Transition Based Dependency Parser for Amharic Language using Transformer Model

A Thesis Presented

by

Meaza Birhanu Kebede

to

The Faculty of Informatics

of

St. Mary's University

In Partial Fulfillment of the Requirements
for the Degree of Master of Science

in

Computer Science

February 2024

ACCEPTANCE

Transition Based Dependency Parser for Amharic Language using Transformer Model

By

Meaza Birhanu Kebede

Accepted by the Faculty of Informatics, St. Mary's University, in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Thesis Examination Committee:

Internal Examiner

External Examiner

Dean, Faculty of Informatics

February 2024

DECLARATION

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

____

Full Name of Student

____

Signature

Addis Ababa

Ethiopia

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

____

Full Name of Advisor

____

Signature

Addis Ababa

Ethiopia

February 2024

Acknowledgment

I begin by offering my heartfelt gratitude to the Almighty God, whose divine guidance and unwavering presence have been the bedrock of strength throughout the journey of completing this thesis on parsing. In moments of challenge and triumph, I felt the grace and blessings that fueled my perseverance.

I extend my sincere appreciation to my thesis advisor, Alembante Mulu (PhD), whose mentorship and expertise have been a beacon of wisdom. I am grateful for the guidance that transcends academic realms and embraces the spiritual dimensions of knowledge.

I extend my gratitude to my colleagues and friends for their camaraderie and stimulating discussions, contributing to the robustness of this thesis.

Special appreciation goes to to my family , my husband and friends, thank you for your unwavering support and encouragement. Your prayers and belief in my abilities have been a constant source of strength.

In acknowledging the divine guidance and the contributions of these individuals and institutions, I humbly submit this thesis with gratitude.

Table of Contents

List of Abbreviation and Acronyms

AP	Average Perceptron
BILSTM	Bidirectional Long Short-Term Memory
CFG	Context Free Grammar
FNN	Feed Forward Network
HMM	Hidden Markov Model
LAS	Labeled Attachment Score
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
MIRA	Average Margin Infused Relaxed
MST	Maximum Spanning Tree
NLP	Natural Language Processing
PA	Passive Aggressive
POS	Part of Speech
UAS	Unlabeled Attachment Score
XDG	Extensible Dependency Grammar

List of Figures

List of Tables

Abstract

Dependency parsing is a fundamental task in natural language processing that involves analyzing the grammatical structure of sentences. This research focuses on advancing dependency parsing techniques for the Amharic language, using Transformer model. Amharic language is rich in linguistic complexities. For the experiment, we utilized a treebank containing 1574 sentences. Out of these, 500 sentences were meticulously crafted by the researcher in collaboration with linguistic experts. The entirety of the sentences originated from works of fiction and various novel genres, chosen to ensure relative structural correctness. While the remaining 1074 Amharic sentence adopted from UD-Amharic Treebank. The research begins with careful data preprocessing to ensure the quality and consistency of the dataset. We perform morphological analysis, POS tag and syntactic relations on collected sentence. The Transformer model is well-known for its success in various natural language processing tasks. The model's ability to capture contextual information and long-range dependencies aligns with the linguistic complexities of Amharic. Comparative analyses are conducted to assess the effectiveness of the Transformer model against traditional parsing algorithms additionally, the Arc-Hybrid algorithm, known for its efficiency in parsing non-projective structures, is integrated to enhance parsing capabilities. The hybrid approach addresses Amharic's complex sentence structures and long-range dependencies. The utilization of Transformer models and the Arc-Hybrid algorithm showcases their potential in advancing the accuracy and robustness of dependency parsing for languages with complex linguistic structures. The proposed system is evaluated and achieves 94.58 % unlabeled attachment score and 84.2% and labeled attachment score.

Key words: dependency parsing**,** transformer model, arc hybrid, unlabeled and labeled attachment score

Chapter One

1. Introduction

1.1. Background

Natural Language Processing (NLP) enables computers to decode and comprehend spoken language. It is the foundation of many of the technologies we use on a daily basis, including grammar checkers, voice assistants, search engines, chatbots, machine translation, and software media monitoring tools. [1, 2]. The main advantage of NLP is that it enables humans to interact with computers without having to translate their queries and instructions into computer language, and the computers can do this by themselves. The goal of natural language processing is achieved by putting downstream NLP tasks such as speech recognition, machine translation, question answering, and information retrieval into practice [3, 4].

Natural language processing was designed to improve human to human or human to machine communication. But natural language is not in a form that can be easily processed or understood by computers due to natural languages needs different level of knowledge such as syntactic level, semantic, morphological, discourse and pragmatic [4,5]. For instance, in Amharic language one sentence can be interpret in multiple ways. Let see the sentence "በኮረብታው ላይ ያለውን ሰው በቴሌስኮፕ አየሁት።". It can be formulated in to "ቴሌስኮፕ ተጠቅሜ በኮረብታው ላይ ያለውን ሰው አየሁ።". From those sentences, we can understand that the linguistic structures of texts are required to be organized at different levels. Therefore, NLP requires an in-depth understanding of various terminologies and concepts to develop downstream NLP tasks to real-world scenarios [4, 5, and 6]. Broadly speaking, to process and understand natural language, it needs to have a knowledge about what the words mean, how words combine to form sentence meanings and so on [6]. In this study, we will cover, analyzing a sentence at the syntactic level typically entails breaking it apart into individual words. The syntactic level also shows which words modify other words, how the words are arranged into phrases, and which words are crucial to the meaning of the sentence [7].

Parsing is a process that used to analysis the syntactic level of natural language text [8, 9]. The process of deducing a text's syntactic structure from its individual words using an underlying grammar (of the language) is known as parsing in natural language processing (NLP). Moreover, parsing in natural language helps to analyze the input sentence in terms of grammatical constitutes, identifying the parts of speech, and syntactic relations [9]. Let consider this sentence "Alemu hits Kebede while he was on moving/ አለሙ ከበደን በእንቅስቃሴ ላይ እንዳለ መታው፡፡This sentence is ambiguous to understand: Who is on moving /ማን እየተንቀሳቀሰ ነው? Form the given sentence, to answer the question "who is on moving?" Parser can help by determining the relationship between words in a given sentence which mean that is that the word አለሙ/Alemu or ከበደ/Kebede can related to the word እንቅስቃሴ/moving on. This shows how parsers can help for developing for downstream NLP tasks like question answering, sentiment analysis, relation extraction, grammar checking, machine translation, and speech recognition [7,8,9].

Dependency parsing and constituent parsing are the two methods of parsing [8]. Constituent parsing is a task that involves grouping (chunking) a given text into phrase-level structure and determining whether the sentences are grammatically correct. In this type of parsing, the given sentence is divided in to constitutes, that is, sub-phrases that belong to a specific category in the grammar. The grammar provides a specification of how to build valid sentence using a set of rules [10].

Direct relationships between words or other tokens in a phrase serve as the foundation for dependency parse trees. Instead of determining if a sentence is grammatically correct, dependency parsing analyzes the representation and relations between words in a sentence and establishes links between the "head" words and the words that modify those words [11]. Recently, dependency parsing can be more useful for several downstream tasks like relation extraction, sentiment analysis or question answering [11, 12]. This is due to dependency parsing makes it easy to extract language structure triples (subject-verb-object for English and subject-object-verb for Amharic) that are often indicates of semantic relations between predicates. A study also prove that dependency parsing is advantageous when working with free word order languages [11].

Therefore, we propose to use dependency parsing in this study. Studies used either graph-based or transition-based parsing ways of techniques to build dependency relation [12]. In graph-based parsing, it uses to find scoring possible dependency graphs for a given sentence, usually by factoring the graphs into their component arcs searching for the highest scoring graph for a given sentence [13, 14]. The second method is Transition-based parsing; a sentence is parsed in linear time by which process the text word by word and build the dependency tree progressively. These methods also efficient on projective type sentence [15, 16].

Because of two factors, we choose to use transition-based dependency parsing in our study rather than graph-based dependency parsing. The first method, which is graphing-based, employs exhaustive search to identify the tree in the graph with the maximum spanning edge. More time and memory are required as a result using graph-based parsing [15]. The second is graph-based parser mainly efficient on non-projective sentences [16]. And the limited availability non-projective sentence in treebank for the Amharic language. So that, Transition-based dependency parsing will be used in this study.

In transition-based parser uses limited local information while making decisions and it uses a greedy nature of algorithm. This results to incorrect choices will lead to incorrect parses since the parser has no opportunity to go back and pursue alternative choices [17]. Incorporating global information about the entire sentence into the decision-making process is one technique to enhance the performance of a transition-based parser. The parser can make better local decisions if it is aware of both the structure it has already constructed and the words that are still to come.

This study aims to investigate how transformer helps to enhance a transition-based parser, particularly in terms of long-distance relationships between words. In this study, we will apply Transformer mechanism for incorporating global information about the entire sentence into the decision making.

1.2. Motivations

Dependency parsing is a fundamental task in NLP that helps to understand the grammatical structure and relationships within a sentence. While there has been significant progress in dependency parsing for widely spoken languages, there is still a lack of resources and tools for under-resourced languages like Amharic. This research can help bridge this gap and provide valuable resources for Amharic NLP.

A reliable dependency parser for Amharic can have practical applications in various domains. It can facilitate automated translation, text summarization, sentiment analysis, and other NLP tasks specific to Amharic. The work can contribute to the development of language technologies that benefit both native Amharic speakers and researchers working on Amharic language processing.

