ADAMA SCIENCE AND TECHNOLOGY UNIVERISTY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTING
DEPARTMENT OF COMPUTING

Design and Development of
Automatic Unlexicalized Statistical Constituency Parser for Tigrigna

Gereziher Hailesilassie Tafere

A THESIS SUBMITTED TO
THE SCHOOL OF GRADUATE STUDIES OF
ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR
THE DEGREE OF MASTER OF SCIENCE IN
INFORMATION SYSTEM

Adama , Ethiopia
June 2017

ADAMA SCIENCE AND TECHNOLOGY UNIVERISTY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTING
DEPARTMENT OF COMPUTING

Design and Development of Automatic Unlexicalized Statistical Constituency Parser for Tigrigna

Gereziher Hailesilassie Tafere

Signature of the Board of Examiners for Approval

	Name	**Signature**	**Date**
Advisor :			
Examiner :			
Examiner :			

Declaration

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

Gereziher Hailesilassie Tafere

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

Dr. Solomon Teffera

Acknowledgements

First and foremost, praise to Almighty God, Jesus Christ, for his name alone is exalted. Jesus Christ is my strength and my light; his love endures forever. Pray to the blessed Virgin Marry, the mother of my lord and my eternal mother. On the bright side, firstly, my special thanks goes to my endowed advisor, Dr. Solomon Teffera, for his rigorous academic guidance and excerted encouragements. I love you so much my advisor, for you lead me with the pin point ideas which become me a break through to the success end of the research. This research would have no value without your concert and very strong comments and suggestions. I loved it to the most, all the way you show me lovableness, honesty and responsibility. Thank you so much my beloved advisor. Nextly, my thankfulness goes to Dr. Martha Yirefu, Dr. Million Meshesha and Mr. Haftom Tesfay, for their lovely encouragements. I am thank full to you, Dr. Martha, for your continuous support that you have kept me from losing of all my concentrations by giving suggestive ideas every of those times I made contact with you. Thank you so much. I am thankful to Dr. Million, for you gave me an insight into smoothing techniques and for your always beloved encouragements. I am always thankful to you, Mr. Haftom Tesfay, for all your consistent advices, and brotherly encouragements. I am very proud by having you for all you show me on how to be a practical person of doing things, which I loved the most. Nextly, my greatest thanks goes to Proff. Steven Bird and Mr. Thorsten Brants, for they hold my research up to have the right research tools and also for their rigorous guidance and encouragement. My special thanks also goes to my linguistic co-advisor, Mr. Teklay Guesh, for his careful corrections with provision of knowledge understanding of modern linguistic syntactic structure for Tigrigna language. I am also thankful for Mr. Teklay Gebre-Egziabiher as well as Tedela Keleta for they provide me with tagged corpora. Last but not least, I am thankful to all who have credited to the success of this research including my families, my colleagues, stuffs in Debre-berhan University as well as Adama Science and Technology University for their significant contribution to the research. To an end, I am thankful to both Debre-berhan University and Adama Science and Technology University for all their support of research facilities.

TABLE OF CONTENTS

Content **Page**

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

LIST OF ALGORITHMS

Algorithm **Page**

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CL	Computational Linguistics
HCI	Human Computer Interaction
NLG	Natural Language Generation
NLP	Natural Language Processing
NLTK	Natural Language ToolKit
NLU	Natural Language Understanding

Abstract

Syntax Parsing is known to be an intermediate step for higher components of Natural Language Processing like Machine Translation. The research on the design and development of automatic unlexicalized statistical constituency parser for Tigrigna language is never attempted. As a result, in this research, a tree bank containing a total of 250 syntactically parsed corpus is made with the help of linguistic professional. Viterbi based bottom up probabilistic context free parsing tool is used for parsing using automatic probabilistic context free grammar induction model. Maximum likelihood estimation technique is used to extract and learn probabilities automatically from context free grammar repositories. Tigrigna word and Affix Segementer, Transliteration system and the Trigram 'n' tags which is efficient language independent tagger, are integrated as inputs in to the parser. The segementer splits morphological affixes as well as complex words into their representative base forms. While the primary role of the parser becomes structural disambiguation, where as the role of Trigram 'n' Tags tagger is to handle lexical or word category disambiguation together with the word and Morphological segementer. After all, the parser is evaluated with standard parser evaluation scoring tool called Evaluate Bracketing. Accordingly, the parser has achieved state of the art accuracy with F−Score 95.12% on on 75−25 percentage split.

CHAPTER ONE
INTRODUCTION

1.1 Background of the Study

The natural language that we use to communicate in our daily life which could be either free word order language as Czech or positional language having SVO word order as English is constructed from a highly-structured system of rules for generating speech or representing linguistic knowledge in terms of syntactic or semantic knowledge. Natural Language Processing (NLP) is a field at the intersection of Artificial intelligence(AI), Computer science(CS) and Computational Linguistics (CL) dedicated to solve problems related to Natural Language Generation (NLG) and Natural Language Understanding (NLU). The sub field of Artificial Intelligence (AI) and CL that focuses on computer systems that can produce understandable texts including how information is best communicated between machine and human using English or other human language is called NLG. As such NLG provides an important and unique perspective on many fundamental problems and questions including the following:

1. How should computers interact with poeple? What is the best way for a machine to communicate information to a human?
2. What kind of linguistic behavior does a person expect of a computer he or she is communicating with, and how can this behavior be implemented? These are basic questions in Human Computer Interaction (HCI), an area of computer science which is becoming increasingly important as the quality of software is judged more and more by its usability.
3. How can the appropriate syntactic constraints be formalised for a given language? What constitutes readable or appropriate language in a given communicative situation?

On the contrary, the sub field of AI and CL that focuses on the study of computer systems that understand English and other human languages is known as Natural language understanding(NLU). NLU is closely related to NLG, both are concerned with computational models of language and its use. However, their difference lies largely on pragmatic level. On pragmatic level, the process of NLG is thought as the reverse

process of NLU. While NLG is the process of mapping internal computer representation of information into human language(mapping semantic representation into human language), where as NLU is the process of mapping human language into internal computer representation(Mapping surface-form sentences into internal semantic representation). The former is called linguistic realization in NLG systems and where as the later is called Parsing in NLU systems. Most fundamentally, as has been observed by McDonald(1992), while the process of NLU is best characterized as one of hypothesis management(i.e., Given some input which of the multiple possible interpretations at any given stage of processing is the appropriate one?), where as NLG is best characterized as a process of choice(i.e., Given the different means that are available to achieve some desired end, which should be used?). The main focus of this research is on the later one, Syntax Parsing in NLU [1]. Syntax parsing is one of the higher components of NLP that determines the syntactic relationships between words in a sentence of a given language L and helps to achieve the syntactic level of separating sentence of a given language L having valid grammatical sequences from a sentence with ungrammatical word order which is not sentence of L. As a result, the output of Syntax analayzer is proper grammatical order or structure of sentence represented with branching tree diagrams(syntax trees) based on the allowable grammar rules of the language[2]. There are most widely used frameworks towards syntax analysis or parsing. The most common ones include constituency parsing frame work and dependency based frame work. Dependency parsers are the extensions of Constituency syntax parsers. To date, vast amount of researches on both frameworks have been conducted for resource rich languages as English language. NLP tools such as Natural Language ToolKit (NLTK) ToolKit and Stanford ToolKit including sophisticated language resources like Word Net, Syn-sets, Tree bank annotated corpora have become available for resource rich languages most notably for English as the first most resource rich language to use those tools [3–5]. Despite of the fact that the area of syntax parsing is mostly studied for resourch rich languages as English, Chinese, Germen, however, for low resourced Ethiopic semitic languges as Amharic and Tigrigna there are no existing parsing tools as well as annotated resources and the research on the area of syntax parsing is not widely studied.

1.2 Statement of the Problem

The problem of not having automatic syntactic parser leads to the difficulty of having robust system for higher components of NLP as Machine Translation, Speech Recognition and Dependency Parsing. Nevertheless, for Ethio-Semitic languages as Tigrigna Language having very rich and complex morphologies, there exist neither annotated parsed corpora(a ready made hand parsed corpora which could be directly used as an input for syntax parser) nor any research attempted on the above stated parsing frame works. In other words, Tigrigna is resource poor or low resourced language. Currently, the existing manually written modern Tigrigna grammar books are based on the theory of generative grammar, they use unlexicalized constituency based parsing framework. Hence, to make the computer based parsing development one step higher for Tigrigna language, it is preferred to start conducting a research on the design and development of unlexicalized parser for Tigrigna with these available resources. For this reason, in this research, unlexicalized constituency based parsing frame work is selected over the others. In order to conduct the research study, the researcher has selected the following key research questions in an application to the design and development of unlexicalized syntax parser for Tigrigna language by adapting tools from resource rich languages to Tigrigna Language.

1. What are the legal tree bank syntactic structures of Tigrigna language sentence useful for the development of parsing model?
2. How NLTK Probablistic Context Free Grammar(PCFG) parser is used to define such set of legal syntactic tree bank structures of Tigrigna language sentences?
3. How PCFG model is induced automatically rather than assigning probabilities to grammars by hands?
4. How can we integrate Parsing Language models with N-gram tagging language models that is with TnT tagger for lexical ambiguity resolution and out of vocabulary (OOV) words handling?
5. Tigrigna is exhibited by its having rich and complex morphologies. How can a parser can handle such complex morphological properties if they can not be handled by a tagger?
6. Given IPA for Tigrigna, what is the best way to represent Tigrigna language by its own Ethiopia Script so that users could be able to understand the syntax of their language using their own scripts?

1.3 Objective of the Study

1.3.1 General Objective

The main goal of this research is to design and develop automatic unlexicalized statistical constituency parser for Tigrigna language.

1.3.2 Specific Objectives

The specfic objectives include :

- To discuss related works.
- To prepare Tigrigna tree bank of correctly parsed sentences.
- To develop an algorithm that induces PCFG model automatically.
- To choose an algorithm that selects the best parse conforming to the PCFG grammars from the treebank.
- To develop word and morphological affixes segementer
- To develop IPA to Ethiopic translitrator.
- To select a best language independent tagger.
- To integrate all the three intermediate inputs to the parser.
- To evaluate the over all performance of the parser on test sentences.
- To forward the conclusion and recommend for future work.

1.4 Methodology of the Study

1.4.1 Approaches to the Study

In order to answer the research questions raised in the statement of the problem as well as to accomplish the specific objectives, both data collection and design science research methodologies are employed in this research work. Data collection methodology is used because annotated corpora are useful for development of syntax parsing as well as evaluation purposes and machine learning purposes. Hence, new Tigrigna tree bank of 250 correctly parsed sentences is prepared in relation to syntax of Tigrigna language. The empirical or design science research methodology is used to employ experimental

analysis in order to evaluate the model induced from the tree bank as the result of the annotated tree bank data collection process. Hence, the empirical research method is used for the discussion of the experimental evaluations based on the architectural design, algorithms used as well as techniques used in the process of developing the deep syntax parser.

1.4.2 Development Tools

In this research study, NLTK 3.1 together with Python 3.4.2 is used for development. NLTK is regarded as Natural Language tool having expressive power for text analytics. It is a leading platform for Text analytics. For this reason, NLTK is widely used by professionals for research purposes. Python 3.4.2 is selected because it works seamlessly with NLTK 3.1. Hence, in this research study, NLTK Bottom-up Viterbi PCFG Parser Tool is selected for the development of deep analysis constituency parser for Tigrigna language, because other nltk parsers used by previous researchers as NLTK chart parser lead to many plasuable ambiguous parses; how ever, Viterbi parser posses the property of Viterbi algorithm which is always based on the selection of the single best or most probable parse with the highest probability form many different parses. Hence, this tool is better for tree bank evaluation purposes in order to calculate the exact parses in terms of standard parser evaluation measures[5]. The reason is that, when tree bank is prepared by linguistic professionals, that tree bank has a single correct parse from which the algorithm learns structures for prediction and espcially viterbi kind of parser makes ease of selecting the best single parse so that that gold parse is compared to the original gold parse made by professional linguistics.

1.4.3 Evaluation Method

Evaluation methods can be broadly divided into non-corpus and corpus-based methods with the latter subdivided into un[annotated and annotated corpus-based methods (Carroll et al., 1999). The non-corpus method simply lists linguistic constructions covered by the parser/grammar. It is well-suited for hand-built grammars because during the construction phase the covered cases can be recorded. However, it has problems with capturing complexities occurring from the interaction of covered cases. Hence, corpus based evaluation methods are most widely used. The most widely used corpus-based evaluation methods are the constituent-based (phrase structure) method and the dependency method. The former method has its roots in the Grammar Evaluation Inter-

est Group (GEIG) scheme (Grishman et al., 1992) developed to compare parsers with different underlying grammatical formalisms. It promoted the use of phrase-structure bracketed information and defined precision, recall, and crossing brackets measures. The GEIG measures were extended later to constituent information (bracketing information plus label) and have since become the standard for reporting automated syntactic parsing performance. There are two major approaches to evaluate parsers using the constituent-based method. On the one hand, there is the expert-only approach in which an expert looks at the output of a parser, counts errors, and reports different measures. On the other hand, using automated software to compare with a gold standard. This replaces the counting part of the former method with a software system that compares the output of the parser to the gold standard, highly accurate data, manually parsed or automatically parsed and manually corrected by human experts. The well standard software system for PARSEVAL evaluation measure used to evaluate the bracketing performance of the output of a parser against a gold standard is called Evalb [6]. Hence, in this research, the well known standard evaluation metric for parsing known as PARSEVAL(Black et al.1991) is used together with the parser evaluation scorer, Evalb. Evalb is a freely available bracket-scoring program that reports the PARSEVAL measures precision, recall, and the number of crossing brackets (Sekine & Collins 2006). It also reports tagging accuracy as the percentage of the POS tags correctly assigned. This program takes the gold standard and parser output files as the inputs, and reports the scores for each sentence separately by printing them to the standard output [7, 8].

1.5 Significance of the Study

The finding or the significance of this study on syntax parsing for Tigrigna language is twofold. At first, Syntax parser, as man-machine interface tool, is useful for understanding the grammatical structure of a language easily. This is significant input for Tigrigna language learning in all level of educational academies beginning from primary and secondary schools to higher educational academies. They could us it to show how the modern grammatical usage of the language works as the parser can show the high level deep syntactic structure of the language automatically using branching tree structures by organizing the syntax into its constituents. Secondly, a parser is considered as the main component of a wide range of NLP systems as MT systems[9, 10], SR and Dependency Parsing. For instance, statistical parsing work has been aimed directly at particular applications like Parsing for English-Chinese Machine Translation[7, 11, 12]. As such,

the following figure shows the architecture including word alignment translation during machine translation as shown in figure 1.1. With regard to this, a parser aims to find the

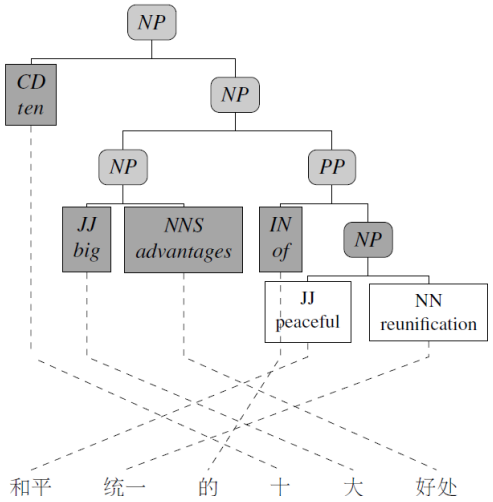


Figure 1.1: Syntactic Reordering

best syntactic tree for a given sentence among all derivations under a grammar, and MT decoder explores the space of all possible translations of the source-language sentence in order to match with sentence in the target language.

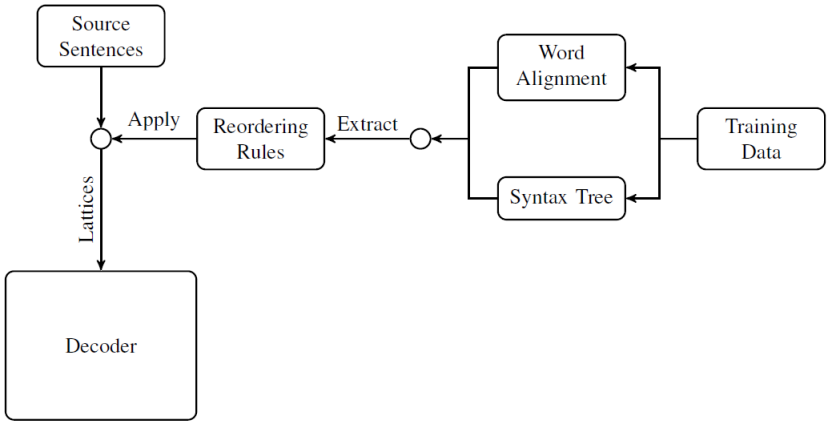


Figure 1.2: Decoder and Aligner

This shows Syntax Parser plays a great role becoming an intermediate input for the higher-level components of NLP as Machine Translation[12]. As a result, NLP community of researchers could take advantage of it. Hence, it becomes a frame work to be used as an input to higher NLP tasks for Tigrigna language.

1.6 Scope of the Study

Generative grammars show the transformation of rules[12]. Tigrigna language, as English and Amharic language, it consists of simple, compound, complex and compound complex sentences [13–15]. Modern Tigrigna grammar follows constituency based

generative grammar formalism and all types of sentences are included in the tree bank including interrogatives and conditional sentences.

1.7 Organization of the Thesis

The thesis is organized into 6 chapters. Chapter one introduces the background NLP. It explains available tools for parsing as NLTK. Ii also tries to explain the significance of Syntax Parser as grammatical syntax analyzer. Generally, It also opens up the door to conduct a research in order to see the beneficiary of adopting parsing tools from resource rich languages for syntax analysis of under resourced but morphologically complex languages as Tigrigna, states the problem area, objective of the research, research methodology, significance of the study as well as scope of the study. Chapter two covers the comparative analysis and review of NLP research works on the are of parsing including discussion of related works of Ethiopic Semitic languages. Chapter three covers an overview of Tigrigna language, Tigrigna tag-set and tree-bank corpus preparation. Chapter four discuss the design of supervised integrated parsing system architecture. Chapter five reports the experimental evaluation performance results. Chapter six concludes with the researches findings and ends up with conclusion and recommendation for future work.

CHAPTER TWO
LITRATURE REVIEW AND RELATED WORKS

2.1 Introdution

This chapter at first reviews the major challenging factors in natural language parsing as well as the existing relevant approaches towards solving the challenges by critically criticizing, comparing and contrasting the relevant literature, defining the concepts, and explaining the knowledge gap. Nextly, it follows with the discussion of related works on parsing of Ethiopic Semitic languages. The result of the discussion made in the litrature review as well as related work is then used as an input for the design and development of supervised integrated parsing system architecture for Tigrigna language.

2.2 Challenges in Parsing

Statistical parsers encounter two major problems in their analysis. These are syntactic ambiguity and the availability of resources for their machine learning.

2.2.1 Ambiguity

In the course of building one or more constituent structures according to the productions of a grammar, syntax parser is faced with a major problem called syntactic ambiguity[3]. There are two types of syntactic ambiguities faced by parsers. These are lexical and structural ambiguities. There are three types of structural ambiguities: Prepositional Phrase Attachment ambiguity (e.g. I saw a man on a hill with a telescope), Coordination ambiguity (e.g., younger cats and dogs) and ambiguity due NP bracketing (e.g., Spanish language teachers). Church and Patil (1982) showed that the Number of parses grows at rate of number of parameterizations of arithmetic expressions which grow with Catalan numbers[11].

$$C(n) = \frac{1}{n+1}\binom{2n}{n} \tag{1}$$

Consequently, as the length of the input sentences grows, the coverage of the grammar increases and the number of parse trees grows rapidly at an astronomical rate[3]. For instance, in the sentence "I saw the man with the binoculars" which has two possible

prepositional phrases "I saw the man with" and "with the binoculars", the parses or meanings (I saw a man with my own binoculars and I saw the man who has binoculars) are generated as in the following figures with trees 2.1 and 2.2

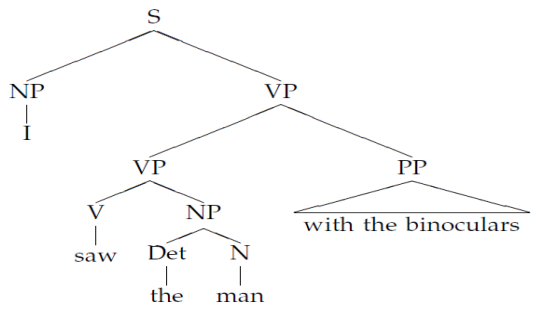


Figure 2.1: T1

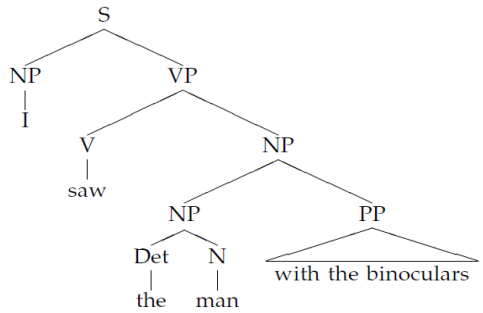


Figure 2.2: T2

Accordingly, the number of parses generated for simple noun phrases as a function of the number of trailing prepositional phrases generated can be calculated with Catalan number results as in the following table. As shown in the table above taking example

Number of PP	*Number of NP*
2	*2*
3	*5*
4	*14*
5	*132*
6	*469*
7	*1430*

Figure 2.3: Number of NPs generated as PP ambiguity grows

above the sentences I saw a man with my own binoculars generates,

$$\frac{(2\times 2)!}{(2+1)!\,2!} = \frac{4!}{3!\,2!} = \frac{(4\times 3)!}{3!\,2!} = 2$$

parses as shown above. As the result, as the tree structure becomes complex the ambiguity can very quickly make it imprudent to keep every possible parse around and so difficult to decide which of the tree structures is likely to correctly represent the text's internal organization of that linguistic input sentences. The kinds of ambiguities called lexical ambiguities also turn out to have highly ambiguous nature especially in constructing broad coverage grammars. The reason is that as soon as we try to construct a broad-coverage grammar, we are forced to make lexical entries highly ambiguous for their part of speech. For instance, in grammars with narrow coverage as a toy grammar, a is only a determiner, dog is only a noun, and runs is only a verb. However, in a broad-coverage grammar, a is also a noun (e.g. part a), dog is also a verb (meaning to follow

closely), and runs is also a noun (e.g. ski runs). Such ambiguity which is unavoidable has its own issues in efficiency leading to inefficiency in parsing seemingly innocuous sentences. Any kind of parser requires the resolution of all syntactic ambiguities that arise during the interpretation of a sentence. The same is true for Tigrigna language. Hence, dealing with ambiguity or ambiguity resolution is a key challenge in developing broad coverage parsers [3, 16].

