural language sentences in conveying meaning, and representation of grammar rules and lexicon in developing parsers are also topics discussed in this chapter.

Deleted: Review of related works that have implemented the selected parsing approach and works done so far on natural language processing of Amharic are included in this chapter.

The third chapter briefly describes the Amharic grammar. Included in the discussion are the different word classes, phrase structures and simple sentences in the language.

The fourth chapter, which is the core of this thesis, discusses the approaches taken to design the parser and the features of the language considered in designing the parser. It also discusses the part of speech tagger developed by Mesfin and how it was enhanced

and embedded in the design and development of the prototype parser. Extraction of grammar rules and probabilistic data, and the design and structures of the different databases implemented for representing the data required by the sentence parser are discussed in this chapter as well.

The lexicon, the probabilistic grammar, and the algorithms used to develop the automatic parser and the computations made to get the lexical and phrase structure probability matrixes are presented in chapter five, which is also another core chapter in this thesis. The database designed to hold the knowledge base, the data structure implemented, the experiments conducted, and the evaluation procedures and the results achieved are also presented in this chapter.

Deleted: made

Chapter six, the last chapter, presents conclusions and recommendations based on the findings of the study. This chapter will also indicate some pointers to future works. A reference list to be used for further reading is also included at the end of this chapter. The appendices attached at the end provide additional information on some of the topics discussed in the different parts of the thesis and are found at the end of the last chapter.

Chapter Two

Automatic Sentence Parsing

Deleted: ¶
¶

2.1 Introduction

As already indicated in the first chapter, the main objective of the study is to design and implement an Amharic sentence parser. A sentence parser for a natural language is a program that diagrams sentences of that language. That is, it supplies for a given sentence a correct grammatical analysis, demarcating its parts (called constituents), labeling each, identifying the part of speech of every word used in the sentence, and usually offering additional information, such as the *semantic class* (e.g., Person, Physical Object) of each word and the *functional class* (e.g., Subject, Direct Object) of each constituent of the sentence. (Black, NY)

In this chapter different techniques and approaches to sentence parsing are discussed. The first part discusses the linguistic features of natural language sentences in conveying meaning. The second part is about the representation of grammar rules and lexicon in developing parsers, & the last one, discusses about different approaches to sentence parsing with a particular emphasis on the selected approach in this study.

Deleted: third part

Deleted: , and the last part, about related works that have implemented the selected parsing approach and works done so far on natural language processing of Amharic.¶

2.2 Sentences and Meaning

A natural language system must use considerable knowledge about the structure of the language itself, including what the words are, how words are combined to form sentences, what the words mean, how word meanings contribute to sentence meanings and so on (Allen, 95). Meanings in a sentence can be represented using different

linguistic units and syntactic structures. There must be a mechanism to capture and represent these rules and units, so that the computer can recognize the meaning of the text. Sentence parsing is one of the major activities towards this end for it performs an analysis of the natural language utterance or sentence and resolves ambiguities to the extent possible using various sources of linguistic and other statistical information. The major purpose of parsing in general and sentence parsing in particular is extracting structural and semantic information from the input text (Abiyot, 2000).

It would not be possible to model or parse sentences without mechanisms to compute the properties of larger constituents from the properties of their parts. That is, the properties of new sentences, which are unlikely to have been seen before, can only be inferred from knowledge of how their parts participate in the sentences. While this point may seem obvious, it has deep consequences in parsing. Any parser must include a generative mechanism or *grammar* that specifies how sentences are built from their parts, and how the information associated to the sentence derives from the information associated to its parts. Furthermore, to be able to cope with previously unseen sentences, any such system must involve *generalization* with respect to the data from which the language model or parser was developed (Pereira, NY).

The linguistic organization of a natural language processing system includes features such as the following (Meyers et al, NY):

- **Grammar and lexicon** - the rules for forming well-structured sentences, and the words that make up the sentences

- **Morphology** - the formation of words from stems, prefixes, and suffixes.

  Example: DägƏ + nät + e = dägƏnäte

  Generous Suffix Suffix

- **Syntax** - the set of all well-formed sentences in a language and the rules for forming them

- **Semantics** - the meanings of all well-formed sentences in a language

- **Pragmatics** (world knowledge and context) - the influence of what we know about the real world upon the meaning of a sentence. e.g., " Fiğnaw tänƏsafäfä"

  (The balloon rose) allows an inference to be made that it must be filled with a lighter-than-air substance.

- The influence of **discourse** context (e.g., speaker-hearer roles in a conversation) on the meaning of a sentence.

- **Ambiguity**
  - Lexical - word meaning choices
  - Syntactic - sentence structure choices
  - Semantic - sentence-meaning choices.

Although all of the above mentioned tasks are crucial in any language understanding system, this study focuses on the development of grammars and lexicon, syntactic

analysis, and lexical and syntactic ambiguity resolution, in developing a parser for the Amharic text/sentences.

2.3 Grammatical Formalisms

A grammar can be defined as a description language plus a set of structural constraints - according to which a parser attempts to analyze the symbol sequences presented to it. Put another way, a parser accepts as input a sequence of words (or their surrogates) in some language and an abstract description of possible structural relations that may hold between words or sequences of words in the language, and produces as output zero or more structural descriptions of the input as permitted by the structural rule set or the grammatical formalism (Merlo, 1996).

Grammar specifies two things:

- A grammar's weak generative capacity: the set of grammatically correct sentences that are contained within the language
- A grammar's strong generative capacity: the structure to be assigned to each grammatical sentence in the language.

Generally speaking, the motivation for parsing is the belief that grammatical structure contributes to meaning and that discovering the grammatical structure of a NL word sequence is a necessary step in determining the meaning of the sentence (Merlo, 1996). In some parsers the construction of a meaning representation is carried out in parallel with the derivation of a structural analysis according to the grammar.

There are different grammatical formalisms proposed so far. Among them the most commonly and widely used formalisms are Context Free Grammar (Chomsky, 1957), Transformational Grammar (Chomsky, 1965; Radford, 1981), Transition Network Grammars (Woods, 1970), Unification Based Grammar (Kay, 1982) and Probabilistic Context Free Grammars (Charnaik, 1993). Each of these formalisms is briefly discussed in the remainder of this section.

2.3.1 Context Free Grammars

A *context-free grammar* (CFG) is a formal system that describes a language by specifying how any legal text can be derived from a distinguished symbol called the *axiom*, or *sentence symbol*. It consists of a set of *productions*, each of which states that a given symbol can be replaced by a given sequence of symbols. Each production of a context-free grammar consists of a symbol to be replaced and the sequence that replaces it. Symbols that are to be replaced are called *non-terminals*, and are always represented by *identifiers*[2]. Every non-terminal must appear before a colon in at least one production. The axiom is a non-terminal that appears before the colon in exactly one production, and does not appear between the colon and the period in any production. There must be exactly one non-terminal satisfying the conditions for the axiom. Symbols that cannot be replaced are called *terminals*, and may be represented by either identifiers or *literals*[3].

---

[2] An identifier is a sequence of letters and digits, the first of which is a letter.

[3] A literal is a sequence of characters bounded by apostrophes ('). An apostrophe appearing within a literal is represented by two successive apostrophes.

CFGs are a very important class of grammars for two reasons: The formalism is powerful enough to describe most of the structure in natural languages, yet it is restricted enough so that efficient parsers can be built to analyze sentences (Allen, 95). An example of a CFG is given below.

S → NP VP

NP → N N

NP→ AdjP N

2.3.3 Transition Network Grammars

2.3.3.1 Recursive Transition Network Grammars

This grammatical formalism is based on the notion of a transition network consisting of nodes and labeled- arcs. One of the nodes is specified as the initial state, or start state. Starting at the initial state, an arc could be traversed if the current word in the sentence is in the category on the arc. Simple transition networks are often called finite state machines(FSMs). FSMs are equivalent in expressive power to regular grammars, and thus are not powerful enough to describe all languages that can be described by a CFG. To get the descriptive power of CFGs, a notion of recursion in the network grammar is needed. A Recursive Transition Network (RTN) is like a simple transition network, except that it allows arc labels to refer to other networks as well as word categories (Allen, 95).

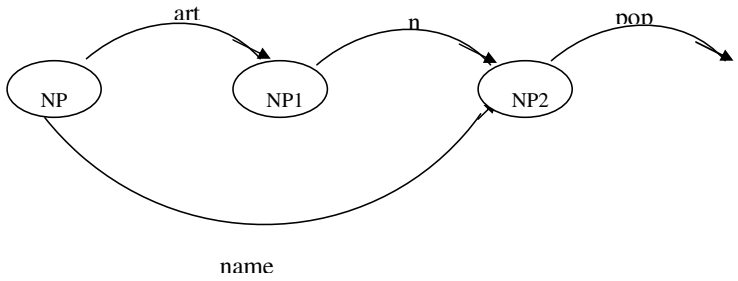
2.3.3.2 Augmented Transition Networks

The Augmented transition network (ATN) formalism introduced by Woods in 1970 has become one of the most popular forms for writing natural language grammars

(Grishman, 1994). A transition network is a representation of regular (or finite-state) grammar. The network is a directed graph whose arcs are labeled by terminal symbols (words or word categories). One node of the graph is designated as the start state; one or more nodes are marked as final states. A sentence is in the language defined by the network 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.

Deleted: to some

Example:



2.3.4 Context Sensitive Grammars

A context sensitive grammar is a phrase structure grammar which satisfies the following constraint: For every production of the form

X -> y

The length of y (i.e. the number of symbols in y) is greater than or equal to the length of x. An alternative notation is sometimes used for context-sensitive grammars. In this notation, each production is of the form

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 z are sequences of zero or more terminal and non-terminal symbols. The meaning of this production 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 (Grishman, 1994).

Example:

(VP) → (V SUBCAT _np_vp:inf)

(NP)

(VP VFORM inf)

This says that a VP can consist of a V with SUBCAT value _np_vp:inf, followed by an NP, followed by a VP with VFORM value inf.

2.3.5 Unification-based Grammars

Almost all computational grammars incorporate feature structures (attribute value structures), the category label being a special attribute singled out for linguistic convenience and not for any formal reasons. These feature structures are manipulated by the operation of unification (i.e. the entire grammar can be specified as a set of constraints between feature structures), hence the term *unification-based grammars*. CFGs or any of the grammars mentioned above can serve as the backbones for the unification-based grammars, CFGs being the most common. As soon as feature structures and unification are added to a CFG, the resulting grammars are no longer polynomially parsable. In practice, conditions are often placed on the possible feature structures which allow the polynomial probability to be restored. The main reason for the excessive power of the unification-based grammars is that recursion can be encoded in the feature structures (Joshi, NY).

Example: X0 → X1 X2

CAT_0 = S
CAT_1 = NP
CAT_2 = VP
$AGR_0 = AGR_1 = AGR_2$
$VFPRM_0 = VFORM_2$

2.3.6 Probabilistic Context Free Grammars (PCFG)

The PCFGs are context-free grammars (CFGs) with associated probabilities. A PCFG sentence is determined by CFG rules and statistical properties of the rules. Compared with the CFG, the PCFG is especially useful for sentence parsing in the case of syntactic ambiguity, ungrammatical sentence analysis, and grammar learning. It gives a better probabilistic model for syntax analysis for statistical properties of natural languages are introduced (Yao and Lua, 1998).

A probabilistic context-free grammar (PCFG) is a four-tuple $\langle W, N, S, R \rangle$, where $W = \{w^1, w^2, ..., w^u\}$ is a set of terminal symbols like words in a sentence, $N = \{N^1, N^2, ..., N^v\}$ is a set of non-terminal symbols, $S = \{N^1\}$ is a set that only has one starting symbol, and $R = \{R^1, R^2, ..., R^w\}$ is a set of grammar rules with probabilities. For a rule $R^m \in R$, it is a context-free grammar (CFG) rule with the form $R^m : N^i \rightarrow \xi^j$, and its probability $P(R^m) = P(N^i \rightarrow \xi^j)$(Yao and Lua, 1998).

The simplest approach to generalize context free grammars is to count the number of times each rule is used in a corpus containing parsed sentences and use this to estimate the probability of each rule being used. For instance, consider category C, where the grammar contains m rules Suppose that there are *m* rules R_1, ..., R_m with left-hand side C. The probability of using rule R_j to derive C can be estimated by

$Pr(R_j \mid C)$ = count(#times R_j used) / Sum i=1,m (#times R_i used) …. (1)

The following grammar (for English), adopted from Allen, 95, shows a probabilistic CFG based on a parsed version of an example corpus:

	Rule	Count for LHS	Count for Rule	Probability
1.	S -> NP VP	300	300	1
2.	VP -> V	300	116	.386
3.	VP -> V NP	300	118	.393
4.	VP -> V NP PP	300	66	.22
5.	NP -> NP PP	1023	241	.24
6.	NP -> N N	1023	92	.09
7.	NP -> N	1023	141	.14
8.	NP -> ART N	1023	558	.55
9.	PP -> P NP	307	307	1

Table 2.1 Probabilistic Context Free Grammar

The formalism can be based on the probability that a constituent C generates a sequence of words w_i, w_{i+1}, ... , w_j, written as $w_{i,j}$. This probability is called the **inside probability** and written $Pr(w_{i,j} \mid C)$. For example, Pr(flower|N) is the inside probability that constituent N is realized as the word flower. (Allen, 1995).

Although PCFGs are advantageous in many aspects such as in structural ambiguity resolution, PCFGs do not, however, take context or lexical co-occurrence into consideration. This means that sometimes some of the parses based on a PCFG may not be the best, because they will have based their plausibility purely on syntactic structures.

2.3.7 Lexicon

A structure called lexicon is used to efficiently store the possible categories for each word in a language. With a lexicon specified, a grammar need not contain any lexical rules of the form N → flower, where N is the POS category for the word flower (Allen, 95).

A *lexicon* is minimally a list of words that associates each word with its syntactic properties. The most important of these properties is its (gross) syntactic category, whether the word is a verb, a noun, an adjective, and so on. In addition, depending on the sophistication of the overall grammar, the lexicon will contain information as to the subcategory of the word (such as, whether a particular verb is transitive or intransitive), other syntactic properties (such as, the gender of a noun in a language that makes gender distinctions), and perhaps also morphological and semantic information (Gazdar and Mellish, 1996).

A simple context free grammar lexicon is shown below as an example.

N → säw

V → mättä

Adj → tƟlƟk

Adv → tolo

P → wädä

Where the left hand sides are POS categories for the words on the right hand sides.

2.4 Approaches to Parsing

As indicated earlier parsing algorithm can be described as a procedure that searches through various ways of combining grammatical rules to find a combination that generates a tree that could be the structure of the input sentence (Allen, 95).

During the past few decades, several methods of syntax analysis[4] were proposed. Among them are: the Generalized Phrase Structure Grammar (GPSG) parsing (Fisher, 1989), the Head-driven Phrase Structure Grammar (HPSG) parsing (J. Lee et al, 1992), the Cascaded Morphological parsing (W. J. Ketty, 1993), the Unification-Based Grammar parsing, the Probabilistic Context-Free Grammar (PCFG) parsing (Charnaik, 1993), etc. Considering their parsing or reasoning mechanism, we can roughly classify the above methods of sentence parsing into two types: (a) pure rule reasoning driven by grammar rules (rule based approach); (b) probabilistic reasoning also driven by grammar rules but associated with statistical probabilities. For pure rule reasoning, there are three basic parsing techniques (Allen, 1995) which are: the top-down parsing technique, the bottom-up parsing method, and the mixed-mode strategy. For probabilistic reasoning, statistical probabilities are introduced to assist. That is, linguistic specifications and statistical regularities of syntax are combined to be used for better syntax analysis.

[4] Determining the structure of sentences

2.4.1 Rule Based Approaches

In parsing sentences, rule based approaches attempt to find a way in which that sentence could have been generated from the start symbol in the grammar. There are two ways that this can be done:

Top-Down Parsing – Begin with the start symbol and apply the grammar rules forward until the symbol at the terminals of the tree correspond to the components of the sentence being parsed.

Bottom-Up Parsing – Begin with the sentence to be parsed and apply the grammar rules backward until a single tree whose terminals are in the words of the sentence and whose top node is the start symbol has been produced (Rich and Knight, 1991).

The advantage of the top-down parser is that, in working on word n+1, it eliminates from consideration symbols which could not occur in any parse tree whose first n terminal symbols match the first n words of the sentence. The bottom-up parser has no such contextual constraints, so it will perform reductions (build partial parses) in many cases where the top-down parser would have avoided the corresponding productions. On the other hand, the top-down parser will try to expand symbols whose expansion always includes a particular word or word category, even if that word or category is not present in the sentence being analyzed. The bottom up parser will avoid the corresponding reductions. The magnitude of these effects depends on the characteristics of the grammar being used (Grishman, 1994).

A clear advantage of bottom-up approach is that a partial parse will be built just once, even if it is subsequently used as a constituent of many different partial parses. In contrast, the top-down algorithm may end up re-expanding a given symbol many times starting at the same word, if that symbol arises in many different contexts (Grishman, 1994).

As a simple top-down parser parses a sentence from top to bottom, this algorithm tries to predict the end string from the given grammar. This can be non-*deterministic*, as each time it chooses the wrong path; it has to backtrack to where it made the wrong decision. One serious problem associated with top-down parsers is that they sometimes get stuck in a loop, if one or more of the grammar rules are left recursive[5] (Allen, 1995). Another is that it is inefficient and has a worst-case exponential run-time. Bottom-up parsers start with the input string and try to reduce the string into its grammar rules. A disadvantage often cited with this type of parsing is that if the grammar has empty productions (e) the parser can also get stuck in a loop (Winograd, 1983).

It is possible to combine some of the advantages of the top-down and bottom-up approaches by taking one of these algorithms and incorporating some of the features of the other. The virtue of the top-down algorithm in building partial parses selectively, based on left context, can be combined with the virtue of the bottom-up algorithm in building each partial parse only once. The table of partial parses is often called a 'well-formed sub string table'. In **chart parsers**, the well-formed sub string table and the tree

[5]Grammar rules are said to be left recursive when a constituent in the right hand side of a grammar rule appears on the left side as well.

built by a top-down parser are unified in a single data structure **- the chart** (Grishman, 1994).

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 a terminal or non-terminal symbol, the starting point of entry and the entry length. The key list is a pushdown 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 larger entries. Chart parsing allows greater efficiency by recording partial parses and avoiding repetition (Charniak, 1993).

Chart based parsers can be considerably more efficient than parsers that rely only on a search because the same constituent is never constructed more than once. For instance, a pure top-down or bottom-up search strategy could require up to C^n operations to parse a sentence of length n, where C is a constant that depends on the specific algorithm used. A chart based parser on the other hand, has a worst case complexity of $K * n^3$, where n is the length of the sentence and K is a constant depending on the algorithm. A chart parser involves more work in each step, therefore K will be larger than C. But still, a chart is much faster than a top down or a bottom up parser (Allen, 95).

In this study a chart-parsing algorithm is implemented. The choice is made because of the advantages it holds over the top-down and bottom-up strategies, as discussed above.

2.4.2 Encoding Structural Ambiguities

There are two different issues that are of concern in encoding structural ambiguities. The first involves improving the efficiency of parsing algorithms by reducing the search but not changing the final outcome, and the second involves techniques for choosing between different interpretations that the parser might be able to find. There are several ways to build efficient parsers which can handle this kind of ambiguities well. Some methods improve the efficiency of search based parsing models, while others specify models of parsing that are inherently deterministic or only partially analyze the sentences.

Some deterministic methods (such as look ahead techniques, partial parsing, etc) use preferences in parsers to select certain interpretations, sometimes at the cost of eliminating other possible interpretations, to handle ambiguities. Evidence for this type of models is drawn from psycholinguistic theories concerning the human parsing process. One method for improving the efficiency of parsers is to pre-compile much of the search into tables. These techniques are used in shift reduced parsers, which are used for programming languages. These techniques can be generalized to handle ambiguity and can be used for natural languages as well (Allen, 1995). There is also a fully deterministic parsing framework that parses sentences without recourse to backtracking. This means that the parser might make mistakes and fail to find interpretations for some

sentences. If the technique works correctly, however, only sentences that people tend to misparse will be misparsed by the system.

Such deterministic methods use several different techniques (Allen, 95):

- Encoding ambiguity within parse states
- Using look ahead techniques to choose appropriate rules
- Using efficient encoding techniques for encoding ambiguity
- Changing the definition of the output so that it ignores ambiguity.

In general, most of these methods employ heuristics to choose between alternate syntactic analyses to resolve syntactic ambiguities. Creating such heuristics, however, is difficult and time consuming. Furthermore, there is no systematic method for evaluating how well the heuristic rules work in practice.

2.4.3 Statistical Parsing

In statistical parsing, grammar rules specify the structures allowable in the language, while probabilities specify the distributional regularities of sentence structures in the language. That is, probabilistic reasoning by way of statistical probabilities is introduced to assist reasoning. It means that linguistic specifications and statistical regularities of syntax are combined to be used for better syntax analysis. The probabilistic reasoning has become much more popular in resent years (Yao and Lua, 1998). This popularity pertains to the fact that large databases, or corpora, of natural language data have become available in the past few years, and hence the “sparse data problem” (i.e.

unavailability of required data) associated with probabilistic methods is solved to a certain extent (Allen, 95).

2.4.3.1 Probabilistic Context-Free Grammars (PCFG) Parsing

In natural language, a sentence is composed of words in a certain order. An *n*-word sentence $w_{1,n}$ is defined as

$$w_{1,n} = w_1 w_2 \ldots w_n$$

where $w_1, w_2, \ldots,$ and w_n are words that make up the sentence, and the subscripts denote the order of the words in the sentence. In this paper, the sentences we processed are Amharic word-segmented sentences (i.e. words in the sentence are separated by a delimiter). For example, a 5-word sentence is shown below

Kassa wädä tƏmƏhƏrƏt bet hedä.