1.3. Statement of the Problem

The language patterns of texts must be arranged at many levels in order to process and comprehend natural languages, as structured text these days boosts the capabilities of NLP applications. [2], [4]. Analysis the structure or syntactic level of text focus on how words are grouping together to form correct sentences and determines what structural role each word plays in the sentence [4,7]. The syntactic level of analysis involves breaking a sentence down into its constituent words and organizing them into a specific syntactic structural unit. In order to determine the true meaning of a sentence, syntactic level analysis must also look at how the words in a sentence relate to its grammatical structure. A crucial component of natural language processing (NLP) is syntactic level analysis, which helps determine the grammatical meaning of each sentence. [5].

Parsing is the second steps of Syntactic level analysis next to part of speech tagging [5, 6]. Parsing is a process of identifying the role played by words in a sentence, interpret the relationship between words, and interpret the grammatical structure of sentences.

Parsing generally helps one comprehend the structural responsibilities of words and how they are placed together to produce sentences or phrases. It also helps to minimize the overall structural complexity of sentences, which is important for many NLP applications [4]. Semantic analysis, grammar checking, automatic abstracting, text summarization, machine translation, etc. are some NLP applications that use parsers as a component.

Amharic is the working language of Ethiopia at present-day, it is still one of less-resourced languages with limited linguistic tools available for Amharic text processing [18]. Since parser are identified as key components in many NLP applications, we will propose to develop transition-based dependency parser for Amharic using Transformer mechanisms. Hence, there have been developed Amharic parser using either ruled-based or data-driven based methods [24, 25, 26 and 27]. However, to the best of our knowledge, there is no similar work conducted on transformer-based dependency parser for Amharic language. As far as we know, transformer is one of the state art techniques of language modeling and it outperforms from other RNN based deep learning approach.

1.4. Research Questions

To this end, the current study attempts to explore and answer the following research questions.

- To what extent the use of transformer mechanism adds performance on Amharic dependency parsing system?
- To what extent the proposed approach extended-arc hybrid enables to design effective dependency parser for Amharic language?

1.5. Objective of the Study

1.5.1. General Objective

The general objective of this research is to develop Transition-based dependency parser for Amharic language using transformer approach.

1.5.2. Specific Objectives

To achieve the general objective, the following specific task are conducted.

- To review related works on Amharic and other languages
- To prepare and collect syntactically annotated Amharic sentence dataset (treebank).
- To investigate the syntactic structures and linguistics features of Amharic language.
- To design a network model for dependency parser for Amharic sentences.
- To evaluate the performance of the model using effectiveness measures.

1.6. Scope and Limitations of the Study

The focus of this work is design and develop transition-based Amharic dependency parser. We will use a manual annotated Amharic sentence dataset (Treebank). Our research needs morphological analyzed and part of speech tagging sentences. However, our work will not develop morphological analyzer and part of speech tagger tools. the proposed parser accepts grammatical correct Amharic sentence. In addition to this, Lack of availability of data and records to be used for data processing., Insufficiency of time and budget are some of the limitations.

1.7. Significance of the Study

Amharic language processing on computers is aided by the use of numerous computational linguistic tools, such as dependency parsers. As a result, the findings of this study have implications for numerous NLP applications involving the Amharic language. The primary beneficiaries, for example, are academics working on machine translation, Amharic question answering, grammar checks, relation extraction, spell checks, text summarizing, etc. Furthermore, the results of this study can be used by linguistic students studying Amharic to automatically parse sentences in the language.

The proposed Amharic parser helps:

- Task of paraphrasing is focus on rewriting the target sentence that exactly match in semantic or meaning but different in syntactic [30]. Let’s consider the sentence “አበበ አባቱ የሞተው በመኪና አደጋ ነው አለ።” and “አበበ በመኪና አደጋ ህይወቱ እንዳለፈ አባቱ ተናገረ።”. The second sentence is paraphrased by replacing the word “አለ” with “ተናገረ” and “የሞተው” with “ህይወቱ እንዳለፈ” and changing the position of words. However, the paraphrased expression is not semantically matched. This is caused by not considering the relationships of the words in the target sentence. Therefore, develop paraphrasing requires parser to determine the relationship between words in a sentence.
- Question Answering: When answering "በአፍሪካ ትልቁ ተራራ የት ይገኛል?" you need to parse the question and use parsed sentences to build the answer.
- Speech Recognition: While not an NLP task parser helps speech recognition involves choosing among many possible strings. Parsing scores the strings with either a pass/fail or a likelihood score, to give a powerful language model for speech recognition. Similarly, parsing could help in spell checking, optical character recognition (OCR), text prediction (T9), handwriting detection etc.
- Machine Translation: Again, parsing allows us to choose between several possible translations. Also, it makes translation of phrases and terms easier.
- Grammar Checking: Well, parsing can help in checking the grammaticality of a document, even though you could get some leverage out of several canned patterns.

1.8. Thesis Organization

The thesis is organized into five chapter. The first chapter starts discussing introduction about Natural language processing and dependency parsing, statement of the problem, objective of the study including general and specific, scope and limitation of the study, methodology followed including research design, data collection and preparation, tools, techniques, and evaluation metrics.

The second chapter discuss about literature on dependency parsing, about Approaches to work on that are categorized into grammar based and data driven. Type of neural network-based dependency parsing like feed forward, recurrent neural network and encoder-decoder and transformer also deals with an overview and discussion of Amharic language, and related work for both local and foreign language are discussed here.

The third chapter discusses the methodology the experimental setup, software tools used, the hardware environment, architecture of the system, the data used for the experimentation of the research.

The fourth chapter discuss about designing processes the experimentation, analysis, and the performance level of the system and discussion about the outcome.

The fifth chapter which is the last chapter discusses about conclusion and recommendation for future work.

Chapter Two

2. Literature review/ Related Work

2.1. Introduction

In this chapter, we begin with a brief introduction of syntax parsing. We present what has been established and accepted in dependency parsing. We also discussed problems or issues that remain unsolved on the area of dependency parsing. In addition to this, we review the emerging trends and new approaches of dependency parsing are presented. A detailed description of how our research extends, builds upon, and departs from previous research is presented. Finally, we presented a detail analysis of related works and summarizes the gaps of those related works.

2.2. Overview of Parsing

Parsing is the task of NLP that used to build syntactic structure of language and dependency structure of a sentence [33]. Based on these syntax parsing is divided in to two, constituent parsing and dependency parsing. Constituent parsing organizes a sentence in nested sequence of constituents or phrases for detecting the correctness of the syntactic structure of a given sentence. While dependency parsing is determining the head-dependent relationship that exist between words in the sentence [9].

Dependency parsing is useful for downstream applications of natural language processing like machine translation, synonym generation and relation extraction [6]. Dependency parsing takes sentence as input and generates dependency tree as output. In dependency tree, arcs (links) indicate certain grammatical relation between words and each word depends on exactly one parent. The tree starts with a root node. Dependency tree must be acyclicity, connectivity, projectivity or non-projectivity and have a single head. Dependency tree can be result to projective or non-projective. Dependency parsing is better over phrased based methods, especially for flexible word order languages [10].

2.3. Approaches Dependency parsing

Parsing looks at the relationships between various words and phrases in a sentence. Constituency parsing and dependency parsing are at least two different types of parsing [49]. Phrasal constituents are taken out of a sentence in a hierarchical manner using constituency parsing. Dependency parsing examines the connections between individual word pairs. Approaches to work on dependency paring are categorized into grammar based and data-driven approaches [34]. Grammar-based approach uses rule-based specification of grammars which is accompanied by a sentence-oriented view on syntax to analysis of a sentence with respect to the given grammar. Whereas data-driven approaches learn to produce dependency tree for sentences only from an annotated corpus [35]. The majority of deep learning applications in parsing that have occurred recently have been in dependency parsing, which has another significant difference in solution types. Graph-based parsing builds multiple parse trees, which are subsequently traversed to identify the right one. The majority of graph-based techniques are generative models, where the trees are built using a formal grammar derived from the natural language [14]. The first person to suggest the graph-based dependency parsing approach was McDonald (2005).

In graph-based models, the parsing process is search for the highest scored tree structure that spans all the words of the sentence and roots at an artificial node “ROOT” [26]. The first-order model, which is also called edge-factored [11], assumes that the score of a tree structure is the sum scores of independent edges. In second-order models, each sub-graph contains a pair of adjacent edges, like the same-side sibling edges [21 Recent years have seen a rise in popularity for transition-based techniques, which typically create a single parse tree, over graph-based techniques. A set of configurations, also known as parser states, and a set of transitions, also known as parse actions, for switching between configurations make up the transition system for dependency parsing in transition-based dependency parsing. [8]. Word by word, we parse a text using transition dependency parsing and gradually construct the dependency graph. The stack and the buffer, two data structures that store words from the sentence and indicate which portions need to be processed further, are what power the parser. While the words from the sentence are copied to the buffer, the stack is initially empty. We create the sentence's dependency tree by moving words from the buffer to the stack using a transition scheme. In transition-based parser, to specify the relationship between two words and eliminate the dependent of this relation from the stack, we employed transition actions or operations. The parsing procedure is complete when the stack is empty and the buffer only contains the root token. Although several changes have been suggested, the conventional approach to transition-based dependency parsing is to build a stack with just the ROOT label and a buffer for every word in the sentence. After that, words are stacked and links, or arcs, are drawn between the two highest-ranked things. After identifying dependencies, words are removed from the stack. Until the buffer is empty and the ROOT label is the only thing left on the stack, the operation is repeated. To control the circumstances under which each of the previously mentioned actions occurs, three main strategies are employed. All dependents are connected to a word before the word is connected to its parent in the arc-standard technique [50], [51]. Words are linked to their parents as soon as feasible in the arc-eager approach [50], [51], regardless of whether or not all of their offspring are linked to them. Lastly, the arc-standard approach is changed to permit position shifting on the stack in the swap-lazy approach [52]. This allows non-projective edges to be graphed.