2.2.2 Availability of Annotated Resources for Machine Learning

The availability of annotated resources or corpora is another factor that leads to performance reduction of syntax parser. The reason is that, it is very difficult to manually define a grammar whose rules determine a single parse for an arbitrary sentence. As a result, with shortage of annotated resources, it is hard to develop broad coverage syntax parsing tool, because with no availability of annotated resources, it is very hard to obtain a realistic estimate of how good parser achieve the aim of providing robust, broad coverage parsing[11].

Parsers can be classified according to the way they resolve ambiguity (hand built-grammar-based, statistical, hybrid or generalized), the way rules in parses are built (selective vs. generative), the parsing algorithm they use (LR, chart parser, Viterbi-based etc,), type of grammar (unification-based grammars, context-free grammars, PCFG, lexicalized context-free grammars, etc.), the representation of the output (bracketed, list of relations, etc.), and the type of output itself (constituency i.e, phrase structure vs dependency i.e, grammatical relations)[7, 11, 16].

2.3 Classification of Parsers-Based on Ambiguity Resolution

Based on the ablity of resolving ambiguity, there are three approaches for parsing. These are rule based approach, statistical approach and hybrid approach.

2.3.1 Rule-Based Approach

Rule based approaches also known as grammar-driven approaches because they formulate knowledge about the syntactic structure of a language in the form of hard-coded linguistic rules. These rules are then applied to parse the input text in order to produce the resulting parse tree (the best parse tree) for a given grammar. Information about individual words such as what parts-of-speech is usually stored in an online dictionary or lexicon. Such information is then accessed by the parser for each word in the input

text prior to applying the linguistic rules. Rule based parsers follow rule based parsing approach. According to a grammar, rule-based parsers search for the best parse tree by hard-coded rules using searching strategies or directions as top-down searching strategy or bottom-up one. However, since the production rules are applied recursively, different parse trees share the same sub parse trees which results in overlapping. To avoid these overlapping sub parse trees being produced more than once, the parser can cache them (i.e., memorisation) during the parsing process. Hence the parse problem can be efficiently solved by dynamic programming (DP) (Cormen et al., 2001). In DP-based parsers, the cache for sub parse trees is called the chart. Consequently, the DP-based parsers are called chart parsers. The CYK algorithm developed by Cocke (1970),Kasami (1965), Younger (1967) and Earley algorithm developed by Jurafsky and Martin (2000) are two examples of rule based chart parsers for Context free grammar(CFG). More over,rule based parser is aimed at resolving ambiguities in case the rules for the language are as simple as to cover. In case there are no simple rules to cover, it is hard for such parsers to resolve ambiguity. The other drawback of rule based parsers is that extensive amount of (dictionary) data and labor (to write the rules) by highly-skilled linguists are required in order to create, enhance and maintain them. This is especially true if the parser is required to have broad coverage (i.e., if it is to be able to parse NL text from many different domains (what one might call general text). The solution to such problems is using statistical approaches which uses statistical parsing algorithms to collect statistical data from correctly parsed sentences, and resolves ambiguity by experience through learning[12, 17].

2.3.2 Statistical Approach

Stastical approaches also called data-driven approaches because instead of being stored in the traditional form of dictionary data and grammatical rules, linguistic knowledge in these parsers is represented as statistical parameters, or probabilities. These probabilities are then commonly used together with simpler, less specified, dictionary data and/or rules, thereby taking the place of much of the information created by skilled labor in rule-based systems. Similar to rule-based parsers, the statistical parsing process can be viewed as a search algorithm. However, instead of using rules to find the correct parse tree, in contrast to rule based approaches, statistical parsing algorithms collect statistical data from correctly parsed sentences, and resolves ambiguity by experience. Instead of using rules, statistical parsers choose the best parse tree from possible candidates based

on the statistical information. Because of this, these approaches are also called learning based approaches. Similar to Statistical Machine Translation(SMT), the statistical parsing process is called a decoding process. The first attempt at statistical parsing was done by Sampson (1986) using a manually designed language model (manually parsed material) and decoding search algorithm to find the highest scoring parse. Pioneered by Geoffrey Leech and Roger Garside (Garside et al., 1987) linguists started to look for ways to define English grammar from examples using human annotated corpora. Meanwhile, study on statistical parsing began[3]. Probabilistic Context Free Grammar (PCFG) based parsers are the examples for statistical parsers. Regarding statistical parsing with CFG, an important early model is the Probabilistic Context Free Grammar (PCFG) which associates each production rule with a probability. With PCFG, a candidate parse tree can be scored by the overall probability of the rules used to generate it. Probabilistic parsing allows us to rank the parses of an ambiguous sentence on the basis of evidence from corpora and the Viterbi algorithm helps to select the best unambiguous parse from the candidate parse trees[18]. This is mainly because rule-based grammars never covered the whole usage of the language, and it was doubtful whether they would ever do so. As the result, with statistical approaches or learning-based approaches, it have been found to be more effective overall, taking into account the total amount of human expertise and effort involved. Thus, the advantage of statistical approach is at first it significantly decreases the amount of rule coding required to create a parser by making a parser to extract statistical information from the same type of text. Secondly, it also covers the whole grammar usage of the language[8, 11, 16]. Briscoe and Carroll (1993) carry the use of traditional rule-based linguistics a step further by using a unification-based grammar as a starting point. Through a process of human-supervised training on a small corpus of text, a statistical model is then developed which is used to rank the parses produced by the grammar for a given input. A similar method of interactive training has been used by Simmons and Yu (1991) to produce favorable results. The disadvantage of this approach is however sometimes it comes up with invalid sequence of parse. Moreover, the main drawback of statistical parsers is that, their performance depends on training corpus used to gather statistical information about the grammar of the language. They require large amounts of training data, often in the form of large NL text corpora that have been annotated with hand-coded tags specifying parts-of-speech, syntactic function. The solution to such problems is using Hybrid approaches[11, 17].

2.3.3 Hybrid Approach

In order to obtain more useful information and thus overcome the need for augmenting corpora with tags, rule based approaches are used together with probabilistic stastical approaches. Such an approach where by statistics-based processing is incorporated into a rule-based processing or rule based processing is incorporated into statistical parser is called bootstrapping method or hybrid (mixed approach). The framework behind this is simple, it is being able to use statistical methods to adapt (or tune) rule-based systems automatically for particular types of text in order to acquire additional linguistic information from corpora and to integrate it with information that has been developed by trained linguists. As such, it uses a rule-based parser to compute part-of-speech and their corresponding stastical rule probabilities while processing a large non-annotated corpus. These probabilities are then incorporated into the very same parser, thereby providing guidance to the parser as it assigns parts of speech to words and applies rules during the processing of new text . Researchers in statistical NLP have experimented with a variety of strategies, some of which employ varying degrees of traditional linguistic abstraction. For instance, Su and Chang (1992) group words in untagged corpora into equivalence classes, according to their possible parts-of-speech. Rather than over the words themselves, statistical analyses over these equivalence classes are performed in order to obtain higher-level trigram language models that will be used later by their statistics-based parser [11, 16]. Brown et al. (1992) have similarly resorted to reducing inflected word forms to their underlying lemmas before estimation of statistical parameters.

2.4 Based on the Parsing Algorithm used to select the best parse

Parsing requires searching a very large space of possible tree structures to connect the node "S" with the input sentences (tree leaves). With no information about which sub trees are more likely to be included in a complete parse, it can take a long time to find even a single parse. As the result search problem is one of the steps in parsing process, which has its own performance effect (the time and space complexity[11, 16]. Hence, during syntactic parsing, selection of best and efficient algorithm is required in order to guide the search of the space of possible tree structures and alleviate the problems of time and space complexity which results in performance optimization. The solutions to these problems are in the hand of Chart based tabular dynamic programming algorithm

called Viterbi algorithm which is the best selection algorithm that leads to efficient parsing[16, 19]. Moreover, based on their search problem solving approach for efficient construction of parse tree that likely represents the input sentences, the most common parsing strategies are Top-down parsing, Bottom-up, and Tabular(Chart based dynamic programming) algorithms.

2.4.1 Top-down Approach

Top down parsing strategy also called goal-driven searching strategy starts by looking at the rules in CFG picking non-terminals first and then picking rules from the grammar to expand the non-terminals in order to build the parse tree at the top (root) "S" (LHS) node and see if the input is compatible with the rules. Accordingly, the top (root) "S" (LHS) node follows with list of constituents to be built replacing the left most non terminal with each of its possible expansions, and rewriting the goals in the goal list by matching one against the LHS of the grammar rules. After all, expansion is made to list of constituents with the RHS toward the leaves on RHS, iterating until all leaves match the sentence to be derived. Finally, it backtracks when candidate POS does not match POS of current word in input string or terminates successfully when the leaves of a tree match the input sentences. The advantage of goal-driven or hypothesis driven approach is it starts with "S" and always produces "S". However top down strategy has also its own drawbacks. Its disadvantage is that it constructs trees that may not match input sentences and also generate structures inconsistent with input, do badly if there are many different rules for the same LHS (No left-recursion), performs useless work by expanding things that are possible top-down but not there and repeated work which means anywhere there is common sub structure. It also fails when no more non-terminals exist to expand in any of the trees. The solution is that, since the best option is to see the input text as quickly as possible, Ravalli(1987) notes that for top-down parsing, depth-first building is the preferred expansion technique, since "reaching the input as fast as possible will give the earliest indication of error in hypothesis" (p21). Top-down algorithm is also theoretically based on the idea of using a generative grammar to produce all possible sentences in a language until one is found which fits the input sentence(Palme1981,p208)[11, 16]. Moreover, Top down parser is hopeless for rewriting parts of speech (preterminals) with words (terminals). Recursive descent parsing is a kind of top-down parsing. Top-down parsers use a grammar to predict what the input will be, before inspecting the input. The solution to such problems is, since the

input is available to the parser all along, it would be more sensible to consider the input sentence from the very beginning. This approach is called bottom-up parsing [3, 16]. Thus, the solution to lexical lookup is always done with bottom-up parsing strategy.

2.4.2 Bottom-up Approach

Bottom up parsing algorithm is also called input-driven approach as the initial goal list of a bottom-up parser is the words (input sentences) to be parsed. Bottom up approach begins with the single word in the input sentence. if a sequence in the goal list matches the RHS of a rule, then bottom up parser replaces the sequence by the LHS of the matching rule. It then continues to combine elements in the input sentence in different ways until a tree covering the whole sentence is found. Finally, bottom up parser terminates successfully when "S" is reached and builds up trees [Palme 1981, p204, Palme 1981, p208][8, 16]. Thus, the advantage of data-driven approach is that it is constrained by the input string and always generates structures consistent with input and handles left recursion. However, the main drawback of bottom -up approach is sometimes it is blind if the RHS of several rules match the goal list. The solution to such problem is making a choice of which rule to apply. Since the best option is to see the input text as quickly as possible, Ravalli (1987) notes that for bottom-up parsing, breadth-first building is the preferred expansion technique, since " they wait to see which other words are present, and how they are structurally related, before climbing up a level" [11, 16]. The other disadvantages with such parser is it fails when no more rewrites are possible. i.e unable to deal with empty categories, termination problem which makes it incomplete. In addition to this it is inefficient when there is great lexical ambiguity. i.e it constructs constituents that may not lead to the goal "S". The solution to this is however, using grammar-driven control as a help and repeated work which means anywhere there is common sub structure. A simple kind of bottom-up parser is the shift-reduce parser. To summarize, as we can see from the above discussion, both bottom-up and top-down approaches suffer from both completeness and efficiency[3]. They are inefficient in that they generate too many useless trees, they do have no idea in resolving ambiguity, and repeated re-parsing of sub trees present which are the major problems for parsers during its analysis [3, 8, 16]. The solutions to these problems include use of Dynamic Programming approach called chart parsing which memorizes table of partial parses found so far for "re-use" if required.

2.4.3 Dynamic Programming Approach

Dynamic programming algorithms store intermediate results and re-uses them when appropriate, achieving significant efficiency gains. In DP-based parsers, the cache for sub parse trees is called the chart. Consequently, the DP-based parsers are called chart parsers. As the result, this technique can be applied to syntactic parsing, allowing us to store partial solutions to the parsing task and then look them up as necessary in order to efficiently arrive at a complete solution Such algorithms come into solution resolving all the three major problems of parsers such as resolving ambiguity, avoiding left recursion and extensive repeated work (overlapping sub parse trees being produced more than once) occurred in both bottom up and top down parsing through memorization during the parsing process, and for ambiguity resolution, (Cormen et al., 2001). There are three types of dynamic programming methods, CKY, Earley and Chart Parsing. CKY (Cocke-Kasami-Younger) algorithm is a pure bottom-up DP based search algorithm, which caches the partial parse trees of all possible subsequences of an input sentence. However, it requires first normalizing the grammar[8, 11, 16]. The solution to this problem is using Earley parser. Earley parser does not require normalizing grammar however it is more complex . Compared to the CYK algorithm, the Earley algorithm combines bottom-up search with top-down prediction. The algorithm processes the input sentences from left to right, and its chart contains the partial parse trees that cover the processed subsequences from the beginning of the sentence (Jurafsky and Martin, 2000) . Chart parsers are the efficient and robust DP approaches that overcome problems of both CKY and Earley. Unlike Earley in which case its chart contains the partial parse trees, chart parsers are efficient methods that retain completed parses in a chart and combine top-down and bottom-up searching. However, the problem with chart parsers is that they result with more than one parses giving a choice of different parse trees . The solution to such problems is using another variant of tabular parsing algorithm called Viterbi algorithm. Viterbi algorithm is the best DP selection algorithm that generates the most possible parse of sentences (Viterbi Parse) given the grammar[18]. Since Viterbi algorithm is efficient algorithm that selects the most probable parse tree resolving the ambiguity, it is termed as Viterbi parse selection[18].

2.5 Based on Parsing Model

Based on machine learning models for parsing, there are two three types of parsing models. Generative Model, Discriminative Model and Generative Model with Discriminative Reranking[18].

2.5.1 Generative Parsing Model

Generative models learn the joint probability distribution P(X,Y) over both observations X=(X1...Xn) and labels Y= (Y1...Yn) on (X, Y) pair and can model P (X, Y). Such models could use P (X,Y) to generate likely (X, Y) pairs. A classic PCFG is a generative parsing model, a model of joint probability P(X,Y) of input and output. P(X,Y) can also be transformed into $P(Y|X)$ by applying Bayes rule and then used for classification. With regard to this, generative models have many advantages, such as the possibility of deriving the related probabilities $P(Y|X)$ and P(X) through conditionalization and marginalization, which makes it possible to use the same model for both parsing and language modeling . The attractive property of such model is the fact that the learning problem for these models often has a clean analytical solution, such as the relative frequency estimation used in Treebank grammars, which makes learning both simple and efficient .Training a generative statistical parser maximizes a quantity usually the joint probability of inputs and outputs in the training set which is only indirectly related to the goal of parsing, which means to maximize the accuracy of the parser on unseen sentences. Given sentences S = (S1..., Sn) and parse tree T, the model that learns the Joint probability distribution function P(S,T) of the input sentences S and output parse tree T on (S,T) pair with

$$P(S,T) = \Pi_{n \in nodes(T)} \times P(expansion(n)|history(n)) \tag{2}$$

and selects a single most likely parse for a sentences S based on

$$T_{best} = argmax_{T:yield(T)=S} P(S,T) \tag{3}$$

is called History based generative model. In this sense, these models transform a sentence into a tree, often grounded on phrase-based grammars. Generative models based on PCFG grammars learned from corpora are still among the best performing 88%-89% labeled precision/recall (Collins 1997/99, Charniak 2000). The first attempts to automatically parse natural language sentences were mainly conducted using generative

models. A wide range of parser was, and still is, based on Probabilistic context-free grammar (PCFGs) (Magerman, 1995; Charniak, 2000; Collins, 2003). Collins' parser uses a simple bottom-up chart parsing algorithm. Collins' parser is comparatively very quick to train and has reasonable efficiency. Collins (1997) further improved the parser by proposing three different statistical models. Charniak lexicalized PCFG included the maximum entropy based model(Charniak, 2000). Petrov & Klein (2007) introduced a method to automatically refine PCFG rules by iteratively splitting them. This method leverages an efficient coarse-to-fine procedure to speed up the decoding process. A key element in the success of PCFGs is to refine the rules with a word lexicon by attaching to PCFGs lexical information called the head-word . Several head-word variants exist, but they all rely on a deterministic procedure which leverages clever linguistic knowledge. However, the main drawback with generative models is that they force us to make rigid independence assumptions, thereby severely restricting the range of dependencies that can be taken into account for disambiguation In this sense, Generative models cannot model long range dependencies within trees. The solution to these problems is using generative with discriminative rerankers. [11, 12, 16].

2.5.2 Discriminative Parsing Model

The models that learn a conditional distribution over words(labels) given sentences (observations or observed data) $P(W|S) = P(S,W) = P(W1\ldots,Wn|S1\ldots,Sn)$ are called discriminative models. Discriminative models make independence assumptions among W but not among S. They directly estimate posterior probabilities $P(W|S)$. We can train such models to maximize the probability of the output given the input. We can also minimize the loose function in mapping inputs to outputs. The approaches where the words of the sentence themselves are seen as constituents over the position they occupy in the sentence and the grammars are viewed as constituents (system constraints) over the correct parse trees are called Discriminative Constituent Parsing models[11, 16, 20]. There are two types of discriminative models. These are pure discriminative model and the conditional one. Discriminative models are called conditional models for they explicitly can model the conditional distribution $P(T|S)$ of outputs (candidate analysis T) given the input sentence S. This means that the distinct advantages of discriminative models over generative models is that we can train the model to maximize the probability of the output given the input[21]. Unlike generative models that will attempt to optimize the joint prediction of S, Pure discriminative models could be directly used to minimize

the error rate of the parser and optimize the mapping from inputs to outputs with-out explicitly modeling a conditional distribution. As the result, while discriminative models learn hard or soft that is implicit or explicit boundary between classes, generative models model distribution of individual classes. Because of this, at conditional prediction tasks logistic regression models tend to outperform naive Bayes models with the same number of parameters which means discriminative models tend to outperform generative models. Conditional models are widely used in phrase structure parsing and related frameworks. The Magerman parser (named spatter) [magerman 1995] uses a history based model, in which a parse is modeled as the sequence of decisions used to build it, with the probability of each decision conditioned on some limited aspect of the previous decisions (the history)[22]. Spatter works in a bottom up fashion, using decision trees to define a distribution over the possible moves available to the parser at each point in the parsing process. Examples of parser moves include assigning a pos tag to a word, or extending a node in a partially built tree by creating a parent-child arc. The discriminative model of magerman learns a distribution $P(T|S)$ of parse trees, given input sentences S and parse tree T, (T,S)pair, estimated using decision trees with d1d2...dn sequence of decisions with the sequence of actions of a shift-reduce parser

$$P(T,S) = \Pi^{n}_{i=1} P(di, d_1 ... d_{i-1}, S) \tag{4}$$

such decision tree models are used to pick out features of the derivational history that are predictive for a certain parsing decision. The accuracy scores achieved by spatter were 84.3% labelled precision and 84.0% labelled recall and overall result accuracy 84.1% on the Parseval metrics, which compare the labelled nodes in the parse tree returned by the parser with the labelled nodes in the gold standard PTB parse. These metrics have become the standard measure for evaluating PTB parsers. However, this model has scored 84.1% which means it fails to reach the state of the art performance. The problem with this model was label bias problem. The discriminative model of [Collins, 1996] consists of labels with lexical dependencies as a model. This model calculates the probability of the labels as product each of the labels given each of the modifiers. This has scored state of the art performance with 85.5% overall result showing some improvements over the previous discriminative model of [magerman 1995]. More recently, Petrov & Klein (2008) proposed PCFG based discriminative parsers that reaches the performance of their generative counterparts. This model has scored

state of the art performance with 90.1% overall result. Among the non-Reranking but discriminative parsers Carreras et al. (2008) currently holds the state-of-the-art. The parser of Carreras et al. (2008) leverages a global-linear model with PCFGs together with various new advanced features. Such a tag based discriminative model has outperformed discriminative parser of [magerman 1995] with state of the art result 91.1% [8, 11, 16]. Generally discriminative models perform lower but still good accuracy 86% - 87% labeled precision/recall in compared to Generative Models. However, the major problems with discriminative models lies in its complexity. This procedure requires a lot of expertise, but it is also tedious and time-consuming. Even after this painstaking process, it is still hard to say whether the selected feature set is complete or optimal to obtain the best possible results. Therefore, discriminative training methods normally require the use of numerical optimization techniques in order to associate weights to lexico-syntactic substructures. Lexico-syntactic substructures are represented as a vector of features. Because of this the performance of discriminative constituent parsing crucially relies on feature engineering[16, 23]. If the feature set is too small, it might under fit the model and leads to low performance. On the other hand, too many features may result in an over fitting problem. Usually an effective set of features have to be designed manually and selected through repeated experiments (Sagae and Lavie, 2005; Wang et al., 2006; Zhang and Clark, 2009). The use of more complex features makes the parsing problem harder and computationally intensive. Hence, the solution to such problems is that learning features automatically with machine learning algorithms as proposed by Lei et al. (2014). Lei et al. (2014) proposed to learn features by representing the cross-products of some primitive units with low-rank tensors for dependency parsing[8, 11, 12]. However, to achieve competitive performance, they had to combine the learned features with the traditional hand-crafted features.

