ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL SCIENCES

DEPENDENCY BASED AMHARIC GRAMMAR CHECKER

Abraham Gebreamlak Gebremariam

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia

September 2019

Addis Ababa University

College of Natural Sciences

Abraham Gebreamlak

Advisor: Yaregal Assabie (PhD)

This is to certify that the thesis prepared by Abraham Gebreamlak, titled: *Dependency Based Amharic Grammar Checker* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

	Name	Signature	Date
Advisor:	Yaregal Assabie (PhD)		
Examiner:	Dagmawi Lemma (PhD)		
Examiner:	Ayalew Belay (PhD)		

Abstract

Nowadays, advancement in computer and software technologies have reached the level of becoming basic necessity. The rise of computer-like gadgets such as smart phones, hand-held device, etc. is making the society paperless. As a result, we compose and exchange information on hourly basis. A grammar checker then come to play a role in identifying grammatical errors efficiently. An Amharic grammar checker was designed and developed through phrase structure grammar formalism which lacks to analyze nonlinear structure of a sentences. Another problems of phrase structure grammar is that as the complexity of phrases increases, it is difficult to treat directly through statistical model such as bigram and trigram models. These factors initiated us to design and develop a dependency based Amharic grammar checker. Accordingly, we propose an Amharic grammar checker integrated with dependency parsing system. The parser is based on dependency grammar formalism through which relationship of a word and its modifier is identified.

The system is implemented through Python 3.7.2, UDpipe 1.0 to obtain and evaluate tokenizing and tagging models; MaltParser 1.9.2 to induce dependency parsing models and MaltEval 1.0 to evaluate the result of parsing model. The models were trained with a dependency treebank for Amharic. Lastly, we reported the performance of the induced models and grammar checker with randomly selected sentences from dependency treebank. The tokenizer and the tagger were also evaluated with raw texts collected from newspapers. We found out the tokenizer performs good with an accuracy of 100%. However the tagger's performance was 43.11% on raw text corpus. The dependency parser were also evaluated during development to select best algorithm; which was Covington Non-projective algorithm. The agreement checker was evaluated for each agreement type. The results are 68.18%, 81.25% and 20% of subject-verb, object-verb, and adverb-verb agreements respectively. These results confirm us the feasibility of a dependency based grammar checker with an integrated dependency parser.

Keywords: Dependency and Phrase structure grammar, Dependency Parser

Acknowledgments

I would like to express my special thanks of gratitude to my advisor Dr. Yaregal Assabie who gave me invaluable idea to initiate on the topic this research work. I would also thank him for his support, understanding, patience and motivation during the entire research work.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Acronyms/Abbreviations

CFG	Context Free Grammar
CONLL	Computational Natural Language Learning
CONLL-U	Computational Natural Language Learning Universal
CS	Complex Sentence
DBAG	Dependency Based Amharic Grammar Checker
DET	Determiner
DOBJ	Direct Object
HMM	Hidden Markov Model
LAS	Labeled attachment score
MST	Maximum Spanning Tree
N	Noun
NLP	Natural Language Processing
NMOD	Noun Modifier
NP	Noun Phrase
NSUBJ	Noun Subject
OBJ	Object of a Sentence
POS	Part of speech tagger
SBJ	Subject of a Sentence
SMT	Statistical Machine Translation
SOV	Subject Object Verb
SS	Simple Sentence
SVM	Support Sector Machine
UAS	Unlabeled attachment score
UD	Universal Dependency
UPOS	Universal Part of Speech
V	Verb
VP	Verb Phrase
XML	Extensible Markup Language
XPOS	Language Specific Part of Speech

Chapter 1: Introduction

1.1. Background

One of the fundamental features of human behavior is natural language. It is a vital means through which we communicate about the world that affects our daily lives. Most human knowledge is recorded using natural languages. Natural languages are then processed to understand and represents human knowledge into meaningful computer understandable forms. The result of which help us to further analyze and represent naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications [1]. These level of linguistic analysis include morphological, syntax, semantic, discourse and pragmatic analysis.

One application of syntactic analysis is grammar checking which deals with identifying grammatically incorrect sentences. Research and development of grammar checking techniques have been carried out since the 1970's [2], mainly for well-resource languages such as English [3, 4, 5, 6], Swedish [7, 8, 9, 10], and so on. The earliest grammar checker for local language was developed in 2011 [11] for Afan Oromo. The grammar checker was designed based on manually created rules. Even if the grammar checker provides promising results, it still generates false alarm when it is provided with incorrectly tagged words [11]. The other grammar checker for local language was Amharic grammar checker [12] which was developed in 2013. The grammar checker for Amharic is a bit complex than that of Afan Oromo as it was implemented with two methods. The first is rule-based approach for simple sentences. The other is statistical one for both simple and complex sentences.

To implement a grammar checker, we need grammar of the specific language and the formalism of the grammar through which the language is represented. Grammar refers to a system of rules describing what correct sentences should look like [13]. A grammar can be formalized through phrase structure

grammar or dependency grammar. A phrase structure grammar is also called constituency grammar which deals with the constituent of a sentences such as noun, noun phrase, verb, verb phrase, etc. [1]. Dependency grammar formalism, on the other hand, is the syntactic representation of natural language based on dependencies between pairs of words, one is designated as a head and the other is its dependent [14]. This refers the pair as a head-modifier dependency relationship.

1.2. Motivation

Currently, we use a grammar checker on word processors to check the grammatical correctness of a text and sometimes to evaluate the result of machine translation. This is a common practice for well-resourced languages such as English. We are guaranteed that a word processor provides suggestions for errors we made while using English language in word processors.

It is becoming increasingly important to have an Amharic grammar checker since more and more electronic Amharic documents are being produced with the advent of computers and word processing applications. In addition to this, we are prone to grammatical incorrectness since Amharic is the second language for most of none-native speakers of Amharic and it is morphologically complex language.

An attempt was made to design and develop an Amharic grammar checker [12]. The research was conducted with the motives of alleviating grammatical errors made during writing in general and designing grammar checking system that suites the language's morphological structure, clause order, etc., instead of directly adopting other languages' grammar checking system in particular. Accordingly, the grammar checker was implemented with phrase structure grammar formalism of Amharic language. This model focuses on the concatenation of strings which result in generation of useless intermediate nodes called phrases [15]. The introduction of intermediate nodes make constituency grammar less efficient than dependency grammar [15]. For

example, if we represent a sentence *"students hate annoying professors"* on both constituency as well as in dependency grammar, we are going to gain considerable performance improvement. As it is depicted in Figures 1.1 and 1.2; four edges must be passed or traversed in order to check agreement between *students* and *hate* in constituency grammar while one edge is enough in dependency framework [15].

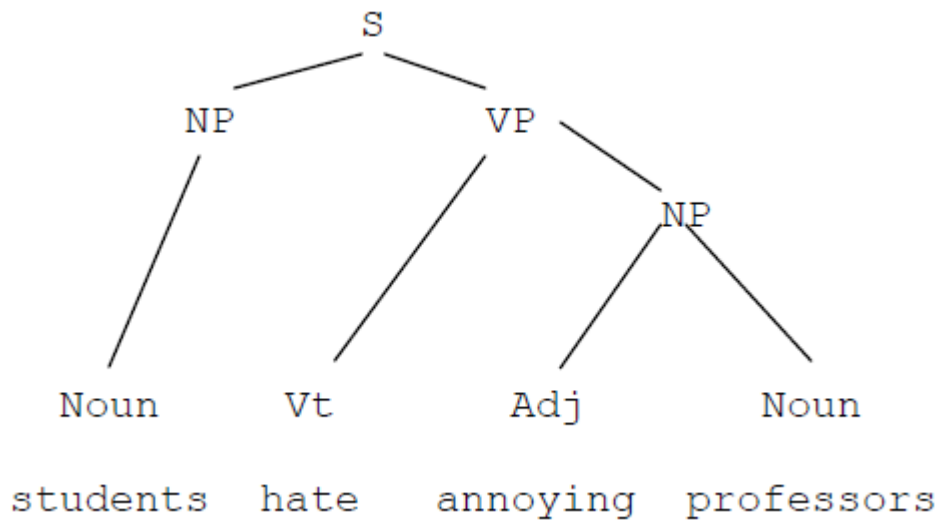

Figure 1.1. Constituency Representation

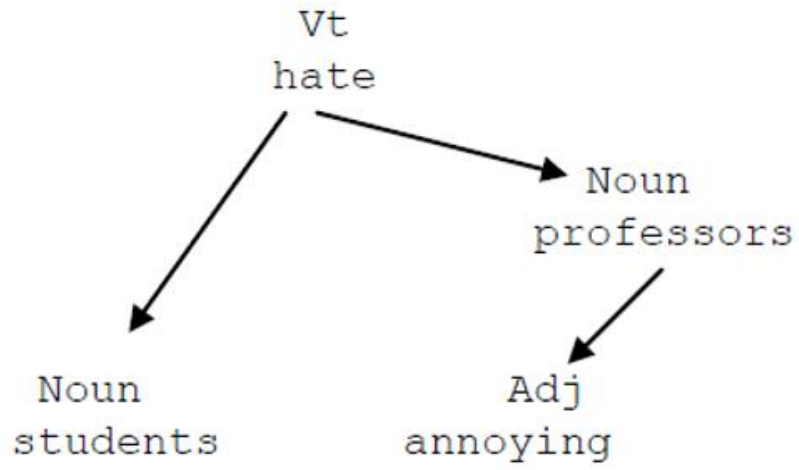

Figure 1.2. Dependency Representation.

The choice of dependency grammar formalism is, therefore, motivated by the following factors.

- Dependency model focuses on functional complementation of syntactic unites rather than intermediate unites [15].
- The path along which features are propagated for unification are longer in constituency tree than in dependency tree [15] [16].
- In constituency grammar, the number of parse tree grows exponentially with the sentence length, which is difficult to control [17].

1.3. Statement of the Problem

Every time information technology advancement is becoming more and more volatile and unpredictable. The emergence of fast and portable computing devices, and widely availability of smart phones that have equal capability as a computer create paperless society. This indicates that volumes of electronic documents are increasing significantly. We compose electronic document on word processors and exchange them through email and social media.

A grammar checker then helps to identify grammatical errors effortlessly and precisely. This is a common practice for well-resourced languages. However, little has been done for Amharic compared to others. Aynadis Temesgen and Yaregal Assabie [12] designed an Amharic grammar checker by taking phrase structure grammar into consideration, where it lacks capabilities of modeling languages that have complex morphological features such as Amharic. Such cases are exhibited while analyzing languages with statistical approach of n-gram model. It is also impossible to manually craft all grammar rules exhaustively. Aynadis Temesgen and Yaregal Assabie [12] claim that Amharic sentences with complex phrase structures are difficult to model directly with bigram or trigram methods. To alleviate such problem and improve the performance of the grammar checker, they recommended to apply a parser before checking grammatical correctness of a sentence.

Another feature that phrase structure grammar lacks is that it treats all sentences as a sequence of words making linear relationship. In other words, it only considers relationship between adjacent words as if no relationship

exist among other nonlinear words of a sentence. It ignores tree-like structure of natural language sentences [18].

This research, therefore, is going to be conducted with the aim of exploring Amharic sentences structure by focusing on dependency grammar formalism to alleviate the problem arise from the complexity of phrase structure and to handle nonlinear relationships of words in a sentence. The research also incorporates dependency parsing to capture dependency relationships among words of which grammatical agreements are going to be checked.

1.4. Objectives

- **General Objective**

The general objective of this research is to design and develop a prototype of a dependency based Amharic grammar checker by integrating dependency parser.

- **Specific Objectives**

To achieve the general objective of the study the following specific objective are identified.

- ❖ Reviewing literature on theoretical subject matter of grammar checker and related works done on other languages;
- ❖ Collecting Amharic corpus and preparing test sets to evaluate performance of the developed system;
- ❖ Studying grammatical structure, morphology and syntax of Amharic language;
- ❖ Designing dependency grammar for Amharic sentences;
- ❖ Developing a prototype for the proposed system and
- ❖ Evaluating the performance of the system.

1.5. Scope and Limitation of the Study

The scope of this research work is concerned with detecting common grammatical errors of Amharic sentences, which include subject-verb disagreement, object-verb disagreement, adjective-noun disagreement and adverb-verb disagreement. However, this research work does not include grammatical error correction of Amharic sentences.

1.6. Method

The methodology that the researcher is going to be conducted comprises four basic activities: conducting literature review, data collection and preparation, prototype development and evaluating the system.

- **Literature Review**

  Literature review has been conducted on theoretical framework that make up grammar checking and Amharic sentence, morphology and syntax. In addition to this, related works are going to be reviewed to show what has been done and how the research works have been done by others on dependency grammar formalism and dependency parsing.

- **Data Collection**

  Amharic sentences have been collected from grammar books to prepare them in the format that suit dependency based analysis. A dependency grammar formalism mostly requires input data to be in CONLL or CONLL-U format in which sentences are represented column-wise with ten fields. The collected data will be used for training and testing the performance of the system.

- **Prototype Development**

  Different tools are used to develop the prototype of the system. These are UDpipe version 1.0 to induce word tokenizer, POS and morphological feature tagger; MaltParser version 1.9.2 to analyze sentences in dependency framework and to induce a dependency parsing model for

Amharic; MaltEval version 1.0 to evaluate the induced model and select the best parsing model. Python is also used to develop the prototype.

- **Evaluation**

  Once the prototype is developed, it is evaluated with appropriate metrics. Since the prototype encompasses tokenizer, tagger, dependency parser and agreement checker, the evaluation metrics vary accordingly. Therefore, we use two group of evaluation metrics.

  The dependency parser is evaluated by labeled attachment score (LAS) and unlabeled attachment score (UAS) [19] using MaltEval [20]. UAS measures the percentage of correctly assigned head. It ignores whether or not the dependency relation is correctly labeled or not. A more encompassing accuracy measure is LAS, which measures the percentage of tokens that are assigned the correct head with the correct relation.

  The tokenizer, tagger and the agreement checker, however, are evaluated for precision and recall. Test case are going to be prepared manually to evaluate the performance of the prototype.

1.7. Application of Result

Digital media is significantly impacting most of our daily lives in one way or another. We communicate through email and compose on social media such as Facebook and Twitter on daily even hourly basis. We also communicate through printed documents. A grammar checker then help us to produce error free works.

This research, therefore, enables Amharic language users, particularly the non-native ones, to effectively prepare official documents, letters, emails, etc. It improves productivity as it saves the time users take for proofreading too.

Apart from these, a grammar checker can also be used as a post processing activity for statistical machine translation (SMT). One problem with standard statistical machine translation system is that its output tends to be

ungrammatical, since it heavily depends on statistical compositions and there is no linguistic knowledge used in the systems. Therefore, grammar checker can be used to evaluate the result of a SMT.

In addition, a grammar checker can be helpful in development of other natural language processing systems like question answering, dialogue generation, and paraphrasing systems etc. [21].

1.8. Organization of the Thesis

This thesis comprises six chapters including the current introductory chapter. Chapter 2 provides an explanation of grammar formalism and grammar checking approaches. It provides an overview of dependency grammar, which is the grammatical analysis we used in designing the grammar checker, along with an explanation of dependency tree, parsing, algorithm, treebank and its data formats. A summary of the nature of the Amharic language and an overview of its distinctive linguistic features are discussed in the end. In Chapter 3, we summarize related works pertinent to developing dependency based grammar checker. These are research works conducted on different types of grammar checker approaches. In Chapter 4, we introduce the design aspects of the dependency-based Amharic grammar checker, its architecture and other linguistic phenomenon that are specific to the Amharic language. We also discuss the components of the grammar checker including sentence preprocessing, dependency parser, relationship extractor and agreement checker. In Chapter 5, we present the result of experiment and the corresponding test cases to evaluate the prototype developed. Finally, in Chapter 6, we report on conclusions and recommendations including possible future works on how to extend and enhance the current work.

Chapter 2: Literature Review

2.1. Introduction

This chapter presents literature review on the concepts that are fundamental for the research. The chapter begins with a brief introduction to grammar, grammar formalism and approaches to grammar checking starting from rule-based method to hybrid one. Constituency grammar and dependency-based grammar formalisms are also discussed briefly with more emphasis given to dependency grammar. The discussion on dependency grammar focuses on dependency relation, dependency tree and treebank, dependency parsing algorithm, as well as universal dependency. Regarding Amharic language, we describe the morphological structure of Amharic words, the type of Amharic sentences and their common grammatical agreements.

This chapter is organized in five sections: Grammar Checking and Approaches, Constituency Grammar, Dependency-Based Grammar, Amharic Language, and Summary.

2.2. Grammar Checking, Formalism and Approaches

Language is a means of exchanging information and knowledge. The information can be specified either in written or spoken forms. The most important thing in information content form is the validity of sentences in the given language. All valid sentences of a language must follow the rules of that language or grammar. Invalid sentences are not worth to convey information especially in written form and therefore rejected immediately.

2.2.1. Grammar

From a formal point of view a language is possibly an infinite set of character strings. The character strings are structured, that is, elementary strings reoccur in various combinations. It is the task of a grammar to define all of the well-formed strings of a language and to describe their structure [15]. A grammar is also defined as the study or use of the rules about how words change their forms and combine with other words to make sentences. In

linguistics, grammar is the set of structural rules governing the composition of clauses, phrases, and words in any given natural language [1].

2.2.2. Grammar Checking

Currently, the day to day activities of human beings are assisted and complemented by computer systems. Computer systems support us in such way that alleviate our problem on different aspects of our life. This includes automating natural language processes. To realize this, it is necessary for computers to get aware of natural languages.

One application of natural language processing is a grammar checker. Grammar checker is a software that verify the syntax of a specific language against the grammatical rules of that language [21]. Correctness and validity of sentences are checked with the help of an underlying grammar of the natural language. The grammar consists of a set of rules that govern the construction of sentences. A valid sentence is one in which all words are compatible to each other in terms of gender, number, person and case. Word order as well as punctuation marks also factor for a sentence to grammatically valid. Grammar checking, therefore, involves testing grammatical agreements of words of a sentences against the language's grammatical rules or compositions.