2.4. Neural Network

The widespread adoption of neural networks in NLP is a result of their success in fields like computer vision, handwriting recognition, and speech recognition. Neural networks develop statistical models that define higher level characteristics as weighted combinations of lower-level features from raw input to accomplish classification tasks. The issue is essentially reduced to learning the best weights to apply across a range of levels of varied size between the raw input and the intended output. Learning such deep structures needed numerous features to be created manually outside of neural networks. This technique is time-consuming and frequently ineffective since it requires hundreds of features, some of which may not accurately represent the problem's parameters in all their complexity or may over specify them and cause over-fitting.

The introduction of dense word vector representations has also drawn attention to this method in NLP [(53).] As a result, it was no longer necessary to discretely represent individual words or word counts, and the issue of coming across terms that weren't part of the initial training set was somewhat resolved. The "curse of dimensionality," or the issue with sparse data, has finally been resolved, and many NLP applications no longer require massive input layers.

Numerous studies on deep learning-based syntactic parsing have been done recently. In the article [17], Chen and Manning presented a Dependency parsing model based on FNN within transfer for the first time in 2014. The feed-forward neural network (FNN) is a straightforward neural network model that performs a one-way transmission from the input layer to the output layer. The gradient, which was generated using a common back-propagation approach, was used to refresh the parameter. Pei et al. [19] developed a solution for the Graph-based dependency parsing model via FNN using a similar methodology. When compared to a previous Dependency parsing approach without neural networks, this one-use word embedding and POS-tag embedding, considerably reducing the amount of feature engineering required. Phrase embedding is calculated using the hidden cell vector of an FNN.

2.4.1. Recurrent Neural Networks (RNN)

One type of neural network used to analyze sequential data is RNN. RNN uses a recurrent network to carry out the same task over all instances of a sequence, with the output being reliant on earlier calculations and outcomes. A fixed-size vector is often created to represent a sequence by feeding tokens into a recurrent unit one at a time. RNNs have consumed memory for prior computations in order to utilize this data now. Many NLP applications, including language modelling, machine translation, speech recognition, and picture captioning, are well suited for recurrent neural networks [46]. RNN networks are susceptible to the vanishing gradient problem, which makes it more difficult to determine and modify the early layer network parameters. Many techniques were used to get over this restriction, including residual networks (ResNets), long short-term memory (LSTM), and gated recurrent units (GRUs). The first two are the most used RNN versions in natural language processing (NLP) applications. [16].

RNN is used in Natural Language Processing to process data sequences by reading a sequence data by data and generating an output at each time step in the sequence of input data. This output is based on the input at the current time step as well as the previous outputs. This enables RNN to handle context information in the outputs. RNNs, however, struggle with long-distance relationships between things. A former item's influence decreases with each subsequent time step until it disappears [40].

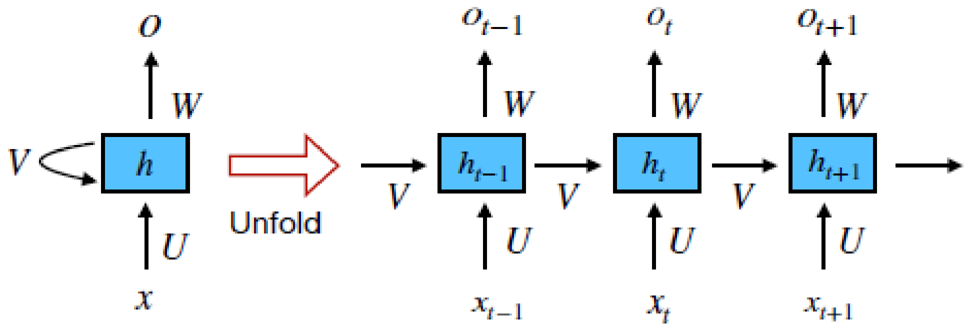


Figure 2.1 - A Visualization of a Recurrent Neural Network (RNN)

(Figure Source: Tom Young, et al. 2018, p 11)

2.4.2. Long Short-Term Memory

For several applications, including speech recognition, translation, and picture captioning, RNN has been successful. RNN has drawback that predictions can only be made once the complete sequence is available. Predicting the subsequent word from the previous words is the task in the case of language models. Given that it anticipates words in the future, RNN is obviously inappropriate. RNN will not provide good accuracy in this application. In addition to solving the problem of exploding/vanishing gradients, LSTMs capture long-term dependencies better than RNN [47, 49]. Long Short-Term Memory is one suggestion for addressing the issues with RNN. [39] introduced Long Short-Term Memory, a specific type of recurrent neural networks. The LSTM network contributes, as shown in figure 2.0.2.

The gradient vanishing issue was addressed with the development of long short-term memory (LSTM). Three "control gates" and one "memory cell" make up the LSTM, which regulates when to "memorize" and when to "forget." One input gate, one forgets gate, and one output gate are typical components of an LSTM. The recording rate at which present moment input is permitted to enter memory cells is determined by the input gate. The forget gate calculates the input moment's forgetting rate. The activation function comes after the forget gate, which first passes the current input and the previous output through the linear layer. The outcome is multiplied by points and applied to the cell state. Second, the cell gate and input gate both functions similarly. They are multiplied by each ingredient.

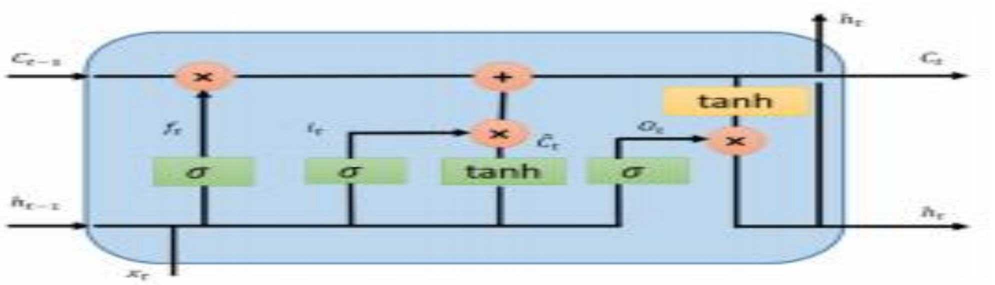


Figure 2.2 - A Visualization Long Short-Term Memory Cell

(Source: Abulwafa, Arwa .2022)

The horizontal line that crosses the top of Figure 2.2 represents the cell state, which is the central concept of LSTMs. LSTMs add or subtract information from the cell state, a process known as "gates." It is possible to define an input gate (i), forget gate (f), and output gate (o) as [49]:

$$f_t = \sigma (W_f . [h_{t-1}, x_t] + b_f) \quad (2.1)$$

$$i_t = \sigma (W_i . [h_{t-1}, x_t] + b_i), \quad (2.2)$$

$$\sim C_t = \tanh (Wc.[h_{c-1}, x_t] + b_c), \quad (2.3)$$

$$C_t = f_t * C_{t-1} + i_t * \sim C_t, \quad (2.4)$$

$$O_t = \sigma (W_o . [h_{t-i}, x_t] + b_o), \quad (2.5)$$

$$h_t = O_t * \tanh (C_t). \quad (2.6)$$

Bidirectional Long Short-Term Memory (Bi-LSTM) For forward and backward inputs produced by two distinct LSTMs, the model maintains two distinct states. The second LSTM handles the backward direction information of the input sequences and starts at the end of the sentence, whereas the first LSTM is a regular sequence that starts from the beginning of the sentence. Two distinct hidden layers are used by the bidirectional LSTM to process the data in both directions before feeding it into the same output layer. Both the forward and the backward concealed sequences are computed. [15, 18].