Kassa went to school.

where " " (space) is the word segmentation symbol. Here, $w_{1,n}$ = Kassa wädä tƏmƏhƏrƏt bet hedä, w_1= Kassa (Kassa, a name), w_2= wädä (to) , w_3= tƏmƏhƏrƏt (education) , w_4 = bet (house), and w_5= hedä (went).

Since a sentence may have several possible different structures, there are some uncertainties in determining its exact structure. These uncertainties can be represented by a certain probabilistic model. Therefore, probabilistic modeling is a general method for modeling and describing statistical properties of sentences and doing syntax analysis (Yao and Lua, 1998). Such a probabilistic model can be well described by probabilistic context-free grammars (PCFGs) which enable description of some of the high probability

structures that arise typically in languages. A probable structure of the sentence “Kassa wädä tƏmƏhƏrƏt bet hedä.” is shown in Figure 2.1.

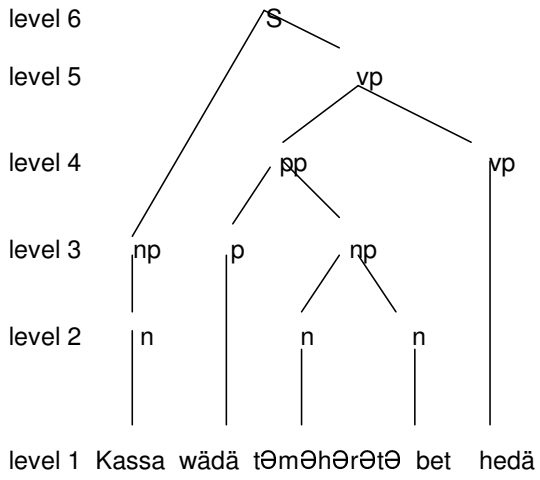


Figure 2.1. A probable structure of the sentence “Kassa wädä tƏmƏhƏrƏtƏ bet hedä.”

The probabilities of specific parse trees for a sentence can be found using a standard chart parsing algorithm, where the probability of each constituent is computed from the probability of its sub constituents and the probability of the rule used. When entering an item E of category C on the chart, using rule i with n sub constituents corresponding to chart entries E_1, ..., E_n:

$$Pr(E) = Pr(\text{Rule } i \mid C) * Pr(E_1) * ... * Pr(E_n)$$ (Allen, 1995).

PCFGs tend to be robust (Manning and Schütze, 1999). They produce a model of a language based on real data, and therefore do not have to worry about things like grammatical mistakes, which occur in real-life situations. Although PCFGs have many advantages, a critical disadvantage is that context is not taken into account at all (Cahill, 2000). In fact a tri gram (sequence of three words in this case) model of a language would probably achieve better results (Charniak, 1993), even though it takes no account of internal structures in the language. Manning and Schütze (1999) also point out that even structurally a PCFG is deficient. A PCFG takes no account of where a phrase appears in a sentence. For instance, the expanding of a subject NP is often different to the expanding of an object NP. Many people (e.g. Charniak, 1997) have noted this disadvantage and many attempts (e.g. embedding context dependent method such as lexicalised parsing) have been made to combine other theories and PCFGs to achieve better results.

2.4.3.2 Best-First Parsing

So far, probabilistic CFGs have done nothing for parser efficiency. Algorithms developed to explore high-probability constituents *first* are called **best-first parsing** algorithms. The hope is that the best parse will be found first and quickly. The chart-parsing algorithm can be modified to consider most likely constituents first. The central idea is to make the agenda a **priority queue** - the highest rated elements are always first in the queue, and the parser always removes the highest-ranked constituent from the agenda before adding it to the chart.

2.4.2.3 A Simple Context-Dependent Best-First Parser

The best-first algorithm improves efficiency but not accuracy. A simple alternative method of computing rule probabilities is one that uses more context-dependent lexical information. The idea exploits the fact that the first word in a constituent is often the *head word* and thus has a big effect on the probabilities of rules that account for the constituent. This suggests a new probability measure for rules that takes into account the first word:

$$\Pr(R \mid C, w) = \frac{\text{Count(\#of times rule R used for cat C starting with w)}}{\text{Count(\# of times cat C starts with w)}} \quad \ldots\ldots (2)$$

How this helps: for instance, in a certain corpus given in the example below, singular nouns rarely occur alone as a noun phrase (i.e. NP -> N), and plural nouns are rarely used as a noun modifier (i.e. starting rule NP -> N N) - this can be seen from the statistics in the table below, adopted from Allen, 95.

Rule	**the**	**house**	**peaches**	**Flowers**
NP -> N	0	0	.65	.76
NP -> N N	0	.82	0	0
NP -> NP PP	.23	.18	.35	.24
NP -> ART N	.76	0	0	0
Rule	**ate**	**bloom**	**like**	**put**
VP -> V	.28	.84	0	.03
VP -> V NP	.57	.10	.9	.03
VP -> V NP PP	.14	.05	.1	.93

Table 2.2 some rule estimates based on the first word

Although a Context-Dependent Best-First Parser is said to be more accurate and efficient (Allen, 1995), for the purpose of this study, the simple PCFG parsing techniques

are utilized, mainly for reasons of: time available for the work, and personal interest in the approach.

In particular, the parsing algorithm implemented is the standard bottom up chart parsing algorithm with slight modification. The algorithm is modified to calculate probabilities as it builds the parse structures and pick the best parse (the one with the highest probability value) using the inside outside algorithm. Also the chart which is holding all possible structures is a two dimensional array of size n x n. The bottom up chart parsing algorithm is selected because of the minimal computational complexity when compared to the other algorithms and also due to the fact that the parser accepts the output of a POS tagger, and it seemed logical to go from bottom up and construct the parse instead of from top down.

Chapter Three

The Amharic Grammar

3.1. Introduction

This chapter tries to briefly discuss the structure of the Amharic word classes, phrase structure and sentence formalisms as a background to the work made. More details could be obtained from Baye(1987), and Getahun(1990). Among the word classes discussed in this chapter are nouns, verbs, adjectives, adverbs, prepositions and conjunctions. Pronouns fall under the noun category. Other Amharic words such as interjections and numerals, phrase structures of the Amharic language such as noun phrases, verb phrases, adjectival phrases, adverbial phrases and prepositional phrases, and sentence formalisms of the language, particularly the simple sentences, are all discussed in this chapter. To render meaning for discussion, the chapter begins with a brief review of the writing system and punctuation marks in Amharic.

Deleted: are also discussed separately but they

The analysis and discussions made in this chapter are based on the data extracted from Mesfin (2001), Abiyot (2000), Baye (1987)[6], Getahun (1990) [1], and Dawkins (1969). Further details on the subject can be obtained from these sources.

[6] The year is according to the Ethiopian calendar

3.2 The Amharic Alphabet and Punctuation Marks

In this study, the Latin letters are used instead of the Amharic "fidel" to represent the Amharic alphabet, as mentioned earlier due to lack of a standard coding scheme for the Amharic writing system. A list of the Amharic alphabet (fidel) adopted from Leslau, and used in this study is found in appendix 1. Detailed discussion on Amharic writing systems may be found in (Dawkins, 1969).

Analysis of Amharic texts reveals that different Amharic punctuation marks are used in Amharic to serve for different purposes. Appendix 2 summarizes such punctuation marks together with the purpose for which they are used.

3.3 Word Categories in Amharic

The basic characteristic of the syntactic structure Amharic is S(Subject) – O(Object) – V(Verb). This structure SOV gives it a characteristic that modifiers in Amharic generally precede the words they modify.

For the purpose of this study, works in the area of word categorization are broadly classified and discussed in two categories, namely, " early" and "recent" works.

In early works (Mersi'hazen, 1934)[7] Amharic words are categorized into the following eight categories (or classes or parts of speech). These are the noun, Verb, Adjective, Adverb, Preposition, Pronoun, Conjunction and Interjection categories.

[7] Date is according to the Ethiopian calendar.

In recent works such as Baye (1987)[8], the early categorization of Amharic words is reduced into five, putting **pronouns** and **conjunctions** under the **noun** and **preposition** categories respectively. In this categorization, **interjections (**which are words with out syntactic functions), are not considered as parts of speech[9].

In this study, the classification by the early scholars is adopted but treating nouns and pronouns in the same part of speech category as suggested by Baye. This preference is made because a direct implementation of a part of speech tagger developed by Mesfin(2001), which is implemented as part of this study, adopts this kind of classification. Mesfin's justification for such classification is that "the early categorization is more exhaustive and it allows the tagger to tag words exhaustively", which I found to be convincing.

3.3.1 The Noun Class

Like English, Amharic nouns are words used to name or identify any of a class of things, people, places or ideas or a particular one of these. For this study, the Amharic noun class is considered to consist of nouns and pronouns.

3.3.2 The verb class

An important property of the Amharic verbs is that any word that comes at the end of a complete grammatical Amharic sentence is a verb. As a consequence of this property a word at the end of such a sentence is expected to be tagged as a verb by an Amharic

[8] Date is according to the Ethiopian calendar

[9] See Baye for such categorization and why interjections are not considered as parts of speech

tagger. Amharic verbs are also known of taking such subject markers as **–hu**, -**e, -h\k, (a) č** and so on.

3.3.3 The Adjective Class

Adjectives in Amharic usually precede the nouns that they modify or describe.

Example: **sänäf** tämari näw "he is a lazy student"

Lazy student is

In this example, the adjective sänäf "lazy" precedes the noun tämari "student" which it modifies. But this does not mean that a word is an adjective just because it precedes a noun. For instance, in **yəh bäg** "This sheep", the word yəh "This" precedes the noun bäg 'Sheep'. Although the word **yəh** functionally shares the feature of an adjective (modifier), it is a pronoun, a demonstrative pronoun.

3.3.4 Prepositions

The term preposition[10] refers to words, which will have meaning only when they are attached or used together with other words such as nouns, verbs, pronouns and adjectives. Prepositions can appear as

- Simple prepositions that stand alone as separate words

Examples **ləlä** təmhərt "because of education"

For education

Wädä bet "to the house or home"

To house

[10] Two types of prepositions can be recognized in Amharic depending on weather they are bound or free. See (Bay, 1987) and (Gethahun, 1990) for definitions of free and bound prepositions.

- Simple prepositions prefixed or attached with other words (e.g. nouns and verbs).[11]

  Examples: **bä**-makina "by car"

  By car

  lä-hizb " to/for the public"

  to/for public

- Compound prepositions consisting of two parts, prepositional prefixes and postpositions put after nouns. The postpositions can either be single prepositions that stand by their own or a preposition not separated from a noun.

  Examples: **bä**-sat'n-u wast "Inside the box"

  Preposition box (noun) a postposition (noun)- inside

  Kä-angät **bä-lay** "Above the neck"

  Preposition noun (neck) preposition noun

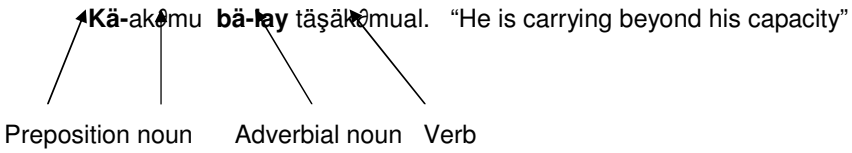
lay in the above example is not a preposition. It is a noun and the whole **bä-lay** is considered as a preposition not separated from a noun. In some cases the whole **bä-lay** is treated as an adverbial noun.

Kä-zaf **lay** "From a tree"

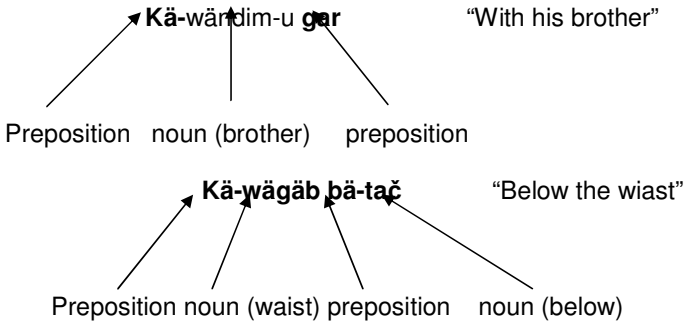
Preposition noun (tree) preposition

[11] According to Cowley et al (1982), this is one distinctive feature of Amharic grammar which is only slightly shared by languages of the same syntactic type.

Here **lay** is used as a preposition rather than as a noun.



While in the above example **bä-lay** is used as an adverbial noun.



The whole bä-tač is a noun adverb composed of a preposition not separated from the noun (Baye, 1987)

3.3.5 The Adverb Class

In Amharic, adverbs can be found in either primitive forms (i.e. as separate words that appear by their own) or in compound forms as combinations of prepositions and some other words but that appear as separate or in rare cases as compound words. In each case they refer to place, time, circumstance etc. A long list of such adverbs is found in (Dawkins, 1969).

There are also what are called adverbs of position, which may be referred to as noun adverbs. Noun adverbs can be used either as a noun or as an adverb depending on their context.

Examples:

wädä **wəst'** gäbä "He entered inside" wəsť here is used as a noun

to inside he entered

(bä) wəst' yəsäral "He works from inside" wəsť is used here as an adverb

from inside he works

As can be seen from the examples mentioned above, noun adverbs (or adverbs of position) such as wəst "inside" can function alone as an adverb or as a noun. But they often have the locative particle **bä** when they are used adverbially. If such noun adverbs are used alone without such prepositions (as **bä, wädä**, **kä**) they are mostly treated as nouns. On the other hand, noun adverbs prefixed with prepositions are treated as adverbs. That is, wəst is a noun while bä-wəsť is an adverb. An exhaustive list of noun adverbs is also found in (Dawkins, 1969).

Adverbs can also be formed from nouns and adjectives by prefixing { **bä-** } as in **bä**-hayəl "by force" and **bä**-dähna "being well" respectively. Adverbs formed in this manner are short adverbial clauses. That is, the resulting words are more of adverbs and are as such considered as adverbs.

Days of the week in Amharic language may also be used as either a noun or as an adverb.

3.3.6 Conjunctions

Conjunctions in Amharic are coordinating or subordinating. They coordinate words, phrases, clause and sentences. A list of Amharic coordinating conjunctions is found in (Dawkins, 1969) together with a detailed discussion on such coordinating and subordinating conjunctions.

Concerning conjunctions there are two views. In early works (e.g. Marsehazn, 1934), conjunctions were viewed as appearing in their own Category. But in recent works (Baye, 1987; Getahun, 1990), they are categorized in the same category as prepositions.

This work adopts the early view that conjunctions and prepositions appear in different categories, the preposition and conjunction parts of speech categories. One problem that arises by categorizing prepositions and conjunctions into different classes is the problem pertaining to distinguish conjunctions from prepositions. The problem in distinguishing the two mainly arise from the fact that the same words are mostly used as both prepositions and conjunctions. S*ə*lä “for”, bä “by”, and kä “with/from” are, for instance, both prepositions and conjunctions. One may think that this will create a problem during the tagging process. But this will not be a problem for it can be minimized if the manual tagging is done very carefully by good annotators, as the two can be identified from the context they are used.

3.3.7 Numerals

These are words representing numbers. They can be **cardinal** or **ordinal** numbers. A list of the Amharic Cardinal numbers is found in (Dawkins, 1969; Leslau, 1973). In Amharic, the ordinal numbers are formed from the cardinal numbers by suffixing the suffix {**–(ä) ňňa }.**

Example	**Cardinal**	**gloss**	**Ordinal**	**gloss**
	hulätt	two	hulät- äňňa	second
	assər	ten	assər- äňňa	tenth

Like English, compound Amharic numerals are put separately. The following are examples to illustrate this.

Example hulät mäto sälasa and "two hundred thirty one"

hulät mäto sälasa and-äňňa "231st "

In Amharic, there are also numerals that indicate distribution. These numerals are called distributive numerals.

Example **hulätt hulatt** "two two"

There are also special numerals in Amharic that correspond to the English "half", "quarter" etc.

Examples of these include **gəmaš** "half" and **siso** "one third".

3.3.8 Interjections

Like English, Amharic has many words or phrases used to express such emotions as sudden surprise, pleasure, annoyance and so on. Such Amharic words are called interjections. These Amharic interjections can stand-alone by themselves outside a sentence or can appear any where in a sentence.

Examples **goš !**

Goš ! abbate mäťťa

abbate mäťťa **Goš !**

sərah ťəru näw **Goš !** ləje.

A long list of Amharic interjections is found in (Dawkins, 1969).

3.3.9 Specifiers, Modifiers and Complements

Sections following this one discuss about phrasal categories and sentences in Amharic. Baye (1987), uses the terms specifier, modifier, and complement, in parsing Amharic sentences. These are not word categories, but are used to specify the function of a certain word or phrase in a sentence. The definitions are given below.

Specifier: a construct used to specifically point out or differentiate a certain thing.

Example: Hulät gize təmməhərt bet hedä. He went to school twice.

Twice education house he went

Here "Hulät gize" (twice) is used to specifically point out the number of times the subject i.e. he, went to school, and is therefore labeled as a specifier, and its word class falls under the adverbial phrases.

Modifier: a construct that builds, enhances the word/phrase following it such as an adverb or an adjective is labeled as a modifier. These labels are mostly used in complex sentences, to label sentences that modify phrases or other sentences in a certain construct.

Example: **bä-mäkina** wädä bet hedä. " He went home by car"



By car to house he went.

Here the prepositional phrase **bä-mäkina** is also labled as a modifier for the rest of the sentence. wädä bet hedä "he went home" is a sentence by itself, and the function of **bä-mäkina** "by car" is to build the sentence by giving more information about the incident. More details on modifiers is found in Baye(1987).

Complement: a construct that is used to make a certain idea complete.

Example: Kassa gädälä. Kassa killed.

Kassa an∂bäsa gädälä. Kassa killed a lion.

Here an∂bäsa "lion" is a complement, which is a noun. It is said to be a complement because it is used to make the sentence "kassa killed" complete, by way of specifying what the object of that sentence is.

3.4 Phrase Structure of Amaharic

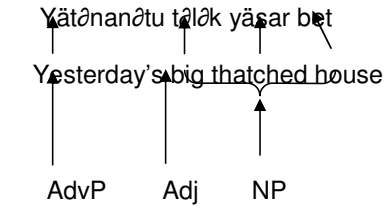
A phrase is a structure in a language, which is constructed from one or more words in the language. In Amharic there are five phrase classes, which are noun phrases, verb phrases, adjectival phrases, adverbial phrases and prepositional phrases Baye(1987). Each of these phrase classes are discusses in this section.

3.4.1 Noun Phrases

Noun phrases(NPs) are used to refer to things: objects, places, concepts, events, qualities and so on. The simplest NP consists of a single noun or pronoun such as ∂su (he), ∂suwa (she), ∂när∂su (they), etc. Noun phrases in Amharic are constituted of a noun and another constituent. The nouns in the noun phrases appear with noun complements, adverbial and adjectival modifiers, and specifiers (Baye, 1987). For example “yä sar bet” is a noun phrase. And it is composed of the noun “bet” and another noun phrase “yä sar”. This would produce the phrase structure rule:

NP → NP N

There are many combinations of constituents that could make up a noun phrase in Amharic. An example is provided below.

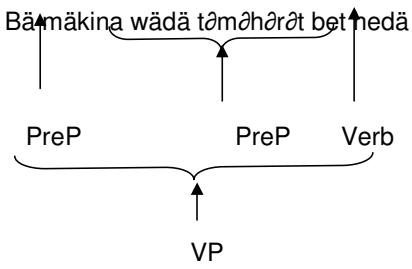


Grammar rule: NP → AdvP Adj NP

3.4.2 Verb Phrases

Verb phrase are composed of a verb and other constituents such as complements, modifiers and specifiers. Some examples of verb phrases are given below.

Example 1:

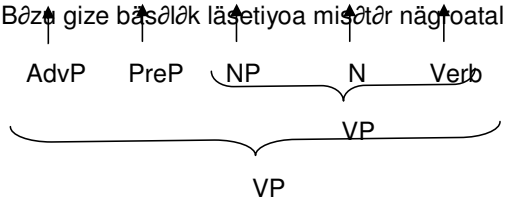


He went to school by car.

Structure rule: VP → PreP PreP Verb

In the above example hedä ”went” is the verb, and since it has no verbal inflection suffix such as –u, -n etc, it implies that the subject is a masculine third person singular. The first prepositional phrase modifies the verb while the second one complements it. Therefore both the prepositional phrases are constituents of the verb phrase.

Example 2: Bɑ̂zɑ̂ gize bäsɑ̂lɑ̂k läsetiyoa misɑ̂tɑ̂r nägroatal.



He told the lady secrete by telephone a lot of times.

Structure Rule: VP → AdvP PreP VP

VP → NP N Verb

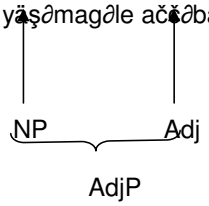
In this example also, the above constituents are all included in the verb phrase since all of them modify or complement the verb nägroatal, which has the root nägärä “tell” and

the suffix –oatal to implicate that the subject is a third person singular masculine and the object is a third person singular feminine.

3.4.3 Adjectival Phrases

The construction of adjectival phrases is the same as that of noun phrases and verb phrases. An adjective and other constituents such as complements, modifiers and specifiers, make up an adjectival phrase. Some examples are given below.

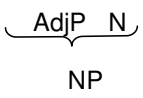
Example 1: yäş*ə*mag*ə*le ač*ə*bär*ə*bari (A crook who is old)



Structure rule: AdjP → NP Adj

Here the noun phrase complements the adjective and hence it is an adjectival phrase.

Example 2: t*ə*l*ə*ku säw*ə*ye "the big man"



Here the adjective t*ə*l*ə*ku "the big" describes the noun "säw*ə*ye" and the above example falls under noun phrases. But the adjectival phrase is composed of an adjective only and the structure rule would be:

AdjP → Adj

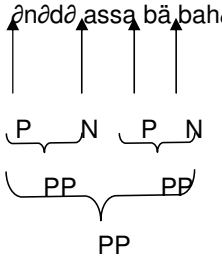
NP → AdjPN

3.4.4 Prepositional Phrases

Prepositional phrases in Amharic are constructed from prepositions(P) and other constituents such as nouns, noun phrases, verbs, verb phrases etc.