To this end, grammar formalism, grammar rules and approaches of implementing grammar checker are required to design a grammar checker.

2.2.3. Grammar Formalism

Grammar formalism is a mechanism of knowledge representation of a language through a formal notation. To elaborate further, it is a way of defining and describing syntactic structure of sentences [15].

The syntactic analysis of a sentence is either directed to composition of expressions where hierarchical (top down) analysis of constituents of an expression, or the relations between elements of the expressions in terms of head and its dependents. The first expression focuses on grouping of words

whereas the later interests in syntactic functions of words, or how do words in the expression relate to each other [22]. Every syntactic theory, therefore, contains Constituency (Phrase Structure) or Dependency Analyses.

2.2.4. Grammar Checking Approaches

A grammar checker can be implemented with three approaches regardless of the type of grammar formalism. These are rule-based, statistical and hybrid approaches [18].

Rule-based Grammar Checking

In this approach, input sentences are checked against manually prepared rules which are represented in the form of machine parsable format [23]. The drawbacks to rule-based system are: the extensive effort required to build the ruleset; regardless of the size of the ruleset, given the variability of the human language it is virtually impossible to capture all possible rules and the large number of exceptions to rules [24]. It also requires linguistic expertise on the specific language [25]. However, one advantage of rule-based approach is the grammar checking is always complete if the grammar rules are complete [23]. The rules are easily configurable [25] in that we can add or remove individual rules without affecting the whole setup.

Statistical Grammar Checking

Statistical grammar checkers build statistical models using syntactic labels that are used for detecting and correcting grammatical errors. The typical statistical approach is to model how likely the occurrence of an event is, given a history of preceding events. Thus, statistical approaches easily adaptable to any language requiring only training data in the form of syntactically labeled corpus [24]. The corpus can be created manually or extracted automatically from books, magazines, documents, or online resources etc. The corpus is supposed to be representing all language features and linguistic phenomenon of the language under consideration [12] [25]. The system is, therefore, resource consuming and highly dependent on large and domain adapted data; and it usually lacks semantic information and favoring high occurring

statistical events which are not always the best way of detecting and correcting grammatical errors [24].

Hybrid Grammar Checking

A hybrid system combines both rule-based and statistical approaches to overcome the weaknesses of the two approaches if the mixture of the two components is done properly. Detection of errors can be achieved statistically and rule-based, the task of the hybrid approach is then to resolve any conflicts that arise between the outputs of the two approaches [24]. So that it achieves higher efficiency and more robustness.

2.3. Constituency Grammar

A constituency (phrase structure) grammar deals with the constituent of a sentences such as noun, noun phrase, verb, verb phrase, and so on [1]. It refers to the analysis that divides sentences into its subparts or phrases. It looks at how an expression is built up [22].

The most commonly used mathematical system for modeling constituent structure in natural languages is the Context-Free Grammar [1]. A Context-Free Grammar consists of a set of rules or productions and a lexicon of words and symbols. Each rule expresses the ways symbols of the language are grouped and ordered together. For example, the following productions express that a noun phrase (NP) can be composed of either a *Proper Noun* or a determiner (*Det*) followed by a *Nominal*; a *Nominal* can be one or more *Noun*s. [1]

NP → Det Nominal

NP → Proper Noun

Nominal → Noun | Nominal Noun

Context-free rules can be hierarchically embedded, so we can combine the previous rules with others like the following which express facts about the lexicon:

Det→ a

Det→ the

Noun → flight

The symbols that are used in a CFG are divided into two classes: terminal and non-terminal symbols. The symbols that correspond to words in the language ("the", "flight") are called terminal symbols; the lexicon is the set of rules that introduce these terminal symbols. The symbols that express clusters or generalizations of these are called non-terminals. In each context free rule, the item to the right of the arrow (→) is an ordered list of one or more terminals and non-terminals, while to the left of the arrow is a single non-terminal symbol expressing some cluster or generalization. The non-terminal associated with each word is its lexical category, or part-of-speech [1].

A context free grammar G is a quadrupled [22]:

G= (V_{NT}, V_T, P, S) where:

- ✓ V_{NT} a set of non-terminal symbols (phrases)
- ✓ V_T a set of terminal symbols (lexical items)
- ✓ P a set of production rules of the form A →α, where A is a terminal symbol and α is a sequence of terminal and non-terminal symbols.
- ✓ S: a designated start symbol, S ∈ V_{NT}

2.4. Dependency Grammar

Dependency grammar structure grew from a theory of structural syntax which focused on connections and grammatical relations between words. Dependency grammar structure consists of lexical elements linked by binary dependency relations [26]. It refers to the analysis looking at the relations that are found in an expression. It looks at what role parts of the expression play to convey the meaning [22].

It represents the syntactic structure of a sentence by means of dependency trees. A dependency tree for a sentence is a directed tree whose nodes are all the words of that sentence. Each arc of a tree represents a single syntactic dependency directed from the head to its modifiers or dependent, and labelled with the specific syntactic function such as SBJ for subject of a sentence, OBJ for object of a sentence, NMOD for modifier of a noun, and so on [14].

Dependencies are motivated by grammatical function. A word depends on another either if it is a complement or a modifier of the latter. For example, a transitive verb “love” requires two dependents, one noun with the grammatical function subject and one with the function object. To demonstrate further, consider a sentence “a man sleeps”. The indefinite article “a” depends on the noun “man” which in turn depends on the verb “sleep”. The verb “sleep” depends on nothing, which is the root of the sentence [27]. This shows that dependency grammar formalism is closer to natural language. In other words, dependency representation is more similar to the human understanding of a language.

A major advantage of dependency grammar is its ability to deal with morphologically rich languages and have a relatively free word order. An additional practical motivation for a dependency-based approach is that the head-dependent relations provide an approximation to the semantic relationship between predicates and their arguments that makes them directly useful for many applications such as co-reference resolution, question answering and information extraction [19].

2.4.1. Dependency Relation

The traditional linguistic notion of grammatical relation provides the basis for the binary relations that comprise the dependency structures. The arguments to these relations consist of a head and a dependent. In constituency or phrase structure grammar, the head word of a constituent is the central organizing word of a larger constituent (e.g., the primary noun in a noun phrase, or verb in a verb phrase). The remaining words in the constituent are either direct or indirect dependents of their head. In dependency-based

approaches, the head-dependent relationship is made explicit by directly linking heads to the words that are immediately dependent on them, bypassing the need for constituent structures [19].

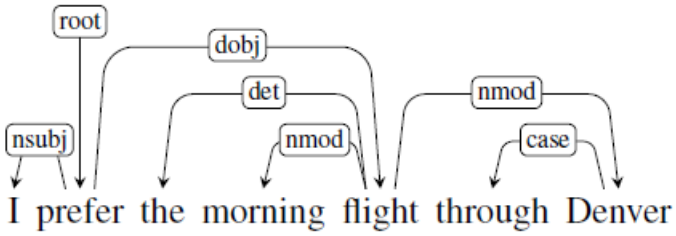

Figure 2.1. Typed Dependency Structure

In addition to specifying the head-dependent pairs, dependency grammars allow us to further classify the kinds of grammatical relations, or grammatical function, in terms of the role that the dependent plays with respect to its head such as subject, direct object and indirect object. In Figure 2.1 above, the relation NSUBJ and DOBJ identify the subject and direct object of the predicate *prefer*, while the relations NMOD, DET and CASE represent modifiers of the nouns *flight* and *Denver*. The directed arcs in the tree represent the head-dependent relations. The arc labels denote functional categories of the words [19].

2.4.2. Dependency Tree

There are two types of dependency tree representation: projective and non-projective. A dependency tree is called projective if its edges can be drawn in the plane above the words of the sentence, without any edges crossing each other. Figure 2.2 is an example of a projective tree.

In a non-projective dependency tree, the edges are crossing each other, as illustrated by Figure 2.3. Non-Projectivity is typically needed to handle long-distance dependencies and flexible word order mostly in complex sentences [19].

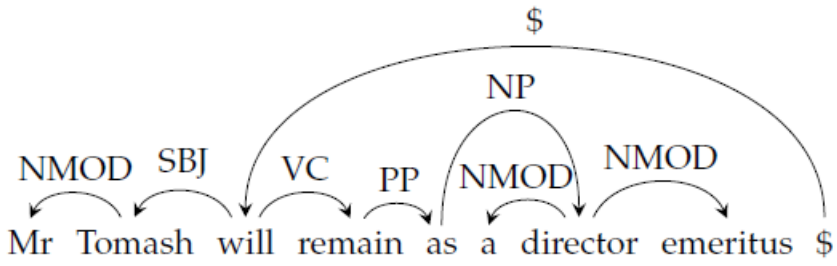

Figure 2.2: Projective dependency tree.

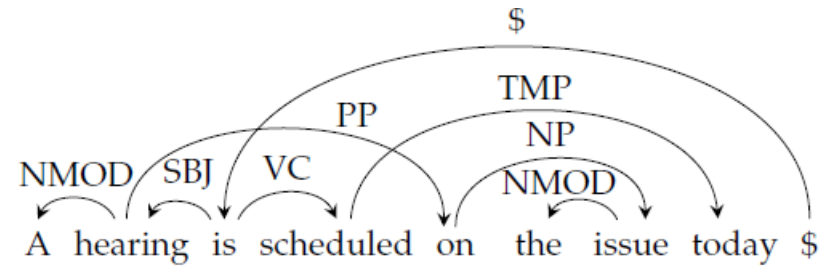

Figure 2.3: Non-projective dependency tree.

2.4.3. Dependency Parsing

In dependency parsing, the syntactic structure of a sentence is described in terms of the words in a sentence and an associated set of directed binary grammatical relations hold among the head and dependent words [19]. The parsing process is an important step of sentence preprocessing for many NLP applications such as grammar checking, segmentation, classification, sequence prediction problem, etc. [28]

Relations among the words are illustrated above the sentence with directed, labeled arcs from heads to dependents. We call this a typed dependency structure because the labels are drawn from a fixed inventory of grammatical relations. It also includes a root node that explicitly marks the root of the tree, the head of the entire structure [19].

The internal structure of the dependency parse consists of directed relations between lexical items in the sentence. These relationships directly encode important information that is often buried in the more complex phrase structure parses. For example in Figure 2.1, the arguments to the verb *prefer* are directly linked to it in the dependency structure, while their connection to the main verb is more distant in the phrase-structure tree. Similarly, *morning*

and *Denver*, modifiers of *flight*, are linked to it directly in the dependency structure [19]. These properties are depicted in a parse tree of phrase structure grammar below.

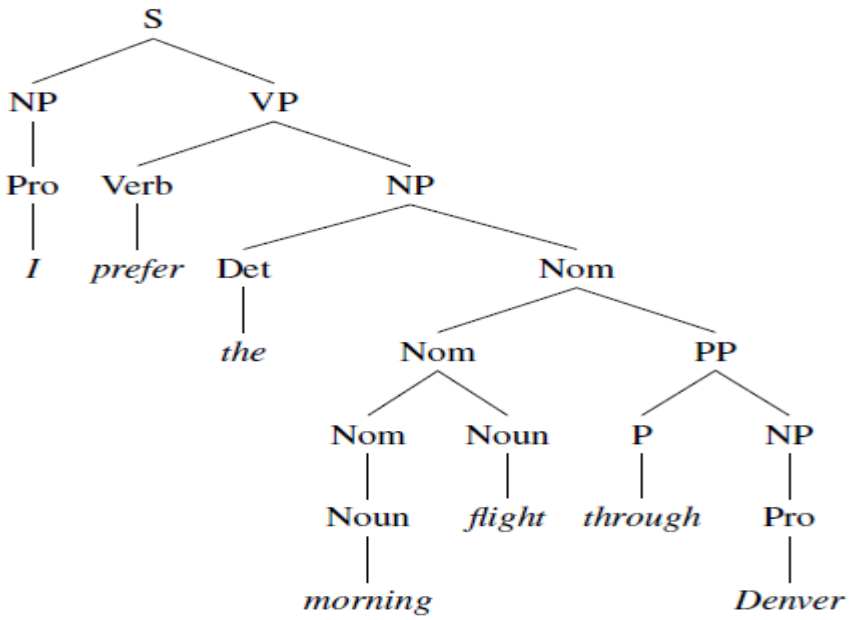

Figure 2.4: Parse Tree of Phrase Structure Grammar

The same is true for Amharic phrase structure grammar. In Figure 2.5 below, the subject "ካሳ" connects the verb "ሰምቷል" through intermediate components CS and VP. However, the two words directly linked each other in dependency grammar. Likewise, the noun "አስቴር" and the verb "እንደሰራች" relate through intermediate constituencies of SS and VP.

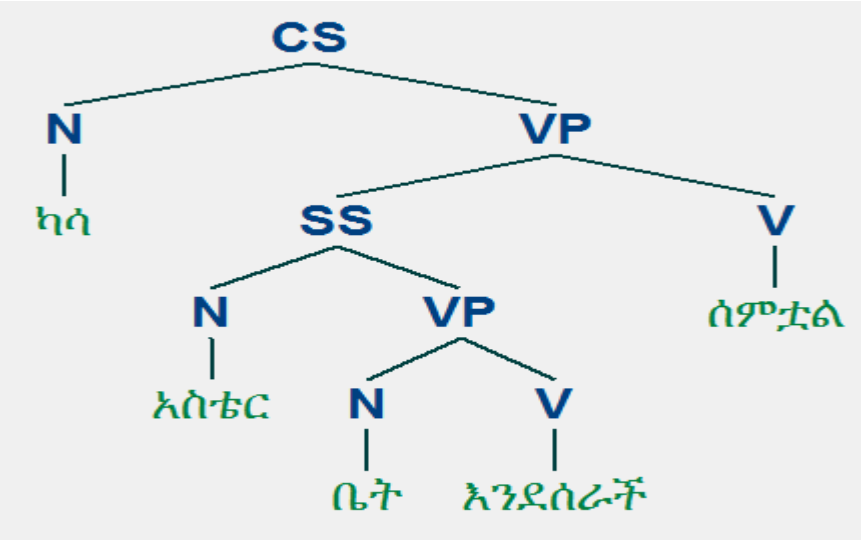

Figure 2.5: Parse Tree of Phrase Structure for Amharic Complex Sentence

2.4.4. Dependency Parsing Algorithm

Syntactic parsing is a process by which grammatical structures are identified and assigned to sentences within a text. Typically, parsing is an automated process and plays a significant role in the development of NLP tools.

An increased availability of linguistic resources has resulted in the development of dependency parsing platforms such as MSTParser [29] and MaltParser [30] [31] [32]. Statistical data-driven parsers learn how to syntactically analyze sentences from a set of examples. This data set is referred to as training data, which is in the form of a treebank. Probabilistic or statistical data-driven dependency parsers predict dependency trees after having learned patterns within the training data (treebank), which help to assess the probability of two words being part of a head-modifier relationship.

Two main approaches dominating data-driven dependency parsing. These are graph-based and transition-based. Graph-based dependency parsing involves the construction of a parse tree by predicting the Maximum Spanning Tree (MST) in the digraph for the input sentence [29]. In this digraph, each word corresponds to a vertex, and these vertices are all connected by directed edges (arcs). Based on frequency counts in the training data (treebank), each arc in the graph is assigned a score at the learning or training stage. Making a common assumption of arc factorization, the score of the graph is the sum of all the arc scores (weights). The challenge of the parser is to find the highest-scoring tree, that is, a subgraph including all vertices and only the minimum number of arcs to be connected, the MST, when choosing from amongst the proposed candidates. MSTParser [29] is a graph-based parser.

On the other hand, in transition-based dependency parsing, the training phase consists of learning the correct parser action given the input string and the parse history. The parsing phase consists of parser actions as dictated by the learned model. These actions are based on a shift-reduce parser, which involves progressing through the input string and moving tokens onto a stack from a buffer (shift) or removing them once they have been fully processed (reduce). MaltParser [31] is an example of a transition-based parser.

In transition-based parsing, the parser moves from left to right through a sentence, making decisions as to which words will make up dependency pairs with the help of a classifier. The transition-based parsing algorithms use a buffer containing the sentence tokens in linear order, a stack onto which each token is pushed as part of the processing step and an arc list that contains the proposed head-modifier relations [32] [30]. In this approach, the parser looks to see what is on the top of the stack and appearing next in the buffer. Due to the fact that it does not look beyond the next item in the buffer nor does it undo any decisions it has already made, it is referred to as a greedy algorithm [19].

MaltParser

MaltParser is a freely available transition-based parsing model for research and educational purpose [30]. With MaltParser, we can induce a new parsing model as it is a language independent parsing tool. The parsing model is sometimes called shift-reduce as is reduces the problem of parsing a sentence to the problem of finding an optimal path through a transition system [33].

MaltParser supports a number of parsing and learning algorithms [30]. The parsing algorithms in MaltParser can be categorized into three families, Nivre's Algorithm, Covington's Algorithm, and Stack Algorithm [33]. A key component in transition-based parsing is the idea of configuration which comprises of a stack, an input buffer of words, or tokens, and a set of relations representing a dependency tree.

Since MaltParser is an inductive dependency parser, it uses learning algorithm to induce a parsing model. The learning problem of MaltParser is to induce a classifier for predicting the next transition given a feature representation in the form of training data that have dependency structure. MaltParser employs two built-in learning algorithm, LIBSVM and LIBLINEAR [34].

The learning type of LIBSVM is support vector machine (SVM) that learns by examining hundreds or thousands of data. As a learning system, LIBSVM

involves two steps: first, training a data set to obtain a module and second, using the module to predict information of a testing data set [35].

The other learning library, LIBLINEAR, utilizes various linear classifiers including SVMs [34]. LIBLINEAR is an open source library for large scale linear classification. It supports regression and linear SVM [36]. Similar to LIBSVM, LIBLINEAR follows two steps to induce a model: training and predicting.

When the performance of the two libraries is being compared, LIBLINEAR is very efficient than LIBSVM for training large-scale problems. A comparison is made between them on a corpus with more than 600,000 examples. The LIBLINEAR takes only several seconds to train a text classification problem, however, the LIBSVM would take several hours [35]. On the contrary, LIBSVM is more memory efficient than LIBLINEAR since it does not store weight vectors explicitly.