2.5.3 Global Discriminative Model

Reranking parsers use global discriminative models. Such models come into solution to manage the complexity of conditional(discriminative) models by limiting their applicability to finite set of candidate structures generated by Generative PCFG parsing models. In such a way, these global discriminative models are used to rerank the n top candidates already ranked by more efficient base parser which is often a generative PCFG parser. The advantage of such models is that the set of candidate parse is small enough to be enumerated efficiently without dynamic programming enabling

global features. For discriminative constituent parsing, Henderson (2004) employed a recurrent neural network to induce features from an unbounded parsing history. The neural network model used by Henderson (2004) learns feature representations in the hidden layer and made parsing predictions based on the learned features in the output layer. In such model after the input units are taken, the back propagation algorithm is used to learn the prediction model parameters and induce features simultaneously. This model achieved significant improvement from pre training on a substantial amount of pre-parsed data Evaluated on standard data sets, the neural network model outperformed with state of the art accuracy 90.1%. Such model outperformed all state of-the-art parsers on Chinese and all neural networks based models on English and particularly effective on cross domain tasks for both Chinese and English Discriminative approaches from Henderson (2004); Charniak & Johnson (2005) outperformed better than standard PCFG-based generative parsers, but only by discriminatively re-ranking the K-best predicted trees coming out of a generative parser[24]. The main draw back with such types of parsers is their models only work by computing statistics of simple grammar rules (over parsing tags) occurring in a training corpus[11]. However, many language ambiguities cannot be caught with simple tag-based PCFG rules. Self-training over unlabeled corpora which means training re-ranker over a generative model is used to label the unlabeled dataset[25]. The original parser is then re-trained with this new "labeled" corpus. Such discriminative classifiers are used to rerank the list of k best trees generated by the generative model. With such re-ranking and other techniques the parsers of (McClosky et al., 2006) outperformed better than all the other models including models of Henderson (2004) and Charniak & Johnson (2005). Based on the knowledge of researchers, the state of the art in syntactic parsing is still held by McClosky et al. (2006) with state of the art accuracy 92.1% leveraging discriminative re-ranking. Magerman (1995) outlines a statistical decision-tree model, which differs from that of of Charniak (1997) mainly in the type of probabilities it considers as part of the probability model. A probabilistic parser is described by Inui et al. (1997). This model was based on an earlier parser by Briscoe and Carroll (1993) but improves on the probability model. It gains some context sensitivity by assigning a probability to each LR parsing action according to its left and right context. There is a large body of work relating to PCFG parsing, as models become more and more complex. There has also been much recent work on automatically extracting 'deep' probabilistic grammars from the Penn Treebank. These include grammars for combinatory categorial

grammar (Hockenmaier and Steedman 2002), lexical functional grammar (Cahill et al. 2002) and head-driven phrase structure grammar (Miyao et al. 2003). However, The most influential research. Smoothing attempts to compensate for sparse data by redistributing the probability mass from observed events to unobserved events. Charniak's parser achieves a 13% error reduction over the results in Collins (1997). It achieves precision and recall of 90.1 on sentences of length less than or equal to 40, with a slight drop in performance on sentences of length less than or equal to 100. The output of the Charniak's (2000)[21] parser has since been improved with the integration of a maximum-entropy reranker (Charniak and Johnson 2005). The reranker takes as input the 50-best parses returned by the Charniak parser and uses features derived from each parse to assign a weight to it, returning the parse among the original best 50 that is assigned the highest weight. This model achieves an f-score of 91.0, which is one of the highest results reported in the literature on Section 23 of the Wall Street Journal to date. The self-training parsing system of McClosky et al. (2006a) currently achieves the best f-score of 92.1 on Wall Street Journal Section 23 of the Penn Treebank. McClosky et al. (2006a) retrain the Charniak parser on both the Wall Street Journal (inserted five times into the training data) and 1.75 million sentences of the North American News Text corpus that had been parsed by the original Charniak parser. In addition, they use the reranking technology described in Charniak and Johnson (2005) to achieve maximum results. From unlexicalized parsers, currently, the best unlexicalized PCFG model is that of Petrov and Klein (2007)[26].

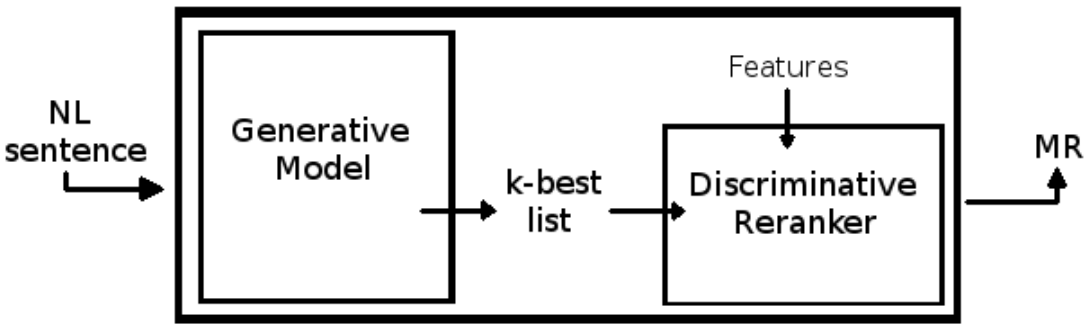


Figure 2.4: discriminative re-ranking

Where MR is meaning representation. Whilst the majority of work on statistical parsing has been for English, the following shows the summery of the state of the art statistical parsing accuracies for English parsers.

Table 2.1: State-of-the-art Parsing Accuracies for English

Model	Parser Name	Developer	Over all accuracy
	Collins parser	Collins,1999	88.2
	Charniak parser	Charniak,2000	89.6
	Berkley Parser	Petrove&klien 2007	90.1
	Stanford PCFG	Klien&Manning,2003a	85.5
	Stanford Factored	Klien&Manning,2003b	86.6
	Factored PCFGs	Hall&klien,2012	89.4
Discriminative	Carreras	Carreras et.al (2008)	91.1
Discriminative Reranking	SSN	Henderson,2004	90.1
	Discriminative Rerankers	Charniak and Jhonson,2005	91.1
	McClosky	McClosky et.al (2005)	92.1

2.6 Related Works on Ethiopic Semitic Languages

2.6.1 Part of Speech Tagger for Amahric Language

Ethiopic Semitic languages as Tigrigna possess quite rich and complex morphology[13, 14]. One way of improving performance of POS tagger is either using Majority vote or using hybrid taggers. However, researchers as [27] showed that, for segemented data the hybrid method becomes unpractical. To date, for Amharic language the researchers [27] have developed a segementer and showed that segmenting words composed of morphemes that have different POS tags is a promising direction to get better tagging accuracy for morphologically rich languages. Currently, there is neither robust morphological segementer nor robust part of speech tagger applied as an input for Tigrigna Parser. As a result, syntax analysis of such language not only requires parsed corpora for machine learning purpose but also it requires robust automatic word as well as morphological segementer and robust language independent part of speech tagger as an intermediate input to the parser. Hence, development of morphological and word segementer is also taken as the main role of this research.

2.6.2 Part of Speech Tagger for Tigrigna Language

Despite of the fact that there is an attempt made on the development of morphological analayzer for few common Tigrigna nouns and few common verbs by Gasser[28], other words, which could not be an input to a robust syntax parser due to rich morphological property of the language, currently, there no research conducted in automatic morphological segementation of Tigrigna language sentence that consist of compound words and verbal morphological affixes. In other words, the morphological analyzer developed with finite state automata by Gasser is very limited to few number of nouns and it doesn't include most of the derivational and inflectional rules of Tigrigna verbs as well as nouns. Hence, Language independent statistical taggers as TnT tagger [29] that don't merely depend on specific language for tagging, are known for their high performance efficiency of tagging for different languages by processing the morpholgy of the languages. Hence, for morphologically rich and complex languages as Tigrigna, robust automatic morphological segementer as well as robust statistical language independent tagger as TnT that could learn and handle such complex morphological features is mandatory to be used as preprocessor or an intermediate input for part of speech tagging Tigrigna words, which is in turn used as intermediate input for syntax parser to provide it with tagged inputs for syntax analysis. Two researchers have conducted a research on the development of Tigrigna part of speech tagger by taking advantage of existing tools. The first researcher is Keleta [30]. Keleta has collected part of speech tagged Tigrigna corpus and uses Conditional random field (CRF) tagger for his experimentation. He has provided more tag sets, which they become an input for other researcher as Teklay. Teklay [31] has conducted a research on the same area but used Hybrid tagging approach using existing nltk taggers, a different approach other than CRF. Teklay has used smaller tag sets than used by Kelta. In spite of the fact that Kelata has come with graphical analysis of the frequency distribution for the occurrence of most common tagsets used in Tigrigna language, the report on the performance accuracy of his tagger is left open. There is no a report of performance analysis on the part of speech tagger developed by Keleta. On the other hand, Teklay has reported the performance of accuracy for his tagger as 95%.

2.6.3 Constituency Based Sentence Parser for Amharic Language

To date, as far as the knowledge of the researcher is concerned, there is no research conducted in the area of constituency based syntax parsing for the Ethiopic Semitic

language, Tigrigna. However, for Amharic language, two researchers have conducted their research in this area, that is parsing with probablistic context free grammars f. The first research is conducted by Atelach[32]. She has conducted her research in parsing of Amharic simple sentence. On the other hand, another researcher, Daniel[33] has conducted his research in parsing of Amharic complex sentences. The main similarity between these two researchers is they have used the same techniques and algorithms in their research. They have used CKY parsing with Inside outside algorithm. The only difference between the two researchers is the type of sentences used together with the corpus size. The type of sentences used by Daniel is only complex sentences and its corpus size larger than collected by Atelach(2001) that she has collected for only simple sentences as shown in the figure 2.2. However, there are gaps left behind to be explored by both of the researchers. With this regard, the gaps that are not covered by both of the researchers include :

1. Both researchers have used Bigram language model for lexical ambiguity resolution. Hence, the need to use Trigram language model is left open to be explored.
2. Extraction of grammar rules and generation of the corresponding probabilities is done manually by both of the researchers. In other words, the research on automatic grammar induction of the corresponding probability computations dynamically is left for future researchers by both of them. Hence, development of automatic grammar induction is the main goal of this research.
3. The sentences collected by Atelach are only simple sentences, and those collected by Daniel are only complex sentences. Hence, the need to prepare a corpus that incorporates all representative types of sentences including simple and complex sentences is again left open to be explored.
4. Ethiopic semitic languages need to be represented by their own script language which Ethiopic script. In other words, Ethiopic semitic languages have their own alphabetical representation known as Geez alphabet as shown table 3.1 which is commonly used in educational academies and other institutions that use Tigrigna language. However, both researchers, Atelach and Daniel, have used the system of transliteration instead of Ethiopic script representation. Hence, the development of a parser that could produce syntax in its own local language representation which is easily understandable for the language users is also left opened. Hence, at foremost, in the process of building Treebank Style Corpus, the anno-

tated corpus should be representative of the local language and not the transliteration one.

Table 2.2: Constituency Based Parsing Accuracies for Amharic Language

Researcher	Type of Sentences	Sampling Technique	Distribution of different Phrases	Corpus Size	Average Accuracy
Atelach (2002)	Simple Sentence with Uniform Construct	Random Sampling	Judgement Sampling Technique	Total=100 Train=80 Test=20	85%
Daniel (2003)	Complex Sentences	Random Sampling	Judgement Sampling Technique	Total=350 Train=280 Test=70	85.6%

2.7 *Summary*

In this chapter, review of literature and related works are done. Since the main goal this research is not only to review the challenges and solutions of parsing but also to fulfill the gaps of previous researches on Amharic syntax parsing, in this chapter, the related works on parsing of Amharic language is discussed including the total amount of data they used for experimental evaluation , the method of their evaluations, the techniques they used for parsing as well as their gaps not explored 2.2. Taggers and word and morphological segmented have high role in handling the morphological complexities of a language and these are also discussed in the related works . To an end, a very efficient parsing algorithms called Viterbi algorithm is selected as a dynamic programming algorithm for calculating the most-probable parse conforming to PCFG induced model. The reason is that, PCFG parser not only is a statistical parser that uses statistical induced PCFG language model for parsing but also it uses the Viterbi algorthm, a chart (or tableau) based parsing algorithm that stores intermediate results, avoiding the need to recompute identical analyses in different parts of the search space. Hence, Viterbi algorithm is well-suited to PCFG parsing. Consequently, in this research the NLTK parsing tool called NLTK Viterbi Bottom-up PCFG parser, is selected. Parser also requires high performance statistical tagger for lexical ambiguity resoulution. Hence, TnT tagger which is high performance language independent tagger is also selected to be integrated to the parser.

CHAPTER THREE
TGRIGNA MORPHO-SYNTAX AND TREE-BANK PREPARATION

3.1 Introduction

This chapter a first begins by highlighting the overview of Tigrigna language then discusses the syntax of the language . This is done because in syntax analysis they are mandatory part of the syntax analyzer,they are helpful to create syntactically structured sentences organized into different phrases. The main significance of this discussion is then to be used as an intermediate input for Building Tigrigna Treebank Annotated Corpus(BTTAC), which in turn is used an immediate input for supervised machine learning of the parser tool used in the research.

3.2 Tigrigna Language

Tigrigna also written as Tigrinya is officially designated language of both Tigray and Eritrea. In Eritrea and in Tigray,Tigrigna language is not only used as primary language of oral and written communication but also it serves as a medium of instruction in the primary and secondary schools. It is also taught as a subject in the junior and senior secondary schools of Eritrea and Tigray. It is a language found in development where by grammar books such as Tigrigna books like Tigrigna grammar Book (John Mason ,1996),Modern Tigrigna Grammar Book (Tesfay Tewolde,2002), Modern Tigrigna Grammar Book (Daniel Teklu,2003) are published in response to the demands of professionals and curriculum designers as well as teachers of the language as reference materials and teacher's guide.

Tigrigna is closely related to Amharic language as (A.L. Schlozer,1781) named the Semitic languages. Similar to Amharic language, Tigrigna language is also rooted with Geez alphabet(i.e Ethiopic syllabary) special writing system. In addition to this both languages posses the same grammatical word order which is Subject-Object-Verb(SOV). Syllable in Tigrigna is considered to be a consonant followed by a vowel. Accordingly, each syllable in the Ethiopic syllabary denotes a combination of a consonant and a vowel with 35 basic characters. According to the vowel with which the basic symbol is combined, each of these characters have different forms called orders. The system of vowels applied to all consonants in tigrigna language is exhibited in the

alphabetical chart commonly called "Ha Hu" after the first two letters as shown in Table 3.1 [13–15]

A BGN/PCGN of 2007 Agreement is employed for Tigrigna system of Romanization by [34, 35].

Currently,modern Tigrigna language grammars uses the concept of Chomsky's theory of Generative Grammars which makes it easy for studying computer based Natural Language Processing for the language. As a result,to develop computer based syntax analayzer for Tigrigna language,the researcher has taken the advantage of such modern linguistic theory with the modern books written and published based on this theory for the modern usage of the language [13–15, 36].

Table 3.1: Tigrignal Language Alphabets

BASIC CHARACTERS

	1st order		2nd order		3rd order		4th order		5th order		6th order[1]		7th order	
1	ሀ	he	ሁ	hu	ሂ	hi	ሃ	ha	ሄ	hie	ህ	h / hĭ	ሆ	ho
2	ለ	le	ሉ	lu	ሊ	li	ላ	la	ሌ	lie	ል	l / lĭ	ሎ	lo
3	ሐ	ḥe	ሑ	ḥu	ሒ	ḥi	ሓ	ḥa	ሔ	ḥie	ሕ	ḥ / ḥĭ	ሖ	ḥo
4	መ	me	ሙ	mu	ሚ	mi	ማ	ma	ሜ	mie	ም	m / mĭ	ሞ	mo
5[2]	ሠ	se	ሡ	su	ሢ	si	ሣ	sa	ሤ	sie	ሥ	s / sĭ	ሦ	so
6	ረ	re	ሩ	ru	ሪ	ri	ራ	ra	ሬ	rie	ር	r / rĭ	ሮ	ro
7	ሰ	se	ሱ	su	ሲ	si	ሳ	sa	ሴ	sie	ስ	s / sĭ	ሶ	so
8	ሸ	she	ሹ	shu	ሺ	shi	ሻ	sha	ሼ	shie	ሽ	sh / shĭ	ሾ	sho
9	ቀ	k'e	ቁ	k'u	ቂ	k'i	ቃ	k'a	ቄ	k'ie	ቅ	k' / k'ĭ	ቆ	k'o
10	ቐ	kh'e	ቑ	kh'u	ቒ	kh'i	ቓ	kh'a	ቔ	kh'ie	ቕ	kh' / kh'ĭ	ቖ	kh'o
11	በ	be	ቡ	bu	ቢ	bi	ባ	ba	ቤ	bie	ብ	b / bĭ	ቦ	bo
12	ተ	te	ቱ	tu	ቲ	ti	ታ	ta	ቴ	tie	ት	t / tĭ	ቶ	to
13	ቸ	che	ቹ	chu	ቺ	chi	ቻ	cha	ቼ	chie	ች	ch / chĭ	ቾ	cho
14	ኀ	ḥe	ኁ	ḥu	ኂ	ḥi	ኃ	ḥa	ኄ	ḥie	ኅ	ḥ / ḥĭ	ኆ	ḥo
15	ነ	ne	ኑ	nu	ኒ	ni	ና	na	ኔ	nie	ን	n / nĭ	ኖ	no
16	ኘ	nye	ኙ	nyu	ኚ	nyi	ኛ	nya	ኜ	nyie	ኝ	ny / nyĭ	ኞ	nyo
17[3]	አ[4]	e / 'e	ኡ	u / 'u	ኢ	i / 'i	ኣ	a / 'a	ኤ	ie / 'i	እ	ĭ / 'ĭ	ኦ	o / 'o
18	ከ	ke	ኩ	ku	ኪ	ki	ካ	ka	ኬ	kie	ክ	k / kĭ	ኮ	ko
19	ኸ	khe	ኹ	khu	ኺ	khi	ኻ	kha	ኼ	khie	ኽ	kh / khĭ	ኾ	kho
20	ወ	we	ዉ	wu	ዊ	wi	ዋ	wa	ዌ	wie	ው	w / wĭ	ዎ	wo
21	ዐ	'e	ዑ	'u	ዒ	'i	ዓ	'a	ዔ	'ie	ዕ	' / 'ĭ	ዖ	'o

3.3 Syntax of Tigrigna Language

Syntax describes the structured system of communication that language speakers can express their thoughts usually interms of tree or phrase structure diagrams. It deals with the system of rules formed from combination of word class phrasal categories which allow word classes to be combined to form different types of phrases and sentences of a language. With this regard, any Tigrigna sentence, as in any other language, consists of two largest immediate constituents known as subject and predicate. All subjects are Noun Phrases where as all predicates are Verb Phrases. In other words,the subject which is used to mention something is thought of as a Noun Phrase(NP) and the predicate which is used to say something true or false about the subject is thought as Verb Phrase(VP). As a result, syntax analayzer usually describes sentences structure in terms of these two largest constituents NP and VP. The former is headed by a Noun(N) where as the latter is headed by a Verb(V) as shown in Figure 3.1. The phrases have one word each ,/hawka/ headed by noun which functions as a subject and the word /mes'iu/ headed by verb which functions as predicates.

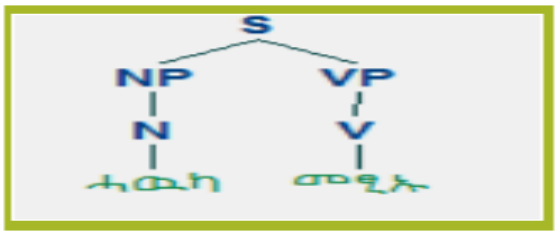


Figure 3.1: Sentences Structure

3.3.1 Basic Rules for Deriving Noun Phrases

Tigrinya has SOV(subject, object, verb) word order. The noun which is used as a subject comes in a sentence first position and is the head of a Noun Phrase(NP). Accordingly, in Tigrigna language, there are basic rules for forming Noun phrases, where by combining one or more of them other Noun phrases could be constructed. Table 3.2 shows the basic rules for forming Noun phrases.
As shown in figures 3.2 and 3.3 ,the phrase may consist of one or more dependents which can be determiners and/or modifiers. It may consist of the head alone.

As we can understand from figure 3.3, Tigrinya noun phrases can be divided into two immediate constituents. These are determiners (DET) and noun phrases which can again be divided into the head noun and its modifiers. Determiners give information relating to definiteness and indefiniteness i.e., they indicate whether what is referred

Table 3.2: Basic Rules for Deriving Noun Phrases

No	Basic Generative Rules for Deriving Noun Phrases
1.	NP→N
2.	NP→NP+N
3.	NP→ADJP+N
4.	NP→S+N
5.	NP→DET+N
6.	NP→DET+NP+N
7.	NP→DET+ADJP+N
8.	NP→DET+S+N
9.	NP→DET+S+NP+N
10.	NP→DET+S+ADJP+N
11.	NP→DET+S+ADJP+NP+N

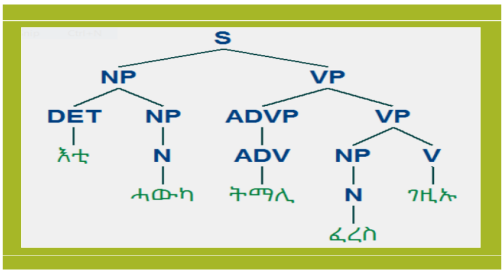


Figure 3.2: እቲ ሓውካ ትማሊ ፈረስ ገዚኡ

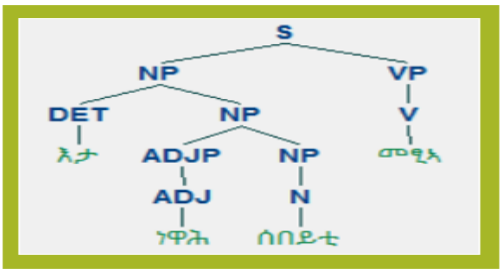


Figure 3.3: Noun Phrase

to by the NP is familiar to both the speaker and the hearer or not. Besides, they give information about quantity and proportion.As shown in the tree diagrams Figure 3.4 ,the predeterminer /kullen/ predetermines the full NP. In this way determiners determine the NP whereas the adjective phrase modifies the NP.