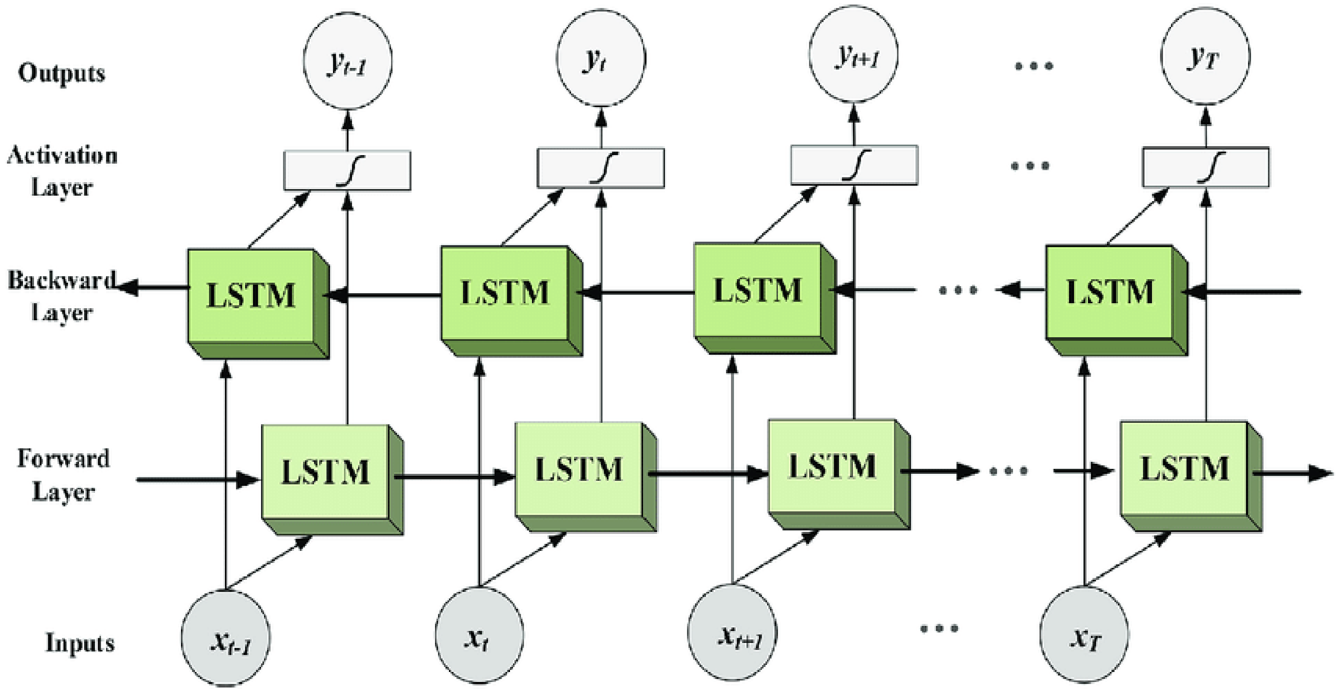


Figure 2.3 - Visualize BiLSTM Network

(source: Yung-Hui Li. et al., 2020 p. 9)

2.4.3. Gated Recurrent Unit (GRU)

With a minor variance in computing cost and model simplicity, GRU outperformed LSTMs; Figure 2.4 illustrates this. In terms of computing cost and complexity, GRUs are tinier variants of RNN techniques compared to ordinary LSTM [48]. GRU blends the cell state, hidden state, and a few more changes into a single update gate by combining the forget and input gates. The following formulas can be used to express the GRU mathematically [48].

$$z_t = \sigma\ (W_z.\ [h_{t-1},\ x_t]),\ (2.7)$$

$$r_t = \sigma\ (W_r.h_{t-1},\ x_t]),\ (2.8)$$

$$\sim h_t = \tanh\ (W.\ [r_t * h_{t-1},\ x_t]),\ (2.9)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \sim h_t.\ (2.10)$$

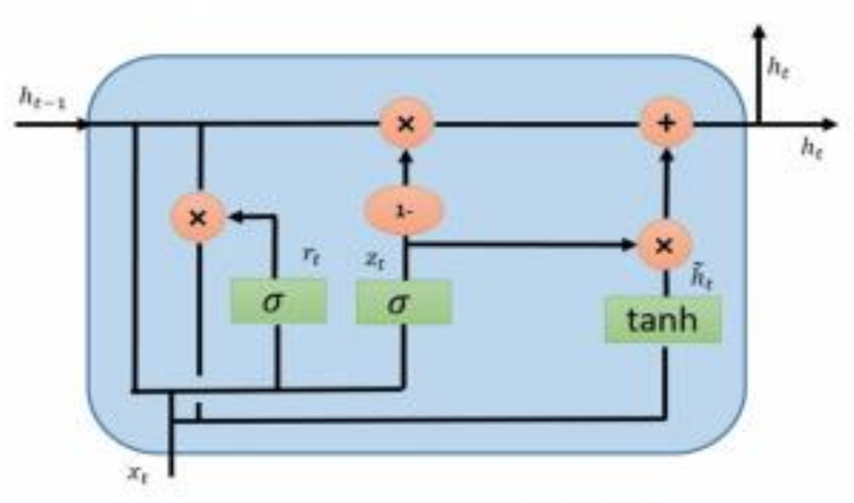


Figure 2.4 - Diagram for Gated Recurrent Unit (GRU).

(source: Jing Chen et al., 2021, p.10)

2.4.4. Transformer

In the paper "Attention Is All You Need" [47], Ashish Vaswani et al. presented the transformer. Transformers were created to deal with sequential data, like text synthesis. Transformers do not, however, process sequential data order, in contrast to other neural networks. Unlike LSTM, Transformer need to process the beginning of a sentence before the end, for instance, if the input data is a natural language sentence. This characteristic makes the Transformer far more parallelizable than RNNs, which previously helped shorten training times. Transformer is an example of an encoder-decoder architecture where input sequences are fed into the encoder, and the decoder predicts each word in turn. By removing RNN and using the attention mechanism, the Transformer can perform better in terms of time complexity and performance [41].

Transformer models without sequence aligned RNNs compute input and output representations exclusively from self-attention. Transformer unquestionably represents a significant advancement over seq2seq models based on RNN. However, it has some restrictions, the first of which is that attention can only handle fixed-length text strings. The second has context fragmentation issues. For instance, if a sentence is broken down the middle, a lot of contexts is lost. Thus, the text is divided without regard for the sentence structure or any other semantic boundaries [42].

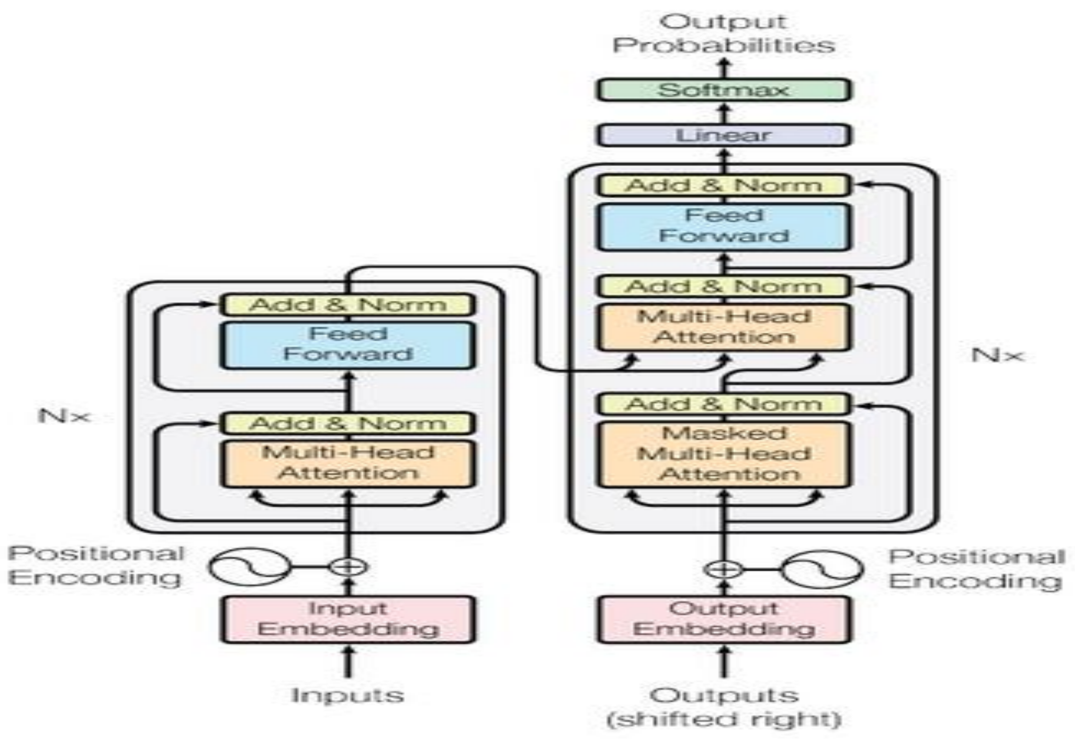


Figure 2.5 - Transformer Model Architecture

(Source: Vaswani A. et al. (2023), p. 3)

As seen in the Figure 2.5, Transformer's architecture has mainly two components: these are Encoder and Decoder parts. A feed-forward neural network and two self-attention mechanisms make up the encoder components. The self-attention mechanism creates a collection of output encodings by weighing the significance of each input encoding that was obtained from the previous encoder. Then, a feed-forward neural network handles each output encoding separately. At last, the subsequent encoder receives these output encodings as input. On the other hand, the decoder section is made up of three main parts: a feed-forward neural network, an attention mechanism over the encodings, and a self-attention mechanism. The input of the first decoder layer is positional data that has been encoded using positional encoding and embedding of the output sequence. The output sequence needs to be partially hidden to stop this reverse information flow, though, as the transformer shouldn't utilize the output's past or present to forecast its future. The final decoder is succeeded by a softmax layer and a final linear transformation, which generate the output probabilities over the vocabulary. [36, 41].

2.4.5. Multi-Layer Perceptron (MLP)

Deep neural network components called MLP are employed in classification jobs. The main component of MLPs is an arbitrary number of hidden layers that are sandwiched between an input layer to receive input data and an output layer to make a decision or prediction about the input. The network's final component, MLP, has output dimensions that correspond to the number of classes. To create a probability distribution over the classes, a Softmax function is frequently applied to the output [48]. Because every unit in a layer is connected to every other unit in the layer above it, MLP has at least fully connected layers. The specifications for each unit in a fully connected layer. MLPs have the same input and output layers but may have multiple hidden layers in between the aforementioned layers, as seen in Figure 2.6 below.