2.4.5. Dependency Treebank

A treebank is a corpus of text that has been annotated with syntactic information describing the grammatical structure of each sentences. One of the syntactic analysis is dependency analysis which is based on extracting sets of labelled relations between pairs of words in a sentence. Treebanks are not only valuable for linguistic research and corpus analysis, but they also provide training data on which statistical parsing models can be built [37].

The information available on a treebank normally contains tokens, index of the tokens in the sentence, lemma of the tokens, part-of-speech tag, morphological data, the index of the head and the description of that attachment or dependency label [38].

Treebanks vary according to different considerations that are taken during development, including type of syntactic representation and labelling schemes [38]. There are a number of syntactic representations or grammar formalisms to choose from while designing a treebank, which are based on varying linguistic theories and formalisms. For example, a phrase structure grammar

representation hierarchically denotes constituents and phrases within sentences, while a dependency grammar labels connections between words within a sentence according to their functional roles.

Treebank development requires labelling schemes that define how linguistic structures are represented and labelled in a treebank. They are often closely linked to the chosen syntactic representation or chosen formalism. They are also influenced greatly by specific linguistic phenomena of the language in question. In addition, there are various ways of assisting treebank development by leveraging from existing NLP resources such as POS taggers, morphological analyzers and existing corpora [38].

In his work, Nivre [26] discusses theoretical assumptions that lay foundation for dependency based analysis. These assumptions address issues including level of representation, nature of lexical element, nature of dependency types and criteria for identifying heads and dependents words in a sentence.

The first assumption is related to the layer of representation. The layer of representation in dependency treebank development can be either mono-stratal or multi-stratal. Mono-stratal representation describes the syntax dimensions of a sentence which are word order, agreement and syntactic valency [39] as it is depicted in Figure 2.6 below. Syntactic analysis requires both a grammar and a parser, the output of which is a representation of the sentence that reveals the structural dependency relationships between words of a sentence [1].

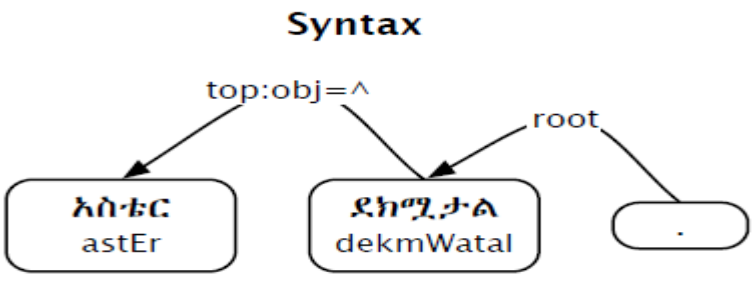

Figure 2.6: Mono-Stratal Representation

On the other hand, multi-stratal representation consists of both syntax and semantic dimensions. The semantic layer handles semantic valency [39]. That means, the semantic layer deals with analysis of meaning of words, phrases and sentences [1]. In Figure 2.7, for example the syntactic and semantic dimensions are illustrated on top and bottom of the words of the sentence respectively.

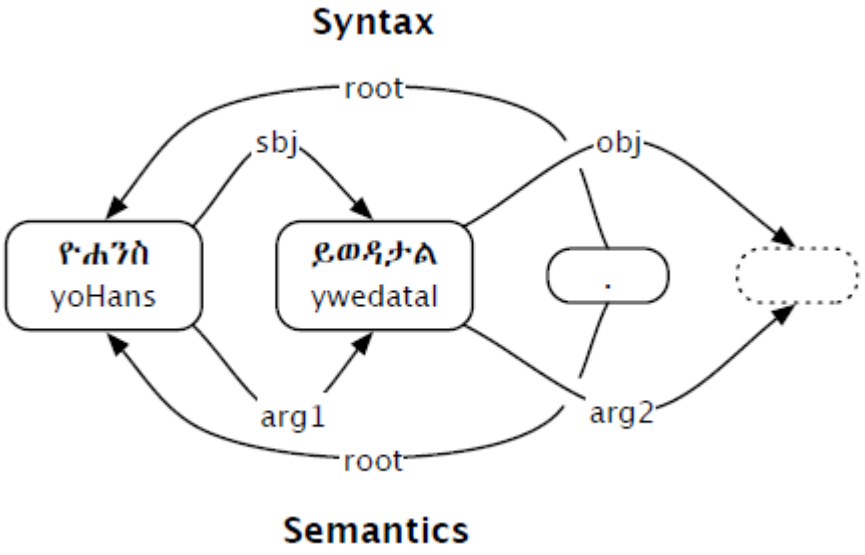

Figure 2.7: Multi-Stratal Representation

For these reasons, mono-stratal representation or syntactic analysis is selected as a design aspect. The nature of dependency type or arc labels also go with the syntactic representation rather than having a semantic role in the abstraction. Grammatical functions such as subject, object, verb, noun modifier, etc. are the choices which are made for the dependency types.

Once we decided the layer of syntactic representation, the next assumption is to identify the nature of lexical element in the abstraction. Is the lexical element morpheme? Word? Or multi-word unit? In this regard, Binyam [40] has identified three problems including consonant length or germination, how to represent compound words and how to segment content and functional words that coupled together.

In case of germination, a word can have two meanings based on the context in which the word is used. For instance, the word, የሚበላ can be read as

/jəmmibəlla/ “the one who is eating” as an active form or read as /jəmmibbəlla/ “the one who is being eaten” as a passive form. Similarly, the word አለ can be read as *alä* “he said” and can be read as *allä* “there is” depending on the situation. In such cases, Binyam [40] propose to segment the words as የ_እም_ይ_በል_ኣ or የ_እም_ይ_ተ_በል_ኣ depending on the context.

Compound words as identified by Binyam [40] can be written in three way. Some words incorporate space between them like ቡና ቤት /”bunna bet”/ “bar” and አየር መንገድ /”ajər məngəd”/ “airline”. Others are separated by hyphen such as ስነ-ስርዓት /”sɨnə-sɨrʔat”/ “procedures” and ስነ-ጥበብ /sɨnə-tʻɨbəb/ “art”. The third way to write a compound word as a single unit. For example, ቤተ ክርስቲያን /bətə kɨrsɨtijan/ “church” can be written as ቤተክርስቲያን /bətəkɨrsɨtijan/ “church” and መስ ሪያቤቶች /məsrija betoʧʧ/ “offices” can also be written as መስሪያቤቶች /məsrijabetoʧʧ/ “offices”. Therefore, compound words are better represented by hyphen or written them as a single word as incorporating space between them make word level analysis difficult.

In Amharic, a single orthographic word may contain functional words such as prepositions, conjunctions, axillaries, and so on. That is, a single word can be a phrase (ለሰው /ləsəw/ “ to human”), a clause (የሚገኙትና /jəmmigəɲɲutɨnna/“and those that are found/available), or even a sentence (አልመጣችም /almətʻtʻaʧʧɨm/ “She did not come.”) [40]. In such cases, we need to segment content and functional words. For instance, an orthographic word ላሸናፊነት /*laʃʃənnafinnət*/ “for a winning” incorporates a preposition ለ *lə* “to/for” a content word አሸናፊነት /*aʃʃənnafinnət*/ “winning” and will be segmented into ለ_አሸናፊነት according to the research work of Binyam.

Since Amharic is morphological reach language [41] and a single verb lexeme can appear in more than 100,000 word forms [39], it is impractical to list all possible word forms in the lexicon. For example, the word “ሰብር” /“sbr”/ “break” can have different forms such as ሰበረ/“sbre”/, ሰብር /“sebr”/, ስብር /“sbr”/, አሰብር /“asebr”/, ተሰበር /“tesber”/, ተሰባር /“tesbar”/, ሰባበር /“sebaber”/, etc… We can even list same word with respect to person, gender and number. The nature of lexical element, thus, will be morphemes as well as words.

Another important assumption of dependency treebank analysis is to distinguishing head and dependent words in a given sentence. To do so, there are some criteria that have been proposed for identifying a syntactic relation between a head H and a dependent D in a construction C [26] [31] [42].

- H determines the syntactic category of C and can often replace C.
- H is obligatory; D may be optional.
- H selects D and determines whether D is obligatory or optional.
- The form of D depends on H (agreement or government).
- The linear position of D is specified with reference to H.

Even if these criteria describe how to identify head and dependent words in syntactic dependencies, it becomes clear that not all heads can satisfy all these criteria in all constructions [43].

To elaborate these principles further, we need to first explain the types of syntactic constructions from the perspective of dependency relations. Syntactic constructions of sentences are either endocentric or exocentric, or both depending on the distribution and relation between words [26] [42]. When we say a construction, it is either a phrase or a sentence.

If a syntactic construction consists of an obligatory word (head) linking with one or more optional words (dependents) that subordinate or attribute to the meaning of the head, the construction is called endocentric. In endocentric construction, the head is functionally equivalent to the whole construction. Such cases are usually exhibited in noun, verb and adjective phrases since the dependents are subordinate to the head noun, verb and adjective in each phrases respectively [26] [42]. In such constructions, the head word is modified by the dependent word.

“ካሳ የሳር ቤት አለው”

/”*Kasa yesar bet alew*”/ (2.1)

“Kasa has a hut”

For example, in the sentence 2.1, “የሳር ቤት” /”*yesar bet*”/ is a noun phrase with “ቤት” /”*bet*”/ is the head of the phrase and “የሳር” /”*yesar*”/ is a dependent word that attributes to the head word. Additionally, the dependent word modifies the head word and told us what kind of house it is. As a result, if we remove the head word “ቤት “ /”*bet*”/from the whole construction “የሳር ቤት “/”*yesar bet*”/, it loses its basic meaning intended to convey. The linear position of the dependent word is also determined with reference to the head word as we cannot say “ቤት የሳር“/” *bet yesar*”/.

On the other hand, the opposite of endocentric construction is exocentric construction where the head does not function like the complete construction [26] [42]. The following prepositional phrase illustrate this.

“እስከ ትልቁ ትምህርት ቤት”

/”*eske tiliku tmhrit bet*”/ (2.2)

“Until the big school”

Here the head of the phrase is “እስከ/”*eske*”/ [44] that functions differently from prepositional phrase. The remaining part of the phrase, conversely, demonstrates endocentric relationship. The word ”ትልቁ”/”*tiliku*”/ is dependent on and modifies ”ትምህርት ቤት”/”*tmhrit bet*”/. Similar to sentence 2.1 above; if we remove the word ”ትልቁ”/”*tiliku*”/ from the phrase, we are going to lose the intended meaning.

Mostly exocentric relationship is exhibited among subject, object and verb of a sentence. The verb of the sentence is the head of both the subject as well as the object of the sentence [31]. The sentence (2.1) exemplifies this property. The subject, object and verb of the sentence are “ካሳ”/”*Kasa*”/, “የሳር ቤት” /”*yesar bet*”/, and “አለው”/”*alew*”/ respectively. Therefore, the exocentric relation exists between “አለው”/”*alew*”/ and “ካሳ”/”*Kasa*”/; and between “አለው”/”*alew*”/ and

“የሳር ቤት” /”*yesar bet*”/. The words complement each other to make up the big sentence.

2.4.6. Treebank Data Format

Parsing systems such as MaltParser [30] and MSTParser [29] are language-independent systems that allow users to build parsing models using their own choice of treebank. Treebanks provide a rich representation of linguistic details of a language and are a solid platform for linguistic analysis. In corpus linguistics, linguists use treebanks to test linguistic theories and study syntactic structures. Treebanks are also invaluable resources for the development of many NLP applications, specifically data-driven parsers. The information available to the parser normally contains: token, index of token, lemma, part-of-speech tag, (optional morphological data), the index of the head (where it attaches) and the description of that attachment (dependency label). The result of parsing can be utilized for grammar checking.

In order to train and test a parsing system, the dependency tree structure needs to be encoded in a suitable format. There are different formats used by various researches including XML format where a sentence spans 3-4 lines of meta-data [30].

```
<sentence>
<word form="This" postag="DT" head="2" deprel="SBJ"/>
<word form="is" postag="VBZ" head="0" deprel="ROOT"/>
<word form="an" postag="DT" head="4" deprel="DET"/>
<word form="old" postag="JJ" head="4" deprel="NMOD"/>
<word form="story" postag="NN" head="2" deprel="PRD"/>
<word form="." postag="." head="2" deprel="P"/>
</sentence>
<sentence>
<word form="So" postag="RB" head="2" deprel="PRD"/>
<word form="is" postag="VBZ" head="0" deprel="ROOT"/>
```

```
<word form="this" postag="DT" head="2" deprel="SBJ"/>
<word form="." postag="." head="2" deprel="P"/>
</sentence
```

Figure 2.8: XML Format

Alternatively, a corpus can also be represented by a column-based format where each token in a sentence is on a new line, each of which containing part-of-speech information, dependency label and attachment information [28]. A sentence consists of one or more tokens each of which are represented on one line, encompasses ten fields separated by a TAB. This representation is called either CoNLL-X [28] or CoNLL-U [45] formats. A CoNLL-U format is a revised version of CoNLL-X format. In CoNLL-U format, the fields dependency relation and miscellaneous replace the obsolete fields projective head and dependency relation to the projective head of the CoNLL-X format [45]. For example the sentence 'They buy and sell books' is represented by CoNLL-U format below:

1	They	they	PRON	PRP	Case=Nom|Number=Plur	2	nsubj	2:nsubj|4:nsubj
2	buy	buy	VERB	VBP	Number=Plur|Person=3|Tense=Pres	0	root	0:root
3	and	and	CONJ	CC	_	4	cc	4:cc
4	sell	sell	VERB	VBP	Number=Plur|Person=3|Tense=Pres	2	conj	0:root|2:conj
5	books	book	NOUN	NNS	Number=Plur	2	obj	2:obj|4:obj
6	.	.	PUNCT	.	_	2	punct	2:punct

Figure 2.9: Column-based format.

2.4.7. Universal Dependency

Universal Dependencies (UD) [46] is an initiative to develop cross-linguistically consistent treebank annotation for many languages, with the goal of facilitating multilingual parser development, cross-lingual learning, and parsing research from a language typology perspective. The notion of UD initiative is to provide a universal inventory of categories and guidelines to facilitate consistent annotation of similar constructions across languages, while allowing language-specific extensions when necessary. The UD project has volunteer researchers from all over the world collaborating to make the

largest multilingual collection of dependency treebanks. Besides providing more treebanks for UD and expanding its reach to more languages, UD contributors actively discuss data representation as well as morphological and syntactic annotation. UD annotation standards are revised with each new iteration in order to achieve treebanks that are more robust, successful, and balanced in terms of the tradeoff between computational tractability and linguistic correctness.

At the time of this writing, the most recent UD release was version 2.0 with more than 100 treebanks and 60 languages [46]. This version is a milestone because of its major leap from the last version, containing a number of radical changes and improvements in both annotation schemes and general data organization to accommodate a more diverse set of studies.

Currently, the data format being used in Universal Dependency is CoNLL-U format [47] [48] [49]. As it is described in section 2.4.6 above, the CoNLL-U formant has ten fields to represent a sentence. A sentence is broken into individual words (tokens), each of which then analyzed based on the standards and guidelines suggested by UD.

While preparing treebanks, tokens are going to be tagged with universal POS tagsets and optionally with language specific POS tagsets. If language specific POS tags are used, the treebank-specific documentation should define a mapping from specific to universal POS tagset [45].

Similarly, tokens are annotated with morphological features such as gender, person, case, definiteness, etc. from universal feature inventory [45].

The syntactic annotation of words describe the dependency relationship of head and dependent words. The dependency relation value should be a universal dependency relation or a language-specific subtype of such a relation defined in the language-specific documentation [45].

2.5. Amharic Language

Amharic /አማርኛ/ *amargna,* is a member of the Semitic branch of the Afro-Asiatic language family. It is spoken by over twenty five millions people and is the working language of the government of Ethiopia. The language has its alphabet, /ፊደል/ fidäl, inherited from the Geez (Ethiopic) language. Geez is an ancient South Semitic language which now serves only as the liturgical language of the Ethiopian Orthodox Tewahedo Church. Fidäl is a syllabary writing system where the consonants and vowels co-exist within each graphic symbol. Fidäl is written from left to right. The writing system consists of 33 consonants, each having seven orders or shapes depending on the vowel with which a given consonant is combined [13].

In Amharic language, a unit of words can be morpheme, root and stem. The smallest minimal unit of word is morpheme, which can either be free or bound. A free morpheme can stand as a word by its own whereas a bound morpheme cannot. An Amharic root is a sequence of consonants and is the citation form of a word. It is the basis for the derivation of verbs. On the other hand, a stem is a consonant or consonant-vowel sequence which can be free or bound where a free stem can stand as a word on its own whereas a bound stem has a bound morpheme affixed to it. In addition, it is part of a word that never changes even when morphologically inflected [50].

2.5.1. Amharic Part of Speech

The dictionary definition of part of speech is a category to which a word is assigned in accordance with its meaning, form or syntactic function. The common part of speech of Amharic languages are noun, pronoun, adjective, verb, adverb, preposition, conjunction and interjection [44].

Nouns

Amharic nouns are words used to identify objects such as a person, place, things and ideas [44]. Amharic nouns have the possibility to have up to two prefix and four suffixes for each stem. Therefore, the suffixes of a noun stem are gender, number, case and definite markers which are depicted below [50].

Table 2.1: Amharic noun marked for gender, number and case

	Gender marker		Number marker		Case marker	
Word	**Masculine**	**Feminine**	**Singular**	**Plural**	**Nominative**	**Accusative**
“ልጅ”/ ”lj”	“ልጅ”/ ”lj”	“ልጅ-ኢት”/ ”lj-‘it”	“ልጅ”/ ”lj”	“ልጅ-አች”/ ”lj-‘oc”	“ልጅ”/ ”lj”	“ልጅ-ን”/ ”lj-n”
“በግ”/ ”beg”	“በግ”/ ”beg”	“በግ-ኢት”/ ”beg-it”	“በግ”/ ”beg”	“በግ-አች”/ ”beg-‘oc”	“በግ”/ ”beg”	በግ-ን/ ”beg-n”

Pronoun

Words that can be used in place of noun are called pronouns. These include personal pronoun, possessive pronoun, interrogative pronoun, demonstrative pronoun and the likes [50] [51] [44].