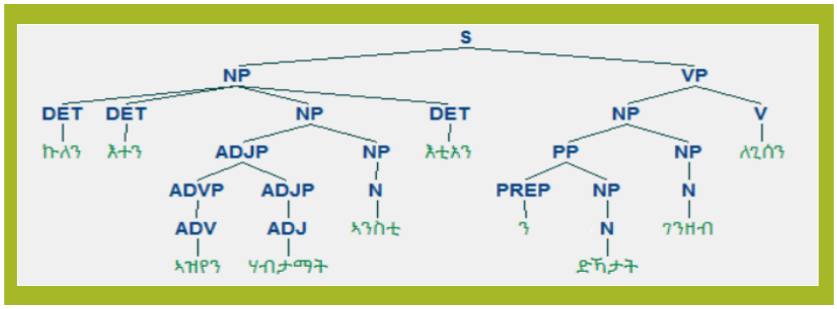


Figure 3.4: predetermines the full NP

3.3.2 Complex Noun Phrase

if the subordinate clause contains a sentence that branches from the subject NP as in figure 3.5 or from the object NP as in figures 3.6 and 3.7 , then the the phrase formed from such clause is called Complex Noun Phrase. Figure 3.5 shows that the subordinate

clause(embeded sentence) /ahatey zigeza'a mes'haf/ branches from the NP^1(i.e., from the subject NP) which is again a sister of the VP^1 from the highest node. Thus, we can see that the head of the subordinate clause is mes'haf. /zi-/ is a complementizer in the subordinate clause.

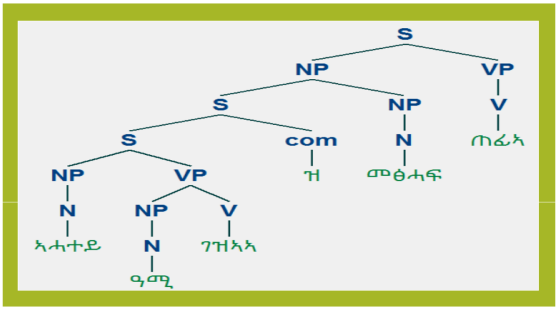


Figure 3.5: CNP 1

In figures 3.6 and 3.7 , the subordinate clauses(embedded sentences) branch from NP. In each of the embedded sentences ,the subject is /ane/ and the object is /mes'haf/. The object NP refered by /mes'haf/ is indicated by pronominal affixes attached to the verbs of embedded sentences(the object pronominal suffixes). Both embedded sentences are relative clauses that modify that NP /mes'haf/ (their head) which is an object to the transitive verb /wesidu/. These subordinate clauses differ from the main clauses mainly because they contain the relativizer /zi-/. In both figures,the complementizer /zi-/ and S(sentence) form subordinate clauses each of which has an object empty position. In the former , an empty object position is indicated by the pronominal object suffix /-a/ 'her' preceded by the geminated epenthetic segment /w/ where as in the later we have an empty object position which is indicated by the pronominal object suffix /-o/ preceded by the geminated epenthetic segment /w/. Moreover, relative clauses modify their respective head nouns. Each of the embeded sentences has a complementizer which occurs attached to a verb. It is /zi-/ if it is prefixed to perfective verbs. But,in the imperfective,it may be realized as /zi-/,/i/ or 0(i.e,complementizers do not occur attached to gerundive, jussive or imperative). The complementizes of complex verb phrases are, however, different from different prepositions such as /kem/ 'as' which may be followed by the complementizers.

3.3.3 Basic Rules for Deriving Adjective Phrases

An adjective which is used to modify a noun comes in a sentence position before and is the head of a Adjective Phrase(ADJP). Accordingly, in Tigrigna language, in Tigrigna language, there are basic rules for forming Adjective phrases, where by combining one

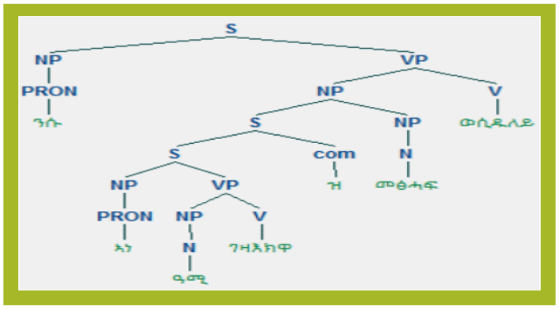


Figure 3.6: Transitive verb

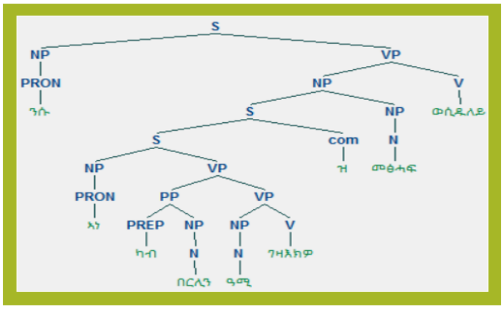


Figure 3.7: clause 2

or more of them other Adjective phrases could be constructed. Table 3.3 shows the basic rules for forming Adjective phrases. The phrase may consist of the head alone, but also it can also be accompanied by dependents. The dependents involve the expression of degree. The modifiers may be degree adverbs or others functioning as degree adverbs. /bit'ami/ is originally a PP, but it is used(functions) as a degree adverb. The adverbial modifiers precede the adjective head.

Table 3.3: Basic Rules for forming Adjective Phrases

No	Basic Generative Rules for Deriving Adjective Phrases
1.	ADJP→DET+ADJ
2.	ADJP →DET+S+ADJ
3.	ADJP →DET+PP+ADJ
4.	ADJP →NP+ADJ
5.	ADJP →DET+ NP+ADJ
6.	ADJP →DET+S+NP+ADJ
7.	ADJP →DET+ PP+NP+ADJ
8.	ADJP → PP+ADJ
9.	ADJP →DET+S+ADJP
10.	ADJP → S+NP+ADJ
11.	ADJP → PP+NP+ADJ

3.3.4 Complex Adjective Phrase

if the subordinate clause contains a sentence that branches from an adjective phrase,then the phrase formed from such clause is called Complex Adjective Phrase. The head of such phrase is an adjective phrase. In the figure 3.8, the embeded sentences /kab kahsay zigedede/ modifies the adjective /has'ir/ and form complex adjective clause. Just as any adjective,the complex adjective clause shown in the figure 3.9 , whose head is an adjective /has's'ir/ can modify a noun and in this case it modifies the noun seb 'person'. The subject of each of in figures the matrix sentences 3.8 and 3.9 is /mannim/ 'none'.

if we change the subject of the sentences from /mannim/ to /nissa/,then the verbs must agree with their respective subject as in figure 3.10.

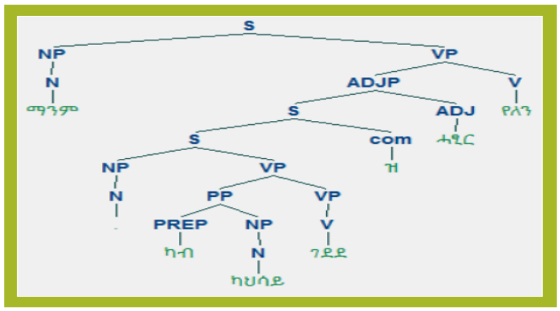


Figure 3.8: CAP

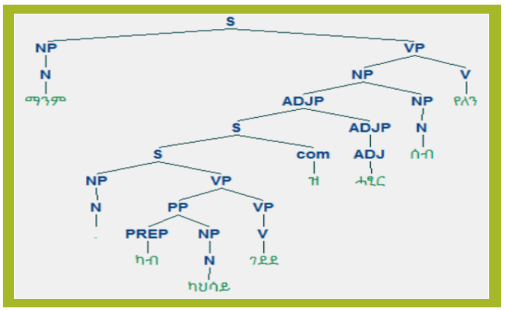


Figure 3.9: CAP 2

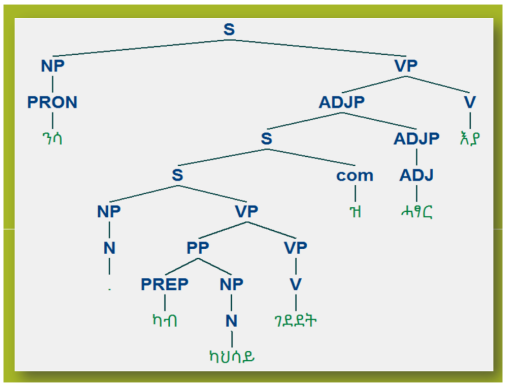


Figure 3.10: CAP 3

3.3.5 Basic Rules for Deriving Verb Phrases

A verb in Tigrigna language comes in a sentence final position and is the head of a verb phrase. The head of Verb Phrase(VP) is a Verb(V). In Tigrigna language, there are basic rules for forming verb phrases, where by combining one or more of them other verb phrases could be constructed. Table 3.4 shows the basic rules for forming verb phrases. As in figures 3.11, 3.12 and 3.13, the verb can be combined with prepositional phrases, S(stand alone) or NP(among other categories) to form a verb phrase (VP). In figure 3.11,the verb /keydu/ combines with the prepositional phrase /nidesye/ to form a VP which is again combined with the PP /bimakkina/ to form a larger VP. The prepositional phrase /bimakkina/ is optional and can occur with almost any verb. Thus,it cannot be used to subcategories the verb. The information added by this phrase is only additional and not essential. In figure 3.12 the verb (the head) stands alone. The intransitive verb /moytu/ does not need a noun to complement it and hence it is both a verb and verb phrase. In figure 3.13, the transitive verb /gezi'u/ needs a direct object to complement

it. Hence, together with the NP it forms a VP.

Table 3.4: Basic Generative Rules for Deriving Verb Phrases

No	Basic Generative Rules for Deriving Adjective Phrases
1.	ADJP→DET+ADJ
2.	ADJP →DET+S+ADJ
3.	ADJP →DET+PP+ADJ
4.	ADJP →NP+ADJ
5.	ADJP →DET+ NP+ADJ
6.	ADJP →DET+S+NP+ADJ
7.	ADJP →DET+ PP+NP+ADJ
8.	ADJP → PP+ADJ
9.	ADJP →DET+S+ADJP
10.	ADJP → S+NP+ADJ
11.	ADJP → PP+NP+ADJ

The PP 'bimakkina' in 3.11 is an adjunct (adjunct adverbial) while the noun /makkina/

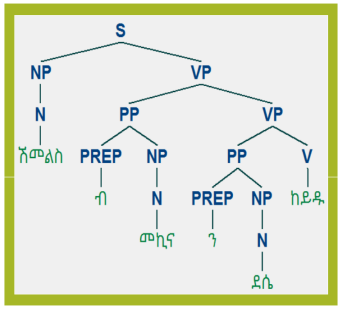


Figure 3.11: VP 3

Figure 3.12: VP 4

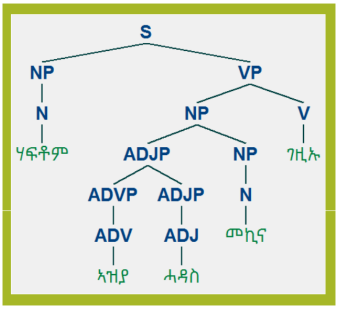


Figure 3.13: VP 5

'car' in 3.13 is a complement(more specifically an object).

3.3.6 Complex Verb Phrase

In a given sentences,if a verb contains sentential complement or modifier ,then the phrase formed from such verb is called Complex Verb Phrase(CVP). The type of verb in such phrase is called complex verb.

For instance figures 3.14 and 3.15 has two sentence: hanna felit'a 'Hanna knew' and hawa tesheleme 'that her brother won'. Figure 3.14 shows that the embeded sentence /hawa kemzimote/ is a complement of the transitive verb /felit'a/. It can be substituted by an object such as /seb/ 'man' as in sentence in the figure 3.15 show that the verb /felit'a/ can be complemented by a noun NP or by a subordinate clause. The subordinate clauses (embeded sentences) can also be modifiers of verbs. In figures 3.16 and 3.17 ,

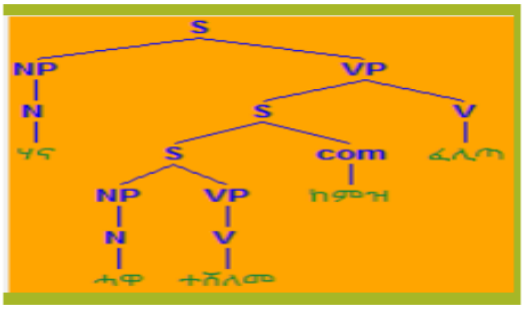


Figure 3.14: CVP

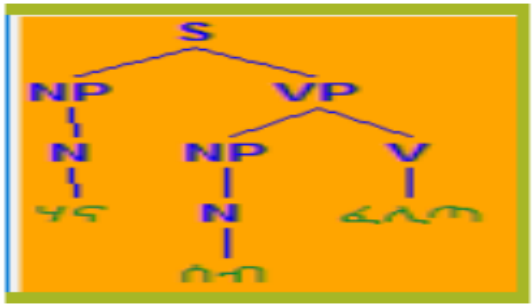


Figure 3.15: CVP 2

the embeded sentences are modifiers and not complements because they are not sisters of the verb. In fact,each of them branch from VP^1 . The verb /hazina/ in figure 3.16 is an intransitive verb and does not need a complement,while the verb /beliE'a/ in figure 3.17 has already a complement /misaha/ and the subordinate clause must branch from a higher node i.e., VP^1. In Figure figure 3.17 , the embeded sentences function as adverb of time Where as in Figure 3.16 and the embeded sentences function as adverb of reason. But, notice that they only function as adverbials. They are not adverbs. Embeded sentences not only have Verbs branching from Sentences but also the embeded

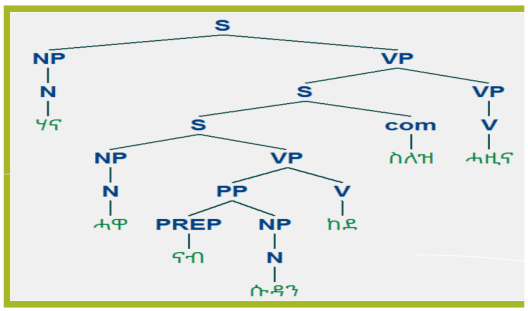


Figure 3.16: CVP 3

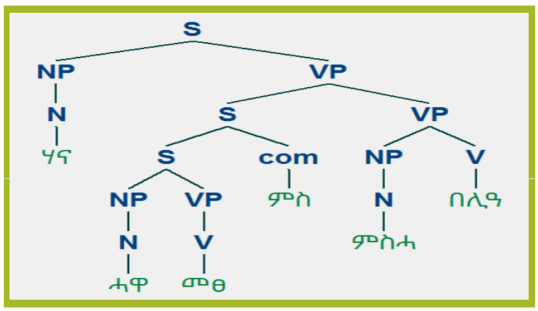


Figure 3.17: CVP 4

sentences could have infinitives instead of verbs as in Figures 3.18 and 3.19.

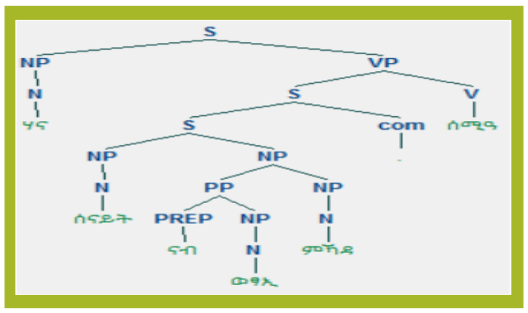


Figure 3.18: ninf

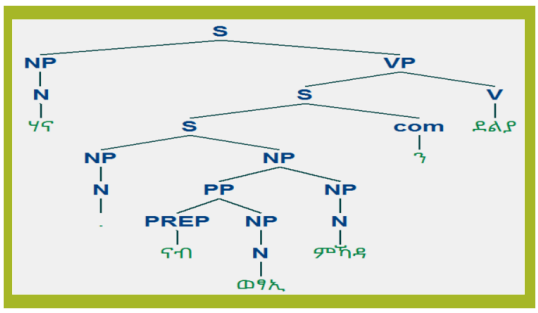


Figure 3.19: ninf 2

3.3.7 Adverb Phrase

As in other languages like English, an adverb in Tigrigna modifies a verb. The head of Adverb Phrase(ADVP) is an Adverb(ADV). The phrase may consist of just the head

alone or may be accompanied by other dependents. An adverb phrase can be a modifier of a verb, an adjective, an adverb, a clause or a sentence. Tigrinya does not have an open word class called adverb. In fact, it has only few words we call adverbs. Hence, the adverbial function is usually expressed by other categories. The adverbial meaning is usually expressed by other words such as nouns, prepositions +nouns or prepositions +adjectives. The adverbial modifiers and complements of Tigrinya can be expressed by nouns, such as /lomi/ 'today' and also prepositions' followed by nouns or adjectives. For instance, the complement nab tembyen 'to Tembien' is formed from a preposition (nab) 'to' and a noun Tembien. But, the function is adverbial. Again the preposition bidinget 'by accident' consisting of a preposition /bi/ 'by' followed by a noun /dinget/ 'accident' is used as modifier. Most of the adverbial functions in Tigrigna are not expressed by adverbs. Modifiers of verbs or verb phrases usually express time, place instrument, manner. Modifiers of adverbs and adjectives commonly express degree for instannce aziyu expresses degree, while adverbs functioning as sentence modifiers usually express the speakers attitude regarding the event spoken of. For instance:
bi'aggat'ami abta makkina zineberu sebat dihinom /the men on the car were saved by chance/
The phrase /bi'aggat'ami/ is functioning as a sentence modifier and expresses the speakers view towards the event.

3.3.8 Prepositional Phrase

In addition to the phrases we have seen so far, Tigrinya has Prepositional Phrases(PP) consisting of adpositions (i.e., Post positions and prepositions), NPs, and ADJPs. The governor of a PP is a preposition. The great majority of Tigrinya adpositions are prepositions and hence we can simply refer to them as prepositions. PPs can be used as modifiers or complements. Consider the illustrations below: The prepositional phrase ?ab may 'in water' is a complement, whereas in the prepositional phrase bihimam 'of illness' is a modifier. Let us also see the following: In figure biqilt'uf is a modifier and branches from the higher VP, while nab nageru is a complement and hence branches from the lower VP.

3.3.9 Constituent Movement-Interrogative Sentence

The theory of movement explores the movement restriction in languages. It explains the restriction that human languages actually place on movement. The syntax of Inter-

rogative sentence takes the behaviour of constituent movement such that in recognition of syntax with interrogative sentences the surface structure (S-structure) is moved to Deep Structure(D-structure) as in figure 3.20

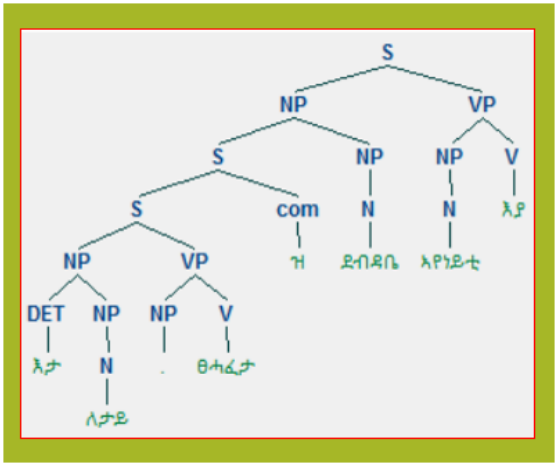


Figure 3.20: Interrogative Sentence

3.4 Building Tigrigna Language Tree-bank

A treebank is a collection of syntactically annotated sentences where each structure has been hand corrected by linguists. The idea is that, a linguist finds it easier to assign an analysis to one particular sentence than to come up with a comprehensive grammar from scratch. We can assume that there is an implicit grammar underlying all the analyses. This grammar is often made quite explicit by means of annotation guidelines A treebank provides a way of using supervised learning to automatically acquire a PCFG. The Penn Treebank (Marcus et al. 1993) is the most widely used treebank in computational linguistics for most resource rich languages including English as a first resource rich language and other resource rich languages as German, Chinese and Arabic. For instance, the parsed syntax structure consists of Brown Corpus consists of one million words from the Wall Street Journal, parts of the Switchboard Corpus and ATIS. The texts are all part-of-speech tagged and a syntactic structure has been assigned to each sentence[37–41]. In this research, a total of 250 parsed sentences are prepared with random data collection method from grammar books and from tigrigna part of speech tagger researcher, teklay, and covering all types of sentences including simple,complex, compound complex ,Interrogative, Conditional sentences and their corresponding clauses based on modern Tigrigna grammar. These sentences are then syntactically annotated and corrected by Mr. Teklay Guesh, Tigrigna linguistic professional at Addis Abeba University. Though the corpus only includes reduced tag sets,

however over 100 tagsets are identified for Tigrigna language and based on the morphological segementer is implemented to handle the pre-processing steps faced by the parser. Hence, these largest tagsets which are over handred tagsets where by most of them have indicators for segmentation which results for the parser to parse complex sentences without having the problem of over tagging. Because, after splitting by the segementer, then they are turned into their reduced tagsets as shown in Table 3.5

Table 3.5: Tigrigna Reduced Tag sets

Reduced Tagset	Description
N	Noun
V	Verb
EC	Empty Constituent
PREP	Preposition
DET	Determiner
NPL	Plural Noun
COM	Complement marker
PRON	Pronoun
ADV	Adverb
C	Conjunction
Aux	Auxiliary

This is advantegous interms of both time and space complexity to run for the parser. Hence segementation can provide a way of handling morphological complexities of morphology rich languages[42] The reduced tagsets used in this research together with their total frequency distribution is shown in 3.21 :

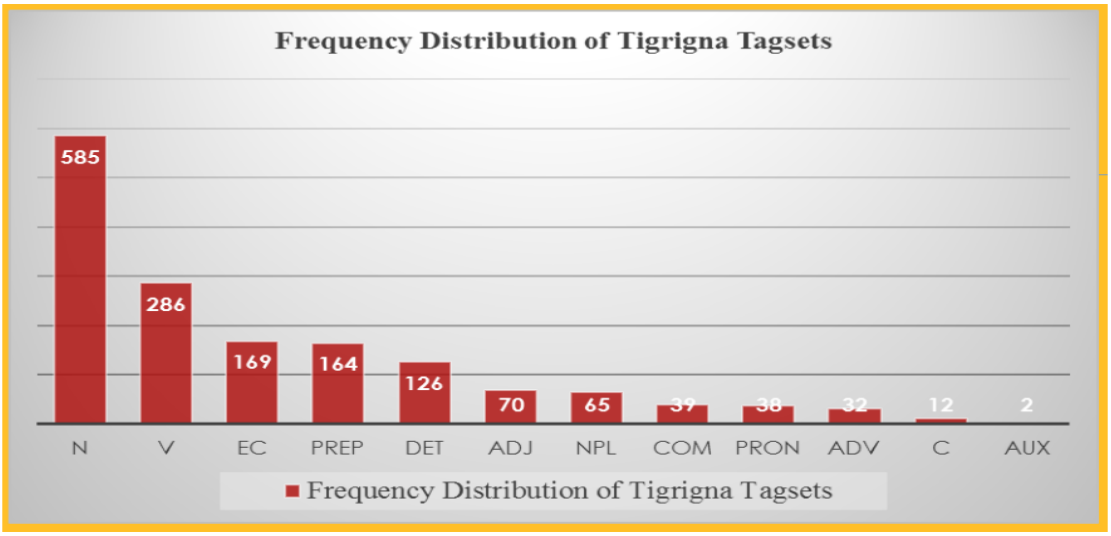


Figure 3.21: Tigrigna Reduced Tag sets Frequency Distribution

3.5 *Summery*

In this chapter detail discussion of Syntax of Tigrigna language is made. A Total of 250 Syntactically annotated Treebank data are prepared by linguistic professional in Addis Abeba University and experimented by the system. The syntax of modern Tigrigna grammar is constructed from the phrases of open and closed word classes. The phrases in Tigrigna modern grammar are Noun Phrase, Verb Phrase, Adjective Phrase, Prepositional Phrase, Adverb Phrase, Complex Verb Phrase, Complex Noun Phrase and Complex Adjective Phrase. Hence, the syntax structure of Tigrigna language is constructed from all these phrases and the Syntax trees for all these phrases are discussed in this chapter. The examples for each Syntax structures given in this chapter are all the result of the Syntax analyzer or the output of the parser that are correctly parsed and matched with the parsed sentences annotated by the linguistic professional. The syntax a total of 11 Tag sets are prepared and used in this research since the separation of complex compound words and affix morphologies into their base forms is made by word and morphological affix segementer developed in this thesis. TnT tagger is used to tag the sentences and the frequency distribution of the tag sets is also the result of the experiment done with the TnT tagger calculated using conditional frequency distribution of the tag sets all over the corpus.