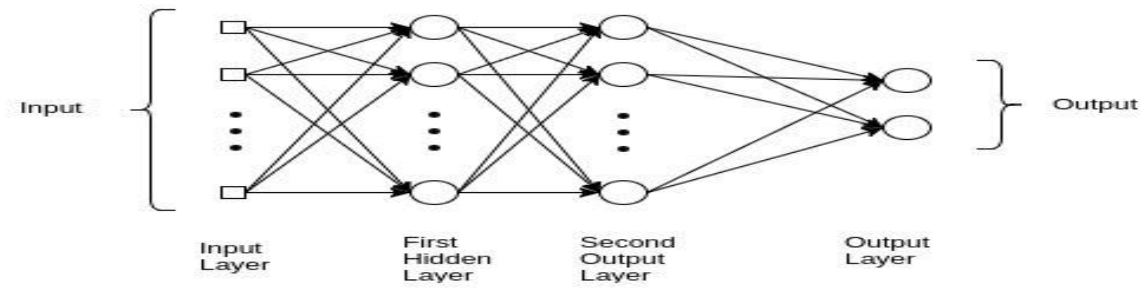


Figure 2.6 - MLP Component

(Source: https://medium.com/@AI_with_Kain/understanding-of-multilayer-perceptron-mlp-8f179c4a135f)

By taking the dot product of the input with the weights that are present between the input layer and the hidden layer, the MLP algorithm processes the input layer and passes the input data. An input value is produced at the hidden layer by this dot product. MLPs use activation functions such rectified linear units (ReLU), sigmoid function, and tanh at each of their computed levels in the hidden layer. The computed output at the hidden layer is then pushed to the next layer in the MLP by taking the dot product with the appropriate weights after it has passed through the activation function. And it continues till it reaches the output layer. In the event of training, the computations at the output layer will be applied to a backpropagation algorithm matching the activation function chosen for the MLP; in the event of testing, a decision will be taken based on the output. [49].

2.4.6. Regularization Techniques

Dropout

One of the most efficient methods for preventing overfitting in a neural network is dropout. Dropout is a parameter that causes the network to accept partial inputs by randomly deleting a portion of its input. Dropout also makes it more difficult for the model to simply memorize the data points. It must instead focus on making the input broader. Dropout is directly applicable to the input values. Dropout can be used on the output of a recurrent neural network or after a feed-forward layer [47].

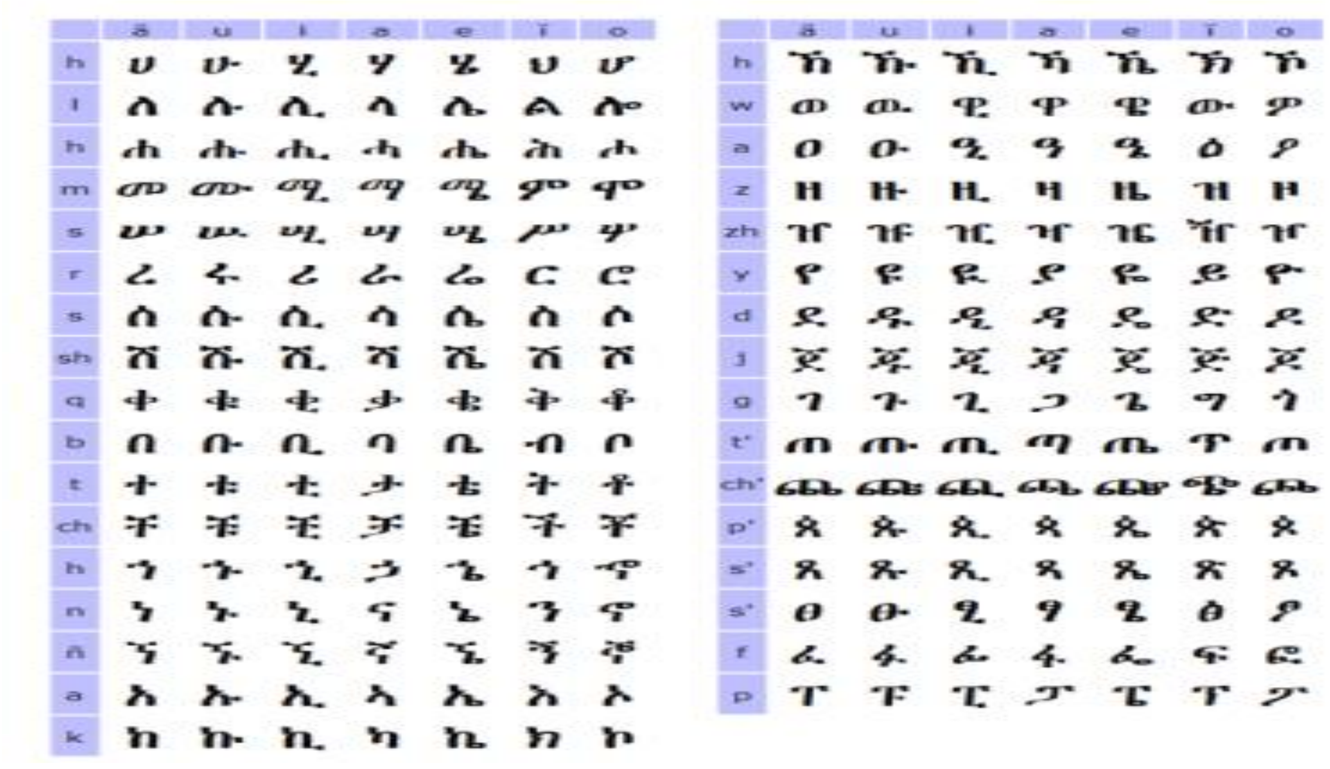
Activation Functions

A neural network may use an activation function to aid in the network's ability to recognize intricate patterns in the input data. The output layer of the neurons is divided using the activation function to determine output [50]. One of the earliest activation functions, the sigmoid function, turns an input number into the range between 0 and 1, giving it an excellent fit to create probability distributions. The hyperbolic tangent function (tanh), however, consistently produces better results. Sigmoid-like in appearance, it turns the values into a range between 1 and 1. The tanh function is frequently employed in recurrent neural networks in particular because its upper bound.

2.5. Amharic language

Amharic (አማርኛAmarigna) is a Semitic language that is spoken mainly in Ethiopia. Though there are many languages that are spoken throughout Ethiopia (including Amharic, Tigrinya, Oromia/Affan Oromo, etc.). Since it is the working language of the Ethiopian government, it has gained an official status, and it is used throughout the country [29]. ፊደል (Fidel) is the Ge'ez script form used to write Amharic. In the syllabary writing system known as Fidel, vowels and consonants coexist inside each graphic sign. The 33 fundamental letters in the language each represent a consonant and have seven form variations that indicate the vowel that comes after the consonant. (see Appendix A).

Amharic language has 11 basic tag sets noun, pronoun, adjective, adverb, verb, preposition, conjunction, interjection, punctuation, numerical and unknown (for words that are difficult to classify) (see Appendix B). Amharic language is morphologically rich languages that makes a challenge to the area of NLP particular in parser. Morphologically-rich languages in the sense that grammatical relations like subject, object, etc. or word arrangements and syntactic information are indicated morphologically or at sentence level [28, 29].



	ä	u	i	a	e	ĭ	o
h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
h	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
m	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
s	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
r	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
s	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
sh	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
q	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
b	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
t	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
ch	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
h	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
n	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
ñ	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
a	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
k	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ

	ä	u	i	a	e	ĭ	o
h	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
w	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
a	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
zh	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
d	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
j	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
g	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
t'	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
ch'	ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
p'	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
s'	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
s'	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
f	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ

Figure 2.7 - Amharic script

(source: https://www.amharicmachine.com/default/alphabet)

2.6. Related Work

2.6.1. Dependency Parsing in English Language

Recently, neural network models have been increasingly focused on for their ability to minimize the effort in feature engineering. With the beginning of deep learning, hand-crafted features were gradually replaced by creating a suitable neural network which makes automatically learns features from the input data.Chenet.al [37] suggested a method for automatically figuring out feature embedding for dependency parsing based on graphs. In a traditional graph-based model, the learned feature embedding is employed as an additional feature. A broad and efficient Neural Network model for graph-based dependency parsing was described in a paper by Pei et al. [19]. By taking advantage of a novel activation function called the tanh cube, their model is able to automatically learn combinations of high-order characteristics using only atomic features.

Additionally, they suggested a straightforward but efficient method for making use of phrase-level data, which is costly to employ in traditional graph-based parsers. A linear scoring function was employed. Additionally, BiLSTM-based architecture without any extra features that relies on the input sequence's embedding was proposed by Kiperwasser et al. [38]. The input embedding was contextualized using a stacked bidirectional LSTM, which was subsequently applied to a multilayer-perceptron with one hidden layer and a tan activation function in between. The optimal course of action is then indicated by the output logits. Additionally, by training a second-stage arc-labeler that shares the same BiLSTM encoder as the unlabeled parser, they presented a novel multi-task learning strategy for labeled parsing. A graph-based dependency parser influenced by Kiperwasser and Goldberg's basic architecture [38] was first mentioned in research by [20]. Adding a biaffine layer, which is utilized to calculate the score of each potential head and their dependents and use the resulting distribution to construct a dependency tree, is one of their contributions. Though the terms "head" and "dependent" in the statement relate to the same token, the dependency tree's directional edges dictate that a word's representation should alter depending on whether it is a head or a dependent. The paper also offers a number of recommendations for improving the model's functionality, such as using embedding dropout.