Deleted: The head word is a preposition while the other constituents are there to modify or complement the preposition.

Example 1: ∂n∂d∂ assa bä bah∂r like fish in sea



Structure Rule: PP → P N

PP → PP PP

∂n∂d∂ "like" in the above example is a preposition and the phrase ∂n∂d∂ assa "like fish" is a prepositional phrase because it has a preposition as a constituent.

Example 2: zaf lay "on a tree"

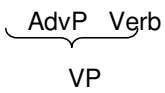


Structure Rule: PP → N P

3.4.5 Adverbial Phrases

Adverbs in Amharic are used in many constructs, such as indicators of degree, in location phrases, to show the manner in which something is done, the time of something, or the frequency of something etc. Adverbial phrases are constructed from one or more adverbs in the language.

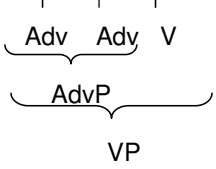
Example 1: tolo mät'ač she came quickly



Structure rule: AdvP → Adv

In the above example, the adverbial phrase is composed of a single adverb, while in the example below it is composed of two adverbs.

Example 2: tolo tolo tämämädä. "He walked briskly"



Here tolo "quickly" appears twice , stressing the point. And the structure rule is:

AdvP → Adv Adv

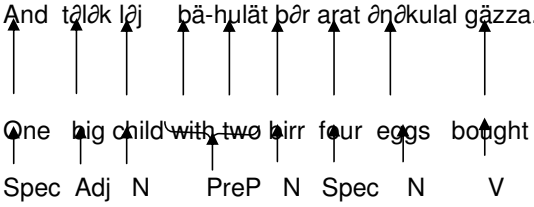
3.5 Sentences in Amharic

Baye (1987) categorizes Amharic sentences into simple and complex. A simple sentence is a complete structure that can convey a complete idea. Complex sentences are used to convey complete ideas like simple sentences but unlike simple sentences they are composed of complex phrases. Complex phrases are phrases that have complements and/or modifiers that are sentences themselves. This study focuses on simple sentences but the techniques applied to parse the simple sentences can be extended for the complex once as well. Simple sentences are discussed in details in the following sections.

Below is an example of a simple sentence in Amharic with the POS tags indicated for each word in the sentence, and the corresponding parse tree for the sentence.

One big child bought four eggs with two birr.

And tələk ləj bä-hulät bər arat ənəkulal gäzza.



One big child with two birr four eggs bought

Spec Adj N PreP N Spec N V

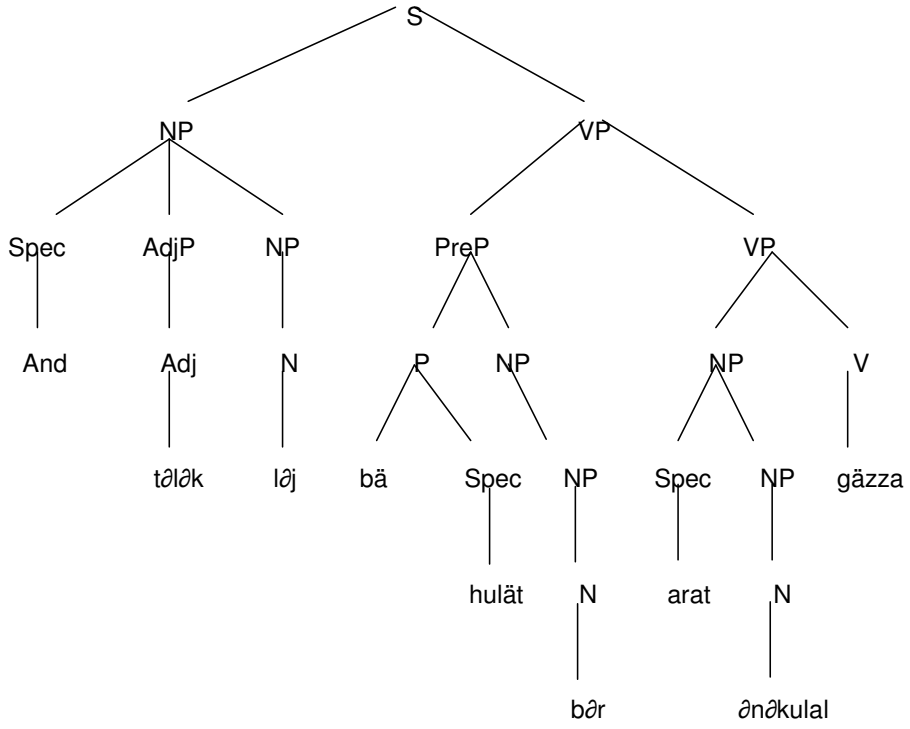


Figure 3.1 A parse structure for the sentence. “And tələk ləj bä-hulät bər arat ənəkulal gäzza”.

Baye(1987) classifies simple sentences into:

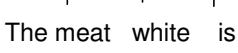
- Declarative Sentences
- Interrogative sentences
- Negative sentences and
- Imperative sentences

Each of these categories is described in this section. Further details about compound and complex sentences may be found in Baye (1987).

3.5.1 Simple Declarative Sentences

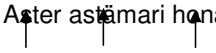
A simple declarative sentence consists of a NP, the subject, followed by a verb phrase, the predicate. These sentences are used to convey ideas and feelings that the speaker has about things, happenings, feelings, etc, that could be physical, mental, real or imaginary.

Example: Sägaw čëoma näw. The meat is white.



The meat white is

Aster astämari honäč. Aster became a teacher.



Aster teacher became

In the first example, a simple declaration is made to describe about a thing, i.e. that the meat is white; and in the second one the declaration is about an incident, i.e. Aster (a lady) becoming a teacher.

3.5.2 Simple Interrogative Sentences

A sentence that questions about the subject, the complement, or the action the verb specifies, is called an interrogative sentence.

Example: Kassa mäče mäta? When did Kassa come?

Here the interrogator knows that it is Kassa who came, but inquires about the time of his arrival.

Kassa mäta woi? Did Kassa come?

Here the interrogator knows that Kassa would come but does not know whether he came. The “woi” in the end implicates that the answer to the question is either yes or no.

Kassa mäta ∂n∂de? Did Kassa come?

This question appears to be the same in idea as the second one, but it does not assure that the answer would be a yes or no like that of the second one.

“∂n∂de” appears at the end of questions, as “woi” does. “∂n∂de” indicates that the interrogator has some prior knowledge about the subject matter at hand.

3.5.3 Simple Negative Sentences

These sentences negate a declarative statement made about something.

Example:	Kassa m∂sa-w∂n bäla.	Kassa ate his lunch.
	Kassa m∂sa-w∂n al-bäla-m.	Kassa did not eat his lunch.

In the above example, the verb in both sentences is bäla “ate”. It is negated by the negation prefix al- and the suffix –m indicating that the subject of the sentence is masculine, third person singular.

3.5.4 Simple Imperative Sentences

These sentences convey instructions. Mostly, the subject of these sentences is a second person pronoun. The subject is also mostly omitted and since Amharic words are highly inflected, the prefixes (subject markers in this case) indicate the specific subject.

Example:	T'ə̂j amə̂t'a!	Bring tej! (Masculin)
	T'ə̂j amə̂čči!	Bring tej! (Feminin)

3.6 Conclusion

This chapter pointed out that, based on recent and early works, Amharic words are categorized in to eight or five parts of speech categories. For the purpose of this study, words in Amharic are categorized into seven parts of speech, namely:

Nouns	verbs	conjunctions	adverbs
Adjectives	prepositions	interjections	

In particular, in this study pronouns are treated as nouns and hence do not have their own part of speech category. This categorization is based on what has been forwarded by early and recent scholars and also considering solely conveniences for tagging. In some cases, there might be cases that may not go in line with structural linguistics.

Phrases in Amharic were also discussed in this chapter. The phrase categories as classified by Baye (1987) were directly adopted for the purpose of this study. According

to Baye, phrases in Amharic fall into five major categories, namely, the noun phrases, verb phrases, adjectival phrases, adverbial phrases and prepositional phrases. The sentence categorization discussed is also an adaptation from Baye's work, which classifies Amharic sentences into simple and complex. The simple sentences, which include simple declarative sentences, simple interrogative sentences, simple negative sentences and simple imperative sentences, were discussed in this chapter.

Chapter Four

Probabilistic Context Free Grammar Extraction

4.1 Introduction

In this chapter the PCFG that has been employed by the sentence parser developed in this study is discussed. The process is based on the syntactic property of the language presented in chapter three and the assumptions and approaches discussed in chapter two.

The chapter begins by discussing the approaches taken to design the parser. The second section discusses the sample text used in this study. The third section discusses the part of speech tagger developed by Mesfin (2001) and how it was embedded in the design and development of the parser. The features of the language considered in designing the parser are discussed in the fourth section. The fifth section deals with the extraction of a probabilistic context free grammar rule from the tagged corpus. The conversion of the rules to the Coonskin Normal Form (CNF) is presented in section six. And the last one concludes and summarizes the chapter.

4.2 Approach Selected to Design the Sentence Parser

As described in the second chapter, the approach considered in the design of the parser is the PCFG bottom up chart parsing. Puffs are Cogs with associated probability values. Context-free grammars are widely used as models of natural language syntax. In their

probabilistic version (PCFG), which defines a language as a probability over strings, they have been used in a variety of applications (Stocked, 1993).

There are many advantages to using Puffs over normal Cogs to determine the structure of a sentence. As grammars get larger and larger, they become more and more ambiguous. In such cases, a PCFG are reported to give an indication of the most likely parse. It can also be programmed to give the n most probable structures, which can be useful when sentences are ambiguous to the reader (Cahill, 2000).

Assuming that a sentence $w_{1,n}$ (where $w_{1,n}$ is a sequence of words w_{11}, … , w_{1n}) has *T* (*n*) possible parse trees (possible structures), and t_i is the ith parse tree that $0 < i \leq \mathrm{T(n)}$, then the probability of the ith parse tree of the sentence $w_{1,n}$ is the product of the probabilities of all rules used in the parsing and is given by (Yao and Lua, 1998)

$$P(t_i) = \prod_{j=1}^{n} P(R^j) \quad \text{.....(3)}$$

And the probability of the sentence $w_{1,n}$ is the sum of the probabilities of all possible parse trees that

$$P(w_{1,n}) = \sum_{t_i = 1}^{\mathrm{T(n)}} P(t_i) \quad \text{.....(4)}$$

In statistical meaning, $P(t_i)$ shows the possibility of the ith parse tree of all possible parses, while $P(w_{1,n})$ shows how grammatical a sentence $w_{1,n}$ could be in the

language. The larger the $P(t_i)$ is, the more reasonable the parse is. What this means is that to find the best parse of all possible parses of a sentence is nothing but to find $\max_{0<t_i \leq \mathrm{T(n)}} \{P(t_i)\}$. Therefore, $P(w_{1,n})$ and $\max_{0<t_i \leq \mathrm{T(n)}} \{P(t_i)\}$ are two important facts in syntax analysis that the first one gives the reasonability of a sentence in PCFGs, while the second one gives the most possible parse of all. In this study, the basic assumption made is, all sentences are considered to grammatical in the language. Accordingly, the focus is on the $P(t_i)$ values, to get the most probable parse structure.

The PCFG parser designed in this study also tried to take lexical co-occurrence[12] into consideration. This was achieved by using the Hidden Markov Model (HMM) Tagger and implemented bi-grams, and hence an attempt is made to include lexical information in the parsing process. Since the tagger, which was developed by Mesfin(2001) has a mechanism of handling words with multiple POS tags by way of their associated probability values, and lexical co occurrence, there was no need to design and implement such a module in the parser designed. The words in an input sentence are automatically tagged first, before they are processed by the parser. This would help in minimizing the computational time since the parser does not need to handle multiple POS words and generate the parse space for each possibility of the tags. Also, since the lexicon is stored separately, (i.e. from the rule base), the size of the PCFG table is minimized and more efficiently managed.

[12] Words occurring together in a sentence

4.3 Sample Corpus

For the purpose of this study, simple declarative sentences, which are composed of four words were considered. This, as described in chapter one, is due to the time constraint and the lack of hand parsed/annotated text for the purposes of grammar induction and training.

100 such sentences were selected from Baye Yimam's "Yä Amarəĝa Säwasəw". This book was chosen as a reference after discussion with the linguistic advisor for the thesis, the author and other individuals from the language department. The number of sentences was limited to 100 because of the time constraint. In the time available only 100 sentences could be hand parsed, and experimented with. The sentences were selected randomly, but with some judgment by the researcher such as, for instance all the sentences constitute two or more of the phrase classes discussed in chapter three.

These sample sentences were transcribed according to the IPA (International Phonetic Alphabet) standard. See the transcribed sentences in Appendix 3. Some variations were made wherever needed. For example, the tagger accepts only .txt files and when the transcribed text is stored in such files, some of the features of the text are lost. The sentences were then automatically tagged using the POS tagger developed by Mesfin (2001) after building the lexicon and training the tagger with it. Refer to Appendix 4 for the special tags identified by Mesfin (2001) and Appendix 5 for the tagged sentences.

The tagger accepts only .txt files. And when the transcribed sentences were saved using this format, some characters are changed. For example the character ə is changed to ? since it is not supported in a .txt file. These kinds of changes were made for the sake of convenience.

The sentences were also hand parsed. Some of the parses for the sentences were given in the textbook, while the others were parsed by the researcher according to the phrase structure rules of the Amharic language. In hand parsing the sentences, comments and suggestions were taken both from the author of the book and the linguistic advisor for this thesis.

Deleted: See Appendix 4.3.

The probability calculations for the words in the sentences, grammar rule induction, and probability assignment to the grammar rules, were conducted using 80 sentences. These 80 sentences were picked randomly from the sample selected. The rest 20 sentences from the original corpus were used as a test set. The sample corpus is given in Appendix 3.

4.4 Part of Speech Tagger

Mesfin (2001) has developed a prototype simple automatic part of speech tagger for Amharic language. In his study, he used the Viterbi algorithm without any modification. A module for sentence splitter was also developed in order to facilitate the preparation of texts in a file to be tagged with appropriate parts of speech. POS tags were assigned on

the basis of the review made regarding the linguistic properties of the Amharic word classes. The study adopted the Stochastic Hidden Markov Model approach to develop the prototype.

The results achieved by the tagger based on the small sample taken (a page long Amharic text), were high, 97% on the training set and approximately 90% on the test set.

Mesfin's tagger had made use of four tables in a database. The tables and their functions are listed below.

Word Code Table:

This table is used to store words from the sample text and a corresponding word code is given sequentially for each word.

Category Code Table:

This table is used to store the 25 special tags identified with a corresponding category code. The total number of special tags identified is 24, the 24th tag being for all punctuations and an additional 1 for all unidentified words.

Deleted: word categories

Deleted: categories

Lexical Probabilities Table:

This table stores the probabilities of words given categories (tags) for each word in a given corpora. This is written as p (word\tag) or p (word\category) or shortly as p (w_i\C_i). For instance, p (bäkälä\N) denotes the (lexical) probability of bäkälä "it grew", which is a common name in the language, to be a **noun,** and p (bäkälä \V) denotes the (lexical)

probability of bäkälä to be a **verb** in a given pre-tagged corpus. The lexical generation probability is estimated simply by counting the number of occurrence of each word by a category. Mathematically this is given by

$$P(W_i \backslash C_i) = \frac{\text{number of times } W_i \text{ appears in category } C_i}{\text{total number words with category } C_i} \quad \text{.....(5)}$$

Transition Probabilities Table:

These are probabilities of a tag given one or more previous tags. Transition probabilities are denoted by $p(C_i \backslash C_{i-1} \ldots C_n)$. Such probabilities tell us, for instance, the probability of a noun to be preceded by an adjective, $p(C_i = \text{Noun} \backslash C_{i-1} = \text{Adjective})$. They can also tell us the probability of a verb to be preceded by a noun, an adjective and a determiner, $p(C_i = \text{verb} \backslash C_{i-1} = \text{Noun } C_{i-2} = \text{adjective } C_{i-3} = \text{verb})$ and so on.

Thus, depending on the value of n, we can have bigram (n=2), trigram (n=3) or in general an n-gram transitional probabilities. If n=2, the model is a bigram model and the notation $p(C_i \backslash C_{i-1} \ldots C_n)$ reduces to $p(C_i \backslash C_{i-1})$. If n = 3, the model is called a trigram model and the notation $p(C_i \backslash C_{i-1} \ldots C_n)$ reduces to $p(C_i \backslash C_{i-1} C_{i-2})$.

The bigram model, as described, considers pairs of two categories (or tags), C_i and C_{i-1}, and calculates the probability that a category C_i will follow the category C_{i-1}, written as $p(C_i \backslash C_{i-1})$. This model assumes that the probability of a particular category occurring depends solely on the one category immediately preceding it. The trigram model pairs three categories, C_i, C_{i-1} and C_{i-2}, and calculates the probability that the category (or tag)

C_i will follow the two immediate preceding categories, C_{i-1} and C_{i-2}. This is written as p ($C_i \backslash C_{i-1} C_{i-2}$).

In practice, given a database of texts tagged with part of speech, the bigram or transitional probabilities can be estimated simply by counting the number of times each pair of categories occurs compared to individual category counts. Mathematically this is written as:

$$P(C_i = \propto / C_{i-1} = \beta) = \frac{\text{count of the number of times } \propto \text{ and } \beta \text{ occur together in the corpus}}{\text{Number of times } \beta \text{ occurs in the corpus}} \quad .. (6)$$

Where $\propto$ and β are parts of speech codes.

The POS tagger developed by Mesfin is used in this study with some enhancement as follows:

1. The tagger was trained with a larger corpus to attain more accuracy. The 209 words lexicon has been upgraded to a 562 words lexicon. The additional words are taken form the sample text selected for the purpose of this study.
2. The transition probability values were recalculated based on the original and additional lexical data.

4.5 Features of the Language Considered in the Design

In his study, Mesfin had identified 24 special tags. These tags fall under the word classes discussed in chapter three. For example, he has classified the nouns into three,

Deleted: word classes
Deleted: classes

the noun, verbal nouns, and noun formed by prefixing a prefix balä (which indicates an owner of an object) to nouns. In this study, such classes are summarized to be in the original category, the noun. This kind of generalizations were made for the sake of convenience in parsing.

He has also identified special tags such as Vprep and Nprep, which are composed of two word classes. For example, he had a word class Nprep, which is a word with a preposition not separated from a noun (e.g. bämäkina "By car", sɔ̈lähägär "about a country"). Baye (1987) treats these kinds of words as prepositional phrases that are composed of a preposition and a noun. Since the parse structures assigned are taken from Baye's book, and since only 4 word sentences were considered, this kind of words were excluded from the sample taken for the purpose of this study. This is because these words are phrases by themselves, therefore during parsing, it would be left as a phrase (PP) at a terminal or a morphological analyzer should be implemented.

Mesfin assigned the category "J" to represent adjectives. And these assignments were replaced by "Adj" for convenience in the current work. These kind of replacements were made wherever deemed necessary.

4.6 Extraction of a Probabilistic Context Free Grammar

The grammar induction algorithms most successful in language modeling include the Inside-Outside algorithm (Lari and Young, 1990; Lari and Young, 1991; Pereira and Schabes, 1992), and a special case of the Expectation-Maximization algorithm (Dempster et al., 1977).

In extracting the PCFGs, the sentences that are used as a training set were first parsed and represented in the following manner for later comparison purposes:

```
(S (NP  (N  Wätadäroču))
   (VP  (PP (P wädä)  NP (N  gibiačäw))
        (V gäbu)))
```

Although there are many ways of doing it, the simplest way to gather statistical information about a grammar is to count the number of times each rule is used in a corpus containing parsed sentences and use this to estimate the probability of each rule being used (Allen, 95). This is the method used in extracting the PCFG in this study, and this approach was selected for its simplicity. The CFG rules were extracted from the parsed sample, and the probability values were assigned to each rule using the formula:

$$P(Rj|C) \sim= \frac{\text{Count (Number of times Rj is used)}}{\sum_{l=1,m}(\text{Number of times Rj used})} \quad \ldots\ldots\ldots(7)$$

where C represents a category at the left hand side of rule Rj

Deleted: If and when a large amount of processed data is available, any of the algorithms mentioned above could be used to extract grammar rules and also increase the efficiency of the parser.¶
¶

An example of a PCFG is represented below. This is an arbitrary example. See Appendix 6 for the full PCFG extracted from the sample sentences.

Rules	Probabilistic Rules
S → NP VP	S → NP VP: 1.0
NP → N	NP → N : 1.0
VP → NP AUX	VP → NP AUX : 0.5
VP → PP V	VP → PP V : 0.5
PP → P NP	PP → P NP: 0.5

Table 4.1 Probabilistic Context Free Grammars

4.7 Conversion to CNF

After the probabilistic context-free grammar was extracted, the next thing that needed to be done was to convert it into Chomsky Normal Form (CNF). This conversion was done for the purpose of simplicity. Any CFG rules can be easily re-written in the CNF, and this restriction will not result in the loss of any real expressive power. It means each rule $R^i \in R$ is in one of the following two forms,

$$R^i : N^i \rightarrow w^j \qquad \text{or} \qquad R^i : N^i \rightarrow N^j N^k$$

where $w^j \in W$ is a terminal symbol and $N^i, N^j, N^k \in N$ are non-terminal symbols. A grammar in CNF would also be easier to manipulate because of the binary structure it attains.

Rules of the form A --> BCDE (p), where p is the probability, are replaced by a set of rules (Abney, 1996):

A → BC' (p), C'→ CD' (1) and D'→ DE (1) ..... (8)

The PCFGs extracted for the Amharic language are converted to CNF manually. This conversion was in line with the phrase structure rules of the language and in actual sense it related to representing it in the exact structure more than converting it to the CNF. That is, since all sentences considered were 4-word sentences, most of the rules were already in the CNF. Examples are given below.