CHAPTER FOUR
SUPERVISED PARSING INTEGRATED SYSTEM ARCHITECTURE

4.1 Introdution

This chapter in detail discusses the parsing integrated system architecture for Tigrigna language designed as by product of the litrature reviews.

The event which led to the large interest in statistical parsing among the NLP community was the release of the Penn Treebank (Marcus et al., 1993). Nextly, Magerman (1995) was among the first to release parsing accuracies on the new tree bank who has introduced the idea of standard splits of the Wall Street Journal (WSJ) part of the PTB for training, development and testing; Sections 2-21 have typically been used for training, section 00 for development, and section 23 for testing.

4.2 Parsing

Parsing is the term used to describe the process of automatically building syntactic analyses of a sentence in terms of a given grammar and lexicon. The task of parsing is a central one in the field of computational linguistics[43]. For further processing (e.g. semantic, prosodic), one would like to be able to have an analysis for the internal structure of a sentence. In many cases, this analysis is in the form of a phrase structure tree [44]. In most contemporary grammatical formalisms, the output of parsing is some-thing logically equivalent to a tree, displaying dominance and precedence relations between constituents of a sentence, perhaps with further annotations in the form of attribute-value equations ('features') capturing other aspects of linguistic description. As such, the resulting syntactic analyses may be used as input to a process of semantic interpretation,(or perhaps phonological interpretation, where aspects of this, like prosody, are sensitive to syntactic structure). There are many different possible linguistic formalisms, and many ways of representing each of them, and hence many different ways of representing the results of parsing. Providing 'deep' analysis for natural languages is a large research area and includes dependency parsing and unification (or constraint-based) grammars.

4.2.1 Important Features of Good Parser

Syntactic parsers mostly refered by the alogrithm they use , should have several important properties in order to be practically useful for large scale text processing and thus be embedded as component in larger systems[7, 8, 18]. Important features of good parser include:

1. Resource-needy : if the parser requires large quantities of memory or it takes a long period of time to parse average sentences making it unusable in interactive systems.
2. State of the art Performance : if the number of correctly parsed sentences is 85% or higher. Standard corpora get their importance especially when evaluation is to be made over syntactic parsers. The common metric for evaluating parsers is called PARSEVAL measures, as proposed by Black et al.(1991), this helps to see how well the developed parser achieves state of the art parsing performance.
3. Self-tagging : whether or not the parser does tagging itself by taking in the raw text since this eliminates the need for extra modules.
4. Robustness : relates to the property of a parser to handle any type of input sentence and return a reasonable output for it and not an empty line or some other useless output. It should behave in a reasonably sensible way when presented with a sentence that it is unable to fully analyse successfully.
5. Long sentences : the ability of the parser to handle sentences longer than 40 words
6. Stability : whether or not the parser can process large collections of text without crashing or freezing.
7. Sound: It should be ' with respect to a given grammar and lexicon; that is, it should not assign to an input sentence analyses which cannot arise from the grammar in question.
8. Complete: It should also be 'complete'; that it, it should assign to an input sentence all the analyses it can have with respect to the current grammar and lexicon.
9. efficient: Ideally, the algorithm should also be 'efficient', entailing the minimum of computational work consistent with fulfilling the first two requirements,

4.3 Linguistic Grammar Formalisms

4.3.1 CFG Grammar Formalism

The most common kinds of grammars are context free. They are popular because they are mathematically straightforward and well-understood. They are widely used to describe computer programming languages, for example. In natural language processing, they can be used to provide a 'shallow' analysis of a natural language string, where 'shallow' here refers to a purely surface syntactic analysis, as opposed to a 'deep' analysis that would also provide some relation between the string and its meaning. There are, however, some drawbacks to simple CFGs. It can be complicated to treat long-distance dependencies with regular CFGs. However, for most computational purposes, given the wide range of efficient algorithms for CFGs, a CFG is generally accepted as being sufficient to at least get a basic syntactic representation for an input string.

4.3.2 Unification Grammar formalism

Unification grammars extend simple context-free formalisms with the addition of feature structures or attribute value matrices (AVM). Members of the unification grammar family include lexical functional grammar (LFG) (Kaplan and Bresnan 1982), functional unification grammars (FUG) (Kay 1985), PATR-II (Shieber 1984) and head-driven phrase structure grammar (HPSG) (Pollard and Sag 1994). There are a number of advantages and disadvantages to both 'shallow' and 'deep' processing. Deep grammars tend to be precise(because they are often hand-crafted), allow semantic composition and are often reversible (the same grammar can be used for parsing and generation). Some of them have wide-coverage, but are not always as robust as their shallower counterparts. For example, deep grammars will often disallow ungrammatical input, while shallow parsers usually provide a parse for any input given to them (one can argue whether this is a desired feature or not). Shallow parsers are usually fast and statistical disambiguation methods can easily be integrated into them. Shallow parsers cannot usually be used for generation: because they can parse ungrammatical input, they will generate ungrammatical output, too. The output of shallow parsers does not usually contain an analysis of long-distance dependencies that is required for further semantic processing.

4.3.3 PCFG Grammar Formalism

With the amount of data available to researchers nowadays, it is becoming increasingly easier to train statistical parsing models. These models offer some advantages over hand-crafted grammars. They are easy to acquire, achieve wide-coverage over the training domain, are robust, almost always providing some analysis and they can be easily adapted to new domains and data (Armstrong-Warwick 1993). Another advantage is the natural inclusion of lexical distributional information. But, one of the main advantages over hand-crafted grammars is in disambiguation. PCFGs can be smoothed[45].A statistical parser can return a ranked n-best list of possible analyses of a string. A hand-crafted system, while in general might return far fewer analyses, has no way of indicating which ones are more likely. As a result, the area of probabilistic phrase structure parsing has been a central and active field in computational linguistics[46]. Stochastic methods in natural language processing, in general, have become very popular as more and more resources become available. One of the main advantages of probabilistic parsing is in disambiguation: it is useful for a parsing system to return a ranked list of potential syntactic analyses for a string [47]. Probabilistic context-free grammars extend CFGs by associating a probability with each production rule. Formally, a PCFG is defined as a five-tuple ($\sum$, V, S, P and D) where $\sum$ is a finite, non-empty set of terminals (the alphabet); V is a finite, non-empty set of non-terminal symbols (category labels) such that $\sum nV = \emptyset; S \in V$ is the start symbol, ; P is a finite set of production rules, $A -> \alpha$ where, $A \in V$ and $\alpha \in (VU\sum)^\star$; and D is a function assigning a probability to each member of P. Moreover, the productions in PCFG specify the set of terminals and non terminals implicitly. In this way, PCFGs impose the constraint that the set of productions with any given left-hand-side must have probabilities that sum to 1 with very small marginalization error that has no effect.

$$\sum_{A->\alpha \in P} D(\alpha|A) = 1;$$

that is, the probabilities for all RHSs for a given LHSs must sum to 1. A PCFG defines the probability of the tree, P(T), as the product of the probabilities of the token occurrences of the rules expanding each LHS to its RHS in the tree:

$$P(T) = \prod_{i=1}^{n} P(RHS_i|LHS_i) \tag{5}$$

The probability of each tree is calculated by multiplying together the probabilities of each of the rules used to derive the tree. If we want to associate probabilities with each of the CFG rules or productions to get their corresponding PCFG probablstic productions, then we can calculate the probability of the trees. 4.1 and 4.2

Currently, there is no latex package that works for ethiopic scrip. Hence ,the transli-

Table 4.1: Parse Tree with lower probablity

∑	V	S	P	D
ሰላም	NP	S	S -> NP VP	D(S -> NP VP)=1.0
ካብ	VP		NP -> NP PP	D(NP -> NP PP) =0.2
መቐለ	PP		NP -> DET NP	D(NP -> DET NP) =0.2
ንእሽተይ	DET		NP -> N	D(NP -> N) =0.6
መኪና	V		VP -> NP V	D(VP -> NP V) =1.0
ገዚኣ	PREP		PP -> PREP NP	D(PP -> PREP NP)=1.0
	N		N -> 'ሰላም'	D(N -> 'ሰላም') =0.333333
			PREP -> 'ካብ'	D(PREP -> 'ካብ' =1.0
			N -> 'መቐለ'	D(N -> 'መቐለ' =0.333333
			DET -> 'ንእሽተይ'	D(DET -> 'ንእሽተይ' =1.0
			N -> 'መኪና'	D(N -> 'መኪና' =0.333333
			V -> 'ገዚኣ'	D(V -> 'ገዚኣ' =1.0

Table 4.2: Parse Tree with higher probablity

∑	V	S	P	D
ሰላም	NP	S	S -> NP VP	D(S -> NP VP)=1.0
ካብ	VP		NP -> DET NP	D(NP -> DET NP) =0.25
መቐለ	PP		NP -> N	D(NP -> N) =0.75
ንእሽተይ	DET		VP -> NP PP V	D(VP -> NP PP V) =1.0
መኪና	V		PP -> PREP NP	D(PP -> PREP NP)=1.0
ገዚኣ	PREP		N -> 'ሰላም'	D(N -> 'ሰላም')=0.333333
	N		PREP -> 'ካብ'	D(PREP -> 'ካብ')=1.0
			N -> 'መቐለ'	D(N -> 'መቐለ')=0.333333
			DET -> 'ንእሽተይ'	D(DET -> 'ንእሽተይ')=1.0
			N -> 'መኪና'	D(N -> 'መኪና') =0.333333
			V -> 'ገዚኣ'	D(V -> 'ገዚኣ') =1.0

tation helps to convert IPA to ethiopic. As a result, for the sentnce in the table 4.2 the corresponding IPA becomes:

selam kab mekelle nieshtey mekina gezia

The probability of each tree is calculated by multiplying together the probabilities of each of the rules used to derive the tree. This is done by expanding each node on the

node catagory. Hence,

P(T1)= 1.0 x 0.6 x 0.333333 x 1.0 x 0.2 x 0.2 x 1.0 x 0.6 x 0.333333 x 1.0 x 1.0 x 0.6 x 0.333333 x 1.0 =0.0003199990.

P(T2) =1.0 x 0.75 x 0.333333 x 1.0 x0.25 x1.0 x0.75 x 0.333333 x 1.0 x 1.0 x 0.75 x 0.333333 x 1.0 =0.0039062382

Choosing the tree with the highest probability which in this case is P(T2) gives us the one on the right. This tree with highest probability is thus representative of the correct syntactic structure for the given sentence as a desired analysis.

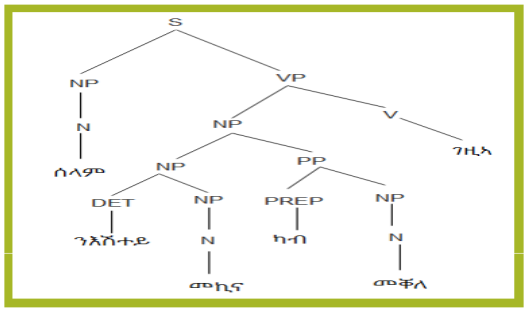


Figure 4.1: S1

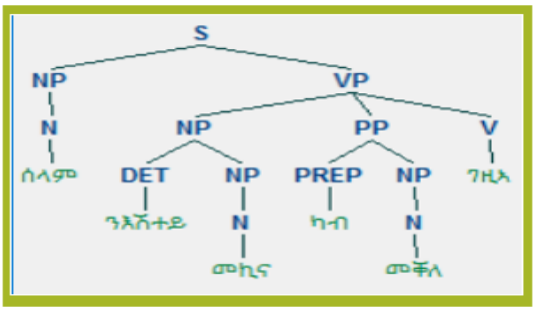


Figure 4.2: S2

4.4 Tigrigna Supervised Parsing Integrated System Architecture

Figure 4.3 shows the over all supervised Tigrigna constituency based PCFG supervised parsing system architecture. This Architecture has two major components, on the right side is the Treebank that contain the grammatical syntax of the language including lexicons and their grammars. On the the left hand side is all the preprocessing steps used as an input for the parser. CFGs are generated from both the left hand side and the right hand side then the CFG repository and PCFG grammar is induced from the CFG repository automatically. The process of inducing PCFG from CFG repository is called Automatic PCFG grammar induction. The PCFG grammar induced in this way is used for the parser a language model. Hence, the next required step is to use a parser with efficient parsing algorithm that could parse any sentences that conform to the PCFG language model, a repository of all syntaxes. The final result is then output syntax parse tree.

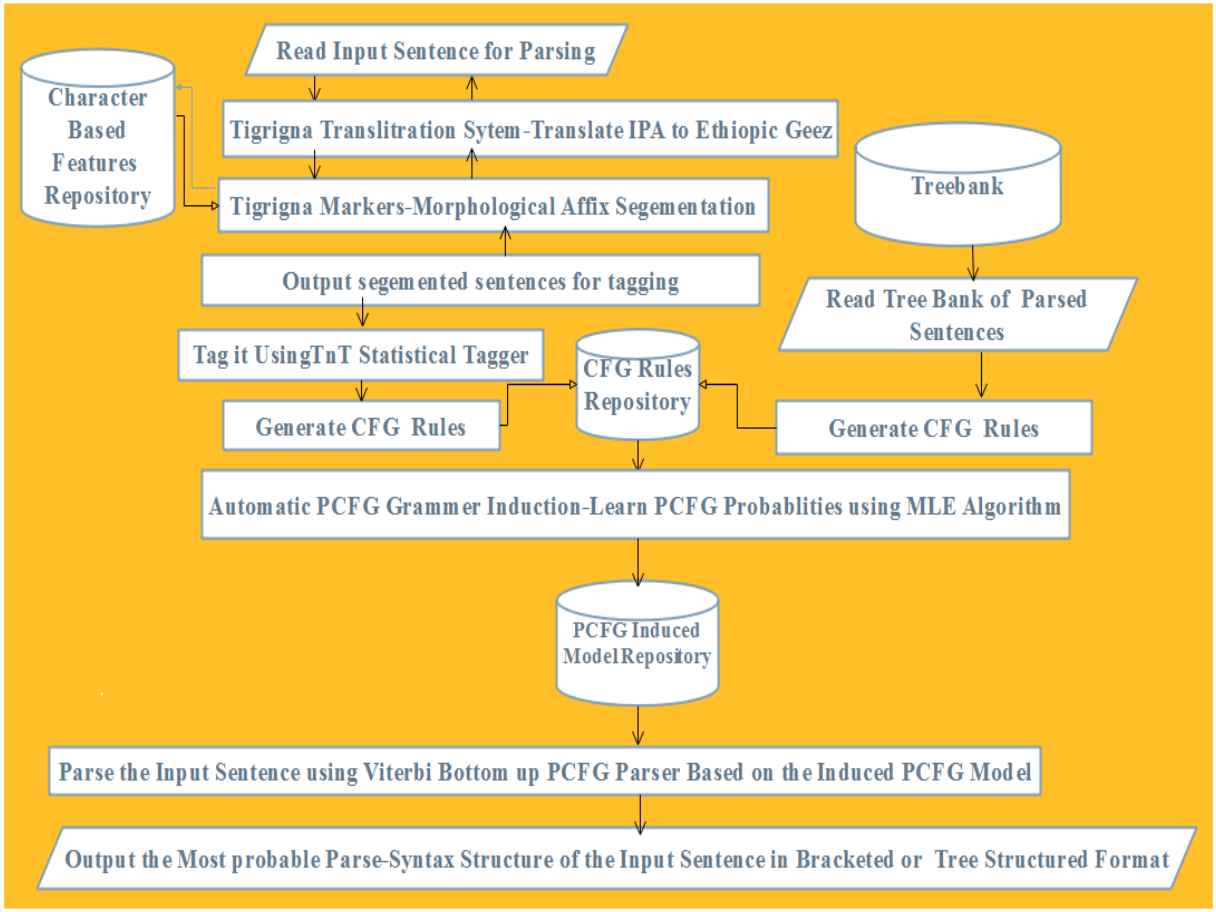


Figure 4.3: Tigrigna Supervised Parsing Integrated System Architecture

4.4.1 Algorithm for Translating IPA to Ethiopic

Ethiopic semitic languages have their own alphabetical representation called Ethiopic script. Hence, representing IPA characters in their coresponding Ethiopic script is found to be crucial especially when we come to syntax parsing and machine translation systems. As a result in this research development of IPA to Ethiopic is also element of the research since it helps to understand the linguistic characteristic level of Tigrigna language wich is written in Ethiopic characters. The algorithm is shown in 4.1. The algorithm is based on segementation of phonetic transcription which means mostly consonants are followed by vowels. In addition to this most of the words in Tigrigna are followed by vowels. Hence the algorithm is based on consonant-vowel pattern system.

Algorithm 4.1: Translate IPA to Ethiopic Script

```
Store IPA letter as Key and their Ethiopic letter as Value in dictionary, mydict:
Then accept input sentence in IPA from the user
For each letter in the input sentence:
    IF letter begins by consonant:
        Insert − segement marker before the consonant until vowel is found
Store the word formed after segement marker is inserted in the variable, text
Split each word in text based on the inserted segement
For each word in splitted text:
    IF word matches with IPA keys in mydict:
        Replace the word with its Ethiopic Script Value
        Insert single space between the segement − marker and the word
        Repeat these steps until the end of the word
        Then store the words in the variable item
        For each word in item:
        IF word matches to the segement marker − :
            Remove the Segement marker − :
Join the sentence and store them in the variable sent
Output the sentence in Ethiopic script which is stored in the variable sent to be an
input for Tigrigna segementer
```

4.4.2 Algorithm for Tigrigna Words and Morphological Affixes Segementer

As discussed in the literature review, Segmentation have higher role in reducing complex words into their base words and complex combination of words with morphological affixes into their base words and morphem levels componets[27]. Hence, in this research an algorithm is developed for segmenting combination of Tigrigna words as in table 4.3 into their components, it splits complex Tigrigna morphems in to their their basic words component morphemes.

The algorithm for word and morphological affix segementation is shown in 4.2

Algorithm 4.2: Tigrigna Words and Morphological Affixes Segementer

Store special character markers in Repository, Rep
Then assign the sentences from the out put of the translitrator to the variable Segementit
For each word in Segementit:
    This is step one, it is used for all cases in other conditions
    IF word starts and ends with same compound word markers or affixes:
        Find the length of the word
        IF length of the word is > 3:
            IF the special character on the the word founds in dependent Rep markers:
                Then don't split this word
            ELSE IF the special character founds in the independent Rep markers:
                Then split this word based on the Tigrigna compound markers or affixes
            ELSE:
                Output the word itself
        ELSEIF length of the word is $\leq$ 3
            Repeat the step in step one
        ELSE:
            Output the word itself
    ELSE IF word starts but not ends with similar word markers or affixes :
        Repeat step one by looking the length and position of the markers
    ELSE IF word not starts but ends with similar word markers or affixes :
        Repeat step one by looking the length and position of the markers
    ELSE IF word starts and ends with different compound word markers or affixes :
        Repeat the step one by looking the length and position of the markers
    ELSE IF word starts but not ends with different word markers or affixes :
        Repeat step one by looking the length and position of the markers
    ELSE IF word not starts but ends with different word markers or affixes :
        Repeat step one by looking the length and position of the markers
    ELSE:
        Output the word itself
Output the segmented Ethiopic sentence to be tagged by the TnT tagger

Table 4.3: Tigrigna words and Affixes segementation

No	Sample words before segmentation	Words formed after segmentation
1.	ብትምህርቶም	ብ + ትምህርቶም
2.	ንወንጀለኛታት	ን + ወንጀለኛታት
3.	ዝገዛእክዎ	ዝ + ገዛእክዎ
4.	እናኸድኩ	እና + ኸድኩ
5.	እንተዘይትሕግዘኒ	እንተዘይ + ትሕግዘኒ
6.	ኣስቴርን ሪሓናን	ኣስቴር + ን ሪሓና + ን
7.	እንትኣትው	እንት + ኣትው
8.	ዘዕረፈትሉ	ዝ + ኣዕረፈትሉ
9.	ስለዝተናደደ	ስለዝ + ተናደደ
10.	ኣይተሓፀበን	ኣይ + ተሓፀበ +ን
11.	ኣይትመፅእን	ኣይ + ትመፅእ +ን
12.	ኣይትትሓፀብ	ኣይት + ትሓፀብ
13.	እናንፀርፀረት	እና + ኣንፀርፀረት
14.	ምስኣባጊው	ምስ + ኣባጊው
15.	እንድሕርሂብክኒ	እንድሕር + ሂብክኒ
16.	እንተተፀንዕ	እንተ + ተፀንዕ
17.	ከምዝተሸለመ	ከምዝ + ተሸለመ
18.	ከምመፀ	ከም + መፀ
19.	ንራሄልን	ን + ራሄል + ን
20.	ብራሄልን	ብ + ራሄል +ን

4.4.3 Algorithm for Integrating Tagging language model with parsing language model-generating CFG

Language models provide a better way for sequence prediction purpose for many NLP systems including Tagging, parsing, speech recognition as well as machine translation. As a result, in this research both language models for tagging and parsing are used. In order to tagging language model to be used for parsing as an intermediate input, it needs a common interface between the two. One way of integrating the language model generated from tagging to parsing is creation of context free grammars from

both. Context free grammars have higher roles for their corresponding probablistic context free grammars. They are used as bench mark from which probabilities can be calculated as in PCFG parsing language model. Hence, in this research an algorithm is developed to gather to ingerate the tagger and the parser, which is preferably through the extraction of context free grammar from both and storing it in repositery from which the parsing language model is the induced. This is because parsing language model is differed from Tagging language model in that , Tagging language model only provides word-tag lexical and tag to tag transition probabilities, while parsing language includes the high level prediction of syntax structures together with the lexicons. That is why corpus based treebank preparation typically of the form Penn Treebank style is needed in statistical machine learning.