2.6.2. Dependency Parsing in Semitic Languages

Semitic languages are morphology rich and word-order patterns are different from that of English. Semitic languages are written right-to-left such as Arabic language, whereas Amharic language follows left-to-right written system. In Modern Semitic languages including Modern Hebrew and many Arabic dialects this has given way to a Subject, Verb-Object (SVO) default structure [44]. A study by Tsarfaty [17] presented the challenges of morphological rich language on parsing process. The first one is morphological rich language is creating lexical challenge on parsing. The second one is architectural challenge. The challenge exists in the preprocessing stages (Segmentation, part of speech tagging, etc.) of parsing. The other one is modeling challenge on parsing system. This challenge is about incorporation of morphological information in syntactic model. Similarly, Goldberg et.al [44] made experiments on Hebrew dependency parsing. In this study in addition to POS, they used morphological information as an input to improve the parsers" performance; inappropriately, the performance of the system did not display a substantial improvement comparative to original parsing systems. A study by Marton et al. [45], presented an experiment on dependency parsing to modern standard Arabic sentences. In their study, they used Maltparser by augmenting inflectional and lexical features) one at a time in a heuristic fashion to explore the contribution of the addition of morphological features to improve the performance of the parser.

2.6.3. Dependency parsing in Amharic Language

When dependency parser models which perform best on English are used to morphologically complex languages like Amharic, they don't work as well.[17]. The main reason is their basic word order which means that Amharic language follow the grammar structure of subject object verb (SOV). Whereas English languages follow the grammar structure of subject-verb object (SVO).
Therefore, there are many researchers conducted on syntax parser for Amharic language. Some of the works are presented as follows: -A study by [23] presented automatic sentence parsing for Amharic text. He develops the first Amharic sentence parser using probabilistic context free grammar (PCFG) and rule-based reasoning. They represented a grammatical rule of the language's phrase structure using probabilistic context-free grammar. Nevertheless, the constructed parser has limitations with regard to corpus size, sentence length, and sentence type. The parser was tested on a dataset of only 100 sentences, and it only accepts simple declarative sentences of length four words. A phrase-based chunkier and a Hidden Markov Model (HMM) were also built in the study [24] to provide a chunkier approach and a bottom-up method with a transformation algorithm to convert the chunk to the parser. The system was trained using 288 sentences in the study, and it was tested using 32 sentences. However, because chunking does not provide a thorough enough examination of texts that are impacted by the morphological complexity of the language, the authors propose chunking as a solution to manageable parsing problems.

A study by Dawud [25] proposed a top-down chart parser for Amharic sentences. They used Context Free Grammar approach for developing grammar rules and top-down chart to build the parses based on developed grammar rules. However, the dataset used in the study is not quite enough to judge the effectiveness of the parser and performance of the morphological analyzer has an impact on the effectiveness of the sentence parser. Gasser [26] proposed to develop a dependency grammar for Amharic using XDG (Extensible Dependency Grammar). The contribution of Gasser is only as the proposed framework was not tested. It offers no evaluation data at all. Furthermore, his work doesn’t consider morphological structure difference between languages. Maltparser [27] is develop as a universal dependency parser. This parser is language independent dependency parser that used to parse sentence. However, attempts to develop language independent parser with complex word structure, flexible word order and multiple level of information yielded unsatisfactory results. Zelalem [28] presented a transition-based dependency parser for Amharic language using deep learning. As Zelalem points out, the parser only works on projective dependencies as Zelalem work's, they used LSTM networks to handle the sequence of sentences.

Therefore, the aim of our study is to design and develop a Transition-based dependency parser for Amharic language using Transformer. Transformer is one of the state art models which address the gaps of LSTM networks. In this study, we will propose to use Transformer for getting attention mechanism while creating dependency relation in a sentence.

Table 2.1 summarizes research works on Amharic syntax parser. We try to present the authors works as well as the approach used. We try to discuss the limitation of author's works and methodology used to conduct the research.

Table 2.1 - Summary of Research Work On Amharic Syntax Parser.

No	Author	Used approach	Limitation of work& approach gap
1	Alemu Atelach	Probabilistic Context Free Grammar (PCFG)	• Limited sentence type: decelerative sentence only • Sentence length: only 4 words • Size of corpus; used only 100 sentences
2	Dawud Abdurohman	Context Free Grammar (CFG) and top-down chart parser	• Limit of research dataset • Their approach is suitable for fixed structure language like English
3	Ibrahim Abeba	Phrase Chunk and Hidden Markov Model (HMM)	• Chunking face on managerial problem on parsing process
4	Gasser	Extensible Dependency Grammar (XDG)	• Their approach is only having theoretical contribution: does not test with test data. • Their work is not considered language structure difference
5	Maltparser	universal dependency parser	• Their work is not considered language structure difference
6	Zelalem Mizanu	Transition based dependency parsing using LSTM	• Their approach mainly efficient for projective dependency relation • Their work also guaranteed to only projective sentences
A study by (Alemu, Dawud, Ibrahim) generally used rule-based approach to parse a given Amharic sentences. However, rule-based approach has a kind of constraint satisfaction problem; there might be more than one parses satisfying the constraints. In addition, the approach is not suitable for free word order language and morphological rich languages [10,11]			

2.7. Summary

In this chapter, we have reviewed literatures on the importance of syntax parsing and approaches syntax parsing. Theoretical backgrounds about Amharic language are discussed in detail. The approaches dependency parsing is also discussed thoroughly. We give detailed definition and analysis deep neural network especially RNN and LSTM architectures. The characteristics and how each layer operate in LSTM was explained thoroughly. We also reviewed Multilayer perceptron classifier, embedding such as word embedding techniques. We have discussed about Amharic language. Finally, we presented the summary related works related to Amharic language.

Chapter Three

3. Methodology

3.1. Introduction

In this chapter, detailed description of the proposed system a transition-based dependency parsing is discussed. The proposed model requires to pass through a series of steps starting from preprocessing Amharic sentences, build parser state configuration, learn pattern of parser state configuration, and determining the dependency relation of words in a sentence. In the following sections, the general description about the proposed system architecture is presented.

3.2. Proposed System Architecture

In this study, we propose Transition based dependency parsing using Transformer neural network. Two distinct phases can be distinguished in dependency parsing: Learning phase: In order to determine the dependency tree of an input phrase, a model must be learned using a text corpus in which every word has been annotated with its head and the kind of reliance it has on it. Parsing phase: Output the dependency tree of an input text using the model. Transition-based parsers apply a series of transitions to a sentence in order to create a dependency tree for it. Creating a model that forecasts the next most likely transition for a given partially parsed text is the task of the learning phase.

The proposed system has four components: preprocessing step, building Transition System, learns the pattern of Transition System, and parsing. In data preprocessing, additional collected Amharic sentences are transforming into Amharic Treebank format. Then, we build the transition state configuration from the given Amharic sentence. In third phase, learn the pattern of transition system configuration through transformer network. Finally, using the trained neural network model, parses the given input sentence.