The rules:

VP → Adv Adv V (0.14)

VP → Adv N V (0.17)

Are converted to the CNF and represented in the following manner.

LHS	RHS1	RHS2	Probabilities
VP	Adj	R1	0.14
VP	Adv	R2	0.17
R1	Adv	V	1
R2	N	V	1

Table 4.2 Probabilistic Context Free Grammars in CNF

4.8 Conclusion

In this chapter, the PCFG parsing approach implemented in the study, and the POS tagger embedded are discussed. The sample text used in this study, the features of the language considered in designing the parser, and the extraction of a PCFG from the sample data were also discussed here. Although there was not much conversion to the CNF, the concept is presented in detail because it is believed that it would provide an insight for future researchers.

Chapter Five

Parsing Algorithm and Experimentation

5.1 Introduction

In this chapter, the algorithms used to develop the parser, and the different data structures and databases implemented are discussed. Also, the details of the experiment conducted and the findings of the study are reported.

The chapter begins with a discussion on the selected algorithm, followed by a discussion on the PCFG parser designed. The data structures implemented, and the databases maintained and their structures are also included. The configuration of the parser, the experimentation and experimental findings, and solutions to identified problems are reported. The method followed in the assessment of the results is also included in the last part.

5.2 The Inside-Outside Algorithm

Since most sentences in natural language are ambiguous (as in the case of Amharic), in practice we need to keep a separate count for each parse of a sentence and weight each partial count by the probability of the parse it appears in. This probability value will then be used for structural ambiguity resolution (i.e. selection of the appropriate parse). The standard algorithm for computing this is called the **Inside-Outside** algorithm, and was proposed originally by Baker (1979). For computing $P(w_{1,n})$ and $\max_{0<t_i \leq \mathrm{T(n)}} \{P(t_i)\}$, the Inside-Outside algorithm is widely used. The parser developed in this study uses this

Deleted: paper

Deleted: is based on

algorithm to estimate the parse probabilities, with a parse chart based on the bottom-up parsing algorithm to assist the parsing.

Deleted: and

To find $P(w_{1,n})$ and $\max_{0<t_i \le T(n)} \{P(t_i)\}$ using the Inside-Outside algorithm, either of the two probability calculations are required. That is, the inside probability or the outside probability, for each node in the parse structure.

For a word sequence $w_{k,l}$ derived directly or indirectly from a non-terminal node $N^j_{k,l} \in N$, the inside probability is:

$$\beta_j(k,l) = P(w_{k,l} | N^j_{k,l}) \quad \text{.....(9)}$$

And the outside probability is:

$$\alpha_j(k,l) = P(w_{1,k-1}, N^j_{k,l}, w_{l+1,n}) \quad \text{....(10)}$$

Where $\beta_j(k,l)$ denotes the probability of the word sequence $w_{k,l}$ in which all the words are inside the node $N^j_{k,l}$, and $\alpha_j(k,l)$ denotes the probability of the wards outside the node $N^j_{k,l}$ (Yao and Lua, 1998).

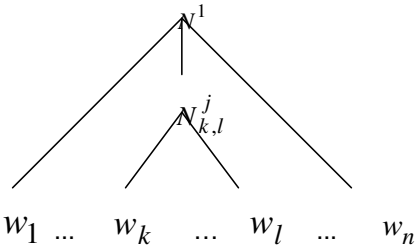


Figure 5.1 The words outside and inside $N^j_{k,l}$.

The above figure shows the words $w_1 \ldots w_{k-1}$ and $w_{l+1} \ldots w_n$ are outside $N^j_{k,l}$ while the words $w_k \ldots w_l$ are inside $N^j_{k,l}$.

The probability of the best parse tree of a sentence which is $\max_{0<t_i \leq \mathrm{T(n)}} \{P(t_i)\}$ can also be obtained by calculating the inside probability or the outside probability.

The algorithm used to calculate the probability of each parse structure for a given sentence is given below.

For each POS tag in the diagonal
Check if a POS tag and the next one occur together at the right hand side of a rule:

If it occurs more than once, get the highest probability of these rules and store its left hand side on the POS matrix, store the rule in the chartentry table and store the probability value in the Probability array

If it occurs once store its left hand side on the POS matrix, store its left hand side on the chart matrix, store the rule in the chartentry table and store the probability value in the Probability array

Else, store 0 in the Probability array and leave the chart matrix value empty

For each possible combination in the chart matrix, calculate the inside probability using the formula:

$$P(t_i) = \prod_{j=1}^{n} P(R^j)$$

Get the maximum of P(ti) and assign it to be the best parse.

Figure 5.2 Algorithm to Calculate the Inside Probabilities

This algorithm is coded in the development of the prototype and is used to calculate the inside probabilities of each parse in the parse tree space. In calculating the probabilities, the algorithm used the modified bottom up chart parsing algorithm, which constructs the parses bottom up.

5.3 PCFG Parsing

The parser designed used the Inside-Outside algorithm to find out the best parse of all possible combinations, and has also a modified standard bottom up chart parsing algorithm by making use of a two dimensional array of POS matrix instead of the active arcs of the original bottom up algorithm. This algorithm, the parse chart, and the configuration are introduced and discussed below.

Deleted: the

5.3.1 Parse Tree

Since the PCFG rules are restricted to the CNF, each possible parse tree of a sentence is a binary tree. Generally, for a parse tree, level 1 is always a set of terminal nodes, which are words, and level 2 is a set of non-terminal nodes, which are the POSs of the corresponding words. According to the CNF, this is written in the form $N^i \rightarrow w^j$, and each node in level 2 only has a descendant in level 1. Therefore a parse tree can simply be considered as a binary tree.

For a non-terminal node N^i, if there is a rule $N^i \rightarrow w^j$, then N^i is supported by the grammar rule $N^i \rightarrow w^j$; and if there is a rule $N^i \rightarrow N^j N^k$, then N^i is supported by the grammar rule $N^i \rightarrow N^j N^k$. Assume that each word in an *n*-word sentence $w_{1,n}$ has one

POS, and $T(n)$ denotes the maximum number of structure trees that $w_{1,n}$ probably have. If each non-terminal node of the trees is supported by a grammar rule, then we can find out that

$$T(n) = \begin{cases} 1, & n = 1 \\ \sum_{i=1}^{n-1} T(i)T(n-i), & n > 1 \end{cases} \quad \text{.....(11)}$$

Clearly, $T(n)$ increases exponentially as n increases. From this we can obtain the parse tree space of a sentence. As an example, the parse tree space of a 4-word sentence is shown below. Actually, $T(n)$ gives the maximum number of possible parse trees a sentence $w_{1,n}$ can have. However, in real use the number of all parse trees of a sentence is less than $T(n)$ since not every node in parse trees can be supported by a grammar rule. That means, some trees in the parse tree space may be ungrammatical for certain sentences.

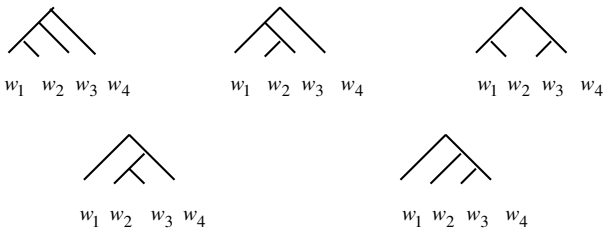


Figure 5.3 Parse tree space of 4-word sentences

The four word sentences in the sample corpus all have these kind of parse tree space. And based on the probability values associated with each grammar rule and the inside outside algorithm, the best parse structure is selected for the input sentences.

5.3.2 Parse Chart

As indicated earlier, $P(w_{1,n})$ shows how grammatical a sentence $w_{1,n}$ could be in the language and $\max_{0<i\leq T(n)} \{P(t_i)\}$ gives the best parse tree of all possible parse trees. In other words, to parse a sentence is to calculate $P(w_{1,n})$ and $\max_{0<i\leq T(n)} \{P(t_i)\}$. Therefore, we can implement sentence parsing by calculating the inside probability or the outside probability to find $P(w_{1,n})$ and $\max_{0<i\leq T(n)} \{P(t_i)\}$. This is nothing but a search in parse tree space.

A parse chart was designed here to implement the Inside-Outside algorithm. It is a matrix but only top-right half will be used since it is a symmetrical matrix. This method is adopted from Yao and Lua(1998). The size of this chart depends on the number of words in the sentence being parsed. It means that for an *n*-word sentence the chart is an $n \times n$ matrix. In this study this matrix is restricted to the size 4x4 as only 4-word sentences are considered. Figure 5.4 shows an example of 4-level-chart, which can parse 4-word sentences. The element $N_{(i,j)}$ ($i, j \in [1, 4]$) in the figure denotes a non-terminal node and the element $N_{(1,4)}$ is the starting symbol if each $N_{(i,j)}$ is supported by a grammar rule. The non-terminal node $N_{(i,i)}$ ($i \in [1, 4]$) in the diagonal supported by the rule $N_{(i,i)} \rightarrow w_i$ is the POS of the word w_i.

j

i

$N_{(1,1)}$	$N_{(1,2)}$	$N_{(1,4)}$	$N_{(1,3)}$
	$N_{(2,2)}$	$N_{(2,3)}$	$N_{(2,4)}$
		$N_{(3,3)}$	$N_{(3,4)}$
			$N_{(4,4)}$

Figure 5.4 4-level-chart

Generally, for an n-word sentence $w_{1,n}$, if each non-terminal node is supported by a grammar rule, there are *n* terminal nodes in level 1 which is the diagonal, *n*-1 non-terminal nodes in level 2, ..., and one non-terminal node (the starting symbol) in level n. Let $P(N_{(i,j)})$ denote the probability of node $N_{(i,j)}$, then each node $N_{(i,j)}$ (*i*=*j*, and *i*, *j*∈[1, 4]) in level 1 is determined by $P(N_{(i,j)}) = P(N_{(i,j)} \to w_k)$, which is completely handed by the tagger.

Deleted:

Calculating the inside probability of $N_{(i,j)}$ we can then obtain $\max_{0<t_i \le T(n)} \{P(t_i)\}$ by searching the whole parse tree space. This will also result in the calculation of $P(w_{1,n})$, if there is an interest to determine how grammatical a sentence is in the language.

Deleted: Based on the chart, to calculate the probability of $N_{(i,j)}$ is to

Deleted: e

5.4 The Configuration of the Parser

- **The Input & Output Interface**

Sentences can be input one by one or all at once from a file for parsing. For example, if we take the sentence,

wätadärocu wädä gibiaĉäw gäbu#

Here # is used as the end of sentence character.

Deleted: .

The tagger tags each word with the appropriate POS and outputs the following format that the parser would use as an input.

wätadärocu\N wädä\PREP gibiacäw\N gäbu\V #\PUNC

This is the actual output of the tagger. When presented with more than one sentence, the tagger outputs the tagged sentences in a sequential manner. But the parser inputs the sentences line by line. Therefore some editing was done manually in order to make the tagged sentences acceptable by the parser. The parser then extracts each word and each POS tag from the tagged sentence and stores it in one-dimensional array variables.

The output of the parser gives the probability of the selected parse structure, the parse result, and the grammar rules used in parsing. For the above sentence, the outputs are:

The probability of the parse structure : 0.126000002026558

The parse result:

S (NP (N wätadärocu))

VP((PP (PREP wädä) (N gibiacäw))(V gäbu)))

In addition to presenting the output in the above format, the parser also presents output in a tree form. This diagram is shown in figure 5.5.

The rules used in parsing:

S --> NP VP

NP --> N E

VP --> PP V

PP --> PREP N

Tree Structure

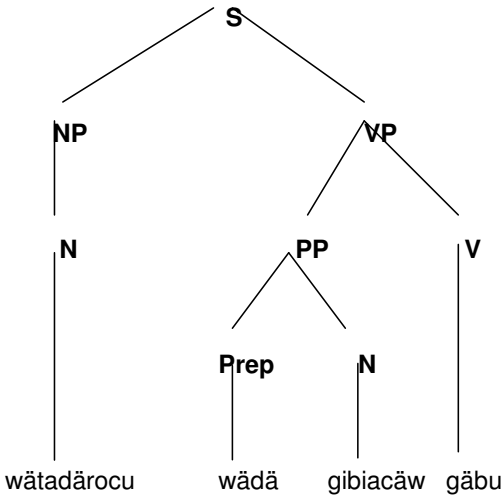


Figure 5.5 A Tree Structure for the parse of the sentence "wätadärocu wädä gibiacäw gäbu".

Deleted: .

The rules used in parsing are the grammar rules used and the rules such as PREP → wädä are not displayed because these kinds of lexical rules are not included in the PCFG since the parser inputs the pretagged sentences and lexical information is handled by the tagger.

- **The Probabilistic Rule Base**

The PCFG rules, which are in CNF, were extracted form the sample corpus and are represented in the database in the following format:

LHS	RHS1	RHS2	Probabilities
NP	N	E	0.35
NP	N	N	0.2
NP	Adj	N	0.12
NP	N	PP	0.01
PP	PREP	N	0.85
PP	N	PREP	0.15

Figure 5.6 PCFG representation

Where E is an empty production.

The field LHS stores the left hand side of the rules while RHS1 and RHS2 represent the first and the second right hand side rules respectively. The probabilities field represents the associated probability values for each of the grammar rules.

Deleted: .

Deleted: For a complete list of the PCFG rules extracted from the corpus see Appendix 5.1.¶

Word information needed in the parsing is stored in the Word code and Lexical Probabilities tables that were implemented by the tagger and which are stored in the same database as the PCFG table.

- **The chart parsing module**

This is the PCFG Inside-Outside algorithm assisted by the bottom up parse chart. During parsing the POSs of words in a sentence are fed one by one into the first level (the diagonal) of the chart. Here, the parse tree space is constructed bottom up and the probability of each parse is calculated as well. The algorithm used in this study, which is a combination of the bottom up chart parsing algorithm and the inside outside algorithm,

with a 4x4 array to act as a chart, is given below. The algorithm below is the modified version of the original algorithm used by Yao and Lua (1998).

Do Until there is no input left

If the agenda is empty, look up the interpretations for the next word in the input and add them to the agenda

Select a constituent from the agenda

For each rule in the grammar of form X → C X_1 …Xn, add an active arc of form X → oC X_1 …Xn from position 1 to 2 .

Add C to the chart using an arc extension algorithm.

Figure 5.7 Bottom Up Chart Parsing Algorithm

To add a constituent C from position 1 to 2:

Insert C into the chart from position 1 to 2.

For any active arc of the form X → X1 … oC … Xn from position 0 to 1, add a new active arc X → X1 … Co … Xn from position 0 to 2.

For any active arc of the form X → X1 … Xn oC from position 0 to 1 , then add a new constituent of type X from position 0 to 2 to the agenda.

Figure 5.8 The Arc Extension Algorithm

For each sentence to be parsed

Extract the words and tags from the output of the tagger.

Insert the POS tags in the diagonal of the array

Call the Inside Probability calculating procedure to fill the POS array.

Get the highest probability from the Probability array.

Get the rules that produced the highest probability from the chartentry table.

Generate the parse structure and display the output.

Figure 5.9 Modified Algorithm

An example is given below to report the implementation of the algorithms. The same example given for the Input-Output is used here for elaboration.

wätadärocu\N wädä\PREP gibiacäw\N gäbu\V #\PUNC

The first thing that the program would do is extracting the words and POS tags. The words are stored in an array for later parse construction. The POS tags are inserted into the diagonal of the POS matrix, which will now have the form:

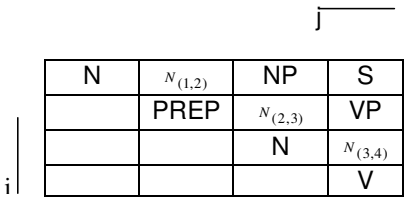


Figure 5.10 Chart Structure During Parsing

$N_{(1,2)}$ will be replaced by a non-terminal from the PCFG table in the database. What happens here is that the records in the table are searched for rules that produce N and

PREP at the right hand side. These rules (could be one or more) and the respective probability values are stored in the chart entry[13] table. The (1,3) entry in the matrix is replaced by “NP”, the (1,4) entry by “S”, and the (2,4) entry by “VP” since all the sentences share these constructions at the top level. The remaining (1,2), (2,3), and (3,4) entries are replaced by the left hand side of the highest probability rules that generated the pair of POS tags in the diagonal. In this example the entry (1,2) is replaced by PP, since the highest probability constituent to produce the sequence N PREP at the right hand side of a rule is the PP (Prepositional phrase). Entries (2.3) and (3,4) are replaced in the same manner. Now the matrix has the form:

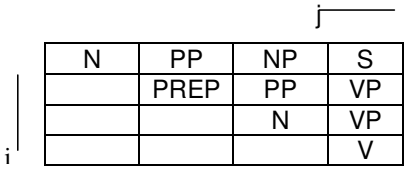


Figure 5.11 A complete POS Matrix

The chart entry table now holds all rules expanded so far. The Inside probability calculation for each possibility of a four word sentence is done at this point. It works as a search through the parse structure space that is represented using the POS matrix above. The calculated probabilities are then stored on a probability array and the rules in the chart entry are marked with the corresponding probability identifier value. Once the highest probability structure is identified, the rules that produced the probability are retrieved from the chart entry table, and the corresponding parse structure is constructed using these rules, the words array and opening and closing brackets.

[13] The chart entry table is a table designed to hold all active arcs, i.e. rules that were expanded so far.

Some of the word categories assigned by the tagger are not consistent with the category assignment used in this study. For example, the tagger assigns “J” for adjectives. This assignment is converted to an “Adj” since the representation in the PCFG table is that of the later one. Also conversions that are needed to summarize categories (e.g. Summarize N, NV, NB to N) are handled before the words are stored in the words array.

The parser is diagrammatically represented in the figure below.

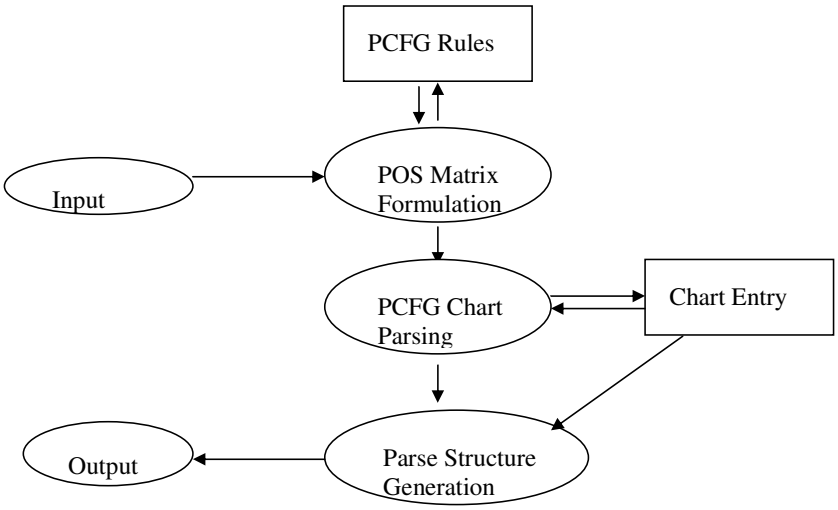


Figure 5.12 Diagrammatic Representation of the Parser

5.5 Experimentation

The sample text selected, which was discussed in chapter four, was used for experimentation. Each sentence in the corpora had been tagged and hand parsed by the researcher, with comments and suggestions from linguists. The 100 sample sentences were selected randomly based on the distribution of the different phrase structures, using judgment sampling technique. One of the criterion set in selection is that the inclusion of two or lesser words in the noun phrase. The probability calculations for the words in the sentences, grammar rule induction, and probability assignment to the grammar rules, which were discussed in chapter 4, were conducted using the 80 sentences randomly picked from the sample corpus. The rest 20 sentences from the original corpus were used as a test corpus.

The first test was done on the 80 sentences which were used in the PCFG induction. Three more tests were made using the test set, a randomly picked set from the sample corpus and a set that was not included in the sample. See the output of the parser for these sets in Appendix 7. The findings are reported in the next section.

5.5.1 Findings

The evaluation technique used in estimating the accuracy of the parser is to simply count the number of correctly parsed sentences and divide it to the total number of parsed sentences. Although there are other evaluation techniques proposed for parsers (such as the PARSEVAL), this technique is employed for the purpose of simplicity. The test on the training set gave a**100% accuracy** since all sentences were parsed correctly. This could be accounted to the fact that the sentences were uniform (have the same

kind of constructs) and that the highest probability constructs are the once that could generate the correct parse structure.

The rules collected from the training corpus seemed to be complete enough to support all the sentences in the test set, and the probabilities were found to be accurate enough to determine the correct parse structure. This conclusion is made due to the results obtained from experimenting with the selected sets and evaluating the outputs (by comparing with the hand parsed sentences and evaluation made by linguists). From the 20 test sentences, kept as a testing set, 19 of them were parsed correctly. The one sentence that was misparsed was due to an error made by the tagger. The parser totally depends on the output from the tagger in the assignment of POS tags. It has no way of counter checking and handling of the mistagged data. In this case the accuracy was found to be:

Accuracy = (19/20) * 100 = 95%

The error that was made by the tagger, which was a misspelled word, was corrected and all sentences were parsed correctly, giving a **100 % accuracy,** also in the test set.

The result obtained from the training set was found to be encouraging. Therefore, 60 more sentences were selected from the sample data, again randomly, and they were tested using the parser. This time, only 2 of them were misparsed. And again, the error occurred due to mistagged and untagged input from the tagger. In this case the accuracy was found to be:

Accuracy = (58/60) * 100 = 96.66%

Again, the error made by the tagger was corrected (which was caused by a mistagged word which was misspelled in the database and an untagged word, which was not included in the database) and a **100 % accuracy** was attained.

To handle the untagged word, an algorithm was designed and the category "UNC" which is assigned by the tagger to words, which are not found in the Word Code table, was guessed. The module was embedded in the parsing program and it made use of bi gram probabilities. This is done by making use of a transition probabilities table. The algorithm designed is given below.

For each word with a UNC category
Get the category for the previous word

If there is no previous word, then

Assign it the category Noun
Else if there is a previous word, then
Assign it the category that has the highest probability of following the category of the previous word in the bi gram transition table.