Personal pronouns comprises a set of pronouns that is associated primarily with a particular person or thing. They are classified based on person, gender, and number [50] [51] [44]. The following table exhibits personal pronouns adapted from the research work of Tigist et al, [50] .

Table 2.2: Amharic Personal Pronouns

Person	Gender	Singular	Plural
1st		“እኔ”/”’nE”	“እኛ”/”’Na”
2nd	Masculine	“አንተ”/”’ante”	“እናንተ”/”’nante”
	Feminine	“አንቺ”/”’anci”	
3rd	Maculine	“እሱ”/”’su”	“እነሱ”/”’nesu”
	Feminine	“እሷ”/”’sWa”	
	Polite	“እርስዎ”/”’rswo”, “አንቱ”/”’antu”	

Demonstrative pronouns show the place where an object is referenced. The place where the object can be near or far from the person’s standing point, who is referring the object [50] [51] [44].

Table 2.3: Amharic Demonstrative Pronouns

Number	Gender	Near	Far
Singular	Masculine	“ይህ” /”yih”/	“ያ” /”ya”/
	Feminine	“ይቺ” /”yci”/ “ይህች” /”yhc”/	“ያቺ” /”yaci”/
Plural		“እነዚህ” /”enezih”/	“እነዚያ” /”eneziya”/

Another type of pronouns are interrogative pronouns which are used for asking questions such as who, whom, whose, which, and what [50] [51] [44].

Table 2.4: Amharic Interrogative Pronouns

	For person	**For things**	**For place**	**For time**	**For condition**	**For reasoning**
	“ማን”/ ”man”	“ምን”/ ”mn”	“የት”/ ”yet”	“መቼ”/ ”mecE”	“እንዴት”/ ””ndEt”	“ለምን”/”lemn ”
Singular	“ማንን”/ ”mann”	“ምንን”/ ”mnn”	“ወዴት”/ ”wedEt”			
	“የማን”/ ”yeman”	“የምን”/ ”yemn”	“ከየት”/ ”keyet”			
	“እነ-ማን”/ ””ne-man”					
Plural	“እነ-ማንን”/ ””ne-mann”					
	“እነ-የማን”/ ””ne-yeman”					
Negation	“ማንም”/ ”manm”	“ምንም”/ ”mnm”	“የትም”/ ”yetm”			
	“የማንም”/ ”yemanm”					

Possessive pronouns replace nouns or noun phrases in a sentence and show ownership. They are formed by prefixing personal pronouns with an Amharic possessive marker “የ” /”ye” [50] [51] [44]. For example, “የእኔ” /”yene”/, “የአንተ”

/"yeante"/, "የእሱ" /"yesu"/, etc are singular and "የእኛ" /"yeNa"/, "የእናንተ" /"yenante"/, etc are plural possessive pronouns respectively.

Verbs

Verbs are words that describe about an action, a state or condition, or occurrence of an event. Amharic verbs are very complex consisting of a stem and up to four prefixes and four suffixes. They are inflected for person, gender, number, and time with the basic verb form being third person masculine singular. Verbs in passive voice are marked by suffixes that depends on person and number.

Adjectives

An adjective is a word that describes or modifies nouns or pronouns and appears before them. An adjective specifies the properties or attributes of a noun or pronoun referent. The attributes of the referent can be shape, behavior, color, etc. An adjective is inflected for gender, number and case in a similar fashion to noun [50]. For example, in a phrase "ነጭ ዶሮ" /"nec doro"/ "white hen", the word "ነጭ" /"nec"/ is an adjective that give extra information about the word "ዶሮ" /"doro"/.

Adverbs

Adverbs are word that modify verbs in terms of time, place, circumstances, etc. For example, the word "በፍጥነት"/"befitnet"/"quickly" modifies the verb "መጣ"/"meta"/"came" in a sentence "ልጁ በፍጥነት"/"liju befitnet meta"/"The came quickly" [50].

Conjunctions

Conjunction is a connecting word that is used to link words, phrases, clauses, sentences, etc. Example:"እና" /"ena"/, "ስለሆነም"/"silehonem"/ "ነገር ግን"/"negergin"/, etc [50].

Prepositions

Prepositions are words that are usually used before nouns to show their relation to another part of a clause and they are limited in number. For example, “ለ”/“le”/, “ከ”/”ke”/, “እንደ”/”ende”/, etc. are prepositions [50].

2.5.2. Amharic Morphology

Amharic is a morphologically rich languages. Like other Semitic languages, it exhibits a root-pattern morphological phenomenon. A root is a set of consonants (also called radicals) which has a basic lexical meaning. A pattern consists of a set of vowels which are inserted among the consonants of a root to form a stem.

In addition to this morphological feature, applying different affixes on Amharic part of speech (noun, verb, adjective, adverb, preposition, pronoun and interjection) create inflectional and derivational word forms.

Derivational morphology is a process of affixation from which words are formed with the same or different part of speech category. But it usually changes the part of speech category.

Verbs

Amharic verbs are derived from nouns as መረዘ /mereze/ ‘he poisoned’ is derived from the noun መርዝ /merze/ ‘poison’. They are also derived from root consonants by inserting vowels patterns. For example, when we insert the pattern ኧ-ኧ (a-a) among the radicals of an Amharic root ሰ-በ-ር /sbr/ means ’break’, we get the stem ሰበር/sabbar-/. Attaching the suffix ኧ (a) on the stem gives ሰበረ /sabbara/ ’he broke’ which is the first form of the verb [41].

Nouns

Amharic nouns are modified by adjectives and serve as subject or object of a sentence. Nouns are derived from other nouns, adjectives, roots, stems, and the infinitive form of a verb by affixation. For example a noun አግረኛ

/Igregna/'pedestrian' is derived from another noun እግር /Igre/'leg' and a morpheme ኛ /agna/ [41].

Adjectives

Adjectives in Amharic include all words that modify nouns and are derived from nouns, stems or verbal roots by adding a suffix ɐnd compound words [41]. Adjectives are derived from nouns, stems or verbal roots by adding a suffix and by intercalation. The suffixes -ኣም/-am/, -ኛ/-anna/, - አዊ/-awi/, and -ኣማ /-ama/ are used in the derivation of adjectives from nouns. For example, it is possible to derive ሀብታም /*habtam*/ (rich, wealthy), ሀይለኛ/haylana/ (powerful, mighty), ዘመናዊ /zamanawi/ (modern) and ድንጋያማ/dIngayama/ (stony) from the noun ሀብት /habt/ (riches, wealth) , ሀይል /hayl/ (power, force) , ዘመን /zaman/ (period, epoch) and ድንጋይ /dInIgay/ (stone) respectively.

The other way of forming a word is called inflectional morphology which adds affixes to mark syntactical functions of a word without changing its original part of speech category. The affixes are prefixes, infixes or suffixes.

Verbs

Verbs are inflected for person (first, second, third), gender (masculine, feminine), number (singular, plural), aspect (perfect, imperfect), tense, and mood (declarative, interrogative, negative and imperative).

The prefix and the suffix for Amharic verbs are placed in four slots before and after the stem. The prefixes are preposition (prep) or conjunction (conj), relative (rel), negative (neg) and subject (sbj) markers that are positioned in first, second, third and fourth sloths respectively. The other four slots after the stem contain subject (sbj), object (obj) or definite (def), negative (neg) or auxiliary (aux) or accusative (acc), and conjunction (conj) markers [12].

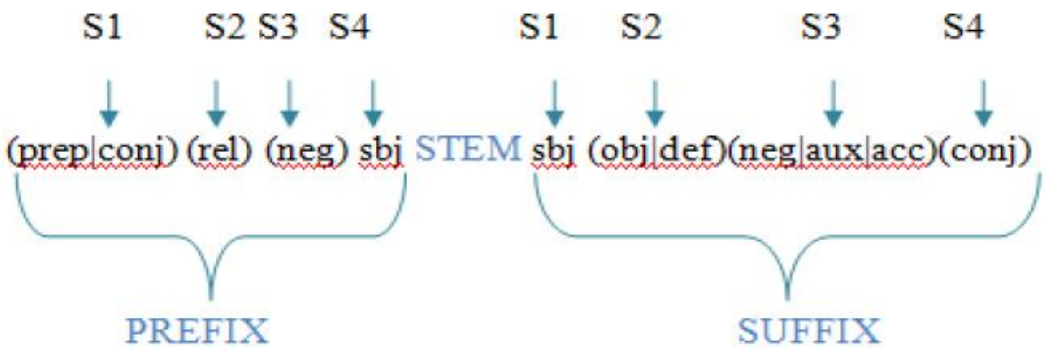

Figure 2.10: Prefix and Suffix Arrangement of Amharic Verbs

Nouns

Amharic nouns are inflected for number (singular/plural), definiteness (definite/indefinite), case (nominative/accusative/genitive/dative), gender (feminine, neuter, masculine), etc. [13]. The suffixes –ኦች /-occ/, -ኦን /-an/ and -ኦት /-at/ are used to inflect nouns for number. For example, ቤቶች/betocc/ houses, መምህራን /mamhIran/ teachers and ህፃናት /hIsanat/ babies are plural forms of ቤት /bet/ house, መምህር /mamhIr/ teacher and ህፃን /hIsan/ baby [41].

Adjectives

Adjectives inflect for case, number, definiteness and gender in a similar fashion to nouns. For example, a singular form of an Amharic adjective ትልቅ /tIlk'/ 'big' can be inflected for number to become ትላልቅ /tilalik/ which is in a plural form [41].

2.5.3. Amharic Sentences

In general, a sentence is a set of words that is complete in itself, typically containing a subject and predicate, conveying a statement, question, exclamation, or command. A sentence is complete in itself means that it expresses a complete thought or idea as it answers all or some of such questions 'Who?' 'What?' 'Where?' 'How?' and 'When?'

The subject of a sentence is the person or thing doing the action or being described. A predicate on the other hand is the rest of the sentence which tells us what the subject does or is and encompasses the verb and object of

the sentence. The object of a sentence is a person or thing that is affected by the action of a verb or involved in the result of an action, or a noun, pronoun or noun phrase that represents that person or thing.

The same is true for an Amharic sentence. It is a set of words structured together to expresses a complete ideas and thoughts [51] [44]. For example:

1. ትልቁ ልጅ /tiliqu lij/ 'The big boy'
2. በሀያ ብር /behaya birr/ 'with twenty Birr'
3. አራት እንቁላል ገዛ /arat enqulal geza/ 'bought four eggs'

If we consider all the three item separately, each of them do not give a complete idea. But all the three make up one complete sentence.

ትልቁ ልጅ በሀያ ብር አራት እንቁላል ገዛ

/tiliqu lij behaya birr arat enqulal geza/

'The big boy bought four eggs with twenty Birr'

The syntactic structure of an Amharic sentence exhibits subject object verb (SOV) structure [12]. Modifiers precede the words or phrases they modify in such structure.

An Amharic sentence is either simple or complex based on the number of constructs involved in the sentence [51] [44].

Simple Sentence

A simple sentence is a sentence with only one verb phrase. The above sentence is a simple one since it has only one verb phrase which is ገዛ / geza/ 'bought'.

There are four types of simple sentences: declarative, interrogative, negative and imperative sentences.

1. Declarative sentence is used to convey ideas, feelings and opinions the speaker wants to express or declare about situations, happenings, activities, actions, etc., that could be physical, mental, real or imaginary.

   Examples:

   ስጋው ጮማ ነው /*"sigaw choma new"*/ 'The meat is fatty"

   አስቴር አስተማሪ ሆነች /*Aster astemary honech/* 'Aster became a teacher'

   ካሳ ወደ ገበያ ሄደ /*Kasa wede gebeya hede/* 'Kasa has gone to the market'

   አስቴር ተሹማለች /*Aster teshumalech/* 'Aster has been appointed'

2. Interrogative sentence is a form of sentence that is used to ask questions about the subject, the complement, or the action the verb specifies. It is usually constructed with pronouns: 'ማን' /man/ 'who', 'ምን' /min/ 'what', 'የት' /yet/ 'where', 'እንዴት' /endet/ 'how' 'መቼ'/meche/ 'when', and 'ለምን' /lemin/ 'why'.

   Example**:** ካሳ ምን አደረገ? /*Kasa min aderege?/*'What did Kasa do?'

3. Negative sentence is a negative form of a declarative statement.

   Example: ካሳ ምሳ አልበላም /*Kasa misa alibelam/* 'Kassa did not eat lunch'.

4. Imperative sentence gives a command or makes a request. The subject is a second person pronoun and implied by the suffix on the verb.

   Example: መስኮቱን ክፈተው /*Meskotun Kifetew/* 'Open the window'

   ልብስ እጠቢ /*libse etebi/* 'Clean the clothes'

   As it is describe in Figure 2.10 on section 2.5.2, an Amharic verb can have four slots before and four slots after the stem. The four slots after stem are suffixes one of which is subject marker of the verb in terms of gender, number, person and definiteness. The stem of the verb ክፈተው /*Kifetew/* is ክፈተ/kefete/. The subject features of ክፈተው /*Kifetew/* are, therefore, third person singular masculine as it is analyzed by HornMorpho [52]. Similarly, the subject features of እጠቢ /*etebi/*are third person singular feminine.

Complex Sentence

A complex sentence on the other hand consists of one or more noun phrases, adjective phrases and verb phrases combining in such a way that simple-complex, complex-simple and complex-complex manner.

- Complex Noun Phrase (NP): contains an embedded sentence that serves as a complement or gives more explanation to the head of the NP.

  For example: [[የካሳ ቤት የመስራት]ዓላማ]

  /Yekasa bet yemesirat alama/

  The goal of Kasa to build a house

  This is a NP whose head is the noun 'ዓላማ' /alama/ 'goal'. The complement sentence 'የካሳ ቤት የመስራት' /Yekasa bet yemesirat/ 'of Kasa to build a house' together with the head word forms a complex NP.

  The embedded sentence can also give more explanation about the head word.

  For example: [ካሳ የሸጠው [የብራና መጽሀፍ]]

  /Kasa yeshetew yebirana metshaf/

  'The Birana book that Kasa has sold'

  The head word is 'መጽሀፍ' /metshaf/ 'book' which forms a simple NP with the word 'የብራና' /yebirana/ 'The Birana'. The dependent clause 'ካሳ የሸጠው' /Kasa yeshetew/ 'Kasa has sold' combines with the simple NP to form a complex NP. The morpheme 'የ' /ye/ 'that' in the dependent clause indicates that the clause is subordinate and cannot stand by itself.

- Complex Verb Phrase (VP): also contain at least one embedded sentence or more than one verb. The embedded sentence a dependent clauses that can be complement or modifier the head word.

For example: [[ካሳ አስቴር ወደ ጎንደር እንደ ሄደች] አውቆዋል

/Kasa Aster wede Gonder ende hedech awkoal/

‘Kasa knew that Aster went to Gonder’

The embedded sentence ካሳ አስቴር ወደ ጎንደር እንደ ሄደች /Kasa Aster wede Gonder ende hedech / is a combination of a sentence አስቴር ወደ ጎንደር ሄደች / Aster went to Gonder/ ‘Aster went to Gonder’ and the preposition ‘እንደ’ /ende/ ‘as’. The existence of ‘እንደ’ /ende/ ‘as’ makes the embedded sentence dependent clause.

2.5.3. Agreement in Amharic Sentences

The dictionary definition of agreement is when the words in a sentence match each other according to the rules of the grammar. In other words, it is the accordance of a verb with its subject and object.

The subject and object agree with the subject and the object features of the verb in terms of number, gender and person. An adjective-noun agreement is with regard to number and gender. An adverb also agree with the verb in terms of time. For example: ‘እሱ እሷን መታት’ /esu esuan metat/ ‘He kicks her’

The subject, object and verb of the sentence are ‘እሱ’ /esu/ ‘እሷን’ /esuan/ and ‘መታት’ /metat/ respectively. The subject is singular in number and third person masculine. The object is also singular, but third person feminine. The subject feature of the verb is third person singular masculine, whereas the object features of the verb is third person singular feminine. Therefore, the subject and the object agree with the verb in the above example.

The common grammatical agreements in Amharic sentences are subject-verb, object-verb, adjective-noun and adverb-verb agreements [50].

Subject and Verb Agreement

A subject is the part of a sentence or clause that commonly indicates what it is about, or who or what performs the action[1]. The subject is typically a noun, a noun phrase, or a pronoun. In Amharic, a subject agrees with the verb in person, gender, and number and appears at the beginning of a sentence [50].

In a sentence, "ካሳ ወደ ገበያ ሄደ" /"*Kasa wede gebeya hede*"/ "Kasa has gone to the market"; the subject "ካሳ"/"*Kasa*"/is third person singular and masculine. Similarly, the verb "ሄደ"/"*hede*"/"gone" is third person singular and masculine. This shows that the morphological properties of both the subject and verb agree. On the contrary, if one of the morphological properties do not agree, the subject and the verb are said to be disagree. For example the subject and verb of the sentence "ካሳ ወደ ገበያ ሄደች" /" Kasa *wede gebeya hedech*"/ "Kasa has gone to the market" are "ካሳ"/"Kasa"/and "ሄደች"/"hedech"/"has gone" respectively. The subject is third person singular and masculine; whereas the verb is third person singular and feminine.

Object and Verb Agreement

An object is either a noun, noun phrase, or a pronoun that is affected by the action of the verb[2]. Similar to subject verb agreement, the object of a sentence should agree with the verb in person, number, and gender [50] [44].For example, in a sentence "ካሳ ቤቱን ሸጠው" /Kasa betun shetew/ "Kasa has sold the house"; the object "ቤቱን"/ betun / "the house" is third person singular masculine agree with the verb "ሸጠው"/shetew/ "has sold" which is also third person singular masculine. If we change the verb "ሸጠው"/shetew/to "ሸጠችው" /shetechew/in the above example, the verb's gender changed to feminine which consequently does not agree with the object.