Algorithm 4.3: Integration of language models

Assign empty list to the variable myproductions, myproductions=[]
Assign NLTK Non terminal to start symbol S, S=NLTK.Nonterminal('S')
Open the file contaning parsed sentence as f
Read the contents and assign them to variable content, content=f.readlines()
For content in f:
    Extract the trees from content using, trees=Tree.fromstring(content)
    Generate the productions from trees using, myproductions=trees.productions()
Open the file containing tagged sentences by TnT Tagger, ff
Assign contents of ff to the variable tagged using, tagged=ff.readlines()
For tagged in ff:
    Store CFG in result using, result.append(tagged.split()[0]+ −> +tagged.split()[1])

Open file cfg for writing to store generated result as cfg
For item in result:
    Write items to cfg using, cfg.write(item) as string
Open the file named cfg to which CFG grammars are written as ff2:
Store the contents of ff2 in a variable called line using line=ff.readlines()
Create empty string st to store the contents of cfg, st=" "
For line in ff2:
    st=st+line
Generate the formatted cfg and store it in g using, g=CFG.fromstring(st)
Generate the productions from the grammar and store it in P, P=g.productions()
Now integrate the CFG generated from TnT tagger and and the CFG generated from Parsed Treebank using,p=p+myproductions

4.4.4 Algorithm for Learning or Inducing PCFGs from CFG Repository automatically-Maximum Likelihood Estimation Algorithm

Machine learning method could be supervised, unsupervised and semi-supervised machine learning[48–50]. Learning with Probablistic models leads to effective natural language parsing [51]. Given annotated tree bank of parsed sentences which contain list of rules together with their lexicons, in this case the 250 syntactically annotated parsed corpora, the technique used for the automatic extraction of stochastic grammars from such tree bank (i.e, collections of hand-corrected syntactic structures) is the maximum likelihood estimation of their count conditional probabilities. Thus, rather than assigning the grammars manually as used by previous researchers [32, 33] for Amharic parser, this research is intended to automatically extract or induce a PCFG from a fully parsed tree bank. The trees on the tree bank contain nodes where by the expansion of each of the node depends only on the node category. This is called a Markov assumption (where the probability of an event depends only on the n previous events) and is commonly referred to as an independence assumption. One of the main advantages of a probabilistic CFG over a regular one is thus, now we can associate each parse of a string with a probability, allowing us to disambiguate to a large extent as shown in tables 4.1 and 4.2. Probabilistic CFGs are also used as language models(Charniak 2001; Roark 2001; Xu et al. 2002) assigning probabilities to strings of words (e.g. in speech recognition). The probability associated with each rule is determined by relative frequency; that is, by counting the number of times a rule occurs in a corpus and dividing it by the number of occurrences of all rules expanding the same LHS. Formally, this is expressed as:

$$P(LHS->RHS_j) = \frac{\#(LHS->RHS_j)}{\sum_{i=1}^{n}\#(LHS->RHS_j)} \tag{6}$$

the probabilities for all RHSs of a given LHS will sum to 1. For example, if a given corpus has three VP rules (VP -> V, VP -> V NP and VP -> V NP NP), the sum of all the probabilities of the VP rules must be 1 with small marginalization error that has no effect. Thus, PCFGs consists of a start state and a set of productions with probabilities. In particular, the probability of a Probabilistic Production essentialy records the maximum likelihood that its right-hand side is the correct instantiation for any given occurrence of its left-hand side. In such a way, the associated probabilities represent how likely it is that these productions will be used. The algorithmic function used to train treebank of parsed corpora or for inducing(extracting) grammars from

treebank automatically is shown in 4.4 For instance, the CFG production rules for

Algorithm 4.4: Algorithm for Automatic PCFG Grammar Induction

```
For each tree in the treebank:
    Get the CFG rules from the tree
    For each such CFG rule R:
      Update the frequency of R
      Update the frequency of LHS(R)
For each tree in the tree bank:
    Let freq(LHS(R)) be the frequency of the LHS of R
    Let prob = freq(R)/freq(LHS(R))
Output: R prob
```

the tree in figure 4.2 is shown in 4.4 together with their corresponding Maximum likelihood rule probability calculation. In this way according to the algorithm 4.4, the PCFG grammar is induced dynamically to generate the likelihood probabilities for all lexical and rule probabilities in contrast to the previous researchers on Amharic syntax parsing, Daniel and Atelach, for which they have assigned the probabilities by their hands which is static and not dynamic.

Extract CFG Rules From Treebank(tree)	CFG Rules Count	Rule probabilities= CFG Rules Count/Frequencies= Maximum Likelihood Estimation(MLE)	Frequencies (Total Count of the occurrence Non-Terminals as left hand side)
S→NP VP	1	1/1=1.0	S=1
NP→DET	1	1/2=0.5 or 0.75	NP=2
NP→N	1	1/2=0.5 or 0.25	
VP→NP PP V	1	1/1=1.0	VP=1
PP→PREP NP	1	1/1=1.0	PP=1
PREP→ከብ	1	1/1=1.0	PREP=1
N→መቐለ	1	1/3=0.33333333	N=3
N→መኪና	1	1/3=0.33333333	
N→ሰላም	1	1/3=0.33333333	
DET→ንእሽተይ	1	1/1=1.0	DET=1
V->ገዚኣ	1	1/1=1.0	V=1

Figure 4.4: Automatic Grammar Induction with MLE

4.5 Smoothing PCFGs

PCFGs induced from the Penn Treebank have many productions with long sequences of non-terminals on the RHS. Probability estimates of the RHS given the LHS are often smoothed by making a Markov assumption regarding the conditional independence of a category on those more than k categories away(Collins, 1997; Charniak, 2000).

$$\begin{aligned} P(X->Y_1\cdots Y_n) &= P(Y_1|X)\prod_{i=2}^{n} P(Y_i|X,Y_1\cdots Y_{i-1}) \\ &= P(Y_1|X)\prod_{i=2}^{n} P(Y_i|X,Y_{i-k}\cdots Y_{i-1}) \end{aligned} \tag{7}$$

Making such a Markov assumption is closely related to grammar transformations required for certain efficient parsing algorithms. Binarized PCFGs are induced from a treebank whose trees have been factored so that n −ary productions with n >2 become sequences of n −1 binary productions.

4.6 Viterbi Parsing Algorithm over a PCFG Parse packed Forest-Bottom up Viterbi PCFG Parser Module

PCFG Bottum up Parser tool is used in this research as it produces a packed edge, which could represent multiple tree. when the Viterbi algorithm is applied over the packed forest represented by an edge (which can contain packed alternatives), only a single a new edge representing the most likely tree is returned. The chart is made up of 'cells' that store the intermediate analyses. Viterbi algorithms are an instance of dynamic programming, and a parsing algorithm that first fills the cells with 'items', a partial analysis (or derivation), where in each cell, all items cover the same part of the input string. In probabilistic chart parsing, when two items with the same LHS are added to the same cell, the algorithm can always choose the item with the highest probability and disregard the lower probability item. This is because in any complete analysis involving the items, the lower probability item will always lead to a lower probability overall derivation. These algorithms are called Viterbi algorithms. The psudeocode of the algotithm for Viterbi Parser is shown in Algorithm 4.6. Viterbi Algorithm uses dynamic programming algorithm to find the single most likely parse for a text. Viterbi algorithm parses texts by filling in a most likely constituent table(MLC) which records the most probable tree representation for any given span and node value. In particular, it has an entry for every start index, end index, and node value, recording the most likely sub tree that spans from the start index to the end index, and has the given node

Algorithm 4.5: Algorithm for Viterbi Parser

```
Create an empty most likely constituent table, MLC.
For width in 1 ··· len(text):
  For start in 1 ··· len(text) − width:
    For prod in grammar.productions:
      For each sequence of subtrees [t[1], t[2], ... , t[n]] in MLC,
        where t[i].label() == prod.rhs[i],
        and the sequence covers [start : start+width]:
          old_p = MLC[start, start +width, prod.lhs]
          new_p = P(t[1])P(t[1])...P(t[n])P(prod)
          IF new_p > old_p:
            new_tree = Tree(prod.lhs, t[1], t[2], ... , t[n])
            MLC[start, start +width, prod.lhs] = new_tree
Return MLC[0, len(text), start_symbol]
```

value. The parser then fills in this table incrementally. It starts by filling in all entries for constituents that span one element of text(i.e., entries where the end index is one greater than the start index). After it has filled in all table entries for constituents that span one element of text, it fills in the entries for constituents that span two elements of text. It then continues filling in the entries for constituents spanning larger and larger portions of the text, until the entire table has been filled. Finally, it returns the table entry for a constituent spanning the entire text, whose node value is the grammar's start symbol. In order to find the most likely constituent with a given span and node value, the parser considers all productions that could produce that node value. For each production, it finds all children that collectively cover the span and have the node values specified by the production's right hand side. If the probability of the tree formed by applying the production to the children is greater than the probability of the current entry in the table, then the table is updated with this new tree.

4.7 The TnT Tagger Module

TnT which stands for Trigrams'n'Tags,is a very efficient and statistical part-of-speech tagger that is trainable on different languages and virtually any tagset. The component for parameter generation trains on tagged corpora. The system incorporates several methods of smoothing and of handling unknown words. TnT is optimized for speed. The Tagger is an implementation of the Viterbi Algorithm for second order Markove Models. The main paradigm used for smoothing is linear interpolation, the respective weights are determined by deleted interpolation. Unknown words are handled by a suffix trie and successive abstraction. It maintains a number of internal freqDist and

ConditionalFreqDist instances based on the training data. These frequency distributions count unigram, bigram and trigram. Then, during tagging, the frequencies are used to calculate the probablities of possible tags for each word. So, instead of constructing backof chain of NgramTagger subclasses, the TnT tagger uses all the n-gram models together to choose the best tag. It also tries to guess the tags for the whole sentence at once by choosing the most likely model for the entire sentences, based on the probablities of each possible tag. It uses second order Markov models for part of speech tagging. The states of the model represent tags, outputs represent the words. Transition probabilities depend on the states, thus pairs of tags. Output probabilities only depend on the most recent category. For a given sequence of words w_1 ... w_T of length T and given elements of the tagset t_1 ... t_T with the additional tags t_{-1}, t_0, and t_{T+1}, which are beginning of sequence and end of sequence markers ,the second order Markove model for TnT tagger is calculated as:

$$\underset{t_1 \cdots t_T}{\arg\max} \left[\prod_{i=1}^{T} P(t_i|t_{i-1}, t_{i-2}) P(w_i|t_i) \right] P(t_{T+1}|t_T) \tag{8}$$

The transition and output probabilities are estimated from a tagged corpu uing maximum likelihood probabilities $\hat{\mathrm{P}}$ derived from the relative frequencies of uni-gram,bigram and tri-gram as follows:

$$\text{Unigrams:} \quad \hat{P}(t_3) = \frac{f(t_3)}{N} \tag{9}$$

$$\text{Bigrams:} \quad \hat{P}(t_3|t_2) = \frac{f(t_2, t_3)}{f(t_2)} \tag{10}$$

$$\text{Trigrams:} \quad \hat{P}(t_3|t_1, t_2) = \frac{f(t_1, t_2, t_3)}{f(t_1, t_2)} \tag{11}$$

$$\text{Lexical:} \quad \hat{P}(w_3|t_3) = \frac{f(w_3, t_3)}{f(t_3)} \tag{12}$$

for all t_1, t_2, t_3 in the tagset and w_3 in the lexicon. N is the total number of tokens in the training corpus. The result of the experiment can be shown in figure5.3

We define a maximum likelihood probability to be zero if the corresponding nominators and denominators are zero. As a second step, contextual frequencies are smoothed and lexical frequences are completed by handling words that are not in the lexicon.

4.7.1 Smoothing Techniques

Trigram probabilities generated from a corpus usually cannot directly be used because of the sparsedata problem. This means that there are not enough instances for each

trigram to reliably estimate the probability. Furthermore, setting a probability to zero because the corresponding trigram never occurred in the corpus has an undesired effect. It causes the probability of a complete sequence to be set to zero if its use is necessary for a new text sequence,thus makes it impossible to rank different sequences containing a zero probability. The smoothing paradigm that delivers the best results in TnT is linear interpolation. Therefore,the trigram probability is estimated as follows:

$$P(t_3|t_1,t_2) = \lambda_1\hat{P}(t_3)+\lambda_1\hat{P}(t_3|t_2)+\lambda_1\hat{P}(t_3|t_1,t_2) \tag{13}$$

Where $\hat{P}$ are maximum likelihood estimates of the probabilities, and $\lambda_1+\lambda_2+\lambda_3 = 1$, so P again represent probability distributions. We use the context independent variant of linear interpolation, i.e., the values of the λs do not depend on the particular trigram. The values of $\lambda_1,\ \lambda_2\ and\ \lambda_3$ are estimated by deleted interpolation. This technique successively removes each trigram from the training corpus and estimates best values for the λs from all other n-grams in the corpus. Given the frequency counts for uni-, bi-, and trigrams, the weights can be very efficiently determined with a processing time linear in the number of different trigrams. The algorithm for deleted interpolation is shown in

Algorithm 4.6: Algorithm for calculating the weights for context-independent linear interpolation $\lambda_1,\ \lambda_2\ and\ \lambda_3$ when the n-gram frequencies are known. N is the size of the corpus. If the denominator in one of the expressions is 0, we define the result of that expression to be 0.

set $\lambda_1 = \lambda_2 = \lambda_3 = 0$
foreach trigram $t_1,\ t_2,\ t_3\ with\ f(t_1,t_2,t_3) > 0$
  depending on the maximum of the following three values:
    case $\frac{f(t_1,t_2,t_3)-1}{f(t_1,t_2)-1}$: $increament\ \lambda_3\ by\ f(t_1,t_2,t_3)$
    case $\frac{f(t_2,t_3)-1}{f(t_2)-1}$: $increament\ \lambda_2\ by\ f(t_1,t_2,t_3)$
    case $\frac{f(t_3)-1}{N-1}$: $increament\ \lambda_1\ by\ f(t_1,t_2,t_1)$
  end
end

4.7.2 Unknown Words Handling

Currently, the method of handling unknown words that seems to work best for in inflected languages is a suffix analysis as proposed in (Samuelsson,1993). Tag probabilities are set according to the word's ending. The suffix is a strong predictor for word classes. The probability distribution for a particular suffix is generated from all words in the training set that share the same suffix of some predefined maximum

length. The term suffix as used here means final sequence of characters of a word" which is not necessarily a linguistically meaningful suffix. Probabilities are smoothed by successive abstraction. This calculates the probability of a tag t given the last m letters l_i of an n letter word: $P(t|l_{n-m+1},...l_n)$. The sequence of increasingly more general contexts omits more and more characters of the suffix, such that $P(t|l_{n-m+2},...n)$, $P(t|l_{n-m+3},...l_n),...P(t)$ are used for smoothing. One has to identify a good value for m, the longest suffix used. The approach taken for TnT is the following: m depends on the word in question. We use the longest suffix that we can find in the training set(i.e., for which the frequency is greater than or equal to 1), but at most 10 characters. This is an empirically determined choice. The recursion formula is

$$P(t|l_{n-i+1},...l_n) = \frac{\hat{P}(t|l_{n-i+1},...l_n)+\theta_i P(t|l_{n-i},...,l_n)}{1+\theta_i} \tag{14}$$

for i = m...0, using the maximum likelihood estimates $\hat{P}$ from frequencies in the lexicon, weights θi and the initialization

$$P(t) = \hat{P}(t) \tag{15}$$

The maximum likely hood estimate for a suffix of length i is derived from corpus frequencies by

$$\hat{P}(t|l_{n-i+1},...l_n) = \frac{f(t,l_{n-i+1},...l_n)}{f(l_{n-i+1},...l_n)} \tag{16}$$

a context-independent approach for θ_i, as we did for the contextual weights λ_i. It turned out to be a good choice to set all θ_i to the standard deviation of the unconditioned maximum likelihood probabilities of the tags in the training corpus, i.e., we set

$$\theta_i = \frac{1}{s-1}\sum_{j=1}^{s}(\hat{P}(t_j)-\bar{P})^2 \tag{17}$$

for all i = 0 ... m-1, using a tagset of s tags and the average

$$\bar{P} = \frac{1}{s}\sum_{j=1}^{s}\hat{P}(t_j) \tag{18}$$

4.8 Algorithm for Integrating TnT Tagger with the Parser Model

Algorithm 4.7: Algorithm for Integrating TnT Tagger with the Parser Model

```
Create new list for holding all the productions,new_p
Read all CFG productions from the treebank corpus,productions
Generate CFG Productions from the TnT tagger,prod
new_p = []
new_p = new_p.append(prod)
new_p + = productions
root = 'S'
Automatic_Grammar_Induction = induce_pcfg(Nonterminal(root), new_p)
```

4.9 *Summery*

In this chapter the over all supervised integrated parsing system architecture is discussed. The chapter begins by discussing important features of good parser as resource needy, state of the art performance, self -tagging, robustness, long sentences, stablity, sound, complete, efficient. Then it followed by the PCFG parsing grammar formalism which is used later as parsing language model. Examples of how PCFG conditional probablities are generated and the tree with highest probablity has correct syntax tree is explaind in this chapter. Nextly, Tigrigna supervised system architecture with each of it major modules is discussed. IPA to Ethiopic script tool is added to help for those users that know IPA but need to understand Ethiopic script. Word and Morphological affix segementer is added to the TnT tagger to assist it at handling complex compound words and complex morphological affixes before its tagging process. Both the IPA to Ehiopic script and word and morpholgical segementer are used on all training and testing sets successfully without any errors. How the segementer segements words and morphological affixes that TnTn tagger couldn't handle is given with tabular example in this chapter 4.3. Maximum likelihood estimation technique is used to induce PCFG model automatically from CFG repository. The table of calculation on how MLE works is also given with example in this chapter 4.4. The viterbi PCFG bottom up NLTK parser tool is used in this research and on how it works its algorithm is discussed in this chapter. TnT tagger is also integrated into the parser and the smoothing techniques and unknown words handling is discussed in this chapter. The experimental evaluation and discussion for both TnT tagger and parser is done on chapter 5. To an end an algorithm is developed to integrate the tagging language model with the parsing treebank. In other words an algorithm is developed to integrate the out put of the tagger with the parsing treebank. This is done by so called CFG Integration. The reason is that PCFG is the extension of CFG adored with probablities hence to assign probablity to the out put of the tagger and to the out put of parsed tree bank, it is necessary to find out the common frame work that make them the parsed tree bank and the output of tagger. It is just we can generate CFG from both and collect it in CFG repository and we can the Induce the corrposponding PCFG only from the CFG repository and thus used as PCFG language model for parsing. Efficient algorithm called Viterbi algorithm is the used to parse sentences based on this PCFG model.

CHAPTER FIVE
EXPERIMENTAL EVALUATION AND DISCUSSION

5.1 Introdution

This chapter is all about the experimental evaluation discussion and results conducted for the TnT tagger and Parser integrated together with three percentage splits for training and testing. The percentage splits are 60−40, 75−25 and 80−20%.

5.2 TnT Tagger

TnT tagger uses the following supervised learning architecture in its process of learning from tagged corpus and then tagging new file based on model parameters generated from the training corpus. The architecture for the tnt parameter generation and tagging process is shown in figure 5.1 below.

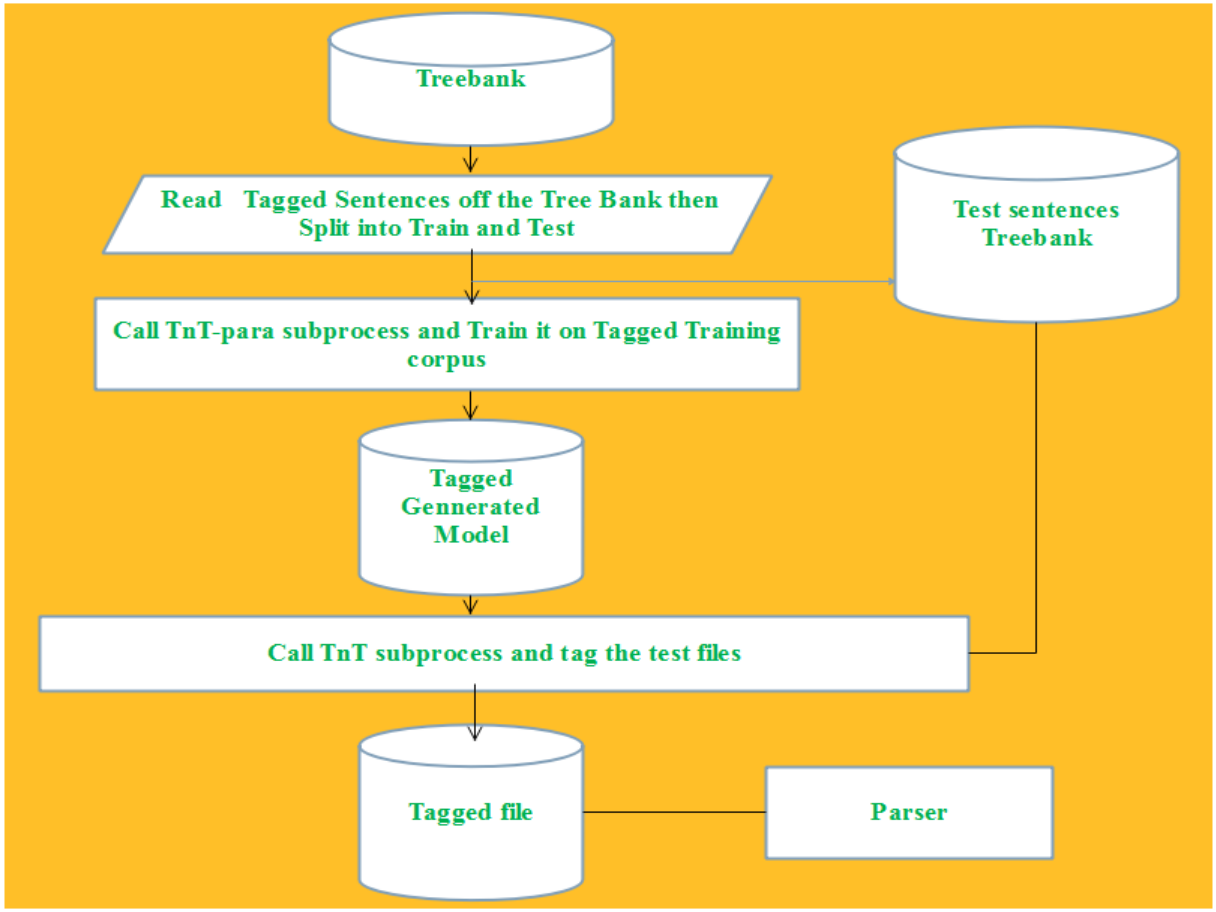


Figure 5.1: TnT tagger Supervised System Architecture

5.2.1 TnT Model Parameter Generation-Training Module

The parameter generation requires a tagged training corpus having extension (.tt). Generation of lexical and contextual model parameters is done as shown in figure 5.2.