Figure 5.13 Algorithm to Guess Unknown Word Categories

This algorithm is implemented and guessed categories for unknown words. It was tested using 50 words selected randomly and it was found to guess 64% of them correctly.

Another set of 22 4-word sentences was collected from people other than the researcher and these sentences were tagged and parsed. This time there was no error due to mistagging, but 5 of the sentences were misparsed. The reason for the error is due to

unavailability of the rule that could generate the correct parse structure. That is, since there were no sentences in the training set with the kind of constructs that these five sentences have, the parser couldn't get the required rule from the PCFG table. An example from the misparsed sentences is given below, see Appendix 7 for details.

Example: The tagged sentence "suri\N yäläbäsäc?w\N l?j\N tam?raläc\V" was parsed wrongly as:

```
.   S (NP (N  suri )
        VP(( NP (N  yäläbäsäc?w ) (N  l?j ))( V  tam?raläc )))
```

The correct parse for the sentence should be:

```
S (NP (N  suri ) (NP (N  yäläbäsäc?w ) (N  l?j))
   VP (V  tam?raläc ))
```

This error occurred since there is no rule in the PCFG extracted that could generate the correct parse. The missing rule for this particular parse was:

NP → N NP

The accuracy for this test is found to be:

Accuracy = (17/22) * 100 = 77%

The result obtained for the first training set was ideal. All the sentences included in this set and the 20-sentence test set are sentences which are composed of one or two words in the noun phrases. None of these sentences had three word noun phrases or noun phrases composed of two nouns. Due to this, the parser parses all sentences which constituted of three words in the noun phrase, and sentences with two nouns in the noun phrase, wrongly. Therefore additional 30 sentences were included in the

training set. These sentences constituted of two or three word noun phrases, which have two or more nouns in the noun phrases. The PCFGs were recalculated using the 110 sentences, and **the training set was tested again**. The test result this time gave an **accuracy of 81%**, since 89 of the 110 sentences were parsed correctly.

The 21 sentences in the training set which were parsed wrongly were all composed of noun phrases with more than one word. Although some sentences in the training set now contained the same type of constructs as the sentences in the second test set, when the second test set was run, the 5 sentences that were misparsed in the first run were misparsed again. This time it was due to a **very low probability assignment** to the rule that could generate the correct parse structure. This low probability was assigned because this kind of rules were quite very few in the training set selected, and the inside probability for such constructs is lower than the one selected by the parser.

5.5.2 Solution to Identified Problems

Some of the errors encountered were due to mistagged and untagged words. To deal with the untagged words problem indicated above, a module was implemented to guess the word categories for untagged words by using bi gram lexical co occurrence probabilities. This solves the problem slightly, but still it has no sure way of handling mistagged words. To make the guessing more accurate, tri gram transition probabilities and morphological information of the words could be utilized.

The problem of misparsing due to omitted rules, could be referred to as the problem of *under generation*. Under generation occurs when a parser is presented with new

sentences, for which the correct analysis cannot be assigned, or in the limit, for which no analysis is possible. These situations may arise, through deficiency in the syntactic rules or lexicon, or simply where the input is ill formed or extra grammatical (Briscoe and Waegner, 1992). One approach to this problem is to iteratively develop the grammar, adding supplementary rules, and re-analyzing the failed sentences. And this would solve the problem stated above.

This method (reanalyzing misparsed sentences and adding supplementary rules) was tried on the second test data. The sentence:

Kassa tolo tolo täramädä

With the POS tags

Kassa\N tolo\ADV tolo\ADV täramädä\V #\PUNC was reanalyzed.

There was no rule to produce the correct parse. Therefore the parser produced

S (NP (N Kassa)
VP((ADV tolo)(VP (ADV tolo)(V täramädä))))

To produce the correct parse the rules given below were required.

AdvP → Adv Adv

VP → AdvP V

These rules were added to the PCFG extracted. And the correct parse given below was obtained.

S (NP (N Kassa)

VP((AdvP (ADV tolo) (ADV tolo))(V täramädä)))

The second problem of misparsing arose due to the low probability assigned to the rules generating the correct parse structure. This could be solved by using algorithms such as the Inside Outside algorithm to automatically extract grammar rules from the training set, and re-estimate these rules by running it iteratively. Due to time limitations, solutions to the problem were not implemented and tested.

Chapter Six

Conclusion and Recommendations

6.1 Conclusion

The purpose of this study was to design and develop a sentence parser for Amharic texts. In this thesis, an attempt is made to describe how to develop an Automatic sentence parser for Amharic language using the probabilistic Inside Outside algorithm supplemented with chart parsing algorithm that implemented Probabilistic Context Free Grammars. A prototype parser was developed for four word sentences in the language.

The thesis began with a brief discussion on the role that NLP plays in enhancing computers' capability to process natural language. In this discussion, it is indicated that all applications of NLP have the common objective of understanding and extracting meaning from a natural language input. This process involves transforming the natural language into a form where the meaning is explicit and is easily usable by the application program. As a way to this end, sentence parsing, a process in which a flat input sentence is converted into a hierarchical structure that corresponds to the units of meaning in the sentence, is discussed as one task in the step towards achieving the above stated objective.

Also discussed are important concepts and terms in relation to sentence parsing. Rule based and statistical approaches, which are the major approaches to sentence parsing, and the different grammatical formalisms used to represent phrase structure rules in a language were briefly reviewed. A discussion on Amharic words, phrases and sentences is also included.

The PCFG parsing was adopted in this research to develop the Amharic parser. Reasons for such preference and, the step-by-step procedures followed to extract the grammar rules, assignment of the probabilities and automatic selection of the best parse from the parse space were discussed. The mathematical formulations related to this approach were also presented in some detail.

A sample corpus prepared as part of (and for the purpose of) this study was used to generate phrase structure rules and to estimate probability values. A POS tagger developed by Mesfin (2001) was used to automatically tag the words in a sentence. The developed parser used the output of the tagger as an input.

The algorithms employed generate all possible parse structures for a given sentence, and calculate the probabilities of each possibility, to return the structure of the most probable parse. A database of PCFGs, which is used by the parser in determining the best parse is also designed and implemented. The thesis presented the algorithms and modules developed for use by the parser to access the database and parse incoming sentences. For this purpose an interface, which allows a user to parse four word sentences, was developed using Visual Basic 6.0. The step-by step description of the system is included in the report.

Experiments were conducted in three phases, one on the training set, the second on the test set, and the third one on a set of four word sentences, different from the ones used as a training and testing set, obtained from different people. The first two sets were obtained from the sample selected for the purpose of the experiment. Evaluation of the

parser performance was made based on judgment of experts, and manual counts. In the study, only one parameter, the percentage of correct parse assignment as compared with the hand parses, was used to measure the performance of the parser.

The results achieved based on the first set of sample sentences was very high, 100% on the training set and approximately 96% on the test set. Before achieving such accuracy, the experiment was repeatedly done on both the training set and the test set with identifying errors and making corrections. This high accuracy is obtained partly due to the small number of words considered for the purpose of the experiment. Another reason is that all the sentences have identical constructions and the highest probability parses were almost always the correct ones.

Another set of sentences which were not included in the sample selected was parsed and the result obtained was 77%. All errors identified with the tagger were due to human made errors (therefore easily fixed) and untagged words. The untagged words problem was slightly handled by a module to guess the POSs of unknown words. The errors made by the parser were due to incomplete PGFGs (under genertion) which were caused by the use of a very small corpus, and low probability assigned rules that could generate the correct parse for the misparsed sentences. In order to deal with the first problem, another set of 30 sentences (with a different construction than that of the first set) was selected from the language and the PCFGs were re-estimated. This time the accuracy of the parser was found to be 81%. And the accuracy of the second test set remained to be 77%, this time the errors were due to low probability assignments.

In general, the experiments conducted, the results achieved, and a discussion on the possible causes of errors were presented with their solutions in the last part of the report.

Although the accuracy of the parser developed in this study is high, additional work is still needed to improve it to handle all kinds of Amharic sentences. In the tests conducted an average 85% accuracy is obtained. Although this figure is very encouraging, it cannot be generalized to determine the performance of a PCFG parser for the language, due to the small sample size taken. Further work is still required to ensure that the PCFGs are representative enough and the probability values are accurate enough.

In this study, the PCFG extracted is limited to a small sample data for various reasons. The reasons being:

- Lack of large corpora in the language annotated with POS tags and sentence parses.
- Manually processing and generating such quantities of data is expensive and time consuming.
- Financial constraint to carry out such huge research.
- Scope of the thesis, which is limited to demonstrating the potential of statistical approaches to develop an automatic parser to Amharic language.

Also, in this study the sentences considered are only four-word Amharic sentences. This again, is due to:

- Shortage of time available for coding and testing
- Shortage of hand parsed and tagged corpus

The researcher had to tag and parse the sample selected. In the time available only a few sentences were selected and processed (i.e. tagged and parsed), grammar rules were generated and the rule probabilities were assigned. The rule probabilities are static, i.e. no dynamic probability re-estimation is used, also due to time constraints.

Apart from these limitations, the research has indicated the possibility to construct or develop a sentence parser for Amharic language. As this work is an initial attempt in the area of probabilistic parsing for the Amharic language, it is hoped that the results found will encourage future researches in natural language processing of Amharic in general and parsing in particular.

It is also hoped that this work in Amharic sentence parsing will encourage Ethiopian students and researchers to take part in the area which ultimately lend to a higher level and more demanding research endeavors such as semantic analysis, discourse integration and machine translation, which are all tasks of NLP in general and NLU in particular.

6.2 Recommendation

The sample taken for this study and the grammar rules generated cannot be taken to be a representative of the language, and future researches should be conducted using a larger set of corpus.

The following areas could be explored further as a continuation of this study.

- Further researches in the NLP of Amharic such as semantic analysis, machine translation, spell checking and grammar, information storage and retrieval systems etc.
- It can be enhanced to handle complex sentences in the language, and to develop a full fledged Amharic sentence parser.
- The performance of PCFGs for natural language processing of the language could be explored, after enhancing it to handle complex sentences first.
- It could initiate the need to develop processed corpus (i.e. hand tagged, parsed, etc data) in the language for experimentation and researches on statistical natural language processing.
- The grammar rules extracted and the corresponding probabilities are all static values. Both the grammar induction and the corresponding probability computations could be made dynamic, i.e. to recalculate the probability values and add new grammar rules, during the parsing of sentences.
- In this study, the bi gram lexical co occurrence was used to guess for unknown words. This technique is not found to be very promising. Morphological analysis can provide some clues about the syntactic category of unknown words. A

research on a word parser for Amharic has already been conducted and also a research on statistical morphological analysis for the language is being conducted by Tesfaye Bayu, a fellow classmate. Although morphological analysis is not sufficient to solve the problem of new words entirely (due to morphological ambiguity), combining statistical lexical co occurrence techniques and morphological analysis could produce a better result. And this area could be explored in future researches.

- Integration of works that are already conducted and are being conducted in the area of NLP of Amharic is necessary in order to ensure the continuity of works and to come up with an end result. An attempt was made in this study to integrate current and previous works and it is hoped that future researches would take an initiative in integrating works.

References

1. Abiyot Bayou. *Design and Development of Word Parser for Amharic Language*. Masters Thesis, Addis Ababa University. 2000.
2. Abney, Steven. *Statistical Methods and Linguistics:* In Judith Klavans and Philip Resnik, eds., The Balancing Act. MIT Press, Cambridge, MA. 1996
3. Allen, James. *Natural language Understanding*. Redwood City: Benjamin/ Cummings. 1995.
4. Baker, James K. Trainable grammars for speech recognition. In Jared J. Wolf and Dennis H. Klatt, editors, *Speech communication papers presented at the 97th Meeting of the Acoustical Society of America*. Acoustical Society of America, MIT Press, June 1979.
5. Baye Yimam. *Yä Amarəňa Säwasəw.* ; E.M.P.D.A, Addis Ababa.1987.
6. Bender, M. L., Sydney W. Head, and Roger Cowley . The Ethiopian Writing System, In Bender et al (Eds.) Language in Ethiopia. London: Oxford University Press. 1976.
7. Black, Ezra. *Evaluation of Broad-Coverage Natural-Language Parsers.* Interpreting Telecommunications Laboratories, ATR, Kyoto, Japan. http://cslu.cse.ogi.edu/HLTsurvey/ch13node6.html#392
8. Briscoe, Ted; Waegner, Nick. Robust Stochastic Parsing Using the Inside Outside Algorithm. *Proceedings of AAAI: Workshop on Probabilistically Based Natural Language Processing Techniques*. San Jose. 1992.
9. Cahill, Aoife. *Probabilistic Context Free Parsing*. 2000. www.compapp.dcu.ie/~away/PROJ3/99-00/cahill.html
10. Charniak, Eugene. *Statistical Language Learning*: MIT Press, Cambridge, MA 1993.

11. Charniak, Eugene. *Statistical Techniques for Natural Language Parsing*. Brown university. 1997.
12. Chomsky, Naom. *Syntactic Structures*. Netherlands: Mouton & Co. 1957.
13. Chomsky, Naom. Aspects of the Theory of Syntax. MIT Press. Cambridge, Massachusetts, 1965.
14. Dawkins, C.H. *The Fundamentals of Amharic*. A.A Sudan interior mission. 1969
15. Dempster, A. P.; Laird, N. M.; Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1--38, 1977.
16. Fisher, *Practical parsing of generalized phrase structure grammar*, Computational Linguistics, Vol.15, No.3, 1989.
17. Gazdar, Gerald; Mellish, Chris. *Natural Language Processing in Prolog.* http://www.cogs.susx.ac.uk/local/books/nlp-in-prolog/ch01/chapter-01-sh-1.6.html#sh-1.6. 1996.
18. Getahun Amare. *Zämänäwi yäamarɔñña Säwasäw bäqälal aqäraräb*. Commercial printing Press: Addis Ababa. 1989
19. Grishman, Ralph. *Computational Linguistics: An Introduction*. New York: Cambridge University Press. 1994.
20. Harris, M. Introduction to Natural Language Processing. Virginia: Reston. 1984.
21. J. Lee, J. C. Dai, and Y. S. Chang, Parsing Chinese nominalizations based on HPSG, *Computer Processing of Chinese and Oriental Languages*, Vol.6, No.2, 1992.
22. Joshi, Aravind. *Parsing Techniques*. Pennsylvania: University of Pennsylvania. http://cslu.cse.ogi.edu/HLTsurvey/ch11node6.html

23. Kay, M. Functional grammar. In *Proceedings of the Fifth Annual Meeting of the Berkeley Linguistic Society*, 1979.
24. Lari, K.; Young, S. J.. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language Processing*, 4:35--56, 1990.
25. Lari, K.; Young, S. J.. Applications of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language Processing*, 5. 1991.
26. Lenat, Douglas B.; Guha, Ramanathan V; Pitmann, Karen; Pratt, dexter; Shepherd, Mary. CYC: Towards Programs with Common Sense. *Communication of the ACM*. V 33 (8). 1990.
27. Leslau, Wolf. *Amharic Textbook.* California: Berkley University. 1968.
28. Leslau, Wolf (1973) English –Amharic context Dictionary otto Harrassowitz, wiesaden, Belgium.
29. Manning, Christopher D and Schütze, Hinrich. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA. 1999
30. Mao, Yonghong. *Natural Language Processing Module: Part of Speech tagging and Sentence Parsing, Laboratory Manual*. MIT, 1997.
31. Merlo, Paola. *Parsing with Principles and Classes of Information*. Dordrecht: Kluwer Academic publishers. 1996.
32. Mersehazen woldeqirqos (1934) yäamarðň säwäsäw, Birhanena Selam Printing Press, Addis Ababa.

33. Mesfin Getachew. *Automatic Part of Speech Tagging for Amharic: An Experiment Using Stochastic Hidden Markov (HMM) Approach*. Masters thesis. Addis Ababa University. 2001.
34. Metzer, Douglas P.; Haas, Stephaine W.; Cosic, Cynthia L; Wheeleer, Leslie H. Constituent Object Parsing for Information Retrieval and Similar Text Processing Systems. *JASIS*, 40(6). 1989.
35. Meyers, Adam; Kosaka, Michiko; Grishman, Ralph. *Chart-Based Transfer Rule Application in Machine Translation.* http://www.cs.nyu.edu/cs/projects/proteus/publication/papers/meyers-coling00.ps.
36. Pereira, Fernando. *Sentence Modeling and Parsing* AT&T Bell Labs, New Jersey, USA. http://cslu.cse.ogi.edu/HLTsurvey/ch3node2.html#Chapter3.
37. Pereira, Fernando C. N.; Schabes,Yves. Inside-outside reestimation from partially bracketed corpora. *ACL*. 1992.
38. Radford, A. *Transformational Syntax*. Cambridge University Press. 1981.
39. Rauch-Hindin, Wendy B. *Artificial Intelligence In Business, Science & Industry: Volume I: Fundamentals.* New Jersey: Prentice-Hall, 1986.
40. Rich, Elaine; Knight, Kevin. *Artificial Intelligence*. 2nd ed. New York: McGraw-Hill. 1991.
41. Schutzer, Daniel. *Artificial Intelligence: An Applications-Oriented Approach*. New York: Van Nostrand Reinhold Company. 1987.
42. Stolcke, Andreas. *An Efficient Probabilistic Context-Free Parsing Algorithm that Computes Prefix Probabilities* Berkley, CA. 1993.

43. Tesfaye Bayu. *Automatic Morphological Analyser: An Experiment Using Unsupervised and Autosegmental Approach*. Masters thesis. Addis Ababa University. 2002.
44. *The 1994 Population and Housing Census of Ethiopia: Results at Country Level: VI Statistical Report*. Federal Democratic Republic of Ethiopia Office of Population and Housing Census Commission. June 1998.
45. Tremblay,J.P.; Sorenson, P.G. *An Introduction to Data Structures with Applications*. New York: McGraw-Hill. 1976.
46. W. J. Ketty Yen, The Design and Implementation of a Cascaded Morphological Parser, *Computer Processing of Chinese and Oriental Languages*, Vol.7, No.2, 1993.
47. Winograd, Terry. *Language as a Cognitive Process Volume 1: Syntax* Adison-Wesley Publishing Company. 1983.
48. Woods, William A.(1970). Transition Network Grammars of natural Language Analysis. *Communication of the ACM*. Reprinted in Barbara J. Grosz, Karen Spark jones, & Bonnie Lynn Webber (eds.). *Readings in Natural Language Processing.* Los Altos, USA: Morgan Kaufnann. 1986.
49. Yao, Yuan; Lua, Kim Teng. *A Probabilistic Context-Free Grammar Parser for Chines.* Department of Information Systems & Computer Science. National University of Singapore. http://www.comp.nus.edu.sg/~luakt/paper/199805.doc 1998.