[1] The definition is taken from: https://www.thoughtco.com/subject-grammar-1692150

[2] The definition is taken from: https://www.thoughtco.com/subject-grammar-1692150

Adjective and Noun Agreement

Adjectives in Amharic include all words that modify nouns and can be modified by the word "በጣም"/batam/ "very, greatly" [41]. Since an adjective is inflected for number and gender [41] [50] [44], it should agree with the noun it modifies with respect to number and gender. To elaborate further consider a phrase "ሀይለኛ"/haylegna/ "powerful" "ንፋስ"/nifas/ "wind". The adjective "ሀይለኛ"/haylegna/ "powerful" modifies the noun "ንፋስ"/nifas/ "wind" and both agree in number and gender, as they are singular masculine.

Adverb and Verb Agreement

Adverbs in Amharic modifies a verb or describes the action of a verb in terms of time mostly [44]. Since an Amharic adverb is not inflicted [41], the agreement between adverb and verb is concerned with time. For example, in a sentence "ካሳ ትናንት ታሞ ነበር" /kasa tinant tamo neber/"Kasa was sick yesterday"; the adverb "ትናንት " /tinant/"yesterday" and the verb "ታሞ ነበር" /tamo neber/"was sick" agree each other in time and both refer past action. If we replace the adverb "ትናንት " /tinant/"yesterday" with "ነገ"/nege/"tomorrow", the verb does not agree with the new adverb which refers future action.

2.5.4. Dependency Grammar for Amharic

Michael Gasser [39] has described dependency grammar for Amharic on how to handle null subjects and objects; subject and object agreement; and relative clauses. Null subjects and objects are handled by introducing empty node in the tree. For example, the following sentence is depicted in figure 2.7 below.

ዮሃንስ ይወዳታል

Yohansywedatal

'Yohannis likes her.'

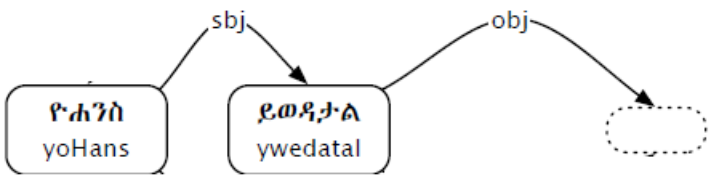

Figure 2.11: Empty node in Amharic Sentence

In Figure 2.11, the transitive verb "ይወዳታል" *ywedatal* has no explicit object so it is linked to an empty node.

An Amharic transitive verb agrees with both the subject and object features of a sentence. The subject agrees with the subject features of the verb and the object agrees with the object features of the verb. Figure 2.8 demonstrates.

አስቴር ዮሐንስን ትወደዋለች

Aster Yohannisn twedewalec

Aster likes Yohannis

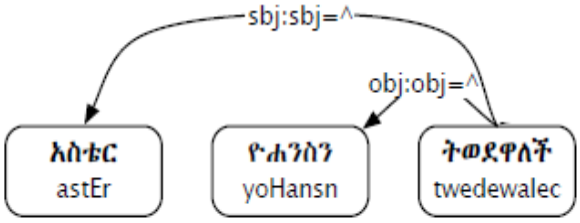

Figure 2.12: Simple subject-verb and object-verb agreement in Amharic.

Relative verbs in a relative clause are treated as the heads of their noun phrase. So verb's subject, object or indirect object features must agree with that noun. For example,

አስቴር የምትጠላው ወንድልጅ ታመመ

Aster yemtTelaw wendlj tameme

The boy that Aster hates got sick

In the sentence above, the object feature of the relative verb *yemttelaw*' that she hates' agrees with the modified noun *wendlj*' boy'; both are third person singular masculine. The subject feature of the main verb *tameme*' he-got-sick'

agrees with the object feature of the relative verb; both are third person singular masculine. The subject feature of the relative verb agrees with its subject *Aster*; both are third person singular feminine. This is illustrated in Figure 2.13.

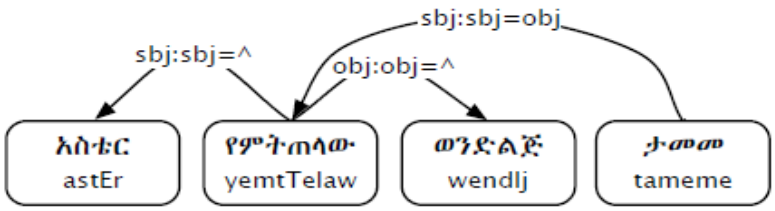

Figure 2.13: A Sentence with Relative Clause

2.6. Summary

In this chapter we have reviewed important concepts and ideas that lay the foundation for the research work. We have discussed what grammar checking is and its implementation approaches along with their pros and cons. We have also discussed constituency and dependency grammar formalisms with which the knowledge of a language is represented. In dependency grammar formalism, we have explained different aspects of the formalism including dependency relation, dependency tree, dependency parsing and algorithm, dependency treebank, dependency data format and universal dependency. Finally, we have seen the grammars of Amharic language which encompasses its morphology, different types of sentence both simple and complex, common grammatical agreements and dependency grammars of the language.

Chapter 3: Related Work

3.1. Introduction

Even if Amharic is the working language of the Federal Democratic Republic of Ethiopia [13] and the second most speaking Semitic language, it is under-resourced [39]. There are researches conducted on grammar checking with different approaches for different languages. In this section, we have reviewed related works on grammar checkers. These are Swedish Grammar Checking, Dependency-Based Rules for Grammar Checking, and Amharic Grammar Checking.

3.2. Swedish Grammar Checking

A hybrid Swedish grammar checker called Granska has been developed by Rickard Domeij *et al* [7] with the aim of better efficiency and quality; and to deal with special features in Swedish grammar and its grammatical deviations. Granska is a hybrid system that uses surface grammar rules to check grammatical constructions in Swedish. The system combines probabilistic and rule-based methods to achieve high efficiency and robustness. The researchers used manually tagged corpus for training. In the process, Granska tokenizes given sentences into word and tagged with part of speech by the tagger module using Hidden Markov Model (HMM). The output of the tagger module is sent to the error rule component where error rules, which are manually prepared, are matched with the text in order to search for specified grammatical problems. The error rule module also generates error corrections and instructional information about detected problems that are presented to the user. Furthermore, the system contains a spelling detection and correction module. The system provides noticeable results especially the tagger module but generates false flags when given to incorrectly tagged words. The performance of the system is the cumulative effect of its individual elements. For example, the tagger module can process 22,000 words per second. However, the whole system can process only 2,800 words per second including the tagger module [7].

3.3. Dependency-Based Rules for Grammar Checking

LanguageTool [18] is a modern rule-based open source grammar checking system that supports 21 languages, nevertheless the number of grammar rules ranges from 4 for Lithuanian to 1810 for French as of April, 2011. This research is a possible extension of LanguageTool with dependency grammar. The grammar checker treats input sentences as a sequence of tokens, ignoring tree-like structure of natural language sentences. For example, an indefinite article in English should never be used with plural noun. However, the existing rules ignore the number of words between the article and the corresponding noun. The proposed extension, therefore, helps to analyze word-word relationships or nonlinear structure of a phrase. So that a variety of grammar errors, including improper use of articles, incorrect verb government, and wrong word form agreement can be handled. LanguageTool uses third-party libraries for sentence splitting and part of speech tagging of the input text. A dependency parser is applied to analyze nonlinear structure of a phrase after which grammatical rules are prepared manually. The system defines an XML-based language for describing rules. In its simplest form, a rule is just a sequence of tokens to be matched in the text. The rules can be authored by any interested contributors.

3.4. Amharic Grammar Checking

Aynadis Temesgen and Yaregal Assabie [12] conducted a research on Amharic grammar checker. The grammar checker employs rule-based and statistical approaches. The rule-based approach is applied on simple sentences. However, the statistical one is used for both simple and complex sentences. The sentences are analyzed based on the phrase structures of their constituent words. The grammar checker uses manually-crafted rules as well as morphological features of words and n-gram (bigram and trigram) based probabilistic methods to check grammatical errors in the two approaches respectively. The errors include adverb-verb disagreement, adjective-noun disagreement, gender disagreement, person disagreement, number disagreement and incorrect word order. Thus, unique sequences of POS tagged words with higher probabilities are considered to be valid, which are

going to be extracted and stored in a repository. The stored sequences of n-gram probabilities are then applied and compared against the sequences of input sentences to check grammatical errors. During testing, the performance of the system indicates that as the number of sequences in the n-gram model increase, precision and recall of the system increase as well. In contrast, it is difficult to detect grammatical errors as the complexity of phrases structure increase. The reason is, the researchers argue, that complex phrase structure cannot be directly modeled by either bigram or trigram models. Other factors that hinder the performance of the grammar checkers are ineffective morphological analyzer and poor quality corpus having wrongly tagged as well as misspelled words. However, the researchers suggested that the performance of the grammar checker can be improved by using a parser, where grammatical error checking can be then done at various levels in line with the parse result [12].

3.5. Summary

In this chapter, we reviewed researches conducted on grammar checker. Both phrase structure and dependency-based grammar formalisms were employed along with rule-based, statistical, and hybrid implementation methods to design the grammar checkers.

Throughout the review, we realized that phrase structure grammar cannot model natural language sentences directly as the complexity of the phrase structure increases. This grammar formalism also lacks to understand or analyze word-word relationships or nonlinear structure of members of a sentences. Conversely, dependency based grammar formalism can handle tree-like structure of a natural language. That is, the grammatical relationship between two words can be handled with dependency-based grammar formalism irrespective of the number of words between them. We also learnt that the relationship between words can be captured by integrating dependency parsing before checking grammatical errors.

Taking these factors into account, we selected dependency based grammar formalism to analyze the structure of Amharic sentences and dependency based parsing algorithm to find word-word relationships.

Chapter 4: DBAG-Checker

4.1. Introduction

This Chapter presents the details of Dependency-Based Amharic Grammar Checker. Architecture of the proposed Model and its individual components are described in this Chapter. Dependency parsing is applied to obtain a pair of head-dependent words. Grammatical agreement checking is finally applied on the pair of words. The types of agreements include Subject-Verb, Object-Verb, Adjective-Noun and Adverb-Verb agreements. The Grammar Checker accepts input sentence, arranges the words of the sentence into ten fields of TAB separated format including POS and morphological features of each word. The result of which is parsed with an induced dependency model that helps to predict head and dependent words along with their dependency relationships. The outcome of the dependency analysis is further examined in accordance with Amharic language grammar structure.

4.2. System Architecture

The diagram depicted on Figure 4.1 shows the proposed architecture of Dependency-Based Amharic Grammar Checker. Formatting input sentence, Dependency Parser and the Grammar Checker are the three major parts of the system. The system first accepts Amharic sentences which need to be formatted in such a way that the dependency parser understands. So, the sentence is tokenized into individual words each of which listed in a new line. These words are then analyzed with tagger to obtain XPOS, UPOS and morphological features of them. The tokenizer and the taggers are induced by UDpipe. Even though the induced dependency parser understands the input sentence in CoNLL-U format with ten columns separated by a single tab character, the outcomes of formatting module only encompasses the first six columns. The seventh and eight columns are predicted by the dependency parser, which are head and dependency relation respectively. Here, the parser is induced by MaltParser after exploring and experimenting the dependency parsing algorithms available in the tool. Therefore, the parser helps to identify a pair of words having dependency relationship. This

relationship is then extracted along with the two words' morphological features in order to check grammatical agreements of these words.

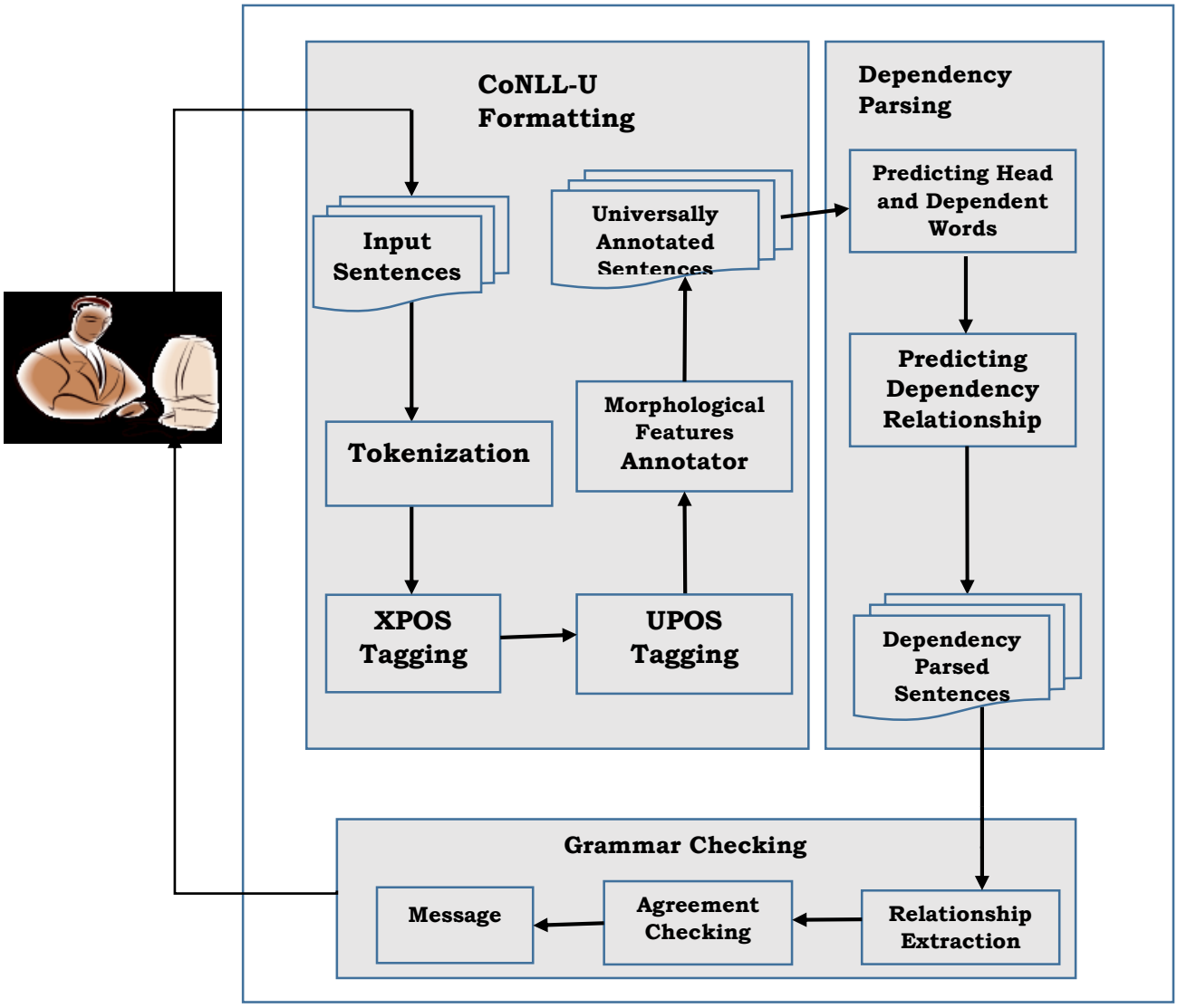

Figure 4.1: Architecture of Dependency Based Amharic Grammar Checker

4.3. CoNLL-U Formatting

The CoNLL-U formatter encompasses tokenizer, XPOS tagger, UPOS tagger and morphological feature annotator. The formatter is concerned with converting the input sentence into suitable format that fit the dependency parser requirement. The data format that the dependency parser understands is either in CoNLL or XML format. In section 2.4.6, we have seen two data formats (XML and CoNLL) that the dependency parser understands. The data format that we decided to use is CoNLL-U since the corpus [47] used to train

the parser is in CoNLL-U format. This activity is going to be done by an induced model using UDpipe [53].

UDPipe is a trainable pipeline for tokenization, tagging, lemmatization and dependency parsing of CoNLL-U files. It is language independent tool which can be trained given only annotated data in CoNLL-U format [53]. By default, UDpipe trains three component models: tokenizer, tagger and parser. Here, the tagger incorporates lemmatization and morphological feature annotation.

The input sentence formatter has got three phases: word tokenizing, POS tagging and morphological feature annotation.

4.3.1. Word Tokenizing

Word tokenization is a process of segmenting an input sentence into individual words based on the space character between them. Each line contains a single word and word index starting from one for each new sentence.

“ታምራት ወደ ጎንደር ሄደ።”

/Tamirat wede Gonder hede/ (4.1)

”Tamirat went to Gonder”

For example, sentence 4.1 is tokenized and indexed as follows.

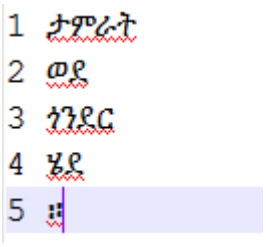

Figure 4.2: Sentence Tokenization

During tokenization, we might encounter content words coupled with functional words. In such cases, we will follow the assumptions discussed in section 2.4.5.

“አበበ ከስራ መጣ።”

/Abebe ksira meta/ (4.2)

"Abebe came from work."

For example, the word ከስራ/” ksira”/“from work” in sentence 4.2 contains content word mingled with functional word. If we treat it as a single word, it creates ambiguity during POS tagging which results in a wrong head-dependent relationship prediction during parsing. The word “ከ”/k/ “from” is a preposition whereas the word “ስራ”/sira/”work” is a noun. Therefore, the functional word “ከ”/k/ “from” should be segmented from the content word “ስራ”/sira/”work” as depicted in Figure 4.3.

1	አበበ	_	_	_	_	_	_	_	_
2	ከ	_	_	_	_	_	_	_	_
3	ስራ	_	_	_	_	_	_	_	_
4	መጣ	_	_	_	_	_	_	_	_
5	።	_	_	_	_	_	_	_	_

Figure 4.3: Content and Functional Words Segmentation

Therefore, the outcome of word tokenization is a representation of words of a sentence in two columns. These are index of the words and word forms or punctuation marks as it is illustrated in Figure 4.3.

4.3.2. Part of Speech Tagging

The first two columns of the dataset is constructed by word tokenization module. Here, POS of the word form and its lemma are tagged based on UD specification.