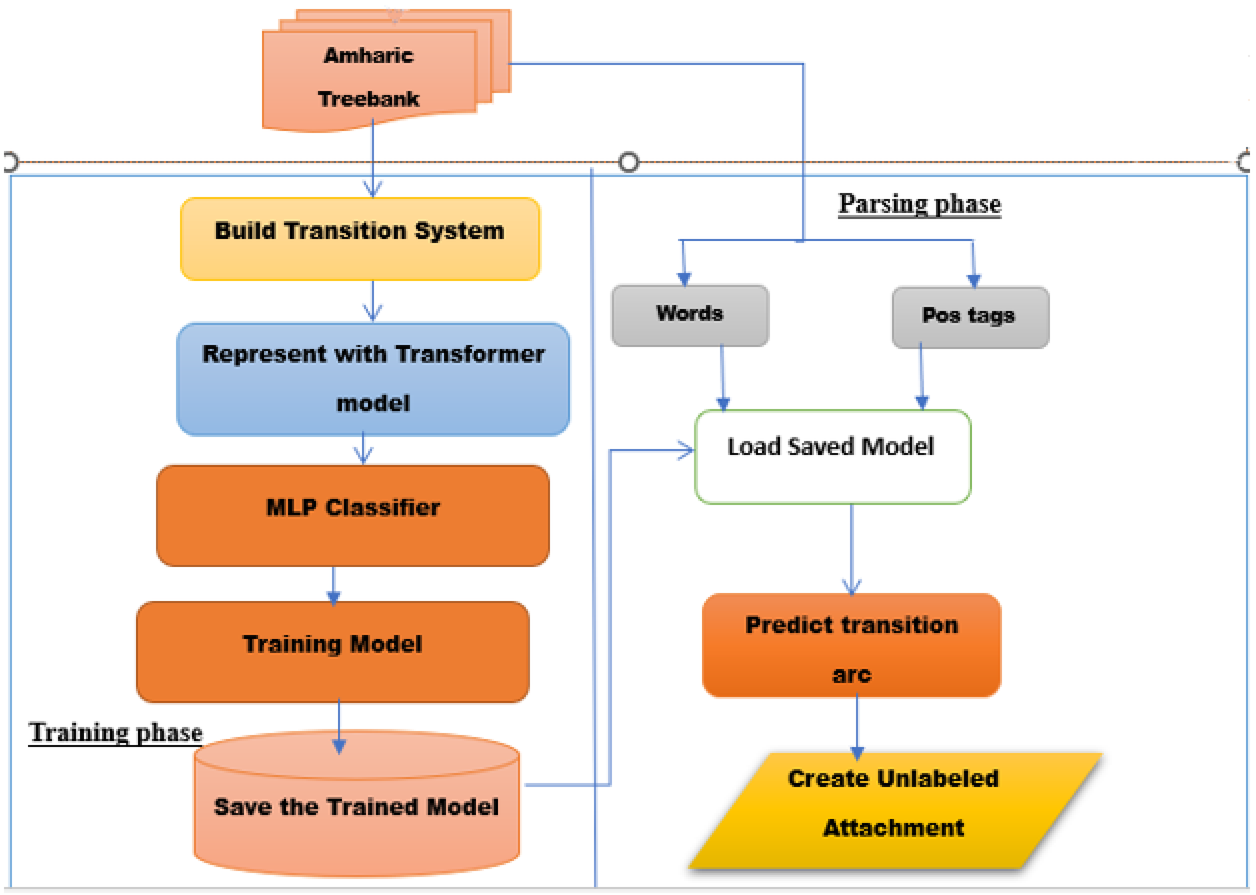


Figure 3.1 - Proposed System Architecture

3.2.1. Data Pre-processing

Data preprocessing is used to transform the raw data or collect Amharic sentence into a useful and efficient format. In this step, we involve two sub-parts steps: data collection and data preparation. In our study, we used Amharic Treebank which adopted from UD-Amharic treebank to train and test proposed neural network model. The Treebank was passed a series of steps starting from collecting grammatical correct Amharic sentence up to annotating POS tags, morphological features, and syntactic relations of the sentence. In addition to the Treebank dataset, additional Amharic sentences were constructed in collaboration with linguistic experts.

Data collection:

In this step, in addition to Amharic Treebank, we also collect 500 additional Amharic sentences from grammar books, fictions, biographies, and news. The collected sentences must be grammatical correct and needs data preparation steps before feed to neural network model.

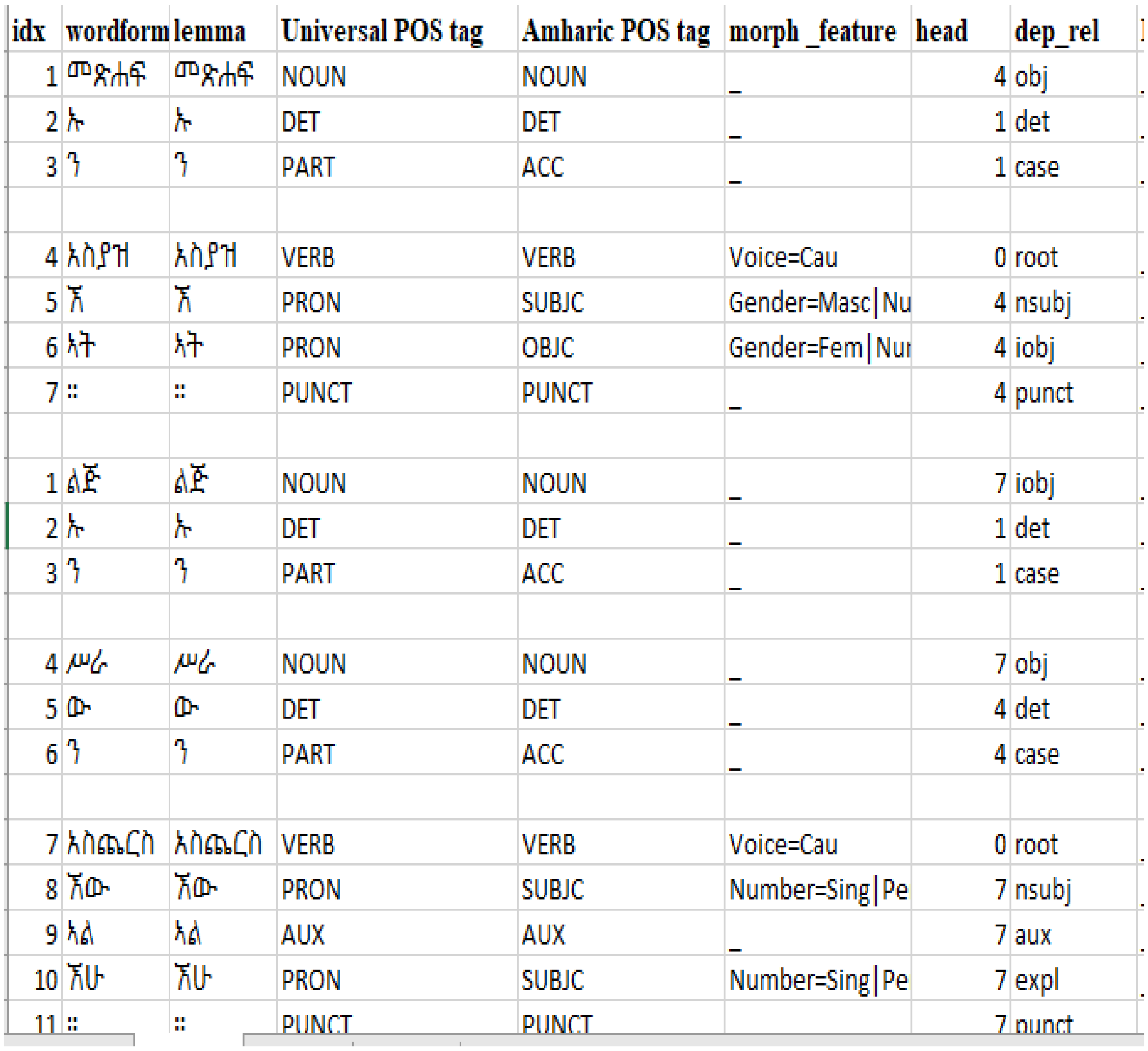
Table 3.1 - Sample Collected Sentence

NO	Sample collected sentence
1	ሚስጥሩ ገባኝ ።
2	በቅሎዎቹ ስለደከሙ ጉዞውን አንቀጥል።
3	እሱ አባቱን ይመስለዋል።
4	ነገ የሚመጣ ይመስለኛል።

Data Preparation:

In this step, we perform morphological analysis, POS tag and syntactic relations on collected sentence. The first stage before syntax analysis is morphological analysis. Morphology analysis used to break strings of language input into sets of tokens corresponding to discrete words, sub-words and punctuation forms [29]. To perform morphological analysis, we have used linguistics experts, in addition to morphological analyzer called Horn Morph (Gasser, 2011).

For instance, the sentence “ልጁን ሥራውን አስጨርሰዋለሁ” to which that is found in our datasets will be segmented into [ልጁን] ልጅ_ኡ_ን [ሥራውን] ሥራ_ው_ን [አስጨርሰዋለሁ] አስጨርስ_ኧው_ኣል_ኧሁ።. Then after, we perform part of speech tag for each word in the morphology analyzed sentences. We have used habit tagger to tag the sentence. The POS tag result of the sentence “ልጅ_ኡ_ን ሥራ _ው_ንአስጨርስ_ኧው_ኣል_ኧሁ።” is ልጅ: NOUN, ኡ: DET, ን: PART, ሥራ: NOUN, ው: DET, ን: PART, አስጨርስ: VERB, ኧው: PRON, ኣል: AUX, ኧሁ: PRON and።: PUNCT. In this step, we also perform the syntactic relationship between words in a sentence. Finally, the dataset is ready to feed to our proposed neural network model. We have used 1574 Amharic sentences to train and test the proposed model.



idx	wordform	lemma	Universal POS tag	Amharic POS tag	morph _feature	head	dep_rel
1	መጽሐፍ	መጽሐፍ	NOUN	NOUN	_	4	obj
2	ኡ	ኡ	DET	DET	_	1	det
3	ን	ን	PART	ACC	_	1	case
4	አስያዝ	አስያዝ	VERB	VERB	Voice=Cau	0	root
5	ሽ	ሽ	PRON	SUBJC	Gender=Masc|Nu	4	nsubj
6	ኣት	ኣት	PRON	OBJC	Gender=Fem|Nu	4	iobj
7	።	።	PUNCT	PUNCT	_	4	punct
1	ልጅ	ልጅ	NOUN	NOUN	_	7	iobj
2	ኡ	ኡ	DET	DET	_	1	det
3	ን	ን	PART	ACC	_	1	case
4	ሥራ	ሥራ	NOUN	NOUN	_	7	obj
5	ው	ው	DET	DET	_	4	det
6	ን	ን	PART	ACC	_	4	case
7	አስጨርስ	አስጨርስ	VERB	VERB	Voice=Cau	0	root
8	ሽው	ሽው	PRON	SUBJC	Number=Sing|Pe	7	nsubj
9	ኣል	ኣል	AUX	AUX	_	7	aux
10	ሽሁ	ሽሁ	PRON	SUBJC	Number=Sing|Pe	7	expl
11	።	።	PUNCT	PUNCT		7	punct