Tnt Parameter Generation Process		
Generate Parameters using	Lexical and Contextual Frequencies Created	
TnT-para <corpus-file>	Lexical	Contextual
tnt-para tigrigna_training_corpus.tt	tigrigna_training_corpus. lex	tigrigna_training_corpus. 123

Figure 5.2: TnT Parameter generation-lexical and contextual frequiencies

Two files containing lexical and contextual frequencies are generated as shown in figure 5.2. When tnt is trained, then total of N tokens , a number of lexicons are generated. In addition to this n-gram is created corpus as shown in figure 5.3

Total lexicons and words		Relative frequencies of unigram, bi-gram and tri-gram		
Lexicons	words	Uni-gram	Bi-gram	Tri-gram
568	1306	14	75	221

Figure 5.3: N-grams created during TnT Parametter generation process

5.2.2 Tagging using TnT

The tagging process requires a new test corpus to be tagged with an extension(.tts). Generation of tagged corpus is done from the mdel created during TnT model parameter generation as shown in 5.4 where model is the name of the language model to be used. In this case, the tagger looks for the models model.lex and model.123 which are previously generated by tnt-para. In this way, Tigrigna test sentences are tagged as

TnT uses	Tagging New corpus tigrigna_test_corpus.tt using TnT
tnt model tigrigna_test_corpus.tts	tnt tigrigna_training_corpus tigrigna_test_corpus.tt >tigrigna_test_corpus.tts
	Tagged out put= tigrigna_test_corpus.tts

Figure 5.4: Tnt Tagger Tagging Process

shown in figure 5.4. As a result the output is written to standard output which in this case is redirected to tigrigna_test_corpus.tts.

This uses the tagger with tri-gram mode; weightes are calculated by using deleted interpolation; unknown words are handled with a suffix trie. For smoothing linear inter-

polation with fixed weightes $\lambda_1 = 0.1, \lambda_2 = 0.2, \lambda_1 = 0.7$ could be used. These pre-processing steps are applied on the tagged file to tag the test file successfully. The out put of the tagged sentences is then send as an intermediate input to the syntax analyzer or parser.

5.3 Standard Parser Evaluation Technique and its Architecture for Tigrigna Statistical Constituency PCFG parser

PARSEVAL measures are the standard way of evaluating parser quality(Black et al. 1991). Hence, the output of PCFG parsing are usually evaluated in terms of PARSEVAL metrics. Evalb is an automatic parser evaluation scoring program or tool which compares output system parsed trees with the Gold Standard trees and computes PARSEVAL measures in terms of three PARSEVAL measures called precision, recall and F-Measure automatically [52]. The architecture for parser evaluation using Evalb is shown in figure 5.5.

In order to evaluate the parser, successive experimental evaluation is made with 60−40,

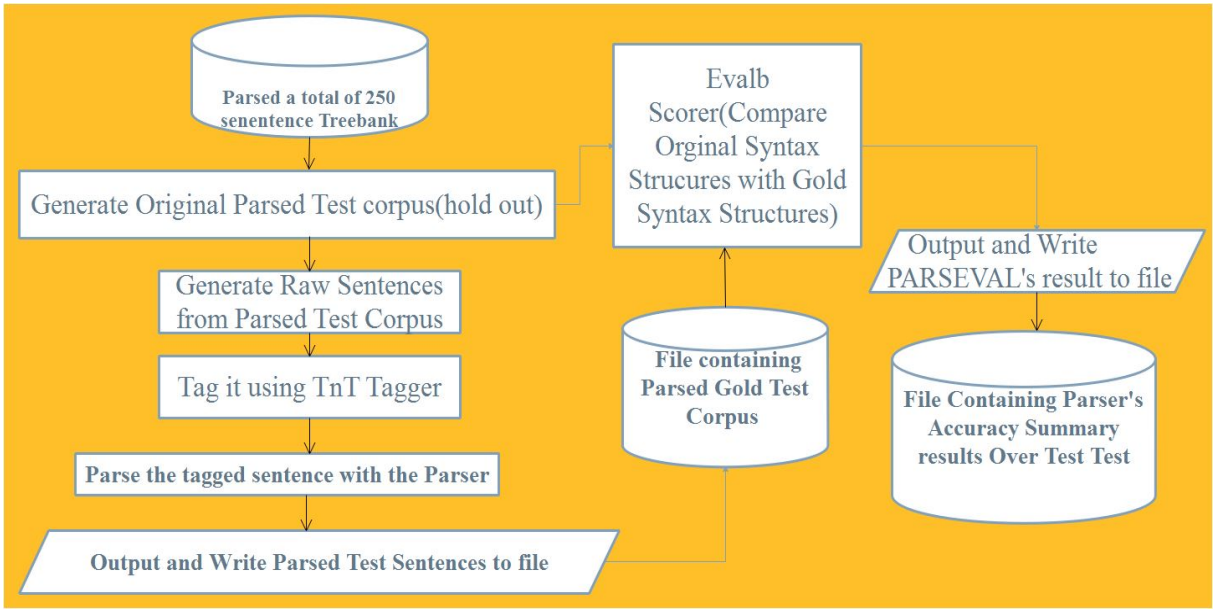


Figure 5.5: Architecture for Parser Evaluation

75−25 and 80−20 percentage splits. The TnT tagger is then integrated as an intermediate input as a tagger with Viterbi based PCFG parser for both the training and testing sets. Then after, test data sets are taken for each splits as test sets, then the original parsed test data sets correctly parsed by linguistic professional are hold out (kept separated) from the training set and from the test set) for latter comparison by evalb scorer. Then, only the raw sentences from these original hold out(kept separated) syntactically parsed test sentences are taken as an input to be parsed by the parser. After the TnT

tagger is trained on each splits and the raw sentences of the test data sets are successfuly tagged by the TnT tagger, the tagger, the Tagger is then integrated into the parser to become an intermediate input for lexical ambiguity resolution and based on each split the parser has parsed each test data sets on each split successfully. However, corpus based statistical machine learning requires large amount of data for learning purposes. As a result, as we can see from figure 5.6, the performance of the parser increases as the amount of data for machine learning increases .

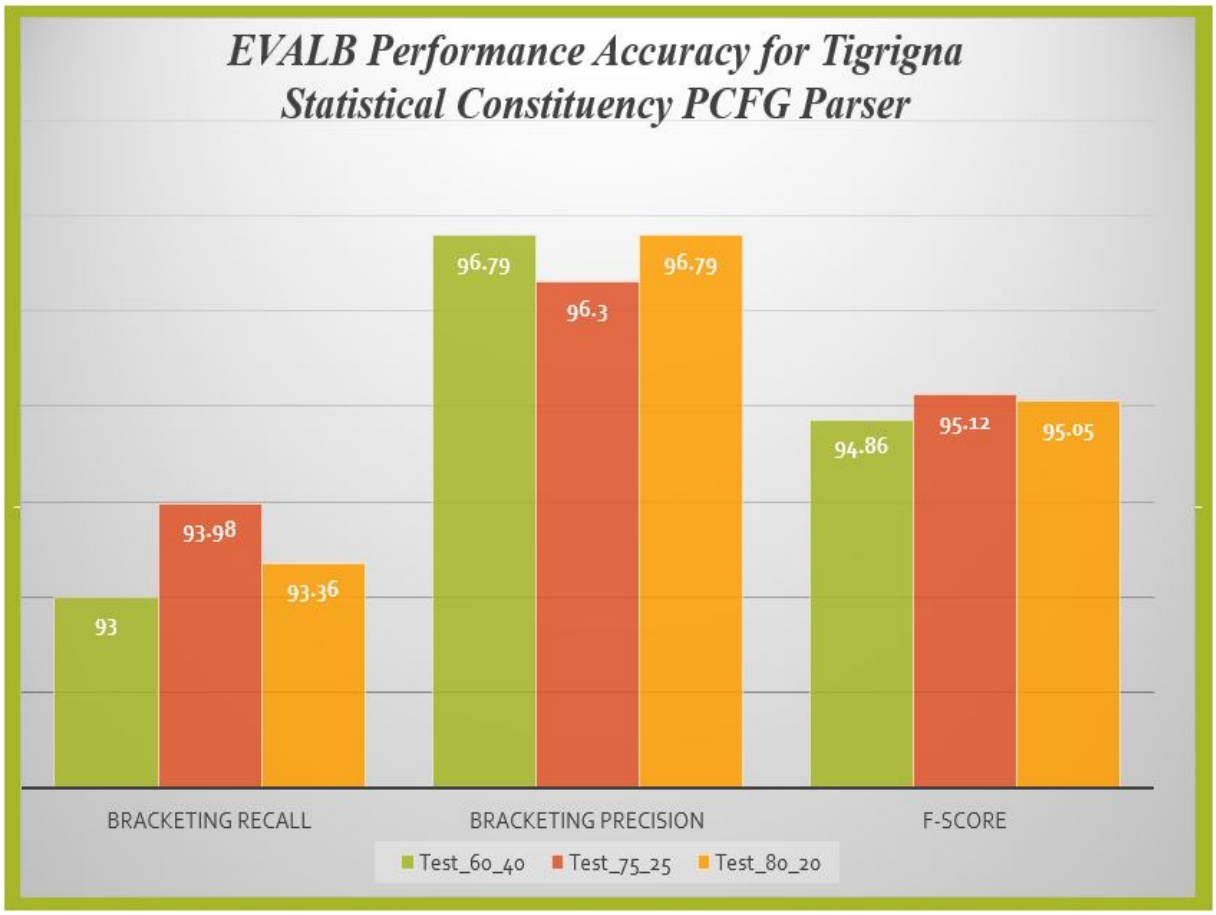


Figure 5.6: Tigrigna Statistical Constituency PCFG Parser Performance Accuracies

Consequently, when the parser is experimented with each of the three percentage splits, the parser has achieved very high performance with the 75 −25 and 80−20 percentage splits. It achieved the highest performance with 75 −25 percentage split. Taking 80−20 as an instance of the experiment, When the parser is evlauated with 80−20 percentage split, the raw sentences of the 80−20 test data sets are given to the TnT tagger and then the TnT tagger tagged them and It is then integrated with the parser and the parser has successfully parsed the input test data sets tagged by the tagger. In this case, out of the

total 250 data sets, 48 of them ares taken as test sets of data. Out of the 48 sentences, 34 of the sentences are parsed exactly and 14 of them are incorrectly parsed as shown in figure 5.9. The corresponding Precision and recall evaluation measures are calculated as as follows for the system parse trees shown in figure5.7.
The three PARSEVAL measures for parser evaluation are: Bracketing Recall: The percentage or ratio of non-terminal bracketing or the correct number of constituents in the systems' parse tree to all constituents appeared in the gold standard parse tree.

$$BR = \frac{\#\ Correct\ \ Constituents}{\#\ All\ \ Constituents\ \ in\ \ the\ \ Gold\ \ Standard\ \ Parse\ \ Tree} \tag{19}$$

Bracketing Precision: The ratio of non-terminal bracketing or the correct number of constituents in the systems' parse tree to the total number of constituents appeared in the system parse tree. A constituent is considered to be correct if it matches a constituent in the Gold standard parse tree.

$$BP = \frac{\#\ Correct\ \ Constituents}{\#\ All\ \ Constituents\ \ in\ \ the\ \ System\ \ Parse\ \ Tree} \tag{20}$$

F-score (or also f-measure): is another common measure for the accuracy of a parser which is calculated as the harmonic mean of precision and recall:

$$F\text{-}score = \frac{2 \times Precision \times Recall}{Precision{+}Recall} \tag{21}$$

$$BR = \frac{\#\ Correct\ \ Constituents}{\#\ All\ \ Constituents\ \ in\ \ the\ \ Gold\ \ Standard\ \ Parse\ \ Tree} = \frac{6}{7} = 0.857 = 0.86$$

$$BP = \frac{\#\ Correct\ \ Constituents}{\#\ All\ \ Constituents\ \ in\ \ the\ \ System\ \ Parse\ \ Tree} = \frac{6}{8} = 0.75$$

$$F\text{-}score = \frac{2 \times Precision \times Recall}{Precision{+}Recall} = \frac{2 \times 0.75 \times 0.86}{0.75 + 0.86} = \frac{1.29}{1.61} = 0.80$$

Generally, the performance of the parser is evaluated based how well the original hold out gold standard test sentences annotated by linguistic professionals matches with the system outputs after the raw test data for each percentage split of 60−40, 75−25 and 80−20. The raw tes data are tagged by TnT tagger and Parsed by the parser. The evaluation is finally made by the standard PARSEVAL scoring tool called Evalb scorer or Evaluate bracketing score. The corresponding gold standard parse trees and system parse tree in figure 5.7 are taken from the correctly annotated gold standard test data set and incorrectly parsed tree by the system shown in id 0 of figure 5.8 and figure 5.9.

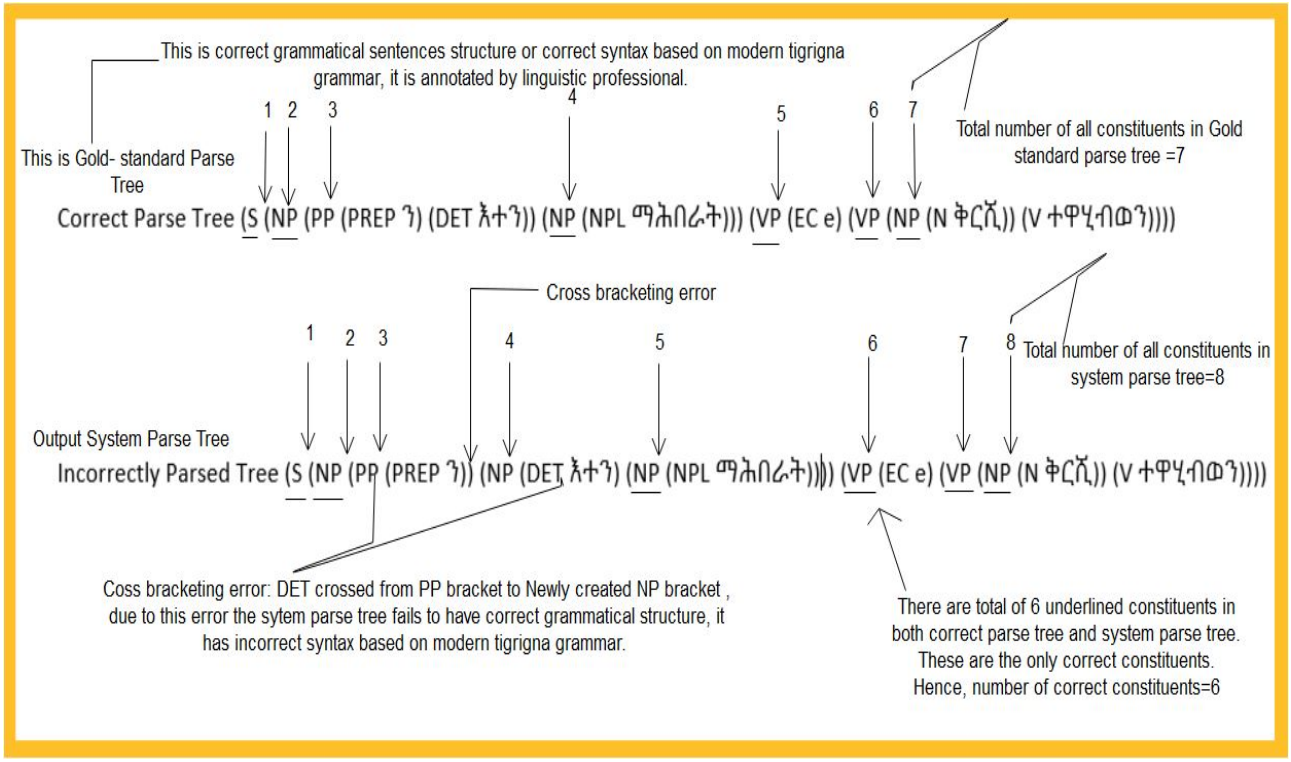
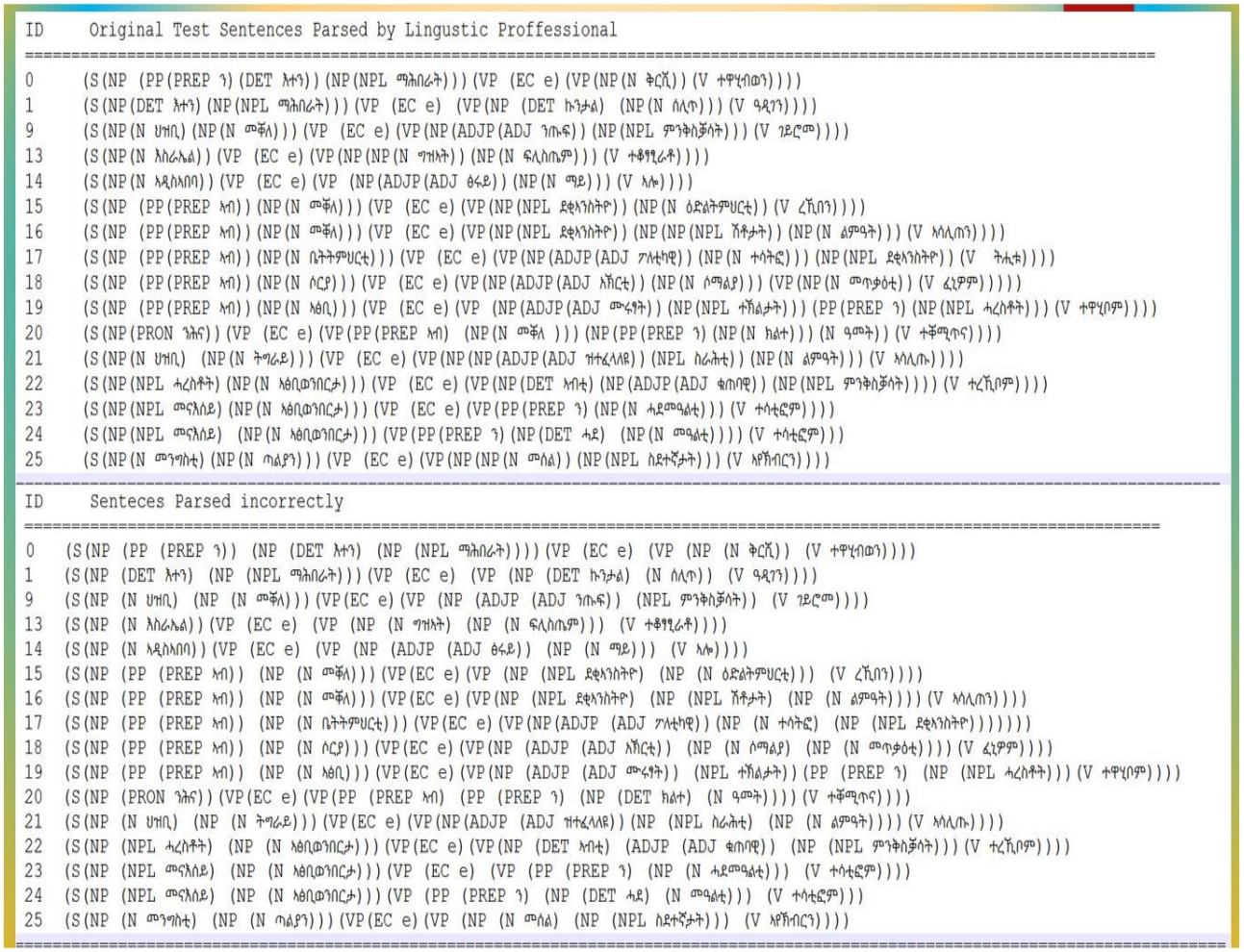


Figure 5.7: Evaluation measurements across precison and recall

```
# Score Result
 ID | length | recall | prec | matched_brackets | gold_brackets | test_brackets | cross_brackets |
---:|-------:|-------:|-----:|-----------------:|--------------:|--------------:|---------------:|
   0|       6|    0.86|  0.75|                 6|              7|              8|               1|
   1|       6|    0.86|  1.00|                 6|              7|              6|               0|
   9|       6|    0.88|  1.00|                 7|              8|              7|               0|
  13|       5|    0.86|  1.00|                 6|              7|              6|               0|
  15|       6|    0.88|  0.88|                 7|              8|              8|               0|
  16|       7|    0.80|  0.89|                 8|             10|              9|               0|
  18|       7|    0.73|  0.80|                 8|             11|             10|               2|
  19|       8|    0.91|  1.00|                10|             11|             10|               0|
  20|       0|    0.00|  0.00|                 0|              0|              0|               0|
  21|       7|    0.89|  0.89|                 8|              9|              9|               1|
  22|       7|    0.89|  1.00|                 8|              9|              8|               0|
  24|       6|    0.86|  1.00|                 6|              7|              6|               0|
  25|       6|    0.88|  1.00|                 7|              8|              7|               0|
================================================================================================
Number of sentence: 48.00
Bracketing Recall:  93.36
Bracketing Precision:   96.79
Bracketing FMeasure:    95.05
```