Appendices

Appendix 1. The Amharic Alphabet, extracted from Leslau (1976)

ሀ	ሁ	ሂ	ሐ	ሄ	ህ	ሆ
hä	hu	hi	ha	he	h	ho
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
Lä	lu	li	la	le	l	lo
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
hä	hu	hi	ha	he	h	ho
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
mä	mu	mi	ma	me	m	mo
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
sä	su	si	sa	se	s	so
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
rä	ru	ri	ra	re	r	ro
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
sä	su	si	sa	se	s	so
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
šä	šu	ši	ša	še	š	šo
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
qä	qu	qi	qa	qe	q	qo
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
bä	bu	bi	ba	be	b	bo
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
tä	tu	ti	ta	te	t	to
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
čä	**ču**	**či**	**ča**	**če**	**č**	**čo**
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
hä	hu	hi	ha	he	h	ho
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
nä	nu	ni	no	ne	n	no
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
ňä	ňu	ňi	ňa	ňe	ň	ňo
አ	ኡ	ኢ	ኣ	እ	ኧ	ኦ
a	u	i	a	e	ə	o
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
kä	ku	ki	ka	ke	k	ko
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
hä	hu	hi	ha	he	h	ho
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
wä	wu	wi	wa	we	w	wo
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ

a	u	i	a	e	ə	o
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
zä	zu	zi	za	ze	z	zo
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
žä	žu	ži	ža	že	ž	žo
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
yä	yu	yi	ya	ye	y	yo
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
dä	du	di	da	de	d	do
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
ğä	ğu	ği	ğa	ğe	ğ	ğo
ገ	ጉ	ጊ	ጋ	ጌ	ጅ	ጆ
gä	gu	gi	ga	ge	g	gop
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
ṭä	ṭu	ṭe	ṭa	ṭe	ṭ	ṭo
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
ćä	ću	ći	ća	će	ć	ćo
ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
ρä	ρu	ρi	ρa	ρe	ρ	ρo
ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
şä	şu	şi	şa	şe	ş	şo
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
şä	şu	şi	şa	şe	ş	şo
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
fä	fu	fi	fa	fe	f	fo
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
pä	pu	pi	pa	pe	p	po

Appendix 2. List of Punctuation Marks In Amharic

No	Punctuation mark	Symbol	Purpose
1	The four dots (or double colon)	::	Marks end of a word and at the same time the end of a sentence
2	Colon	(:)	Separate individual words in a sentence, the
3	White space		Separate individual words in a sentence, the current practice
4	Question mark	∂	Marks the end of an interrogative sentence
5	Exclamation	!	Used at the end of such sentences or interjections that express such emotions as…
6	Semi-colon		Serves roughly the same function as comma
7	Semi-colon		It separates related meanings. That is, serves roughly the same function as semicolon
8	Semi colon		It separates unrelated meanings. That is, it serves roughly the same functions as comma
9	Three dots	…	Marks deliberate omission of words, phrases or sentence
10	Quotation marks	“ “ ‘ ’	Used at the beginning and end of words that are being quoted
11	Parenthesis	( )	Encloses elaboration of Amharic meanings
12	Stroke	|	Separates date, month, year on official let’ers of an organization
13	Vertical stroke	‘	Indicates stress or gemination in Amharic.
14	Mocking mark		Placed at the end of a mocking sentence

Appendix 3. Transcribed Sample Sentences

First Training Set

1. wätadäroĉu wädä gibiaĉäw gäbu #
2. Asəter likämänəbärun läsəra t'äraĉəw #
3. Ayälä təndəš injära bäla #
4. Läma bäsal mäməhər näw #
5. Bälät'ä təndəš bet aläw #
6. kokäboĉ bänəgat bät'äm yabärralu #
7. Kassa läAsəter gänəzäb sät'at #
8. Kassa läsäwəyew misət'ər nägäraĉew #
9. Kassa bet w?sət' alä #
10. Kassa bät'äm ələhäňa näw #
11. Ləjitu bät'äm amäläňa näĉ #
12. Bälät'ä wädä Gojam hedä #
13. Bälät'ä ĉəqa lay wädäqä #
14. säwəyew bet dəräs šäňäň #
15. ləjoĉu ənədä wäf yəbäralu #
16. Asəter ənədä həte wädäqäĉ #
17. Kassa təlləq bäg gäza #
18. Asəter hulät ləjoĉ aluwat #
19. ləjoĉu kəfəl wəsət' alu #
20. Kassa wädä gäbäya hedä #
21. Asəter wädä bet gäbaĉ #
22. ləjoĉu zaf lay wät'u #
23. ləjoĉu bäru lay qomu #
24. Asəter aləgaw lay wädäqäĉ #
25. dorowoĉu qot'u lay adäru #
26. əsaĉäw wəhaw lay wädäqu #

27. asətämariw lätämariwoĉu mäşähaf säťaĉäw #
28. Kassa säw əjəg yakäberal #
29. Kassa bätäsiyat məsawən bäla #
30. Kassa bet wəsəť näbär #
31. Kassa təndəš mäşähaf šäťä #
32. Asəter läləju gänəzäb lakäĉəlät #
33. Asəter ləjitəwan şägurəwan säraĉat #
34. Asəter wädä Gonədär hedäĉ #
35. Asəter əjəg wäfəram nat #
36. ləjətəwa kurəť ənatəwan təmäsəlaläĉ #
37. Asəter təllək wänədəm alat #
38. Asəter tanaš əhət alat #
39. Asəter bäť äwat buna täfälaläĉ #
40. ləju abatun bähayəl täsadäbä #
41. Bäläť ä wädä Gojam hedä #
42. Bäläť ä zaf lay wäť ä #
43. säwəyew bet dəräs šäňäň #
44. säwəyew bäťam räjəm näw #
45. ənsəsoĉu ənədä assa yəwaňalu #
46. Kassa tələq mäşähaf yanäbal #
47. Kassa läləju gänəzäb lakälät #
48. ənate ləjitəwan şägurəwan säračat #
49. əhəte wädä Gonədär hedäč #
50. Asəter tənanət bädənəgät täťäläfäč #
51. Kassa tənanət məsawən bälətoal #
52. əne anəbäsa gädəye näbär #
53. Kassa zare anəbäsa yəgädəlal #
54. Kassa nägä anəbäsa yəgädəlal #
55. Kassa zäwätər anəbäsa yəgädəlal#
56. Kassa ənəčät əyäfälätä näbär #
57. säwoĉu ələqit lay naĉäw #

58. wätadäroĉu gəbəža lay naĉäw #
59. Kassa wärätäňa säw näw #
60. Kassa Asəterən ənəĉĉät asabärat #
61. ləju wädä betu rot'ä #
62. əne qonjo\J ləj ayähu #
63. säwoču bädənəgät rot'äw wät'u #
64. bəzu amätat Gojam norealähu #
65. Asəter təlləq ənəsəra atäbäč #
66. Asəter bät'am qonjo nat #
67. əhəl gotäraw wəsət' alä #
68. Kassa qonjo bet aläw #
69. Asəter bəzu t'äj t'ät'ač #
70. təlləqu säwəye bäčəkola hedu #
71. Asəter qonəjo ləj nat #
72. As?ter təquwaqur šuraboč alwat #
73. ənatəwa qonjo ččama gäzulat #
74. Kassa məsawən əyäbäla näw#
75. Kassa tənanət bähayəl dänägätä#
76. Kassa bəzu nägär tänagärä #
77. Kassa bet wəsət' näbär #
78. nägadew gänəzäb bät'am yəwädal #
79. Kassa bəzu nəbərät täräkäbä #
80. Betty məsawan ənəjära bälaĉ #

Second Training Set

1. tənəŝu wänədəme ələhäňa näw #
2. tələkua əhəte gäbäya hedäĉ #
3. qonəjəyewa ləj zare təmät'aläĉ #
4. ənia ŝəmagəle säwəye motu #

5. yäwän∂d∂me l∂j mis∂t agäba #
6. yäkäbäleaĉ∂n likämän∂bär s∂b∂säba t'ära #
7. yämäs∂ria betaĉ∂n halafi mät'u #
8. yä∂g∂r kuas ĉĉäwata täjämärä #
9. alämaqäf yäsetoĉ qän täkäbärä #
10. t∂n∂n∂ŝ l∂joĉ ĉĉäwata y∂wädalu #
11. t∂l∂l∂q l∂joĉ man∂bäb y∂wädalu #
12. bärari wäfoĉ bät'am yam∂ralu #
13. säkaramu säw∂ye t∂nan∂t aräfu #
14. yäbaläbete ∂nat zare y∂mät'alu #
15. t∂l∂qu l∂je mat∂rik aläfä #
16. yäzän∂d∂ro l∂joĉ bät'am qäb∂t'äwal #
17. yägojam mar bät'am y∂t'af∂t'al #
18. yätägodut wätadäroĉ ∂r∂data agäňu #
19. arän∂guade qäläm däs y∂lal #
20. t∂n∂ŝua l∂j ∂yärot'äĉ mät'aĉ #
21. t∂l∂qu wän∂d∂me s∂ra qäyärä #
22. yät∂m∂h∂r∂t betu däwäl tädäwälä #
23. t∂l∂ku yäabäba mas∂qämäĉĉa täsäbärä #
24. yäbabur hadidu s∂ra tät'änaqäqä #
25. gotäraw w∂s∂t' ∂h∂l alä #

26. betu wəsət' ayət'oĉ alu #

27. yägojam yämar t'äj yət'afət'al #

28. yäBälät'ä asətämari zare yəhedal #

29. yäagote ləj bät'äna tamual #

30. yäEthiopia həzəb mäzəmur täzämära #

First Test Set

1. Ethiopia aränəguade kämis läbäsäĉ #
2. Ayälä ərasun bät'am tamämä #
3. ənate wädä gäbäya hedäĉ #
4. säwəyew bämaläda tänäsəto hedä #
5. wäfoĉu bälelit mäzämär jämäru #
6. Kibur bät'am tatari näw #
7. Kinəfä sägurun mäkorät' yəwädal #
8. Abäbä dabo gäzəto mät'a #
9. fätanu ĉĉala ĉĉube ĉĉäbät'ä #
10. ənatəwa bät'am asəkäyami nat #
11. əne təllək bäg gäzahu #
12. ləju ləbəs əyat'äbä näbär #
13. abate nägä agäru yəhedal #
14. ləjoĉu ənədä ahəya yəragät'alu #
15. Asəter konəjo kämisoĉ aluwat #
16. Asəter säguruan əyätat'äbäĉ näw #
17. Ethiopia bəzu tarik alat #
18. Fäläqä təru səra agäňä #
19. Kassa kosasa ləj näw #
20. Kassa märet lay täňa #

Second Test Set.

1. lɔju wätät bät'am yɔwädal #
2. Kassa tolo tolo täramädä #
3. yäaläm wanĉĉa tɔnanɔt täjämärä #
4. Näbɔrit ɔnɔkɔlɔfuan satɔtäñä adäräĉ #
5. Täsɔfaye šäbäla lɔj näw #
6. abate wädä gäbäya hedu #
7. yäĉayɔna ĉama gäbäyäwɔn at'ɔläkäläkäw #
8. säwɔyew tɔllɔk wɔša aläw #
9. lɔju t'ara lay wät'ä #
10. Qonjit tɔnanɔt mät'ɔta näbär #
11. Käbädä wädä sɔra hedä #
12. säwoĉu foq lay naĉäw #
13. tɔllɔqua setiyo tɔnanɔt motu #
14. hɔşanat bɔzu t'ɔnɔqaqe yɔšalu #
15. guadäñaye tɔru ĉĉäwata yawɔqal #
16. därasiw maqäw maqäw motu #
17. mänoria bete säfi näw #
18. balua bät'am säkaram näw #
19. šɔmagɔlew bät'am atalay naĉäw #
20. bɔrɔqɔyewoĉu yädur arawitoĉ täsädädu #
21. yäɔgɔr kuas ĉĉäwata ɔwädalähu #
22. suri yäläbäsäĉɔw lɔj tamɔraläĉ

Appendix 4. Special Tags Identified by Mesfin (2001)

No.	TAG	DESCRIPTION
1	N	Nouns including all pronouns, invariant for number, gender and case except for verbal nouns and such nouns formed using the prefix balä(e.g. lɔ̈j “chiled”, lɔ̈j-očč “ Children”, ɔ̈ssu “He” ,dägɔ̈nät “kindness”)
2	NV	Verbal nouns (e.g. mäblat “Eating”, mäťäťät “ Drinking.”
3	NB	Noun formed by prefixing the prefix balä to nouns (e.g. balä-bäg “The Šheep owner”, balä-suq “Šhop keeper”, balä-lɔ̈bs “owner of the clothing”)
4	NPrep	A word with a preposition not separated from a noun (e.g. b**ä**m**ä**kina “By car”, sɔ̈lähägär “a bout a country”)
5	NC	A noun suffixed with a conjunction, i.e. a word with noun not separated from a conjunction (e.g. lomina bɔ̈rtukan “Lemon and orange”, zäyts “how about oil”)
6	V	Verb in any form except auxiliary verbs, compound verbs and other forms of the auxiliary and compound verbs (e.g. gäddälä “He killed”, gäddlo “after he killed”, gäddäläč “še killed”)
7	AUX	Auxiliary verbs and all their other forms. This does not include compounds of allä, adärrägä, assäňä and all their other forms (e.g. alä “He, It present” ,näbbär “He, It was”, näw “It is”, nä**č** “Še is”,na**č**äw “They are”)
8	VCO	Compound verbs, i.e. compounds of allä, adärrägä, assäňä, and all their other forms (e.g bɔ̈dɔ̈q allä “He appears”, zɔ̈m assäňto “In a way of making them silent”, bɔ̈dɔ̈g adärrägä “He take it up.”)
9	VPreP	Any verb (main or auxiliary) headed by a preposition. The preposition not separated from the verb (e.g. sɔ̈lämäťťa “Since he came”, kähedä “If he went”)
10	VC	Any verb suffixed or prefixed (i.e. headed) by a conjunction. That is, a word with the conjunction not separated from the verb (e.g. mäťťana “He come and” , sɔ̈mäťťa “When he comes”, sɔ̈zänb “when

		it rains", ∂skt∂ċärs "Until you finiš")
11	J	An adjective which is preceded by neither prepositions nor conjunctions (e.g. däg "Kind", k∂fuňa "Dangerously", t∂ll∂k "Big")
12	JC	An adjective not separated from a conjunction (e.g. däg∂na yäwah "Kind and Innocent", ť∂qurna näčč∂ "Black and white")
13	JNU	A numeral that function as an adjective (e.g hulät b∂rr∂ċ∂qqo " two glasses", amm∂st guräno "five Šeds")
14	JPN	A preposition not separated from a noun but that function as an adjective (e.g. yäť äla b∂rr∂ċ∂qqo "A glass for "tella" ", yäčayna sah∂n "A china made plate")
15	JP	A word with a preposition not separated from the adjective. That is, the adjective is headed by a preposition (e.g. bädäh∂na "In a fine way")
16	PREP	A preposition that appear being not at'ached with other words (e.g. kä "From", lä "To", s∂lä "For Sb/Sth.", ∂ndä "Like")
17	ADV	An adverb (e.g. tolo "In a hurry", t∂lant∂nna "Yesterday", zare "Today", hulg∂ze "Always")
18	ADVC	An adverb which has a conjunction suffixed to it (e.g. ahun∂m "Even now")
19	C	Coordinating conjunctions that appear being not at'ached with other words (e.g. nägär g∂n "However", wäy∂ss "Or")
20	REL	A word which is a relative clause (e.g. yätäsäräqäbät "one who is stolen", yäqomut "those that stand")
21	ITJ	Interjections (e.g. goš! "Wonderful", wa! "Take care! Be careful! Watch out!")
22	ORD	Ordinal number (e.g. amm∂stäňa "The fifth" ,ass∂räňa "Tenth")
23	CRD	Cardinal number (e.g. amm∂st "Five", ass∂r "Ten")
24	PUNC	Punctuation (e.g. **÷, :, !,∂)**
25	UNC	Unrecognized word, i.e. a word not found in the lexicon of the tagger

Appendix 5. Tagged corpus

Training Set

1. wätadärocu\N wädä\PREP gibiacäw\N gäbu\V #\PUNC
2. As?ter\N likämän?bärun\N läs?ra\N ttärac?w\V #\PUNC
3. Ayälä\N t?n?sh\J injära\N bäla\V #\PUNC
4. Läma\N bäsal\J mäm?h?r\N näw\AUX #\PUNC
5. Bälättä\N t?n?sh\J bet\N aläw\AUX #\PUNC
6. kokäboc\N bän?gat\N bättäm\ADV yabärralu\V #\PUNC
7. Kassa\N läAs?ter\N gän?zäb\N sättat\V #\PUNC
8. Kassa\N läsäw?yew\N mis?tt?r\N nägäracäw\V #\PUNC
9. Kassa\N bet\N w?s?tt\PREP alä\AUX #\PUNC
10. Kassa\N bättam\ADV ?l?häna\J näw\AUX #\PUNC
11. L?jitu\N bättam\ADV amäläna\J näc\V #\PUNC
12. Bälättä\N wädä\PREP Gojam\N hedä\V #\PUNC
13. Bälättä\N c?qa\N lay\PREP wädäqä\V #\PUNC
14. säw?yew\N bet\N d?räs\PREP shänän\V #\PUNC
15. l?jocu\N ?n?dä\PREP wäf\N y?bäralu\V #\PUNC
16. As?ter\N ?n?dä\PREP ?h?te\N wädäqäc\V #\PUNC
17. Kassa\N t?ll?q\J bäg\N gäza\V #\PUNC
18. As?ter\N hulät\J l?joc\N aluwat\V #\PUNC
19. l?jocu\N k?f?l\N w?s?tt\PREP alu\V #\PUNC
20. Kassa\N wädä\PREP gäbäya\N hedä\V #\PUNC
21. As?ter\N wädä\PREP bet\N gäbac\V #\PUNC
22. l?jocu\N zaf\N lay\PREP wättu\V #\PUNC
23. l?jocu\N bäru\N lay\PREP qomu\V #\PUNC
24. As?ter\N al?gaw\N lay\PREP wädäqäc\V #\PUNC
25. dorowocu\N qottu\N lay\PREP adäru\V #\PUNC
26. ?sacäw\N w?haw\N lay\PREP wädäqu\V #\PUNC
27. as?tämariw\N lätämariwocu\N mäsähaf\N sättacäw\V #\PUNC
28. Kassa\N säw\N ?j?g\ADV yakäberal\V #\PUNC

29. Kassa\N bätäsiyat\ADV m?saw?n\N bäla\V #\PUNC
30. Kassa\N bet\N w?s?tt\PREP näbär\V #\PUNC
31. Kassa\N t?n?sh\J mäsähaf\N shättä\V #\PUNC
32. As?ter\N läl?ju\N gän?zäb\N lakäc?lät\V #\PUNC
33. As?ter\N l?jit?wan\N sägur?wan\N säracat\V #\PUNC
34. As?ter\N wädä\PREP Gon?där\N hedäc\V #\PUNC
35. As?ter\N ?j?g\ADV wäf?ram\J nat\AUX #\PUNC
36. l?j?t?wa\N kur?tt\J ?nat?wan\N t?mäs?laläc\V #\PUNC
37. As?ter\N t?ll?k\J wän?d?m\N alat\V #\PUNC
38. As?ter\N tanash\J ?h?t\N alat\V #\PUNC
39. As?ter\N bättäwat\ADV buna\N täfälaläc\V #\PUNC
40. l?ju\N abatun\N bähay?l\ADV täsadäbä\V #\PUNC
41. Bälättä\N wädä\PREP Gojam\N hedä\V #\PUNC
42. Bälättä\N zaf\N lay\PREP wättä\V #\PUNC
43. säw?yew\N bet\N d?räs\PREP shänän\V #\PUNC
44. säw?yew\N bättam\ADV räj?m\J näw\AUX #\PUNC
45. ?ns?socu\N ?n?dä\PREP assa\N y?wanalu\V #\PUNC
46. Kassa\N t?l?q\J mäsähaf\N yanäbal\V #\PUNC
47. Kassa\N läl?ju\N gän?zäb\N lakälät\V #\PUNC
48. ?nate\N l?jit?wan\N sägur?wan\N säracat\V #\PUNC
49. ?h?te\N wädä\PREP Gon?där\N hedäc\V #\PUNC
50. As?ter\N t?nan?t\ADV bäd?n?gät\ADV tättäläfäc\V #\PUNC
51. Kassa\N t?nan?t\ADV m?saw?n\N bäl?toal\V #\PUNC
52. ?ne\N an?bäsa\N gäd?ye\V näbär\V #\PUNC
53. Kassa\N zare\ADV an?bäsa\N y?gäd?lal\V #\PUNC
54. Kassa\N nägä\ADV an?bäsa\N y?gäd?lal\V #\PUNC
55. Kassa\N zäwät?r\ADV an?bäsa\N y?gäd?lal\V # \PUNC
56. Kassa\N ?n?ccät\N ?yäfälätä\V näbär\V #\PUNC
57. säwocu\N ?l?qit\N lay\PREP nacäw\V #\PUNC
58. wätadärocu\N g?b?zza\N lay\PREP nacäw\V #\PUNC
59. Kassa\N wärätäna\J säw\N näw\AUX #\PUNC

60. Kassa\N As?ter?n\N ?n?ccät\N asabärat\V #\PUNC
61. l?ju\N wädä\PREP betu\N rottä\V #\PUNC
62. ?ne\N qonjo\J l?j\N ayähu\V #\PUNC
63. säwocu\N bäd?n?gät\ADV rotäw\V wättu\V #\PUNC
64. b?zu\J amätat\N Gojam\N norealähu\V #\PUNC
65. As?ter\N t?l?q\J ?n?s?ra\N atäbäc\V #\PUNC
66. As?ter\N bättam\ADV qonjo\J nat\V #\PUNC
67. ?h?l\N gotäraw\N w?s?tt\PREP alä\AUX #\PUNC
68. Kassa\N qonjo\J bet\N aläw\AUX #\PUNC
69. As?ter\N b?zu\J ttäj\N ttättac\V #\PUNC
70. t?l?ku\J säw?ye\N bäc?kola\ADV hedu\V #\PUNC
71. As∂ter \N qon?jo\J l?j\N nat\AUX #\PUNC
72. As?ter\N t?quwaqur\J shuraboc\N alwat\V #\PUNC
73. ?nat?wa\N qonjo\J ccama\N gäzulat\V #\PUNC
74. Kassa\N m?saw?n\N ?yäbäla\V näw\AUX #\PUNC
75. Kassa\N t?nan?t\ADV bähay?l\ADV dänägätä\V #\PUNC
76. Kassa\N b?zu\J nägär\N tänagärä\V #\PUNC
77. Kassa\N bet\N w?s?tt\PREP näbär\V #\PUNC
78. nägadew\N gän?zäb\N bättam\ADV y?wädal\V #\PUNC
79. Kassa\N b?zu\J n?b?rät\N täräkäbä\V #\PUNC
80. Betty\N m?sawan\N ?n?jära\N bälac\V #\PUNC

Second Set

1. t?n?shu\J wän?d?me\N ?l?häna\J näw\AUX #\PUNC
2. t?l?kua\J ?h?te\N gäbäya\N hedäc\V #\PUNC
3. qon?j?yewa\N l?j\N zare\ADV t?mättaläc\V #\PUNC
4. ?nia\N sh?mag?le\N säw?ye\N motu\V #\PUNC
5. yäwän?d?me\N l?j\N mis?t\N agäba\V #\PUNC
6. yäkäbäleac?n\N likämän?bär\N s?b?säba\N ttära\V #\PUNC
7. yämäs?ria\N betac?n\N halafi\N mättu\V #\PUNC
8. yä?g?r\N kuas\N ccäwata\N täjämärä\V #\PUNC

9. alämaqäf\N yäsetoc\N qän\N täkäbärä\V #\PUNC
10. t?n?n?sh\J l?joc\N ccäwata\N y?wädalu\V #\PUNC
11. t?l?l?q\J l?joc\N man?bäb\V y?wädalu\V #\PUNC
12. bärari\J wäfoc\N bättam\ADV yam?ralu\V #\PUNC
13. säkaramu\N säw?ye\N t?nan?t\ADV aräfu\V #\PUNC
14. yäbaläbete\N ?nat\N zare\ADV y?mättalu\V #\PUNC
15. t?l?qu\J l?je\N mat?rik\N aläfä\V #\PUNC
16. yäzän?d?ro\N l?joc\N bättam\ADV qäb?ttäwal\V #\PUNC
17. yägojam\N mar\N bättam\ADV y?ttaf?ttal\V #\PUNC
18. yätägodut\N wätadäroc\N ?r?data\N agänu\V #\PUNC
19. arän?guade\J qäläm\N däs\V y?lal\V #\PUNC
20. t?n?shua\J l?j\N ?yärottäc\V mättac\V #\PUNC
21. t?l?qu\J wän?d?me\N s?ra\N qäyärä\V #\PUNC
22. yät?m?h?r?t\N betu\N däwäl\N tädäwälä\V #\PUNC
23. t?l?ku\J yäabäba\N mas?qämäcca\N täsäbärä\V #\PUNC
24. yäbabur\N hadidu\N s?ra\N tättänaqäqä\V #\PUNC
25. gotäraw\N w?s?tt\PREP ?h?l\N alä\AUX #\PUNC
26. betu\N w?s?tt\PREP ay?ttoc\N alu\V #\PUNC
27. yägojam\N yämar\N ttäj\N y?ttaf?ttal\V #\PUNC
28. yäBälättä\N as?tämari\N zare\ADV y?hedal\V #\PUNC
29. yäagote\N l?j\N bättäna\ADV tamual\V #\PUNC
30. yäEthiopia\N h?z?b\N mäz?mur\N täzämära\V #\PUNC