Binyam [47] defines POS tagsets based on UD specification. As it is depicted on the table below, the universal POS tagset and Amharic-specific tagset have been mapped to create cross-linguistically consistent annotation schema. In other words, it is possible to convert Amharic-Specific tags into corresponding UD tags.

Table 4.1: UD POS tag and Amharic-Specific tag-sets

UD POS	Amharic tag-set	examples
ADJ	ADJ	ትልቅ “big ”
ADP	ADP	ከ “from”
ADV	ADV	በጣም “very”
AUX	AUX	አል “verb to be”
CCONJ	CCONJ	ግን “but”
DET	DET	ይህ “this”
INJ	INJ	ሆ “oh”
NOUN	NOUN	በግ “sheep”
PART	ACC	ን “accusative case”
	NEG	አለ_ሴት “without a woman”
	RLP	የ_መጣ "who came”
	IRLP	አም_ይ_መጣ "who will come”
	NCM	አል_መጣ_ም "He didn’t come”
PRON	PRON	አንተ "you”
	OBJC	ነገር_ኩ_አት "I told her”
	SUBJC	ሄድ_ኧ “he went”
	POSM	ቤት_ኤ“my house”
PROPN	PROPN	ካሳ “Kassa”
PUNCT	PUNCT	። "period/fulstop”
SCONJ	SCONJ	ስለ "because”
SYM	SYM	€፣£፣$
VERB	VERB	በላ “eat”
X	X	other

Continuing the construction, this module augments the third, fourth, and fifth fields on the result of word tokenization. So the tokenized word of the sentence 4.1 and 4.2 are tagged based on the annotation schema defined.

```
1    ታምራት    ታምራት    NOUN    NOUN
2    ወደ    ወደ    ADP ADP
3    ጎንደር ጎንደር NOUN    NOUN
4    ሄደ    ሄደ    VERB    VERB
5    ።    ።

1    አበበ    አበበ NOUN NOUN
2    ከ    ከ    PREP    PREP
3    ስራ    ስራ    NOUN    NOUN
3    መጣ    መጣ    VERB    VERB
4    ።    ።    PUNC    PUNC
```

Figure 4.4: POS Tagging of Sentence 4.1 and 4.2

4.3.3. Morphological Feature Annotation

The six column in CoNLL-U representation is the FEATS field. It contains a list of morphological features separated by vertical bar (|) and with underscore to represent empty list.

Morphological features are represented as attribute-value pair, with an equal sign separating the attribute from the value sorted alphabetically by attribute name. There are cases where an attribute can have multiple values which are separated by comma alphabetically sorted: case=Acc, Det. In sorting, uppercase letters are considered identical to their lowercase counterparts [46].

Similarly, morphological features for Amharic are defined in Universal Dependencies for Amharic [47]. Morphological features of Amharic words result from the process of either inflection or derivation.

Table 4.2: Morphological Features

Category	Features	Tag	Description
Nominal	Gender	Mas	Masculine
		Fem	Feminine
		Com	Common gender
	Number	Sing	Singular
		Plur	Plural
		Coll	Collective
Verbal	Verb Form	Conv	Converb
		Inf	Infinite
		Vnoun	Verbal noun
	voice	Pass	Passive
		Mid	Middle
		Rcp	Reciprocal
		Cas	Causative
	Tense	NPas	Future/Present
		Past	Past
	Aspect	Imp	Imperfect
		Perf	Perfect
		Prog	Progressive
		Presp	Prospective
	Polarity	Neg	Negative
		Pos	Affirmative

The morphological features of sentence 4.1 and 4.2, therefore, are:

1	ታምራት	ታምራት	NOUN	NOUN	Gender=Masc|Number=Sing|Person=3
2	ወደ	ወደ	PREP	PREP	_
3	ጎንደር	ጎንደር	NOUN	NOUN	Number=Sing|Person=3
4	ሄደ	ሄደ	VERB	VERB	Gender=Masc|Number=Sing|Person=3|Tens=Past|Voice=Act
5	።	።	PUNC	PUNC	_
1	አበበ	አበበ	NOUN	NOUN	Gender=Mas|Number=Sing|Person=3
2	ከ	ከ	PREP	PREP	_
3	ስራ	ስራ	NOUN	NOUN	Number=Sing|Person=3
3	መጣ	መጣ	VERB	VERB	Gender=Mas|Number=Sing|Person=3|Tens=Past|Voice=Act
4	።	።	PUNC	PUNC	_

Figure 4.5: Morphological Features of Sentence 4.1 and 4.2

4.4. Dependency Parsing

Once input sentence are changed into CoNLL-U format, the parser identifies the head and dependent words together with their dependency relationships. Here, the parser is a data-driven model which is trained by a dependency treebank adapted from the research work of Binyam et al [47].

The parsing system used to model the parser is MaltParser [30]. It is a language independent system that generates a parsing model by learning the statistical composition of treebanks. The model then parses an input sentences based on previously learned sentence structure. In other words, it predicts the head and dependent words of the given sentence deterministically. Likewise, the parser predicts dependency relationship of head and dependent words that learned from the treebank. For example, in a sentences ልጁ መጣ ' the boy comes', ልጁ /*liju*/'the boy' is the subject and መጣ /*meta*/'comes' is verb of the sentence respectively. Therefore, the head and dependents of the sentences are መጣ and ልጁ respectively, and their relationship is represented as a pair of words in the form of nsubj (መጣ, ልጁ).

The parser works in such a way that it accepts sentences in CoNLL-U format in which all fields are populated with the required information except HEAD and DEPREL fields. The parser then predicts HEAD and DEPREL values.

Since MaltParser implements a number of algorithms, it is required to carefully design and rigorously experiment to find out a better combination of the learning and parsing algorithms. Details of the experiment is presented in Chapter 5. Therefore, we have tailored the following procedure to design and induce the parsing mode.

1. Create a separate paring model with MaltParser for each algorithms that MaltParser implements.
2. Train each model with a corpus taken from Universal Amharic Dependency Treebank [47] for development which is 20% of the whole corpus
3. Test each model with a similar corpus but manually tagged ones and compare the results
4. Select the best model based on the evaluation and train it with the remaining 80% of the corpus. This would give us the final parsing model.

Continuing the modeling process on sentence 4.1 and 4.2, they are parsed as follows with the selected parsing model.

```
1   ካሳ  ካሳ  NOUN    NOUN    Gender=Masc|Number=Sing|Person=3    4   nsubj   _   _
2   ወደ  ወደ  PREP    PREP    _   3   adjmod  _   _
3   ጎንደር ጎንደር NOUN    NOUN    Number=Sing|Person=3    4   obl _   _
4   ሄድ  ሄድ  VERB    VERB    Gender=Masc|Number=Sing|Person=3|Tens=Past|Voice=Act    0   root    _   _
5   ።   ።   PUNCT   PUNCT   _   4   punct   _   _

1   አበበ  አበበ  NOUN    NOUN    Gender=Masc|Number=Sing|Person=3    4   nsubj   _   _
2   ከ   ከ   PREP    PREP    _   3   adjmod  _   _
3   ስራ  ስራ  NOUN    NOUN    Number=Sing|Person=3    4   obl _   _
4   መጣ  መጣ  VERB    VERB    Gender=Masc|Number=Sing|Person=3|Tens=Past|Voice=Act    0   root    _   _
5   ።   ።   PUNCT   PUNCT   _   4   punct   _   _
```

Figure 4.6: Parsing Results of Sentence 4.1 and 4.2

The corresponding parse trees are:

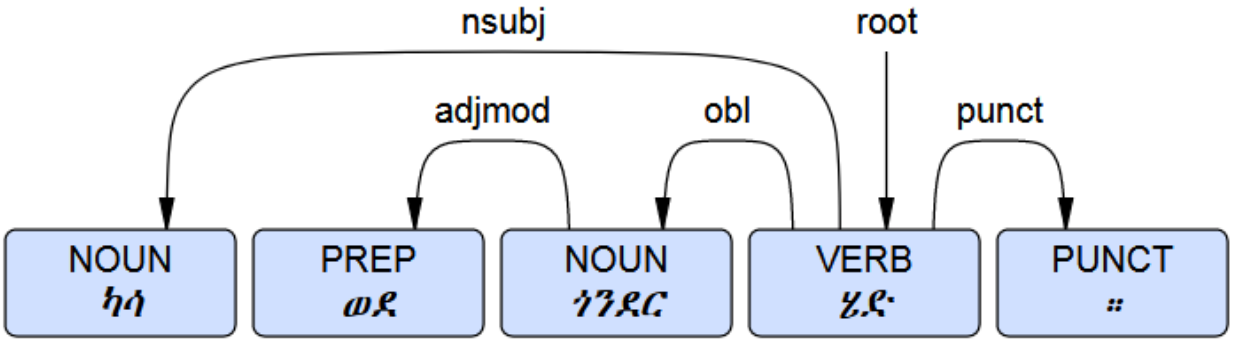

Figure 4.7: Parse tree of Sentence 4.1

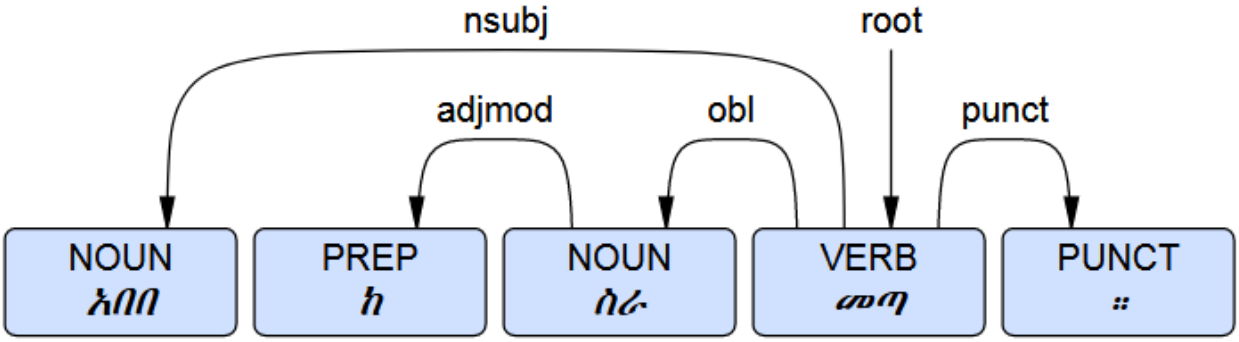

Figure 4.8: Parse tree of Sentence 4.2

4.5. Grammar Checking

The grammar checker consists of two basic modules – the module of relationship extractor and the module of agreement checker. Since the approach to the problem of grammar checking is dependency syntax, the relationship of the pair of words is going to be extracted from the result of dependency parsing. The result of relationship extractor makes it possible to find out the grammatical agreement of the paired words.

4.5.1. Relationship Extraction

At this point, the dependency structure of sentences have been recognized through CoNLL-U formatter and dependency parser, but the information is not fine-tuned to evaluate if head-dependent words are in agreement. Therefore, we need to make the result suitable for agreement checking in the form of relationship which is then defined by five parameters for each word extracted from the parsing result. The parameters are ID, FORM, UPOS, FEATS and DEPREL. The ID field specifies where the word exists in the

sentence. The FORM parameter is the actual lexical word. The UPOS is the universal POS tag of the word, the FEATS is the morphological features of the word such as gender, number, person, tens, aspect, etc. And the DEPREL parameter is the relationship between the head and dependent words. As a result, a relation is defined by five parameters of the dependent word and five parameters of head word as illustrated in Figure 4.9 below. The function getHead () returns the id of the head of the current word. So that, we can check the agreements between the paired words in terms of number, gender, and person.

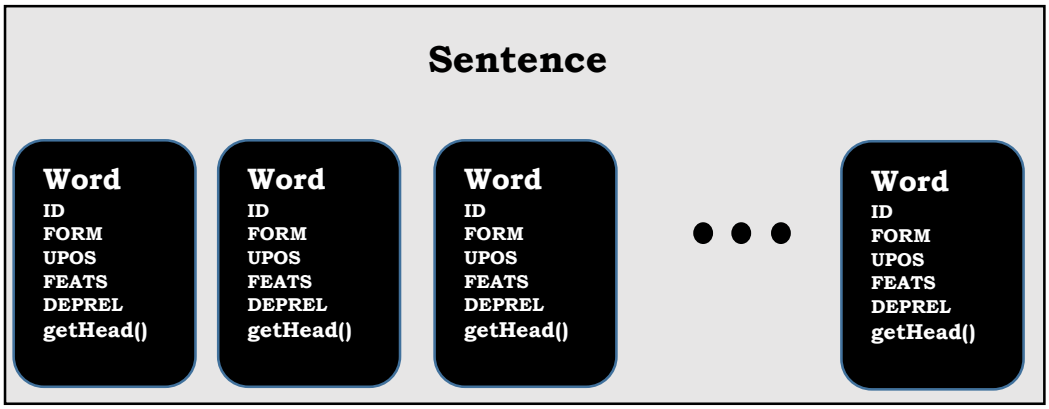

Figure 4.9: Illustration of Sentence Object

Based on the relational model depicted above, a parsed sentence definitely have more than one relational pair that determine the size of the relational dataset during construction. For example Sentences 4.1 has four pair of words (ካሳ , ሄደ) (ወደ , ጎንደር) (ጎንደር , ሄደ) (ሄደ ,።) and (።,ሄደ) and their corresponding dependency relations are nsubj, adjmod, obj, root and punct respectively.

```
nsubj ('ካሳ', 'ሄደ')
adjmod ('ወደ', 'ጎንደር')
obl ('ጎንደር', 'ሄደ')
root ('ሄደ', '።')
punct ('።', 'ሄደ')
-----------------------------
```

Figure 4.10: Head-Dependent Relationship of Sentence 4.1

The root and punct relationship which exist between root word “ሄደ” and a punctuation mark “።”; and between the punctuation mark “።” and the root word “ሄደ”. Here we ignore the root and punct relationships as they do not have significance for agreement checking. Therefore, the head-dependent relationship of Sentence 4.1 encompasses three relationships:

```
nsubj ('ካሳ', 'ሄደ')
adjmod ('ወደ', 'ጎንደር')
obl ('ጎንደር', 'ሄደ')
------------------------------
```

Figure 4.11: Filtered Head-Dependent Relationship of Sentence 4.1

So the head-dependent relationship of input sentences are extracted through Algorithm 4.1.

BEGIN
READ parse sentence from file
INIT train to sentence, heads list[], forms list[],relations list[]
FOR each value in train
    FOR each token in train
        ADD head token in heads list
        ADD form token in forms list
        ADD deprel token in relations list
SET x to the length of heads list
SET i to 0
FOR i in range of x
  SET h to the i^{th} head minus one word in the forms list
  SET d to the i^{th} word in the forms list
  SET rel to the i^{th} relation in the relations list
  CHECK IF rel of i is not root or punct
    OUTPT rel, d and h //which represent the dependent and head words respectively
END

Algorithm 4.1: Head-Dependent Relationship Extractor

4.5.2. Agreement Checking

In section 2.5.3, we have seen common grammatical agreements in Amharic sentences of SOV structure. The type of agreements are in accordance with the type of grammatical relationships returned from the parser.

The parser returned four dependency relationships including nsubj, obj, advmod and amod in the above example. These relationships are mapped to the agreements identified previously. The nsubj is subject verb agreement, for instance.

Table 4.3: Grammatical Agreement-Dependency Relationship Mapping

No	Common Grammatical Agreement	Dep. Relationships
1	Subject-Verb-Agreement	nsubj
2	Object-Verb-Agreement	obj
3	Adverb-Verb-Agreement	advmod
4	Adjective-Noun-Agreement	amod

In order to check agreements based on dependency relationship mapping, we need to first extract morphological features of head and dependent word and represent them in a dictionary. Dictionary is a way of representing information in the form of key-value pair data structure. Each key-value pairs are separated by a comma. The keys and values however are separated by a colon. All the key-value pairs are enclosed by a curly braces to make up a single dictionary. The value of a particular item is traversed through the associated key item.

$\{key_1:value_1, key_2:value_2, key_3:value_3 \ldots key_n:value_n\}$

Figure 4.12: Dictionary Data Structure of Morphological Features

For example, the linguistic information of “ካሳ” /Kasa/ are third person singular masculine and its head is “ሄደ”/Hede/ which is also a third person

singular masculine but in past active voice verb form. Therefore, their dictionary representations respectively are:

{Gender: Masc, Number: Sing, Person: 3}

{Gender: Masc, Number: Sing, Person: 3, Tens: Past, Voice: Act}

Next we are going to look for the common keys of the two dictionaries. Since we do not specify the same morphological feature value more than once for the same word during input sentence formatting, the keys of the dictionaries are unique. The common keys of "ካሳ" /Kasa/ and "ሄደ"/Hede/are Gender, Number and Person.

```
BEGIN
READ parse sentence from file
INIT train to sentence, heads list[], forms list[],relations list[], morph_featurs[]
DEFINE dep_morp_feats[],head_morp_feats[], common_key []
FOR each value in train
      FOR each token in train
            ADD head token in heads list
            ADD form token in forms list
            ADD deprel token in relations list
            ADD feats token in morph_featurs list
//Search for morphological features of dependent words
FOR i in range of list of words
  ADD the ith item in morph_featurs to dep_morp_feats
//Search for the morphological features of head words
SET x to the length of heads list
SET i to 1
FOR i in range of x
   SET j to the ith head word in the head list
   ADD the (j-1)th morph_featurs to head_morp_feats
//Looking for common keys
FOR i in dep_morp_feats
   IF i in head_morp_feats AND ith key of dep_morp_feats = ith key of head_morp_feats
   ADD i to common_key
END
```

Algorithm 4.2: Searching for Common Keys

```
----------------Common Keys-----------------------
Number
Gender
Person
Number
Person
--------------------------------------------------
```

Figure 4.13: Results of Algorithm 4.2

The result of Algorithm 4.2 is the common keys of each relation. As depicted above, there are duplicate values. But these keys are for different relations. The first three keys are for relation nsubj that occurs between the subject and the verb. The other two keys common keys of the object and the verb of the sentence.