Figure 5.8: Evalb Scorer Ananlysis Results on Tigrigna Test Sets for incorrectly parsed sentence on 80/20 percentage split



```
ID     Original Test Sentences Parsed by Lingustic Proffessional
=============================================================================================================================
0      (S(NP (PP(PREP ነ) (DET እቱነ)) (NP(NPL ማሕበራት))) (VP (EC e) (VP(NP(N ቅርሺ)) (V ተዋሂብወነ))))
1      (S(NP(DET እቱነ) (NP(NPL ማሕበራት))) (VP (EC e) (VP(NP (DET ኩንታል) (NP(N ሰለጥ))) (V ዓዲገነ))))
9      (S(NP(N ህዝቢ) (NP(N መቐለ))) (VP (EC e) (VP(NP(ADJP(ADJ ንጡፍ)) (NP(NPL ምንቅስቓሳት))) (V ገይሮም))))
13     (S(NP(N እስራኤል)) (VP (EC e) (VP(NP(NP(N ግዝኣት)) (NP(N ፍሊስጤም))) (V ተቆፃፂራቶ))))
14     (S(NP(N ኣዲስኣበባ)) (VP (EC e) (VP (NP(ADJP(ADJ ፅፉይ)) (NP(N ማይ))) (V ኣሎ))))
15     (S(NP (PP(PREP ኣብ)) (NP(N መቐለ))) (VP (EC e) (VP(NP(NPL ደቂኣንስትዮ)) (NP(N ዕድልትምህርቲ)) (V ረኺቦን))))
16     (S(NP (PP(PREP ኣብ)) (NP(N መቐለ))) (VP (EC e) (VP(NP(NPL ደቂኣንስትዮ)) (NP(NP(NPL ሽቶታት)) (NP(N ልምዓት))) (V ኣሳሊጠን))))
17     (S(NP (PP(PREP ኣብ)) (NP(N ቤትትምህርቲ))) (VP (EC e) (VP(NP(ADJP(ADJ ፖለቲካዊ)) (NP(N ተሳትፎ))) (NP(NPL ደቂኣንስትዮ)) (V ትሒቱ))))
18     (S(NP (PP(PREP ኣብ)) (NP(N ሶርያ))) (VP (EC e) (VP(NP(ADJP(ADJ ኣሸርቲ)) (NP(N ሶማልያ))) (VP(NP(N መጥቃዕቲ)) (V ፈነዎም)))))
19     (S(NP (PP(PREP ኣብ)) (NP(N ኣፅቢ))) (VP (EC e) (VP (NP(ADJP(ADJ ሙሩፃት)) (NP(NPL ተኽልታት))) (PP(PREP ነ) (NP(NPL ሓረስቶት))) (V ተዋሂቦም))))
20     (S(NP(PRON ነሕና)) (VP (EC e) (VP(PP(PREP ኣብ) (NP(N መቐለ ))) (NP(PP(PREP ነ) (NP(N ክልተ))) (N ዓመት)) (V ተቐሚጥና))))
21     (S(NP(N ህዝቢ) (NP(N ትግራይ))) (VP (EC e) (VP(NP(NP(ADJP(ADJ ዝተፈላለዩ)) (NPL ስራሕቲ)) (NP(N ልምዓት))) (V ኣሳሊጡ))))
22     (S(NP(NPL ሓረስቶት) (NP(N ኣፅቢወንበርታ))) (VP (EC e) (VP(NP(DET ኣብቲ) (NP(ADJP(ADJ ቁጠባዊ)) (NP(NPL ምንቅስቓሳት)))) (V ተረኺቦም))))
23     (S(NP(NPL መናእሰይ) (NP(N ኣፅቢወንበርታ))) (VP (EC e) (VP(PP(PREP ነ) (NP(N ሓደመዓልቲ))) (V ተሳቲፎም))))
24     (S(NP(NPL መናእሰይ) (NP(N ኣፅቢወንበርታ))) (VP(PP(PREP ነ) (NP(DET ሓደ) (NP(N መዓልቲ)))) (V ተሳቲፎም)))
25     (S(NP(N መንግስቲ) (NP(N ጣልያን))) (VP (EC e) (VP(NP(NP(N መሰል)) (NP(NPL ስደተኛታት))) (V ኣየኽብርን))))
=============================================================================================================================
ID     Senteces Parsed incorrectly
=============================================================================================================================
0    (S(NP (PP (PREP ነ)) (NP (DET እቱነ) (NP (NPL ማሕበራት)))) (VP (EC e) (VP (NP (N ቅርሺ)) (V ተዋሂብወነ))))
1    (S(NP (DET እቱነ) (NP (NPL ማሕበራት))) (VP (EC e) (VP (NP (DET ኩንታል) (N ሰለጥ)) (V ዓዲገነ))))
9    (S(NP (N ህዝቢ) (NP (N መቐለ))) (VP(EC e) (VP (NP (ADJP (ADJ ንጡፍ)) (NPL ምንቅስቓሳት)) (V ገይሮም))))
13   (S(NP (N እስራኤል)) (VP (EC e) (VP (NP (N ግዝኣት) (NP (N ፍሊስጤም))) (V ተቆፃፂራቶ))))
14   (S(NP (N ኣዲስኣበባ)) (VP (EC e) (VP (NP (ADJP (ADJ ፅፉይ)) (NP (N ማይ))) (V ኣሎ))))
15   (S(NP (PP (PREP ኣብ)) (NP (N መቐለ))) (VP(EC e) (VP (NP (NPL ደቂኣንስትዮ) (NP (N ዕድልትምህርቲ))) (V ረኺቦን))))
16   (S(NP (PP (PREP ኣብ)) (NP (N መቐለ))) (VP(EC e) (VP(NP (NPL ደቂኣንስትዮ) (NP (NPL ሽቶታት) (NP (N ልምዓት)))) (V ኣሳሊጠን))))
17   (S(NP (PP (PREP ኣብ)) (NP (N ቤትትምህርቲ))) (VP(EC e) (VP(NP(ADJP (ADJ ፖለቲካዊ)) (NP (N ተሳትፎ) (NP (NPL ደቂኣንስትዮ))))))))
18   (S(NP (PP (PREP ኣብ)) (NP (N ሶርያ))) (VP(EC e) (VP(NP (ADJP (ADJ ኣሸርቲ)) (NP (N ሶማልያ) (NP (N መጥቃዕቲ)))) (V ፈነዎም))))
19   (S(NP (PP (PREP ኣብ)) (NP (N ኣፅቢ))) (VP(EC e) (VP(NP (ADJP (ADJ ሙሩፃት)) (NPL ተኽልታት)) (PP (PREP ነ) (NP (NPL ሓረስቶት))) (V ተዋሂቦም))))
20   (S(NP (PRON ነሕና)) (VP(EC e) (VP(PP (PREP ኣብ) (PP (PREP ነ) (NP (DET ክልተ) (N ዓመት)))) (V ተቐሚጥና))))
21   (S(NP (N ህዝቢ) (NP (N ትግራይ))) (VP(EC e) (VP(NP(ADJP (ADJ ዝተፈላለዩ)) (NP (NPL ስራሕቲ) (NP (N ልምዓት)))) (V ኣሳሊጡ))))
22   (S(NP (NPL ሓረስቶት) (NP (N ኣፅቢወንበርታ))) (VP(EC e) (VP(NP (DET ኣብቲ) (ADJP (ADJ ቁጠባዊ)) (NP (NPL ምንቅስቓሳት))) (V ተረኺቦም))))
23   (S(NP (NPL መናእሰይ) (NP (N ኣፅቢወንበርታ))) (VP (EC e) (VP (PP (PREP ነ) (NP (N ሓደመዓልቲ))) (V ተሳቲፎም))))
24   (S(NP (NPL መናእሰይ) (NP (N ኣፅቢወንበርታ))) (VP (PP (PREP ነ) (NP (DET ሓደ) (N መዓልቲ))) (V ተሳቲፎም)))
25   (S(NP (N መንግስቲ) (NP (N ጣልያን))) (VP(EC e) (VP (NP (N መሰል) (NP (NPL ስደተኛታት))) (V ኣየኽብርን))))
=============================================================================================================================
```

Figure 5.9: Evalb Scorer: Comparision of Orignial Parsed test sets with Test sets of sentences Parsed by the parser

5.4 *Summary*

This chapter is where most of the experimental evaluation is successively made on both the tagger and the parser. Firstly, out of the tree bank three percentage splits are made with 60−40, 75−25, 80−20 on both parsing and tagging. Then on tagging the segementer developed segements if any complex morphological affix or any compound word exists. After this step, the output is given to the tagger for tagging this is single sentences and since the tagger is trained on the training sets based on the split percentage, then this tagged output is given to the parser and finally the parser out puts the grammatical syntax of this single sentences. The same is for the whole corpus of test data. The whole corpus of test data is given to the segementer then the whole output of the segementer is given to the tagger to be tagged. The process on how the TnT tagger is trained and tags new corpus is done with the experiments and general architecture given in this chapter. Then after, the tagger tags the test data successfuly, the out put of the tagger which is

tagged test data is given to the parser. Then, the parser parses the the tagged data. This parsed data is system parse. But at first their is comparable real annotatd corpus exclusivly taken as test data which is parsed by the linguistic proffessional. Hence, at the end this real data was never seen to the parser and the system parse output is compared with this real gold data. The comparison of matching gold standard test data before parsing and after their tagged are parsed is made with the tool called EValb scorer. This scorer then outputs the test data efficiency measure called F-score at the end. It out puts for all the three percentage splits on each successive experiment. The calculation of how the corresponding recall, precision and F-measure is calculated is explained with an example in this chapter in detail. The example is taken from the gold data kept and from system parsed data as could be seen from id 0 of the figure 5.7, figure 5.8, figure 5.9.

CHAPTER SIX
CONCLUSION AND RECOMMENDATION

6.1 Contribution

The findings of the research include:

1. *Tree bank Preparation. In this research, a tree bank with a total of 250 correctly parsed sentence or syntactically annotated corpus is prepared by linguistic professional for statistical machine learning purpose.*
2. *Automatic PCFG Grammar Induction: Static assignment of probability to grammars was done by the previous Amharic researchers on constituency parsing. In this research, an algorithm is developed to induce probabilities from CFG repository automatically. This is used as a language model for parsing.*
3. *Integration: Integration of the n-gram including Tri-gram language model is done indirectly with the PCFG parsing language model. In this way as the tagger becomes intermediate input to the parser for lexical disambiguation, the parser then handles the grammatical syntactic disambiguation. This is very useful as the parser is fully automated being as grammar checker since it formerly knows the syntax of the language.*
4. *Development of Segementer algorithm and IPA to Ethiopic Script : Tigrigna is morphologicaly rich and complex language. Hence, word and morphological segementer is developed to assist for the tagger ease of complexities. An algorithm to represent IPA in Ethiopic script is also developed. This crucial to represent the language by its own script rather than merely representing its by IPA because users who know IPA representation of Tigrigna can also understand sentences written in Ethiopic alphabets.*
5. *Standard Performance Evaluation Technique. In this research standard parser evaluation technique called PARSEVAL is used. Hence, the performance accuracy of the parser is measured with standard scoring tool for PRSEEVAL called Evalb Scorer.*

6.2 Conclusion

The performance accuracy of the syntax analyzer developed in this research has F-score 95.12%. Hence, NLP community of researchers can use it as an input for machine translation. Tigrigna language researchers can use it for grammar checking and morphological segmentation since it is fully automated and generally developed based modern Tigrigna grammar.

6.3 Recommondation

The following are recommendation of the researcher:

1. Unsupervised learning Algorithm required:

   Unsupervised learning has an advantage over supervised learning in that one does not need to hand-annotate sentences, and thus would provide a cost-effective way of developing parsing resources for minority languages for which there are no treebanks. Hence, in this research even though the method for automatically acquiring a PCFG from an unannotated corpus (unsupervised learning) is made through the integration of TnT tagger with the PCFG Aumtomatic Grammar induction that was at first used to induce the PCFG probablities, from the hand parsed or annotated corpora, how ever with no availablity of annotated corpora, it would be difficult to know the structure of these induced grammars. Hence, development of unsupervised learning algorithm that could result the success rate of PCFG machine learning method or better is left for future researchers.

2. Dependency :

   In this research the PCFG statistical parsing is applied for the constituency based parsing frame work. However, since there is no research conducted on the dependency frame work for Tigrigna language, the researcher recommends future researchers to conduct on dependency parser for Tigrigna language by preparing large annotated tree bank corpus or develop unsupervised metho one. Because better than constituency frame work, the dependency frame work adds to the structures some useful clue that would be helpful in the extraction of predicate-argument information that includes includes additional functional labels such as SBJ for subject.

3. Lexicalization : PCFG parsing in general has certain known weaknesses, such as its inability to take lexical information or structural context into account. This is due to the independence assumptions that are inherent in the formalism, as it assumes that the expansion of any non-terminal label is independent of any other expansion. Hence, this research is focused on Unlexicalized parsing with PCFGs. However, in order to maintain computational simplicity over other more complex parsing methods, PCFG parsers need to be lexicalized with parent transformations just as that Johnson (1999) investigates the idea of a 'parent-transformation'

where each node N of a tree is annotated with its parent category label P to give N^P. The category label NP, for example, becomes NP^S, if it occurs under an S node. A training corpus can be transformed automatically in this manner before extracting a parent-annotated grammar. This grammar now has additional contextual information that a basic PCFG does not encode. For example, it is now possible to distinguish between NPs occurring as subjects of a sentence and NPs occurring as objects of a verb: NPs in subject position are daughters of S nodes, and NPs in object position are daughters of VP nodes. Subject NPs will be annotated NP^S and object NPs will be annotated $NP^V P$. In such way, as Hindle and Rooth (1993) demonstrateed, lexical dependencies become crucial for resolving ambiguities.

4. Treebank Development:

   Treebanks get their significant in natural language processing when supervised machine learning techniques are required and higher level NLP components are to be developed off for processing. In this research an attempt of development or building of Treebank annotated corpus is made by adapting methods from resource rich as well as low resourced languages. Hence, preparation of linguisticaly hand parsed tree banks with morphological features and large data sets is left for future researchers.

5. Splitting Method:

   future researchers could use 10 cross validation technique on their development of robust parser than done on this research.

References

[1] E. Reiter, R. Dale, and Z. Feng, *Building natural language generation systems*. MIT Press, 2000, vol. 33.

[2] Q. Wang, "Drawing tree diagrams: Problems and suggestions," *Journal of Language Teaching and Research*, vol. 1, no. 6, pp. 926–934, 2010.

[3] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[4] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit." in *ACL (System Demonstrations)*, 2014, pp. 55–60.

[5] S. Bird, "Nltk: the natural language toolkit," in *Proceedings of the COLING/ACL on Interactive presentation sessions*. Association for Computational Linguistics, 2006, pp. 69–72.

[6] "Bracket scoring program," http://nlp.cs.nyu.edu/evalb/, accessed: 2016-10-15.

[7] C. F. Hempelmann, V. Rus, A. C. Graesser, and D. S. McNamara, "Evaluating state-of-the-art treebank-style parsers for coh-metrix and other learning technology environments," in *Proceedings of the second workshop on Building Educational Applications Using NLP*. Association for Computational Linguistics, 2005, pp. 69–76.

[8] T. Kakkonen, *Framework and resources for natural language parser evaluation*. Tuomo Kakkonen, 2007.

[9] Y. Liu and Q. Liu, "Joint parsing and translation," in *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 707–715.

[10] K. Yamada and K. Knight, "A syntax-based statistical translation model," in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2001, pp. 523–530.

[11] S. Clark, "Statistical parsing," *Handbook of Computational Linguistics and Natural Language Processing*, pp. 333–363, 2010.

[12] Y. Zhang, "Chinese-english statistical machine translation by parsing," 2006.

[13] M. Jhon, *Book, Tigrinya grammar*. American Evangelical Mission, First Red Sea Press, Inc., Edition,1996.

[14] T. Tewolde, *Book, A modern grammar of Tigrigna*. Tipografia U. Detti – via G. Savonarola Roma, 2002.

[15] D. Reda, *Book, A modern grammar of Tigrigna*. Mega Publishing Enterprise, 2006.

[16] D. Jurafsky and J. H. Martin, "Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition," 2000.

[17] B. Roark, "Probabilistic top-down parsing and language modeling," *Computational linguistics*, vol. 27, no. 2, pp. 249–276, 2001.

[18] D. Klein and C. D. Manning, "A parsing: fast exact viterbi parse selection," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, 2003, pp. 40–47.

[19] S. Petrov and D. Klein, "Improved inference for unlexicalized parsing." in *HLT-NAACL*, vol. 7, 2007, pp. 404–411.

[20] M. Collins and T. Koo, "Discriminative reranking for natural language parsing," *Computational Linguistics*, vol. 31, no. 1, pp. 25–70, 2005.

[21] E. Charniak, "A maximum-entropy-inspired parser," in *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*. Association for Computational Linguistics, 2000, pp. 132–139.

[22] D. M. Magerman, "Statistical decision-tree models for parsing," in *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1995, pp. 276–283.

[23] J. Le Roux, B. Favre, A. Nasr, and S. A. Mirroshandel, "Generative constituent parsing and discriminative dependency reranking: Experiments on english and french," in *ACL 2012 Joint Workshop on Statistical Parsing and Semantic Processing of Morphologically Rich Languages (SP-Sem-MRL 2012)*, 2012, p. q.

[24] E. Charniak and M. Johnson, "Coarse-to-fine n-best parsing and maxent discriminative reranking," in *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2005, pp. 173–180.

[25] D. McClosky, E. Charniak, and M. Johnson, "Effective self-training for parsing," in *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics, 2006, pp. 152–159.

[26] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, 2003, pp. 423–430.

[27] M. Y. Tachbelie, S. T. Abate, and L. Besacier, "Part-of-speech tagging for underresourced and morphologically rich languages—the case of amharic," *HLTD (2011)*, pp. 50–55, 2011.

[28] M. Gasser, "Hornmorpho: a system for morphological processing of amharic, oromo, and tigrinya," in *Conference on Human Language Technology for Development, Alexandria, Egypt*, 2011.

[29] T. Brants, "Tnt: a statistical part-of-speech tagger," in *Proceedings of the sixth conference on Applied natural language processing*. Association for Computational Linguistics, 2000, pp. 224–231.

[30] Y. K. Tedla, K. Yamamoto, and A. Marasinghe, "Tigrinya part-of-speech tagging with morphological patterns and the new nagaoka tigrinya corpus," *International Journal of Computer Applications*, vol. 146, no. 14, 2016.

[31] T. G. ABREHA, "Part of speech tagger for tigrigna," Master's thesis, ADDIS ABABA UNIVERSITY, 2010.

[32] A. Atelach, "Automatic sentence parsing for amharic text an experiment using probabilistic context free grammars," Master's thesis, AAU, 2002.

[33] D. Gochel, "An integrated approach to automatic complex sentence parsing for amharic text," Master's thesis, AAU, 2003.

[34] "Romanisation systems," https://www.gov.uk/government/publications/romanisation-systems, accessed: 2015-09-30.

[35] "Romanisation systems," https://www.loc.gov/catdir/cpso/romanization/tigrinya.pdf, accessed: 2015-09-30.

[36] N. Chomsky, *Book,Syntactic structures*. Walter de Gruyter, 2002.

[37] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: The penn treebank," *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[38] A. Bies, M. Ferguson, K. Katz, R. MacIntyre, V. Tredinnick, G. Kim, M. A. Marcinkiewicz, and B. Schasberger, "Bracketing guidelines for treebank ii style penn treebank project," *University of Pennsylvania*, vol. 97, p. 100, 1995.

[39] E. K. Ringger, R. C. Moore, E. Charniak, L. Vanderwende, and H. Suzuki, "Using the penn treebank to evaluate non-treebank parsers." in *LREC*, 2004.

[40] A. Taylor, M. Marcus, and B. Santorini, "The penn treebank: an overview," in *Treebanks*. Springer, 2003, pp. 5–22.

[41] M. Al-Emran, S. Zaza, and K. Shaalan, "Parsing modern standard arabic using treebank resources," in *Information and Communication Technology Research (ICTRC), 2015 International Conference on*. IEEE, 2015, pp. 80–83.

[42] Y. Goldberg and M. Elhadad, "Word segmentation, unknown-word resolution, and morphological agreement in a hebrew parsing system," *Computational Linguistics*, vol. 39, no. 1, pp. 121–160, 2013.

[43] R. Tsarfaty, "Syntax and parsing of semitic languages," in *Natural Language Processing of Semitic Languages*. Springer, 2014, pp. 67–128.

[44] M. Johnson, "Pcfg models of linguistic tree representations," *Computational Linguistics*, vol. 24, no. 4, pp. 613–632, 1998.

[45] T. Deoskar, M. Rooth, and K. Sima'an, "Smoothing fine-grained pcfg lexicons," in *Proceedings of the 11th International Conference on Parsing Technologies*. Association for Computational Linguistics, 2009, pp. 214–217.

[46] F. Jelinek, J. D. Lafferty, and R. L. Mercer, "Basic methods of probabilistic context free grammars," in *Speech Recognition and Understanding*. Springer, 1992, pp. 345–360.

[47] A. Cahill, "Treebank-based probabilistic phrase structure parsing," *Language and linguistics Compass*, vol. 2, no. 1, pp. 36–58, 2008.

[48] R. Hwa, "Supervised grammar induction using training data with limited constituent information," in *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics, 1999, pp. 73–79.

[49] A. Clark and S. Lappin, "Unsupervised learning and grammar induction," *The Handbook of Computational Linguistics and Natural Language Processing*, vol. 57, 2010.

[50] N. N. Pise and P. Kulkarni, "A survey of semi-supervised learning methods," in *Computational Intelligence and Security, 2008. CIS'08. International Conference on*, vol. 2. IEEE, 2008, pp. 30–34.

[51] P. Skórzewski, "Effective natural language parsing with probabilistic grammars," in *Computer Science and Information Technology (IMCSIT), Proceedings of the 2010 International Multiconference on*. IEEE, 2010, pp. 501–504.

[52] S. Sekine and M. Collins, "Evalb bracket scoring program," *URL: http://www. cs. nyu. edu/cs/projects/proteus/evalb*, 1997.