Test Set

1. Ethiopia\N arän?guade\J kämis\N läbäsäc\V #\PUNC
2. Ayälä\N ?rasun\N bättam\ADV tamämä\V #\PUNC
3. ?nate\N wädä\PREP gäbäya\N hedäc\V #\PUNC
4. säw?yew\N bämaläda\ADV tänäs?to\V hedä\V #\PUNC
5. wäfocu\N bälelit\ADV mäzämär\V jämäru\V #\PUNC
6. Kibur\N bättam\ADV tatari\V näw\AUX #\PUNC
7. Kin?fä\N sägurun\N mäkorätt\V y?wädal\V #\PUNC

8. Abäbä\N dabo\N gäz?to\V mätta\V #\PUNC
9. fätanu\J ccala\N ccube\N ccäbättä\V #\PUNC
10. ?nat?wa\N bättam\ADV as?käyami\J nat\AUX #\PUNC
11. ?ne\N t?ll?k\J bäg\N gäzahu\V #\PUNC
12. l?ju\N l?b?s\N ?yattäbä\V näbär\V #\PUNC
13. abate\N nägä\ADV agäru\N y?hedal\V #\PUNC
14. l?jocu\N ?n?dä\PREP ah?ya\N y?ragättalu\V #\PUNC
15. As?ter\N kon?jo\J kämisoc\N aluwat\V #\PUNC
16. As?ter\N säguruan\N ?yätattäbäc\V näw\AUX #\PUNC
17. Ethiopia\N b?zu\J tarik\N alat\V #\PUNC
18. Fäläqä\N t?ru\J s?ra\N agänä\V #\PUNC
19. Kassa\N kosasa\J l?j\N näw\AUX #\PUNC
20. Kassa\N märet\N lay\PREP täna\V #\PUNC

Second Test Set

1. l?ju\N wätät\N bättam\ADV y?wädal\V #\PUNC
2. Kassa\N tolo\ADV tolo\ADV täramädä\V #\PUNC
3. yäaläm\J wancca\N t?nan?t\ADV täjämärä\V #\PUNC
4. Näb?rit\N ?n?k?l?fuan\N sat?tänä\V adäräc\V #\PUNC
5. Täs?faye\N shäbäla\J l?j\N näw\AUX #\PUNC
6. abate\N wädä\PREP gäbäya\N hedu\V #\PUNC
7. yäcay?na\J ccama\N gäbäyäw?n\N att?läkäläkäw\V #\PUNC
8. säw?yew\N t?ll?k\J w?sha\N aläw\AUX #\PUNC
9. l?ju\N ttara\N lay\PREP wättä\V #\PUNC
10. Qonjit\N t?nan?t\ADV mätt?ta\V näbär\V #\PUNC
11. Käbädä\N wädä\PREP s?ra\N hedä\V #\PUNC
12. säwocu\N foq\N lay\PREP nacäw\V #\PUNC
13. t?ll?qua\J setiyo\N t?nan?t\ADV motu\V #\PUNC
14. h?sanat\N b?zu\J tt?n?qaqe\N y?shalu\V #\PUNC
15. guadänaye\N t?ru\J ccäwata\N yaw?qal\V #\PUNC
16. därasiw\N maqäw\V maqäw\V motu\V #\PUNC

17. mänoria\J bete\N säfi\J näw\AUX #\PUNC
18. balua\N bättam\ADV säkaram\J näw\AUX #\PUNC
19. sh?mag?lew\N bättam\ADV atalay\J nacäw\V #\PUNC
20. b?r?q?yewocu\J yädur\N arawitoc\N täsädädu\V #\PUNC
21. yä?g?r\N kuas\N ccäwata\N ?wädalähu\V #\PUNC
22. suri\N yäläbäsäc?w\N l?j\N tam?raläc\V #\PUNC

Appendix 6. Extracted Probabilistic Context Free Grammar

PCFG generated using the first 80 sentences

Rules	Count l	Total	Probabilities
S → NP VP	80	80	1.0
NP → N E	78	99	0.788
NP → Adj N	21	99	0.212
PP → N PREP	16	27	0.593
PP → PREP N	11	27	0.407
AdjP→ Adv Adj	5	5	1.0
VP → PP V	27	105	0.257
VP → N VP	15	105	0.143
VP → N V	17	105	0.162
VP → NP V	21	105	0.2
VP → ADV V	7	105	0.067
VP → AdjP V	5	105	0.048
VP → ADV VP	9	105	0.086
VP → V V	4	105	0.038

PCFG generated using the 110 sentences

Rules	Count l	Total	Probabilities
S → NP VP	110	110	1.0
NP → N	78	139	0.561
NP →Adj N	31	139	0.223
NP → N N	19	139	0.137
NP → N NP	3	139	0.022
NP → NP N	5	139	0.036
NP → Adj NP	1	139	0.007
NP → N PREP	2	139	0.014
PP → N PREP	16	27	0.593
PP → PREP N	11	27	0.407
AdjP→ Adv Adj	5	5	1.0
VP → PP V	27	136	0.199
VP → N VP	17	136	0.125
VP → N V	27	136	0.199
VP → NP V	20	136	0.147
VP → ADV V	15	136	0.11
VP → AdjP V	5	136	0.037
VP → ADV VP	9	136	0.066
VP → V V	6	136	0.044
VP → V	9	136	0.066
VP → Adj V	1	136	0.007

Appendix 7. Parses

Training Set

1. S (NP (N wätadärocu)
   VP((PP (PREP wädä) (N gibiacäw))(V gäbu)))

2. S (NP (N As?ter)
   VP((N likämän?bärun)(VP (N läs?ra)(V ttärac?w))))

3. S (NP (N Ayälä)
   VP((NP (Adj t?n?sh) (N injära))(V bäla)))

4. S (NP (N Läma)
   VP((NP (Adj bäsal) (N mäm?h?r))(V näw)))

5. S (NP (N Bälättä)
   VP((NP (Adj t?n?sh) (N bet))(V aläw)))

6. S (NP (N kokäboc)
   VP((N bän?gat)(VP (ADV bättäm)(V yabärralu))))

7. S (NP (N Kassa)
   VP((N läAs?ter)(VP (N gän?zäb)(V sättat))))

8. S (NP (N Kassa)
   VP((N läsäw?yew)(VP (N mis?tt?r)(V nägäracäw))))

9. S (NP (N Kassa)
   VP((PP (N bet) (PREP w?s?tt))(V alä)))

10. S (NP (N Kassa)
    VP((AdjP (ADV bättam) (Adj ?l?häna))(V näw)))

11. S (NP (N L?jitu)
    VP((AdjP (ADV bättam) (Adj amäläna))(V näc)))

12. S (NP (N Bälättä)
    VP((PP (PREP wädä) (N Gojam))(V hedä)))

13. S (NP (N Bälättä)
    VP((PP (N c?qa) (PREP lay))(V wädäqä)))

14. S (NP (N säw?yew)
    VP((PP (N bet) (PREP d?räs))(V shänän)))

15. S (NP (N l?jocu)
VP((PP (PREP ?n?dä) (N wäf))(V y?bäralu)))

16. S (NP (N As?ter)
VP((PP (PREP ?n?dä) (N ?h?te))(V wädäqäc)))

17. S (NP (N Kassa)
VP((NP (Adj t?ll?q) (N bäg))(V gäza)))

18. S (NP (N As?ter)
VP((NP (Adj hulät) (N l?joc))(V aluwat)))

19. S (NP (N l?jocu)
VP((PP (N k?f?l) (PREP w?s?tt))(V alu)))

20. S (NP (N Kassa)
VP((PP (PREP wädä) (N gäbäya))(V hedä)))

21. S (NP (N As?ter)
VP((PP (PREP wädä) (N bet))(V gäbac)))

22. S (NP (N l?jocu)
VP((PP (N zaf) (PREP lay))(V wättu)))

23. S (NP (N l?jocu)
VP((PP (N bäru) (PREP lay))(V qomu)))

24. S (NP (N As?ter)
VP((PP (N al?gaw) (PREP lay))(V wädäqäc)))

25. S (NP (N dorowocu)
VP((PP (N qottu) (PREP lay))(V adäru)))

26. S (NP (N ?sacäw)
VP((PP (N w?haw) (PREP lay))(V wädäqu)))

27. S (NP (N as?tämariw)
VP((N lätämariwocu)(VP (N mäsähaf)(V sättacäw))))

28. S (NP (N Kassa)
VP((N säw)(VP (ADV ?j?g)(V yakäberal))))

29. S (NP (N Kassa)
VP((ADV bätäsiyat)(VP (N m?saw?n)(V bäla))))

30. S (NP (N Kassa)
VP((PP (N bet) (PREP w?s?tt))(V näbär)))

31. S (NP (N Kassa)
VP((NP (Adj t?n?sh) (N mäsähaf))(V shättä)))

32. S (NP (N As?ter)
VP((N läl?ju)(VP (N gän?zäb)(V lakäc?lät))))

33. S (NP (N As?ter)
VP((N l?jit?wan)(VP (N sägur?wan)(V säracat))))

34. S (NP (N As?ter)
VP((PP (PREP wädä) (N Gon?där))(V hedäc)))

35. S (NP (N As?ter)
VP((AdjP (ADV ?j?g) (Adj wäf?ram))(V nat)))

36. S (NP (N l?j?t?wa)
VP((NP (Adj kur?tt) (N ?nat?wan))(V t?mäs?laläc)))

37. S (NP (N As?ter)
VP((NP (Adj t?ll?k) (N wän?d?m))(V alat)))

38. S (NP (N As?ter)
VP((NP (Adj tanash) (N ?h?t))(V alat)))

39. S (NP (N As?ter)
VP((ADV bättäwat)(VP (N buna)(V täfälaläc))))

40. S (NP (N l?ju)
VP((N abatun)(VP (ADV bähay?l)(V täsadäbä))))

41. S (NP (N Bälättä)
VP((PP (PREP wädä) (N Gojam))(V hedä)))

42. S (NP (N Bälättä)
VP((PP (N zaf) (PREP lay))(V wättä)))

43. S (NP (N säw?yew)
VP((PP (N bet) (PREP d?räs))(V shänän)))

44. S (NP (N säw?yew)
VP((AdjP (ADV bättam) (Adj räj?m))(V näw)))

45. S (NP (N ?ns?socu)
VP((PP (PREP ?n?dä) (N assa))(V y?wanalu)))

46. S (NP (N Kassa)

VP((NP (Adj t?l?q) (N mäsähaf))(V yanäbal)))

47. S (NP (N Kassa)
VP((N läl?ju)(VP (N gän?zäb)(V lakälät))))

48. S (NP (N ?nate)
VP((N l?jit?wan)(VP (N sägur?wan)(V säracat))))

49. S (NP (N ?h?te)
VP((PP (PREP wädä) (N Gon?där))(V hedäc)))

50. S (NP (N As?ter)
VP((ADV t?nan?t)(VP (ADV bäd?n?gät)(V tättäläfäc))))

51. S (NP (N Kassa)
VP((ADV t?nan?t)(VP (N m?saw?n)(V bäl?toal))))

52. S (NP (N ?ne)
VP((N an?bäsa)(VP (V gäd?ye)(V näbär))))

53. S (NP (N Kassa)
VP((ADV zare)(VP (N an?bäsa)(V y?gäd?lal))))

54. S (NP (N Kassa)
VP((ADV nägä)(VP (N an?bäsa)(V y?gäd?lal))))

55. S (NP (N Kassa)
VP((ADV zäwät?r)(VP (N an?bäsa)(V y?gäd?lal))))

56. S (NP (N Kassa)
VP((N ?n?ccät)(VP (V ?yäfälätä)(V näbär))))

57. S (NP (N säwocu)
VP((PP (N ?l?qit) (PREP lay))(V nacäw)))

58. S (NP (N wätadärocu)
VP((PP (N g?b?zza) (PREP lay))(V nacäw)))

59. S (NP (N Kassa)
VP((NP (Adj wärätäna) (N säw))(V näw)))

60. S (NP (N Kassa)
VP((N As?ter?n)(VP (N ?n?ccät)(V asabärat))))

61. S (NP (N l?ju)
VP((PP (PREP wädä) (N betu))(V rottä)))

62. S (NP (N ?ne)

VP((NP (Adj konjo) (N l?j))(V ayähu)))

63. S (NP (N säwocu)
VP((ADV bäd?n?gät)(VP (V rotäw)(V wättu))))

64. S (NP (Adj b?zu) (N amätat)VP (N Gojam) (V norealähu))

65. S (NP (N As?ter)
VP((NP (Adj t?l?q) (N ?n?s?ra))(V atäbäc)))

66. S (NP (N As?ter)
VP((AdjP (ADV bättam) (Adj qonjo))(V nat)))

67. S (NP (N ?h?l)
VP((PP (N gotäraw) (PREP w?s?tt))(V alä)))

68. S (NP (N Kassa)
VP((NP (Adj qonjo) (N bet))(V aläw)))

69. S (NP (N As?ter)
VP((NP (Adj b?zu) (N ttäj))(V ttättac)))

70. S (NP (Adj t?l?ku) (N säw?ye)VP (ADV bäc?kola) (V hedu))

71. S (NP (N As?ter)
VP((NP (Adj qon?jo) (N l?j))(V nat)))

72. S (NP (N As?ter)
VP((NP (Adj t?quwaqur) (N shuraboc))(V alwat)))

73. S (NP (N ?nat?wa)
VP((NP (Adj qonjo) (N ccama))(V gäzulat)))

74. S (NP (N Kassa)
VP((N m?saw?n)(VP (V ?yäbäla)(V näw))))

75. S (NP (N Kassa)
VP((ADV t?nan?t)(VP (ADV bähay?l)(V dänägätä))))

76. S (NP (N Kassa)
VP((NP (Adj b?zu) (N nägär))(V tänagärä)))

77. S (NP (N Kassa)
VP((PP (N bet) (PREP w?s?tt))(V näbär)))

78. S (NP (N nägadew)
VP((N gän?zäb)(VP (ADV bättam)(V y?wädal))))

79. S (NP (N Kassa)
VP((NP (Adj b?zu) (N n?b?rät))(V täräkäbä)))

80. S (NP (N Betty)
VP((N m?sawan)(VP (N ?njära)(V bälac))))

Second Set

1. S (NP (Adj t?n?shu) (N wän?d?me)VP (Adj ?l?häna) (V näw))

2. S (NP (Adj t?l?kua) (N ?h?te)VP (N gäbäya) (V hedäc))

3. S (NP (N qon?j?yewa)
VP((N l?j)(VP (ADV zare)(V t?mättaläc))))

4. S (NP (N ?nia)
VP((NP (N sh?mag?le) (N säw?ye))(V motu)))

5. S (NP (N yäwän?d?me)
VP((NP (N l?j) (N mis?t))(V agäba)))

6. S (NP (N yäkäbäleac?n)
VP((NP (N likämän?bär) (N s?b?säba))(V ttära)))

7. S (NP (N yämäs?ria)
VP((NP (N betac?n) (N halafi))(V mättu)))

8. S (NP (N yä?g?r)
VP((NP (N kuas) (N ccäwata))(V täjämärä)))

9. S (NP (N alämaqäf)
VP((NP (N yäsetoc) (N qän))(V täkäbärä)))

10. S (NP (Adj t?n?n?sh) (N l?joc)VP (N ccäwata) (V y?wädalu))

11. S (NP (Adj t?l?l?q) (N l?joc)VP (V man?bäb) (V y?wädalu))

12. S (NP (Adj bärari) (N wäfoc)VP (ADV bättam) (V yam?ralu))

13. S (NP (N säkaramu)
VP((N säw?ye)(VP (ADV t?nan?t)(V aräfu))))

14. S (NP (N yäbaläbete)
VP((N ?nat)(VP (ADV zare)(V y?mättalu))))

15. S (NP (Adj t?l?qu) (N l?je)VP (N mat?rik) (V aläfä))
16. S (NP (N yäzän?d?ro)
VP((N l?joc)(VP (ADV bättam)(V qäb?ttäwal))))

17. S (NP (N yägojam)
VP((N mar)(VP (ADV bättam)(V y?ttaf?ttal))))

18. S (NP (N yätägodut)
VP((NP (N wätadäroc) (N ?r?data))(V agänu)))

19. S (NP (Adj arän?guade) (N qäläm)VP (V däs) (V y?lal))

20. S (NP (Adj t?n?shua) (N l?j)VP (V ?yärottäc) (V mättac))

21. S (NP (Adj t?l?qu) (N wän?d?me)VP (N s?ra) (V qäyärä))

22. S (NP (N yät?m?h?r?t)
VP((NP (N betu) (N däwäl))(V tädäwälä)))

23. S (NP (Adj t?l?ku) (N yäabäba)VP (N mas?qämäcca) (V täsäbärä))

24. S (NP (N yäbabur)
VP((NP (N hadidu) (N s?ra))(V tättänaqäqä)))

25. S (NP (N gotäraw)
VP((PP (PREP w?s?tt) (N ?h?l))(V alä)))

26. S (NP (N betu)
VP((PP (PREP w?s?tt) (N ay?ttoc))(V alu)))

27. S (NP (N yägojam)
VP((NP (N yämar) (N ttäj))(V y?ttaf?ttal)))

28. S (NP (N yäBälättä)
VP((N as?tämari)(VP (ADV zare)(V y?hedal))))

29. S (NP (N yäagote)
VP((N l?j)(VP (ADV bättäna)(V tamual))))

30. S (NP (N yäEthiopia)
VP((NP (N h?z?b) (N mäz?mur))(V täzämära)))

First Test Set

1. S (NP (N Ethiopia)
   VP((NP (Adj arän?guade) (N kämis))(V läbäsäc)))

2. S (NP (N Ayälä)
   VP((N ?rasun)(VP (ADV bättam)(V tamämä))))

3. S (NP (N ?nate)
   VP((PP (PREP wädä) (N gäbäya))(V hedäc)))

4. S (NP (N säw?yew)
   VP((ADV bämaläda)(VP (V tänäs?to)(V hedä))))

5. S (NP (N wäfocu)
   VP((ADV bälelit)(VP (V mäzämär)(V jämäru))))

6. S (NP (N Kibur)
   VP((ADV bättam)(VP (V tatari)(V näw))))

7. S (NP (N Kin?fä)
   VP((N sägurun)(VP (V mäkorätt)(V y?wädal))))

8. S (NP (N Abäbä)
   VP((N dabo)(VP (V gäz?to)(V mätta))))

9. S (NP (Adj fätanu) (N ccala)VP (N ccube) (V ccäbättä))

10. S (NP (N ?nat?wa)
    VP((AdjP (ADV bättam) (Adj as?käyami))(V nat)))

11. S (NP (N ?ne)
    VP((NP (Adj t?ll?k) (N bäg))(V gäzahu)))

12. S (NP (N l?ju)
    VP((N l?b?s)(VP (V ?yattäbä)(V näbär))))

13. S (NP (N abate)
    VP((ADV nägä)(VP (N agäru)(V y?hedal))))

14. S (NP (N l?jocu)
    VP((PP (PREP ?n?dä) (N ah?ya))(V y?ragättalu)))

15. S (NP (N As?ter)
    VP((NP (Adj kon?jo) (N kämisoc))(V aluwat)))

16. S (NP (N As?ter)
VP((N säguruan)(VP (V ?yätattäbäc)(V näw))))

17. S (NP (N Ethiopia)
VP((NP (Adj b?zu) (N tarik))(V alat)))

18. S (NP (N Fäläqä)
VP((NP (Adj t?ru) (N s?ra))(V agänä)))

19. S (NP (N Kassa)
VP((NP (Adj kosasa) (N l?j))(V näw)))

20. S (NP (N Kassa)
VP((PP (N märet) (PREP lay))(V täna)))

Second Test Set

1. S (NP (N l?ju)
VP((N wätät)(VP (ADV bättam)(V y?wädal))))

2. S (NP (N Kassa)
VP((ADV tolo)(VP (ADV tolo)(V täramädä))))

3. S (NP (Adj yäaläm) (N wancca)VP (ADV t?nan?t) (V täjämärä))

4. S (NP (N Näb?rit)
VP((N ?n?k?l?fuan)(VP (V sat?tänä)(V adäräc))))

5. S (NP (N Täs?faye)
VP((NP (Adj shäbäla) (N l?j))(V näw)))

6. S (NP (N abate)
VP((PP (PREP wädä) (N gäbäya))(V hedu)))

7. S (NP (Adj yäcay?na) (N ccama)VP (N gäbäyäw?n) (V att?läkäläkäw))

8. S (NP (N säw?yew)
VP((NP (Adj t?ll?k) (N w?sha))(V aläw)))

9. S (NP (N l?ju)
VP((PP (N ttara) (PREP lay))(V wättä)))

10. S (NP (N Qonjit)
VP((ADV t?nan?t)(VP (V mätt?ta)(V näbär))))

11. S (NP (N Käbädä)

VP((PP (PREP wädä) (N s?ra))(V hedä)))

12. S (NP (N säwocu)
VP((PP (N foq) (PREP lay))(V nacäw)))

13. S (NP (Adj t?ll?qua) (N setiyo)VP (ADV t?nan?t) (V motu))
14. S (NP (N h?sanat)
VP((NP (Adj b?zu) (N tt?n?qaqe))(V y?shalu)))

15. S (NP (N guadänaye)
VP((NP (Adj t?ru) (N ccäwata))(V yaw?qal)))

16. S (NP (N därasiw) (V maqäw)VP (V maqäw) (V motu))

17. S (NP (Adj mänoria) (N bete)VP (Adj säfi) (V näw))

18. S (NP (N balua)
VP((AdjP (ADV bättam) (Adj säkaram))(V näw)))

19. S (NP (N sh?mag?lew)
VP((AdjP (ADV bättam) (Adj atalay))(V nacäw)))

20. S (NP (Adj b?r?q?yewocu) (N yädur)VP (N arawitoc) (V täsädädu))

21. S (NP (N yä?g?r)
VP((NP (N kuas) (N ccäwata))(V ?wädalähu)))

22. S (NP (N suri)
VP((NP (N yäläbäsäc?w) (N l?j))(V tam?raläc)))

The erroneous parses are underlined.