These common keys help us to compare the values of the two dictionaries. When we traverse through the two dictionaries of the paired words using the common keys, we compare the associated values of the two dictionaries. For example, the first key among the common items is Gender. Then, we compare the values of Gender of the two words - "ካሳ" /Kasa/ and "ሄደ"/Hede/. The Gender comparison returns true for nsubj relation between "ካሳ" /Kasa/ and "ሄደ"/Hede/. It continues the comparison for person and number. However, there is no comparison for tens and voice as there are no corresponding items for the noun "ካሳ" /Kasa/. We make similar comparison for the remaining common keys and relations which is depicted below in Table 4.4.

Table 4.4: Comparison of Words of Input Sentences 4.1

Relation	**Dependent**	**Head**	**Comparison Keys**		
			Gender	Person	Number
nsubj(ካሳ , ሄደ)	ካሳ	ሄደ	True	True	True
adjmod(ወደ , ጎንደር)	ወደ	ጎንደር	NA	NA	NA
obj(ጎንደር , ሄደ)	ጎንደር	ሄደ	NA	True	True

If the two words in the relation do not have common keys, we do not compare the relation exist between them. For example, the relation adjmod between

“ወደ”/wede/ and “ጎንደር”/Gonder/ does not have common keys for comparison. The morphological features of “ወደ”/wede/ is empty set. Since an empty set is a subset of every other set, the common set of the adjmod relation is empty set. Therefore, the adjmod is not considered to determine the grammatical agreement of sentence 4.1. Similarly, the common key Gender is not used to compare the object verb relation as it is not the morphological features of the objet “ጎንደር”/Gonder/ even if it is of the verb “ሄደ”/hede/.

```
BEGIN
READ parse sentence from file
INIT train to sentence, heads list[], forms list[],relations list[], morph_featurs[]
DEFINE dep_morp_feats[],head_morp_feats[], common_key []
FOR each value in train
      FOR each token in train
            ADD head token in heads list
           ADD form token in forms list
           ADD deprel token in relations list
           ADD feats token in morph_featurs list
//Search for morphological features of dependent words
FOR i in range of list of words
  ADD the ith item in morph_featurs to dep_morp_feats
//Search for the morphological features of head words
SET x to the length of heads list
SET i to 1
FOR i in range of x
   SET j to the ith head word in the head list
   ADD the (j-1)th morph_featurs to head_morp_feats
//Looking for common keys
FOR i in dep_morp_feats
   IF i in head_morp_feats AND ith key of dep_morp_feats = ith key of head_morp_feats
   ADD i to common_key
//Agreement checking
FOR i in range of word list
   Assign the ith common_key to k
  IF the kth element of head_morp_feats is equal to the kth element of dep_morp_feats
  OUTPUT The agreement based on the relation and comparison key
//Disagreement checking
FOR i in range of word list
   Assign the ith common_key to k
  IF the kth element of head_morp_feats is equal to the kth element of dep_morp_feats
  OUTPUT The disagreement based on the relation and comparison key
END
```

Algorithm 4.3: Agreement and Disagreement Checking

```
The subject and the verb agree on{'Gender': {'Masc'}, 'Number': {'Sing'}, 'Person': {'3'}}
The object and the verb agree on{'Number': {'Sing'}, 'Person': {'3'}}
```

Figure 4.14: Agreement Checking Result of Sentence 4.1

Finally, it returns a message about the agreement type along with the linguistic information that the pair of words agree with. As depicted in Figure 4.14, the subject and the verb agree with respect to Gender, Number and Person. That is both the subject and the verb are third person singular masculine. The object and the verb also agree on number and person as both are third person singular.

Chapter 5: Experiment

5.1. Introduction

One of the objectives of this study is to develop a prototype for dependency based Amharic grammar checker. The prototypes include tokenizer, part of speech tagger, dependency parsing models and agreement checking of a pair of words of a sentence. In this chapter, therefore, we are going to discuss evaluation metrics, dataset preparation for testing and inducing tokenizing, tagging and parsing model as well as the grammar checker. We also discuss the experiment how an efficient models are selected, and finally discuss the respective test results.

5.2. Evaluation Metrics

The evaluation metrics employed in this study are accuracy, precision, recall, and F1. They are used for assessing the performance of tokenizer, tagger and agreement checker. These metrics are measured based on the result of the predictions of an induced models. The prediction of a model can either be true, which is correctly classified, or false representing misclassified result. Here, the result of the classification is four in number which are described by confusion matrix as shown in Table 5.1. The row of the table represents the predicted class, while the column represents the actual class. From this confusion matrix, TP and TN denote the number of positive and negative instances that are correctly classified. Meanwhile, FP and FN denote the number of misclassified positive and negative instances, respectively [54].

Table 5.1: Confusion Matrix

	Actual Positive Class	**Actual Negative Class**
Predicted Positive Class	True Positive (TP)	False Negative (FN)
Predicted Negative Class	False Positive (FP)	True Negative (TN)

The evaluation metrics can be determined based on the confusion matrix:

- Accuracy measures the ratio of correct predictions over the total number of instances evaluated and is defined by:

  $Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$, where TP is true positive, TN is true negative, FP is false positive and FN is false negative
- Precision is used to measure the positive patterns that are correctly predicted from the total predicted patterns in a positive class. It is determined by:

  $Precision = \frac{TP}{TP+FP}$
- Recall is used to measure the fraction of positive patterns that are correctly classified and can be determined by:

  $Recall = \frac{TP}{TP+TN}$
- F1 represents the harmonic mean between recall and precision values. It is defined by:

  $F1 = \frac{2 * Precison * Recall}{Precision + Recall}$

For dependency parser, we are going to use labeled attachment score (LAS) and unlabeled attachment score (UAS) [20].

- LAS (Both Right): A token is counted as a hit if both the head and the dependency label are the same as in the gold-standard data.
- UAS (Head Right): A token is counted as a hit if the head is the same as in the gold-standard data.

5.3. Development of Tokenizing and Tagging Model

Development of tokenization and tagging model require training data in CoNLL-U format. Among the morphological fields of CoNLL-U format, XPOS, UPOS, lemma or stem and morphological features are used by UDpipe to build tokenization and tagging model depending on the availability of the fields in the training data.

UDpipe makes use of CoNLL-U formatted sentences to induce a model [53]. When developing a tokenizer and a tagger, we utilized Universal Amharic Dependency Treebank [47]. The Amharic treebank contains 1075 sentences

among which 80% or 860 sentence are used for training the model whereas the remaining 20% or 215 are used to evaluate the induced models. In addition the models are evaluated by row corpus collected from Ethiopian Reporter News Paper (online).

During development, we apply different options depending on the model we build. For example, the tokenizer recognizes options such as epochs, batch size, learning rate and dropout. The tagger also recognizes hyperparameters that need to be optimized. These are lemma, XPOS, features, guesser rules and guesser dictionary [53].

During hyperparameter search, the options of tokenizer have the following values:

- Epochs: its default value is 100
- Batch size is uniformly chosen between 50 and 100. Its default value is 50.
- Learning rate is range between 0.0005 and 0.01. The default value is 0.005.
- Dropout: its default value is 0.1.

Similarly, the options of the tagger have different values. The guesser rule and guesser dictionary have default values of 8 and 6 respectively.

5.3.1. Testing and Evaluation

There are six models experimented for tokenization based on the properties defined on Table 5.2.

Table 5.2: Hyperparameter values for Tokenizer

Hyperparameters	**Tokenization Models**					
	A	B	C	D	E	F
Epoch	50	60	70	80	90	100
Batch_size	50	60	70	80	90	100
Learining_rate	0.0005	0.0025	0.0045	0.0065	0.0085	0.01
dropout	0.1	0.2	0.3	0.4	0.5	0.6

For tagging, we obtained five models by varying guesser rules and dictionaries staring from their default values for experimentation. The properties of the models are presented on Table 5.3.

Table 5.3: Hyperparameter values for Tagger

Hyperparameter	**Tagging Models**				
	A	B	C	D	E
Guesser_rules	8	9	10	11	12
Guesser_dictionary	6	7	8	9	10

We use UDpipe for evaluating the result of the models which depicted on Table 5.4 below. UDpipe uses precision, recall and F1 to evaluate tokenizer. The performance of the models are evaluated against the data reserved for testing.

Table 5.4: Evaluation result of Tokenizer Generation

Models	**Precision**	**Recall**	**F1**
A	98.16	40.96	57.80
B	98.16	40.96	57.80
C	98.16	40.96	57.80
D	98.16	40.96	57.80
E	98.16	40.96	57.80
F	98.16	40.96	57.80

Similarly, the performance of the taggers are evaluated against the test sentences obtained from Amharic Dependency Treebank. UDpipe is used to evaluate the accuracy of the models which are described on Table 5.5.

Table 5.5: Evaluation result of Tagger

Tagger	**UPOS tag**	**XPOS tag**	**feats**	**Lemma**
A	84.84	83.43	88.54	100
B	84.16	82.86	86.61	100
C	84.42	83.22	87.39	100
D	84.68	83.38	88.38	100
E	83.59	82.39	86.87	100

5.3.2. Discussion

The evaluation result of the tokenizer shows that all the models experience the same result irrespective of the hyperparameters value variation. The precision, recall and F1 of the model are 98.16%, 40.96 and 57.80 respectively. This implies that we can use one of the models for tokenizing input sentences.

Conversely, the evaluation of the tagger shows different results when we vary the hyperparameter values. The evaluation matrix for tagging is accuracy of UPOS tag, XPOS tag, feature annotation and lemmatization. Based on the result, the same characteristics is returnee for the lemmatization guesser. Therefore, the result of lemmatization is not considered to select the best model.

Accordingly, the model that returns best accuracy is TaggigModel_A. Its accuracy for UPOS tag, XPOS tag and feature annotation are 84.84%, 83.43% and 88.54% respectively.

Therefore, a single model has been induced for tokenizer and tagger by combining the properties of the best models generated during experiment.

When we apply the induced models on raw text collected from newspaper, the evaluation result is different. The text contains 13 paragraphs, 24 sentences and 682 tokens including punctuation marks. The tokenizer showed good results given a single sentence on new line and a space between successive tokens. It exhibits nearly 100 % accuracy provided that the input text manually separated sentences. However, it mingles one sentence with the other if it is provided with the raw text as it is. Specifically, it does not recognize sentence end marker such as four points. Another drawbacks of the tokenizer is that it tokenizes a word together with a punctuation mark as a single token if there is no space between them. Its evaluation result, therefore, varies depending on how we provide the sentences to the tokenizer. If each sentences are in a new line and, the tokenizer recognizes each sentences and tokenize them accordingly.

In a similar fashion, the tagger has been evaluated. The result of the tagger is poorer than the tokenizer. It correctly tags 294 tokens out of 682 tokens. To be precise, its accuracy is 43.11% as shown in Table 5.6 below.

Table 5.6. POS Tagging Result

POS Tag	**Correct Tag**	**Incorrect Tag**	**Total**	**Accuracy = Correct Tag / Total**
ACC	12	33	45	26.67
ADJ	1	25	26	3.85
ADP	17	41	58	29.31
ADV	5	11	16	31.25
AUX	2	30	32	6.25
CCONJ	6	4	10	60.00
DET	5	31	36	13.89
INTJ	0	5	5	0.00
NOUN	180	40	220	81.82
NUM	8	4	12	66.67
OBJC	0	20	20	0.00
POSM	0	5	5	0.00
PRON	0	9	9	0.00
PROPN	6	16	22	27.27
PUNCT	46	0	46	100.00
RLP	0	1	1	0.00
SCONJ	0	3	3	0.00
SUBJC	0	55	55	0.00
VERB	6	55	61	9.84
Total	**294**	**388**	**682**	**43.11**

During experiment of selecting the best tagger model, it experiences an accuracy of up to 84.84%. This is because the data used for experiment was manually tagged treebank. The word distribution of the training and evaluation treebanks were also homogenous. That means similar words found

on both treebanks. These increases the accuracy of the model during experiment. Whereas, the accuracy of the model with raw text is almost half of that of the accuracy getting during experiment.

As the output of tokenizer and tagger module is the input for remaining modules, the efficiency of the overall will also suffer. As a result, we decided to use manually tagged sentences for dependency parser and agreement checking.

5.4. Development of Dependency Parsing Model

As a design aspect, we have incorporated a parsing model in the grammar checker architecture. A MaltParser is used to obtain the model. As it is described in section 2.4.4, MaltParser implements a number of algorithms including projective and non-projective. Therefore, we need to experiment and investigate which algorithm does best perform and suit the Amharic language structure.

To optimize MaltParser, Miguel et al [33] suggests two heuristic methods that one should follow in order to have an optimized parsing model. These methods are depicted in the form of decision trees for projective and non-projective algorithms.

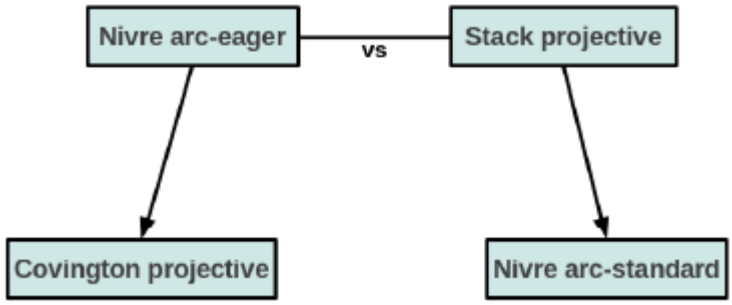

Figure 5.1. Decision tree for best projective algorithm

The decision tree for projective algorithm described in Figure 5.1 has four nodes representing projective algorithms. As Miguel et al [33] proposes, we have to first compare the results of Nivre arc-eager and Stack projective algorithms. If Nivre arc-eager is better than Stack projective, then Covington projective will be tested. The final output of this wing is the comparison result

of Nivre arc-eager and Covington projective algorithms. On the other hand, the Nivre arc-standard is going to be tested provided that Stack projective outperforms Nivre arc-eager algorithm. Similarly, the best algorithm of this wing will be decided after comparing Stack projective and Nivre arc-standard.

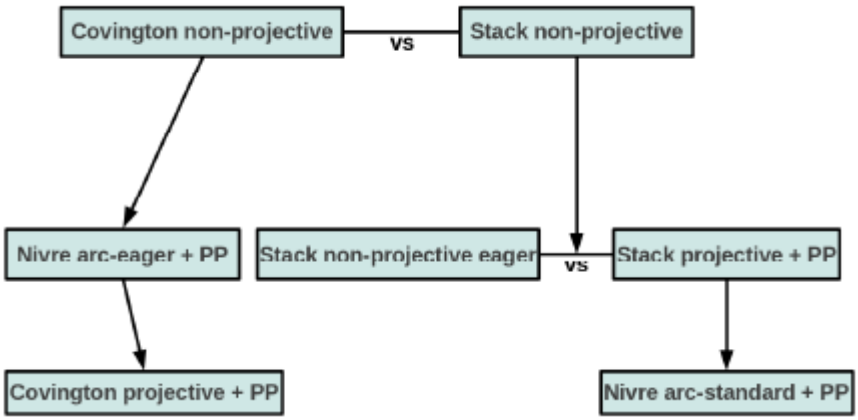

Figure 5.2: Decision tree for best non-projective algorithm

For non-projective algorithms, the decision tree in Figure 5.2 incorporates a couple of nodes that represent the algorithms. In the first move, Covington and Stack non-projective algorithms are compared. If Covington non-projective better performs, then Nivre arc-eager followed by Covington projective are going to be tested combined with pseudo-projective algorithms to handle non-projective structure of the language [33]. The pseudo-projective algorithms include head, path or head plus path [34]. Conversely, if Stack non-projective overtakes then we made a comparison between Stack non-projective eager and Stack projective combined with pseudo-projective algorithms. If Stack non-projective eager beats, we stop here and compare all the results. If Stack projective combined with pseudo-projective algorithms is better, we further test Nivre arc-standard combined with pseudo-projective algorithms.

5.4.1. Testing and Evaluation

The experiment to model the dependency parsing module contains two treebanks. One for development training to look for an efficient dependency parsing algorithm and the other is for testing the induced model during experimental development. The testing data set is considered as a gold corpus

against which the parsing results of each algorithms are going to be checked. The two datasets are similar in that both have equal number of tokens and similar sentences. The first corpus is unparsed one whereas the second one is manually parsed. The corpuses are taken from Universal Amharic Dependency Treebank [47] which is divided into two groups with 4:1 ratio for training and for testing purpose respectively.

We have obtained seven parsing model each of which having the name of the parsing algorithms implemented by MaltParser. However, the learning algorithm applied is only LIBLINEAR due to the fact that the other learning algorithm, LIBSVM, is not available on the current version of MaltParser 1.9.2.

The parsing results are tested against the corpus assigned as a gold reference. We have used MaltEval 1.0 [20] to evaluate the parser. The results of the evaluations is described on Table 5.7 below.

Table 5.7: Evaluation of parsing algorithms

Item No	Algorithm	UAS	LAS
1	nivrestandard	98.90	94.80
2	nivreeager	98.30	94.40
3	covproj	98.10	94.60
4	covnonproj	98.30	95.30
5	stackproj	98.70	94.70
6	stackeager	99.00	95.10
7	stacklazy	99.00	94.80

5.4.2. Discussion

Based on the evaluation result depicted on table 5.7, a parsing model induced by Covington non-projective algorithm along with LIBLINEAR learning algorithm has been selected as a parsing model. This is because, it has the maximum LAS which is 95.3%. Therefore, this parsing model has been trained with the remaining 80% of the dataset reserved for final training.

5.5. Development of the Grammar Checker

As described in design phase, an agreement of a pair of words of a sentence, that have dependency relationship, are going to be checked with respect to morphological features of words. For this we have developed a prototype which is developed with Python programing language. The main purpose of the prototype is to demonstrate and evaluate the grammar checker.

5.5.1. Testing and Evaluation

For agreement checking, we used 20 randomly selected sentences of the Amharic treebank. Since the sentences are manually prepared by linguistic experts, they are grammatically correct. These sentences are intentionally taken as test cases to check whether the grammar checker correctly predicts their grammatical agreement or not.