Appendix 8. The Code Written in VB 6.0.

```
Private words(1 To 4) As String
'Stores the words of the sentence to be parsed
Private tags(1 To 4) As String
'Stores the tags of the words
Private ProbArray(1 To 5) As Double
'Stores the probability of each possibility
Private POSArray(1 To 4, 1 To 4) ' POS matrix
Public MaxProb As Double
'Stores teh maximum probability
Public CurrentFile As String 'File Input
Public ParseNext As Boolean
'To determine if the user wants the next sentence to be parsed

Private Sub Command1_Click()
Dim inCnt As Integer 'String input counter
Dim inFnd As Integer
'To check for the position of input string
Dim sword, snextfile, textline As String
Dim spos, PassWrdCnt As Integer
Const delim = " "
Dim Max As Single
Dim UNCArray(6) As Single 'UNC Transition Probabilities
Set db = OpenDatabase("d:\Atelach\Parse\sampledb.mdb", True)
'Opens Database
Set rst = db.OpenRecordset("PCFGCNF", dbOpenDynaset)
'Sets RecrdSet
Set rst'PP = db.OpenRecordset("TransProbParse", dbOpenDynaset)
Open CurrentFile For Input As #1 'Input from file
ParseNext = True
'Input from file
1: While Not EOF(1)
        Line Input #1, textline
        Text1.Text = textline
        Debug.Print ("Text is = " & Text1.Text)
        If Len(Text1.Text) = 0 Then Exit Sub
          inCnt = 1
          inFnd = InStr(inCnt, Text1.Text, delim)
          wordcnt = 0
          While inFnd <> 0
             sword = Mid$(Text1.Text, inCnt, inFnd - inCnt)
             If Mid(sword, 1, 1) = "#" Then
               parse 'call the parsing procedure
              If MsgBox("Parse next sentence", vbQuestion + vbYesNo) = vbYes    Then
               GoTo 1
```

```
  Else: Exit Sub
  End If

Else
    'Store words and tags
    wordcnt = wordcnt + 1
    spos = InStr(1, sword, "\")
    words(wordcnt) = Mid(sword, 1, spos - 1)
    tags(wordcnt) = Mid(sword, spos + 1)
    'Summarize categories from the tagger
    If tags(wordcnt) = "J" Or tags(wordcnt) = "JPN" Then
       tags(wordcnt) = "Adj"
    ElseIf tags(wordcnt) = "AUX" Then
       tags(wordcnt) = "V"
    'Guess Categry
    ElseIf tags(wordcnt) = "UNC" Then
       If wordcnt = 1 Then
         tags(wordcnt) = "N"
       ElseIf wordcnt = 4 Then
          tags(wordcnt) = "V"
       ElseIf wordcnt = 2 Or wordcnt = 3 Then

       Max = 0
       If wordcnt = 2 Or wordcnt = 3 Then
          If Not rst'PP.BOF Then rst'PP.MoveFirst
             For I = 1 To 6
             UNCArray(I) = rst'PP.Fields(tags(wordcnt - 1)).Value
             If Max < UNCArray(I) Then
             Max = UNCArray(I)
             j = I
             End If
             rst'PP.MoveNext
             Next I
          End If
          If j = 2 Then
             tags(wordcnt) = "N"
          ElseIf j = 3 Then
             tags(wordcnt) = "V"
          ElseIf j = 4 Then
             tags(wordcnt) = "Adj"
          ElseIf j = 5 Then
             tags(wordcnt) = "ADV"
          ElseIf j = 6 Then
             tags(wordcnt) = "PREP"
          End If

          If j = 2 Then
```

```
                    tags(wordcnt) = "N"
                ElseIf j = 3 Then
                    tags(wordcnt) = "V"
                ElseIf j = 4 Then
                    tags(wordcnt) = "Adj"
                ElseIf j = 5 Then
                    tags(wordcnt) = "ADV"
                ElseIf j = 6 Then
                    tags(wordcnt) = "PREP"
                End If
            End If
        End If
        List1.AddItem Mid$(Text1.Text, inCnt, inFnd - inCnt)
        inCnt = inFnd + 1
        inFnd = InStr(inCnt, Text1.Text, delim)
    Wend
  If inCnt < Len(Text1.Text) Then
    List1.AddItem Mid$(Text1.Text, inCnt)
  End If
Wend
Close (1)
End Sub

Public Sub parse()
Dim ProbCat(1 To 3) As Double
'To store non-diagonal categories from the POS Matrix
Dim I, j, k As Integer
Dim n, w, X, y, z As Double
Dim found As Boolean
'Fill the POS Matrix
For I = 1 To 4
  For j = 1 To 4
   If I = j Then
     POSArray(I, j) = tags(I)
   ElseIf I = 1 And j = 4 Then
     POSArray(I, j) = "S"
   ElseIf I = 1 And j = 3 Then
     POSArray(I, j) = "NP"
   ElseIf I = 2 And j = 4 Then
     POSArray(I, j) = "VP"
   Else
     POSArray(I, j) = "E"
   End If
  Next j
Next I
Set db = OpenDatabase("d:\Atelach\Parse\sampledb.mdb", True)
```

```
Set rst = db.OpenRecordset("PCFGCNF", dbOpenDynaset)
'Open table PCFGCNF which stores PCFGs in CNF
Set rstCh = db.OpenRecordset("BPtrCats", dbOpenDynaset)
'Open table BPtrCats which holds the expanded rules for
'non-diagonal categories
Set rstCh1 = db.OpenRecordset("BPtrProb", dbOpenDynaset)
'Open table BPtrCats which holds the expanded rules for
'diagonal categories

'Creat the parse space and calculate probabilities for each parse
For I = 1 To 3
   If Not rst.BOF Then rst.MoveFirst
   While Not rst.EOF
      If rst!RHS1 = tags(I) And rst!RHS2 = tags(I + 1) Then
        POSArray(I, I + 1) = rst!LHS
        ProbCat(I) = rst!probabilities
        'write to the next table
              rstCh.AddNew
              rstCh!Category = I
              rstCh!LHS = rst!LHS
              rstCh!RHS1 = rst!RHS1
              rstCh!RHS2 = rst!RHS2
              rstCh.Update
      End If
      rst.MoveNext
   Wend
Next I
For I = 1 To 4
   For j = 1 To 4
     Debug.Print POSArray(I, j)
   Next
Next

'Start position1
ProbArray(1) = ProbCat(1) * ProbCat(3)

'Start position 2
If Not rst.BOF Then rst.MoveFirst
found = False
   While Not rst.EOF And Not found
   'Debug.Print rst!LHS & rst!RHS1 & rst!RHS2
   If (rst!LHS = "VP" And rst!RHS1 = POSArray(2, 2) And rst!RHS2 = POSArray(3, 4))
Then
      y = rst!probabilities
              rstCh1.AddNew
              rstCh1!ProbabilityCntr = "2b"
              rstCh1!LHS = rst!LHS
```

```
                rstCh1!RHS1 = rst!RHS1
                rstCh1!RHS2 = rst!RHS2
                rstCh1.Update
      found = True
   End If
   rst.MoveNext
   Wend
If Not rst.BOF Then rst.MoveFirst
   found = False
   While Not rst.EOF And Not found
      If (rst!LHS = "NP" And rst!RHS1 = POSArray(1, 1) And rst!RHS2 = "E") Then
      ProbArray(2) = rst!probabilities * y
                rstCh1.AddNew
                rstCh1!ProbabilityCntr = "2a"
                rstCh1!LHS = rst!LHS
                rstCh1!RHS1 = rst!RHS1
                rstCh1!RHS2 = rst!RHS2
                rstCh1.Update
      found = True
   Else
      ProbArray(2) = 0
   End If
   rst.MoveNext
   Wend
'End position2

'Start position3
If Not rst.BOF Then rst.MoveFirst
found = False
   While Not rst.EOF And Not found
   If (rst!LHS = "NP" And rst!RHS1 = POSArray(1, 1) And rst!RHS2 = "E") Then
      X = rst!probabilities
                rstCh1.AddNew
                rstCh1!ProbabilityCntr = "3a"
                rstCh1!LHS = rst!LHS
                rstCh1!RHS1 = rst!RHS1
                rstCh1!RHS2 = rst!RHS2
                rstCh1.Update
      found = True
   End If
   rst.MoveNext
   Wend
If Not rst.BOF Then rst.MoveFirst
found = False
   While Not rst.EOF And Not found
      If (rst!LHS = "VP" And rst!RHS1 = POSArray(2, 3) And rst!RHS2 = POSArray(4, 4))
Then
```

```
        n = rst!probabilities
        ProbArray(3) = n * X
                rstCh1.AddNew
                rstCh1!ProbabilityCntr = "3b"
                rstCh1!LHS = rst!LHS
                rstCh1!RHS1 = rst!RHS1
                rstCh1!RHS2 = rst!RHS2
                rstCh1.Update
        found = True
    Else
        ProbArray(3) = 0
    End If
    rst.MoveNext
    Wend
'End position3
'Start position4
If Not rst.BOF Then rst.MoveFirst
found = False
    While Not rst.EOF And Not found
    If (rst!LHS = "NP" And rst!RHS1 = POSArray(1, 1) And rst!RHS2 =  POSArray(2, 3))
Then
        z = rst!probabilities
                rstCh1.AddNew
                rstCh1!ProbabilityCntr = "4a"
                rstCh1!LHS = rst!LHS
                rstCh1!RHS1 = rst!RHS1
                rstCh1!RHS2 = rst!RHS2
                rstCh1.Update
        found = True
    End If
    rst.MoveNext
    Wend
If Not rst.BOF Then rst.MoveFirst
found = False
    While Not rst.EOF
        If (rst!LHS = "VP" And rst!RHS1 = POSArray(4, 4) And rst!RHS2 = "E") Then
        ProbArray(4) = rst!probabilities * z
                rstCh1.AddNew
                rstCh1!ProbabilityCntr = "4b"
                rstCh1!LHS = rst!LHS
                rstCh1!RHS1 = rst!RHS1
                rstCh1!RHS2 = rst!RHS2
                rstCh1.Update
        found = True
    Else
        ProbArray(4) = 0
    End If
```

```
   rst.MoveNext
   Wend
'End position 4

'Start position 5
If Not rst.BOF Then rst.MoveFirst
found = False
   While Not rst.EOF
   If (rst!LHS = "NP" And rst!RHS1 = POSArray(1, 2) And rst!RHS2 = POSArray(3, 3))
Then
      w = rst!probabilities
            rstCh1.AddNew
            rstCh1!ProbabilityCntr = "5a"
            rstCh1!LHS = rst!LHS
            rstCh1!RHS1 = rst!RHS1
            rstCh1!RHS2 = rst!RHS2
            rstCh1.Update
      found = True
   End If
   rst.MoveNext
   Wend
If Not rst.BOF Then rst.MoveFirst
found = False
   While Not rst.EOF
      If (rst!LHS = "VP" And rst!RHS1 = POSArray(4, 4) And rst!RHS2 = "E") Then
      ProbArray(5) = rst!probabilities * w
            rstCh1.AddNew
            rstCh1!ProbabilityCntr = "5b"
            rstCh1!LHS = rst!LHS
            rstCh1!RHS1 = rst!RHS1
            rstCh1!RHS2 = rst!RHS2
            rstCh1.Update
      found = True
   Else
      ProbArray(5) = 0
   End If
   rst.MoveNext
   Wend

'Calculate Max probability
MaxProb = 0
For k = 1 To 5
   If MaxProb < ProbArray(k) Then
      MaxProb = ProbArray(k)
   End If
Next k
```

```
For I = 1 To 5
Debug.Print ProbArray(I)
Next I
Debug.Print MaxProb
Text4.Text = MaxProb

'Expanded rules for position 1
If MaxProb = ProbArray(1) Then
      Text3.Refresh
      If Not rstCh.BOF Then rstCh1.MoveFirst
      While Not rstCh.EOF
        If rstCh!Category = 1 Then
           Text3.Text = "S --> NP VP" & vbCrLf & _
           rstCh!LHS & " --> " & rstCh!RHS1 & "  " & rstCh!RHS2
        ElseIf rstCh!Category = 2 Then
            Text3.Text = Text3.Text & vbCrLf & rstCh!LHS & " --> " & rstCh!RHS1 & "  " &
rstCh!RHS2
        End If
      rstCh.MoveNext
      Wend
End If

'Expanded rules for position 2
If MaxProb = ProbArray(2) Then
      Text3.Refresh
      If Not rstCh1.BOF Then rstCh1.MoveFirst
      While Not rstCh1.EOF
        If rstCh1!ProbabilityCntr = "2a" Then
           Text3.Text = "S --> NP VP" & vbCrLf & _
           rstCh1!LHS & " --> " & rstCh1!RHS1 & "  " & rstCh1!RHS2
        ElseIf rstCh1!ProbabilityCntr = "2b" Then
            Text3.Text = Text3.Text & vbCrLf & rstCh1!LHS & " --> " & rstCh1!RHS1 & "  "
& rstCh1!RHS2
        End If
      rstCh1.MoveNext
      Wend

      If Not rstCh.BOF Then rstCh.MoveFirst
      While Not rstCh.EOF
        If rstCh!Category = 3 Then
           Text3.Text = Text3.Text & vbCrLf & _
           rstCh!LHS & " --> " & rstCh!RHS1 & "  " & rstCh!RHS2
        End If
      rstCh.MoveNext
      Wend
End If
```

```
'Expanded rules for position 3
If MaxProb = ProbArray(3) Then
Text3.Refresh
   If Not rstCh1.BOF Then rstCh1.MoveFirst
      While Not rstCh1.EOF
        If rstCh1!ProbabilityCntr = "3a" Then
           Text3.Text = "S --> NP VP" & vbCrLf & _
           rstCh1!LHS & " --> " & rstCh1!RHS1 & "  " & rstCh1!RHS2
        ElseIf rstCh1!ProbabilityCntr = "3b" Then
            Text3.Text = Text3.Text & vbCrLf & rstCh1!LHS & " --> " & rstCh1!RHS1 & "  "
& rstCh1!RHS2
         End If
      rstCh1.MoveNext
   Wend

   If Not rstCh.BOF Then rstCh.MoveFirst
      While Not rstCh.EOF
        If rstCh!Category = 2 Then
           Text3.Text = Text3.Text & vbCrLf & _
           rstCh!LHS & " --> " & rstCh!RHS1 & "  " & rstCh!RHS2
        End If
      rstCh.MoveNext
      Wend
 End If
'Expanded rules for position 4

If MaxProb = ProbArray(4) Then
Text3.Refresh
If Not rstCh1.BOF Then rstCh1.MoveFirst
      While Not rstCh1.EOF
        If rstCh1!ProbabilityCntr = "4a" Then
           Text3.Text = "S --> NP VP" & vbCrLf & _
           rstCh1!LHS & " --> " & rstCh1!RHS1 & "  " & rstCh1!RHS2
        ElseIf rstCh1!ProbabilityCntr = "4b" Then
            Text3.Text = Text3.Text & vbCrLf & rstCh1!LHS & " --> " & rstCh1!RHS1 & "  "
& rstCh1!RHS2
         End If
      rstCh1.MoveNext
      Wend

If Not rstCh.BOF Then rstCh.MoveFirst
      While Not rstCh.EOF
        If rstCh!Category = 2 Then
           Text3.Text = Text3.Text & vbCrLf & _
           rstCh!LHS & " --> " & rstCh!RHS1 & "  " & rstCh!RHS2
        End If
      rstCh.MoveNext
```

```
        Wend
End If
'Expanded rules for position 5

If MaxProb = ProbArray(5) Then
        Text3.Refresh
        If Not rstCh1.BOF Then rstCh1.MoveFirst
        While Not rstCh1.EOF
          If rstCh1!ProbabilityCntr = "5a" Then
             Text3.Text = "S --> NP VP" & vbCrLf & _
             rstCh1!LHS & " --> " & rstCh1!RHS1 & "  " & rstCh1!RHS2
          ElseIf rstCh1!ProbabilityCntr = "5b" Then
              Text3.Text = Text3.Text & vbCrLf & rstCh1!LHS & " --> " & rstCh1!RHS1 & "  "
& rstCh1!RHS2
          End If
        rstCh1.MoveNext
        Wend

        If Not rstCh.BOF Then rstCh.MoveFirst
        While Not rstCh.EOF
          If rstCh!Category = 1 Then
             Text3.Text = Text3.Text & vbCrLf & _
             rstCh!LHS & " --> " & rstCh!RHS1 & "  " & rstCh!RHS2
          End If
        rstCh.MoveNext
        Wend
End If

'delete the contents of the tables BPtrProb and BPtrCat
If Not rstCh.BOF Then rstCh.MoveFirst
While Not rstCh.EOF
  rstCh.Delete
  rstCh.MoveNext
Wend

If Not rstCh1.BOF Then rstCh1.MoveFirst
While Not rstCh1.EOF
  rstCh1.Delete
  rstCh1.MoveNext
Wend
GetParseStructure
End Sub

Public Sub GetParseStructure()
'Construct the parse structure for the maximum probability parse
If MaxProb = ProbArray(1) Then
Text2.Refresh
```

```
Text2.Text = "S (NP (" & tags(1) & "  " & words(1) & " )" _
            & " (" & tags(2) & "  " & words(2) & " )" & _
            "VP (" & tags(3) & "  " & words(3) & " )" _
            & " (" & tags(4) & "  " & words(4) & " )"
ElseIf MaxProb = ProbArray(2) Then
Text2.Refresh
Text2.Text = "S (NP (" & tags(1) & "  " & words(1) & " )" & Chr(10) _
            & "  VP((" & tags(2) & "  " _
            & words(2) & " )" & "( " & POSArray(3, 4) & " (" & tags(3) & "  " & words(3) & ")"
_
            & "( " & tags(4) & "  " & words(4) & " ))))"
ElseIf MaxProb = ProbArray(3) Then
Text2.Refresh
Text2.Text = "S (NP (" & tags(1) & "  " & words(1) & " )" & Chr(10) _
            & "  VP(( " & POSArray(2, 3) & " (" & tags(2) & "  " _
            & words(2) & " )" & " (" & tags(3) & "  " & words(3) & " ))" _
            & "( " & tags(4) & "  " & words(4) & " )))"

ElseIf MaxProb = ProbArray(4) Then
Text2.Refresh
Text2.Text = "S (NP ((" & tags(1) & "  " & words(1) & " )" & _
            "(" & POSArray(2, 3) & "((" & tags(2) & "  " & words(2) & ")" & _
            "(" & tags(3) & "  " & words(3) & ")))" & _
            "VP (" & tags(4) & "  " & words(4) & "))"
ElseIf MaxProb = ProbArray(5) Then
Text2.Refresh
Text2.Text = "S (NP (" & POSArray(1, 2) & " (" & tags(1) & "  " & words(1) & " )" & _
            "(" & tags(2) & "  " & words(2) & ")" & _
            "(" & tags(3) & "  " & words(3) & ")))" & _
            "VP (" & tags(4) & "  " & words(4) & "))"
End If

End Sub

Private Sub Command2_Click()
   Unload Me
End Sub

Private Sub Command3_Click()
'To parse sentences in a file contineuosly
Dim inCnt As Integer
Dim inFnd As Integer
Dim sword, snextfile, textline As String
Dim spos, PassWrdCnt As Integer
Const delim = " "
Set db = OpenDatabase("d:\Atelach\Parse\sampledb.mdb", True)
Set rst = db.OpenRecordset("PCFGCNF", dbOpenDynaset)
```

```
Open CurrentFile For Input As #1
ParseNext = True
While Not EOF(1)
        Line Input #1, textline
        Text1.Text = textline
       If Len(Text1.Text) = 0 Then Exit Sub
          inCnt = 1
          inFnd = InStr(inCnt, Text1.Text, delim)
          wordcnt = 0
          While inFnd <> 0
             List1.AddItem Mid$(Text1.Text, inCnt, inFnd - inCnt)
             sword = Mid$(Text1.Text, inCnt, inFnd - inCnt)
             If Mid(sword, 1, 1) = "#" Then
               parse
             Else
                wordcnt = wordcnt + 1
                spos = InStr(1, sword, "\")
                words(wordcnt) = Mid(sword, 1, spos - 1)
                tags(wordcnt) = Mid(sword, spos + 1)
                If tags(wordcnt) = j Then
                   tags(wordcnt) = "Adj"
                ElseIf tags(wordcnt) = "AUX" Then
                   tags(wordcnt) = "V"
                End If
             End If
             inCnt = inFnd + 1
             inFnd = InStr(inCnt, Text1.Text, delim)
          Wend
    If inCnt < Len(Text1.Text) Then
       List1.AddItem Mid$(Text1.Text, inCnt)
    End If
Wend
Close (1)
End Sub

Private Sub Command4_Click()
'Graphic Parse Tree representation
Form2.SHow
Form2.Label1.Caption = words(1)
Form2.Label2.Caption = words(2)
Form2.Label3.Caption = words(3)
Form2.Label4.Caption = words(4)
End Sub

Private Sub dirFile_Change()
'Set path for file input
   File1.Path = dirFile.Path
```

```
   File1.Pat'ern = "*.txt;*.rtf"
End Sub

Private Sub drvFile_Change()
'Select drive for input
    Me.dirFile.Path = Me.drvFile.Drive
End Sub

Private Sub File1_Click()
'Select file for input
   CurrentFile = File1.Path & "\" & File1.filename
End Sub
```