To determine the effectiveness of the grammar checker, the sentences are given to the grammar checker one by one. The number of agreements returned are then counted and compared against the actual available agreements that should be identified. The grammar checker governs major types of agreements such as subj-verb, obj-verb, adj-type and adv-verb agreements. This would help us to figure out the accuracy of the grammar checker. The accuracy is, therefore, computed by dividing the identified count by the actual count.

The result obtained from the evaluation of the grammar checker is depicted on Table 5.8.

Table 5.8. Grammar Checker Evaluation Result

Agreement Type	Identified Count	Actual Count	Accuracy
Sub-Verb	15	22	68.18
Obj-Verb	13	16	81.25
Adv-Verb	1	5	20.00
Total	29	43	67.44

5.5.2. Discussion

From the evaluation result, the grammar checker has identified 68.18% of subject-verb agreement, 81.25% of object-verb agreement and 20% of adverb-

verb agreement. The grammar checker identifies more subject-verb agreement that the other two.

The grammar checker sometimes returns nothing as if there is no agreements between head and dependent words. This is because if one of the two words does not have morphological features in the input CoNLL-U file, the grammar checker considers them as if they disagree. The grammar has also limitation as the number of tokens of a sentence increases. It works fine for simple sentences.

Chapter 6: Conclusions and Recommendations

6.1. Conclusions

The aim of this research work was design and develop a dependency based Amharic grammar checker. To this end, the thesis incorporated syntactic parsing of dependency grammar and pre-processing models to convert input sentences to CoNLL-U format.

The grammar checker's modular components are CoNLL-U Formatter (Tokenizer and Tagger), Dependency Parse and Agreement Checker. The CoNLL-U Formatter and Dependency Parse modules are induced by third party NLP tools which require input sentences in CoNLL-U format to induce the corresponding models. We have used Amharic Dependency Treebank to train the language model. However, the size of the treebank was limited. This affects the performance of the CoNLL-U formatter and the Dependency Parser. Therefore, the performance of the grammar checker depends on the performance of third party tools used to generate the language models.

6.2. Contribution

The Dependency based Amharic grammar checker provides clues that lay foundation for dependency based research work. The main contributions of the research work are:

- The architectural design of the grammar checker can serve as a basis for the study of statistical approach with the same grammar formalism
- Integrated dependency parsing with rule-based approach
- Relationship extraction
- Amharic dependency parsing language model
- Amharic tokenizing and tagging language model for CoNLL-U formatting

6.3. Future Work

This research can be enhanced and upgraded by improving the performance of each components. This can be achieved through (1) utilizing large treebank to train an induced model (2) rigorously experimenting the language models

applied in this research work to obtain optimum models (3) adding other grammatical agreements of Amharic language (4) incorporating statistical approaches of grammar checker.

Specifically, the future work arising from this study are:

- **Large treebank**: The availability of larger treebanks will improve linguistic models. As treebanks grow in size, they will become more useful in literary and historical studies, where the linguistic structure of texts will vary and become investigable in a more precise way.
- **Language independent tools**: universal dependency framework employs different tools for consistency annotation, format conversions and dependency parsing. In this study, we have used MaltParser and UDpipe to build dependency parser as well as tagger models. However, the latter modules performance was poor and need to be investigated by other tools to come up with enhanced model.
- **Statistical Approach**: Currently, the grammar checker implemented with rule-based approach in that morphological features of head-dependent words are compared. Thus, statistical approach improves the performance of the grammar checker.

References

[1] Martin, Daniel Jurafsky & James H., ” Speech and Language Processing” A Text Book An introduction to natural language processing,Computational linguistics, and speech recognition, 2006.

[2] Sylvana Sofkova Hashemi, Robin Cooper and Robert Andersson, "Positive Grammar Checking a Finite State Approach," Department of Linguistics Goteborg University, Goteborg, Sweden.

[3] Moré, Joaquim, "A Grammar Checker Based on Web Searching," Researcher at the Internet Interdisciplinary Institute (IN3) of the UOC, May 2006.

[4] Philip S. Kernick and David M.W. Powers, "A Statistical Grammar Checker," Flinders University of SA, Adelaide SA, August 1996.

[5] Cornelia Tschichold, Franck Bodmer, Etienne Cornu, Franqois Grosjean, Lysiane Grosjean, Natalie Ktibler, Nicolas Lrwy & Corinne Tschumi, "Developing a new grammar checker for English as a second language," Universit6 de Neuch~tel, Avenue du Premier-Mars 26.

[6] Lawley, Jim, "The Development of a Grammar Checker for Spanish Secondary Student of English as a Foreign Language," Universidad Nacional d Educacion s Distancia.

[7] Rickard Domeij, Ola Knutsson, Johan Carlberger, Viggo Kann, "Granska- an efficient hybrid system for Swedish grammar checking," Dept. of Linguistics, Stockholm University, Stockholm.

[8] J. Carlberger, R. Domeij, V. Kann, O. Knutsson, "The Development and Performance of a Grammar Checker for Swedish: A Language Engineering Perspective," Cambridge University Press, Stockholm, Sweden, December 2004.

[9] J. Sj¨obergh, "The Internet as a Normative Corpus Grammar Checking with a Search Engine," KTH Nada, Stockholm, Sweden.

[10] L. A. Sara Stymne, "Using a Grammar Checker for Evaluation and Postprocessing of Statistical Machine Translation," Department of Computer and Information Science Link¨oping University, Sweden.

[11] D. Tesfaye, "A rule-based Afan Oromo Grammar Checker," *(IJACSA) International Journal of Advancec Computer Science and Applications,* vol. 2, 2011.

[12] Aynadis Temesgen and Yaregal Assabie, "Design and Development of Amharic Grammar Checker Using Morphological Features of Words and N-Gram Based Probabilistic Methods," in *The 13th International Confrence on Parsing Technologies(IWPT2013) pp 106-112*, Nara Japan, 2013.

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

[14] Alexander Clark, Chris Fox, and Shalom Lappin, The Handbook of Computational Linguistics and Natural Language Processing, United Kingdom: A John Wiley & Sons, Ltd., Publication, 2010.

[15] V. Agel, L.M. Eichinger, H.-W. Eroms, P. Hellwig, H.-J. Heringer, H. Lobin., "An International Handbook of Contemporary Research. Chapter 79 Parsing with Dependency Grammars," *Dependency and Valency,* vol. II, pp. 1081-1108, 2006.

[16] Gerold Schneider and Prof. Dr. Michael Hess, "A Linguistic Comparison of Constituency, Dependency and Link Grammar," University of Zurich Department of Informatics, Zurich, 1998.

[17] "LXMLS 2011 LISBON MACHINE LEARNING SCHOOL Learning For The Web," 19 July 2011. [Online]. Available: http://lxmls.it.pt/2011/guide_day4.pdf. [Accessed 15 February 2019].

[18] Maxim Mozgovoy, Tsuruga, Ikki-machi, Aizu-Wakamatsu, "Dependency-Based Rules for Grammar Checking with LanguageTool," in *Proceedings of the Federated Conference on Computer Science and Information Systems pp. 209–212*, Fukushima, Japan, 2011.

[19] Daniel Jurafsky & James H. Martin, "Chapter 14 Dependency Parsing," in *Speech and Language Processing.*, 2017.

[20] J. Nilsson, "User Guide for MaltEval 1.0 (beta)," Vaxjo University, Vaxjo, Sweden, February 6, 2012.

[21] Leekha Jindal, Vijay Rana, "Grammar Checking Using Pattern Matching Approach," *International Journal of Interdisciplinary Research and Innovations,* vol. 6, no. 3, pp. 142-147, 2018.

[22] A. Fokkens, *Dependency Grammars Syntactic Theory,* Saarland, Germany: Department of Computational Linguistics Saarland University, 27 October 2009.

[23] D. Naber, ""“A Rule-Based Style And Grammar Checker”," Diplomarbeit. Technische Fakultät Bielefeld, , 2003..

[24] Tiberiu Boros, Stefan Daniel, Adrian Zafiu, Dan Tufis, Verginica Mititelu Barbu, Paul Ionuţ Văduva, "A hybrid approach to Grammatical Error Correction," in *CoNLL-2014 Eighteenth Conference on Computational Natural Language Learning PP 53-58*, Baltimore, Maryland, USA, June 26-27, 2014.

[25] Blossom Manchanda, Vijay Anant Athvale, Sanjeev Kumar Sharma, "Various Techniques used for Grammar Checking," *Internation Journal of Computer Application & information Technology,* vol. 9, no. 1, 2016.

[26] Joskim Nivre, Inductive Dependency Parsing, Sweeden: Springer, Netherlands www.springer.com, 2006.

[27] R. Debusmann, "An Introduction to Dependency Grammar," University of Saarlandes, January 2000.

[28] Sabine Buchholz, Erwin Marsi, "CoNLL-X shared task on Multilingual Dependency Parsing," in *2 Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X), pages 149–164*, New York, June 2006.

[29] McDonald, R., Pereira, F., Ribarov, K., and Haji_c, J., "Non-projective dependency parsing using spanning tree algorithms," in *In Proceedings of Human Language Technology Conference on Empirical Methods in Natural Language Processing*, Vancouver, British Columbia, Canada, 2005.

[30] Joakim Nivre, Johan Hall, Jens Nilsson, "MaltParser: A Data-Driven Parser-Generator for Dependency Parsing," in *In Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, Genoa-Italy, 2006.

[31] Joakim Nivre, Sandra Kubler, "Dependency Parsing Tutorial at COLING-ACL," 2006. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/citations?doi=10.1.1.184.5064. [Accessed April 2018].

[32] Johan Hall, Joakim Nivre, "MaltParser User Guide," [Online]. Available: http://www.maltparser.org/userguide.html#inout. [Accessed August 2018].

[33] Miguel Ballesteros and Joakim Nivre, "MaltOptimizer: A System for MaltParser Optimization," in *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC)*, Istanbul-Turkey, 2012.

[34] Nivre, Joakim & Hall, Johan., "A Quick Guide to MaltParser Optimization," 2010.

[35] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: A Library for Support Vector Machines," Department of Computer Science National Taiwan University, Taipei, Taiwan, Taipei, Taiwan, Initial version: 2001 Last updated: March 4, 2013.

[36] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, Chih-Jen Lin cjlin@csie.ntu.edu.tw, "LIBLINEAR: A Library for Large Linear Classification," *Journal of Machine Learning Research,* vol. 9, pp. 1871-1874, 2008.

[37] Joakim Nivre Ed. by Anke Lüdeling and Merja Kytö., "Treebank," *An International Handbook of Corpus Linguistics,* vol. Vol. 1, p. 225–241, 2008.

[38] Teresa Lynn and (Dr.) Jennifer Foster, "Irish Dependency Treebanking and Parsing," School of Computing Dublin City University, Department of Computing Macquarie University, Dublin, January 2016.

[39] Michael Gasser, "A Dependency Grammar for Amharic," School of Informatics and Computing Indiana University, Bloomington, Indiana USA.

[40] Binyam Ephrem Seyoum, Yusuke Miyao, Baye Yimam Mekonnen, "Morpho-syntactically Annotated Amharic Treebank," in *Corpus Linguistics*, Bloomington, Indiana University, 2016.

[41] Martha Yifiru, Wolfgang Menzel, "Morphology-Based Language Modeling for Amharic," University of Hamburg, Hamburg, August 2010.

[42] Wenliang Chen, Min Zhang, Semi-Supervised Dependency Parsing, Singapore: www.springer.com url: https://books.google.com.et/books?id=zkkwCgAAQBAJ&pg=PA3&dq=%EF%82%A7%09H+determines+the+syntactic+category+of+C+and+can+often+replace+C&hl=en&sa=X&ved=0ahUKEwiLjuPl_OXhAhUoposKHfsCBUAQ6AEIKDAA#v=onepage&q&f=true, 2015.

[43] N. G. Silveira, "Designing syntactic Representations For NLP: An Empirical Investigation," Stanford University http://purl.stanford.edu/kv949cx3011, 2016.

[44] B. Yimam, Amharic Grammar Third Edition, Addis Ababa: Addis Ababa University Press, 2009.

[45] "universaldependencies.org," [Online]. Available: http://universaldependencies.org/format.html. [Accessed August 2018].

[46] "Universal Dependency v2.0," [Online]. Available: http://universaldependencies.org/introduction.html. [Accessed September 2018].

[47] Binyam Ephrem Seyoum, Yusuke Miyao, Baye Yimam Mekonnen, "Universal Dependencies for Amharic," in *LREC proceedings*, Tokyo, Japan, 2018.

[48] "CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies," 10 September 2018. [Online]. Available: http://universaldependencies.org/conll17/. [Accessed 10 September 2018].

[49] "CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies," 19 September 2018. [Online]. Available: http://universaldependencies.org/conll18/. [Accessed 19 September 2018].

[50] Tigist Tensou, Yaregal Assabie, "Word Sequence Prediction for Amharic Language," Addis Ababa University, Addis Ababa, 2014.

[51] B. Yimam, Amharic Grammar, Addis Ababa, 2000.

[52] M. Gasser, "HornMorpho: A System for Morphological Processing of Amharic, Oromo, and Tigrinya," in *Human Language Technology for Development*, Alexandria, Egypt, 2011.

[53] *UDpipe 1.0 Manual,* Czech Republic: Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University in Prague, 2016.

[54] Hossin, M. and Sulaiman, M.N., "A REVIEW ON EVALUATION METRICS FOR DATA CLASSIFICATION EVALUATIONS," *International Journal of Data Mining & Knowledge Management Process (IJDKP),* vol. 5, no. 2, March 2015.

[55] J. Nivre, "Incrementality in deterministic dependency parsing," in *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, Barcelona-Spain, July 2004.

[56] M. A. Covington, "A Fundamental Algorithm for Dependency Parsing," in *39th Anual Association for Computing Machinery Southeast Conference pp. 95–102,* U.S.A., 2001.

Annexes

Annex A: CONLL-X Format Field Description

No	Field	Description
1	ID	Token counter, starting at 1 for each new sentence.
2	FORM	Word form or punctuation symbol.
3	LEMMA	Lemma or stem (depending on the particular treebank) of word form, or an underscore if not available.
4	CPOSTAG	Coarse-grained part-of-speech tag, where the tagset depends on the treebank.
5	POSTAG	Fine-grained part-of-speech tag, where the tagset depends on the treebank. It is identical to the CPOSTAG value if no POSTAG is available from the original treebank.
6	FEATS	Unordered set of syntactic and/or morphological features (depending on the particular treebank), or an underscore if not available. Set members are separated by a vertical bar (|).
7	HEAD	Head of the current token, which is either a value of ID, or zero ('0') if the token links to the virtual root node of the sentence. Note that depending on the original treebank annotation, there may be multiple tokens with a HEAD value of zero.
8	DEPREL	Dependency relation to the HEAD. The set of dependency relations depends on the particular treebank. The dependency relation of a token with HEAD=0 may be meaningful or simply 'ROOT' (also depending on the treebank).
9	PHEAD	Projective head of current token, which is either a value of ID or zero ('0'), or an underscore if not available. The dependency structure resulting from the PHEAD column is guaranteed to be projective (but is not available for all data sets), whereas the structure resulting from the HEAD column will be non-projective for some sentences of some languages (but is always available).
10	PDEPREL	Dependency relation to the PHEAD, or an underscore if not available

Annex B: CONLL-U Format Field Description

No	Field	Description
1	ID	Word index, integer starting at 1 for each new sentence; may be a range for multiword tokens; may be a decimal number for empty nodes.
2	FORM	Word form or punctuation symbol.
3	LEMMA	Lemma or stem of word form.
4	UPOS	Universal part-of-speech tag.
5	XPOS	Language-specific part-of-speech tag; underscore if not available.
6	FEATS	List of morphological features from the universal feature inventory or from a defined language-specific extension; underscore if not available.
7	HEAD	Head of the current word, which is either a value of ID or zero (0).
8	DEPREL	Universal dependency relation to the HEAD (root iff HEAD = 0) or a defined language-specific subtype of one.
9	DEPS	Enhanced dependency graph in the form of a list of head-deprel pairs.
10	MISC	Any other annotation.

Annex C: Universal POS Tags

Universal POS tags	Description
ADJ	Adjective
ADP	Adposition
ADV	Adverb
AUX	Auxiliary
CCONJ	Coordinating conjunction
DET	Determiner
INTJ	Interjection
NOUN	Noun
NUM	Numeral
PART	Particle
PRON	Pronoun
PROPN	Proper noun
PUNCT	Punctuation
SCONJ	Subordinating conjunction
SYM	Symbol
VERB	Verb
X	Other

Annex D: Universal Features Inventory

Universal Features Inventory		
Inflectional Features		**Lexical Features**
Verbal	**Nominal**	
VerbForm	Gender	PronType
Mood	Animacy	NomType
Tense	Number	Poss
Aspect	Case	Reflex
Voice	Definite	
Person	Degree	
Negative		

Annex E: Universal Dependency Relations

Universal Dependency Relations			
Relation	**Description**	**Relation**	**Description**
acl	clausal modifier of noun (adjectival clause)	goeswith	goes with
advcl	adverbial clause modifier	iobj	indirect object
advmod	adverbial modifier	list	List
amod	adjectival modifier	mark	marker
appos	appositional modifier	mwe	multi-word expression
aux	Auxiliary	name	Name
auxpass	passive auxiliary	neg	negation modifier
case	case marking	nmod	nominal modifier
cc	coordinating conjunction	nsubj	nominal subject
ccomp	clausal complement	nsubjpass	passive nominal subject
compound	Compound	nummod	numeric modifier
conj	Conjunct	parataxis	parataxis
cop	Copula	punct	punctuation
csubj	clausal subject	remnant	remnant in ellipsis
csubjpass	clausal passive subject	reparandum	overridden disfluency
dep	unspecified dependency	root	Root
det	determiner	vocative	Vocative
discourse	discourse element	xcomp	open clausal complement
dislocated	dislocated elements		
dobj	direct object		
expl	expletive		
foreign	foreign words		

Declaration

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

Declared by:

Name: Abraham Gebreamlak Gebremariam

Signature: _____

Date: _____

Confirmed by advisor:

Name: Yaregal Assabie (PhD)

Signature: _____

Date: _____