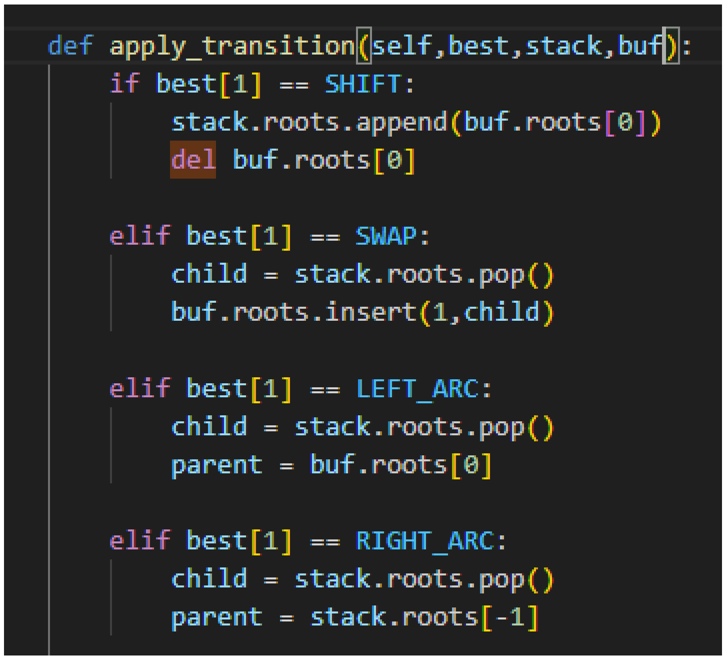
Figure 3.2 - Sample Amharic Sentences from Collected Dataset

3.2.2. Build Transition System

Arc-standard, arc-eager, and extended arc-hybrid are a few models that can be used in transition-based parsing to develop transition systems [32, 35]. We have employed extended arc-hybrid in our context scenario, which enables more robust decision-making when creating word attachments. Additionally, it uses a wider range of global data to determine the transition and can handle non-projective trees, in which a word might have several parents via the SWAP transition that Nive first proposed. [34].

The extended arc-hybrid tracks words and their dependence relationships to create a parse tree that illustrates the syntactic relationships between words in a phrase using a stack, buffer, transition rules (reduction, Left Arc, Right Arc, swap, and shift), and configuration [36, 37].

The algorithm utilizes a number of transitions to get from the beginning configuration to the terminal configuration. The root token is stored on the buffer after the entire phrase in the start configuration. The transitions include the following operations: SHIFT, LEFT ARC, RIGHT ARC, and SWAP, which is a reordering operation. [48]



```python
def apply_transition(self,best,stack,buf):
    if best[1] == SHIFT:
        stack.roots.append(buf.roots[0])
        del buf.roots[0]

    elif best[1] == SWAP:
        child = stack.roots.pop()
        buf.roots.insert(1,child)

    elif best[1] == LEFT_ARC:
        child = stack.roots.pop()
        parent = buf.roots[0]

    elif best[1] == RIGHT_ARC:
        child = stack.roots.pop()
        parent = stack.roots[-1]
```

Figure 3.3 - Visualizations of the Extended Arc-hybrid Transition System

Table 2.2 - A Processing Trace of a Sentence Using Extended Arc-hybrid Transition System

No	**Stack information**	**Buffer information**	**Transition arc**
1.	[root]	ገመድ, ኡ,ን, አስረዘም, ኧ, ው, ።	Shift
2.	[root, ገመድ]	ኡ,ን, አስረዘም, ኧ, ው, ።	Sw
3.	[root]	ኡ,ገመድ, ን, አስረዘም, ኧ, ው, ።	Sh
4.	[root, ኡ]	ገመድ, ን, አስረዘም, ኧ, ው, ።	LA
5.	[root]	ገመድ,ን, አስረዘም, ኧ, ው, ።	Sh
6.	[root, ገመድ]	ን, አስረዘም, ኧ, ው, ።	Sw
7.	[root]	ን, ገመድ,አስረዘም, ኧ, ው, ።	Sh
8.	[root, ን]	ገመድ,አስረዘም, ኧ, ው, ።	LA
9.	[root]	ገመድ,አስረዘም, ኧ, ው, ።	Sh
10.	[root, ገመድ]	አስረዘም, ኧ, ው, ።	LA
11.	[root]	አስረዘም, ኧ, ው, ።	Sh
12.	[root, አስረዘም]	ኧ, ው, ።	Sh
13.	[root, አስረዘም,ኧ]	ው, ።	RA
14.	[root, አስረዘም]	ው, ።	Sh
15.	[root, አስረዘም,ው]	።	RA
16.	[root, አስረዘም]	።	Sh
17.	[root, አስረዘም, ።]	-	RA
18.	[root, አስረዘም]	-	RA
19.	[root]	-	-

3.2.3. Proposed Transformer model

Modeling the transition state is the key to good performance in transition-based parsing. Building transition states typically comprise two memories, a buffer, and a stack, from which tokens can be pushed or popped. Nevertheless, earlier attempts at Amharic dependency parsers use LSTM-based transition-based algorithms that predict transitions by using local sentence information [38]. In this work, we applied a transformer model that leverages both global and local information to generate parse, particularly with respect to long-distance word relations. According to the publication Attention Is All You Need [39], the Transformer was proposed. In our context, we feed the output of transition state configuration into Transformer network for encoding the sequence of stack, buffer, and transition arc states. The proposed Transformer architecture has two parts, the Encoder (left) and the Decoder (right).

The input of the encoder components is the sentence's stack, buffer, and transition arc information; the output is a set of encoded vectors for each word. To express meaning, each word in the input stack, buffer, and transition state is transformed into an embedding. The context of the word in the sentence is then added by adding a positional vector. The Encoder attention block receives these word vectors and uses them to calculate the attention vectors for each word. A feed-forward network receives these attention vectors in simultaneously, and the result is a set of encoded vectors for each word.

After receiving inputs from the input embedding layer, the positional encoders apply relative positioning information. Word vectors with positional information that is, the meaning of the word and its context in the sentence are produced by this layer. The lines "አበበ ውሻን መታው።" and "ውሻ አበበን መታው" should be considered. In the absence of context, the embeddings of the two sentences would be nearly identical. However, we are aware that this is untrue certainly untrue for አበበ. The transformer model's last layer is in charge of forecasting the result for a specific input sequence. Using a linear layer with a softmax activation function is required for this.

3.2.4. MLP Classifier

Multi-layer Perceptron (MLP) is an artificial neural network model that used mapping the given input data onto a set of appropriate outputs. It consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, other layer uses a nonlinear activation function. MLP needs a combination of back propagation and gradient descent for training [42, 44].

In this study, we used MLP. It is used to predicting score of Arc and label pairs of word in a sentence. To determine score of arcs as head or dependent word in a sentence. Then MLPs applies rectified linear unit’s activation functions to generate the outputs. Then push the calculated output at the current layer. Push the calculated output at the hidden layer through the activation function, then use the dot product with the relevant weights to move it to the next layer in the MLP. This process continues until the output layer is reached. The score of head and dependent word pairs will be generated at the output layer.

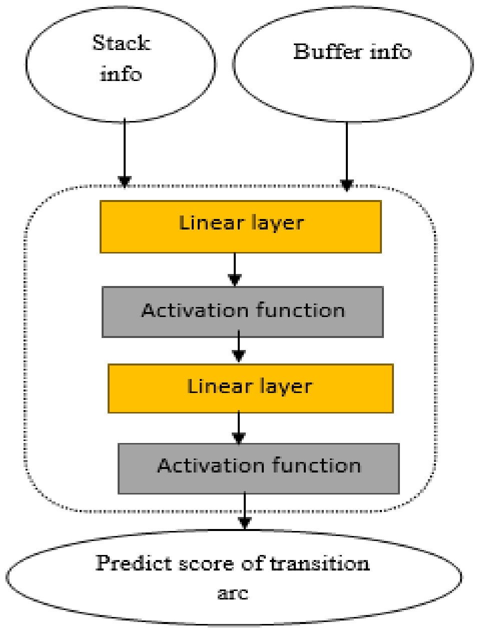


Figure 3.4 - Visualize MLP Classifier in our Parser

3.3. Parsing phase

In this phase, the proposed model is finding a sequence of transition that leads from the start state to the desired goal state. In the start state, the model initialized the stack with the root node, and buffer initialized with words in sentence. In addition to this, the model set the dependency relation set with empty. In goal state, the trained model finalized with stack and buffer empty and generate the set of dependency relations.

Token representations from a sentence are initially fed through the Transformer network during parsing in order to produce contextualized token vectors. After then, the parsing loop is continued until the buffer only holds the root token and the stack is empty, which is the final configuration.

Using trained network models, we create a dependency relation for a new Amharic text in this phase. In addition to parsing a sentence, we also calculate scores for each potential transition arc and use the greedy approach to choose the ideal transition tree. The parsing process is shown in the following steps, which we go into additional detail about in the ensuing sections.

Step 1: **input Amharic sentence**: -feed as sequence of Amharic sentence to the parsing. Sample sentence taken from the Amharic Treebank: ካሳ አስተማሪ ነው።".

Step2: **Morphological analysis and POS tagging**: The input sentence is subjected to part-of-speech tagging and morphological analysis in this step. We carry out this operation manually with the technique described on UD-Amharic Treebank because it is outside of our purview.. For the sentence given above, the output of this step is shown Table 3.3

Table 3.3 - The Result of a Morphology Analysis and POS Tag for Sentence: “ካሳ አስተማሪ ነው::”

Word	Morphology Analysis	POS tag
ካሳ	ካሳ	PROPN
አስተማሪ	አስተማሪ	NOUN
ነው	ን	AUX
	ኧው	SUBJC
::	።	PUNCT

Step 3: Initializing Words and POS tag: in this step; the network model takes words and corresponding POS tags as input. Therefore, we need initialize word and POS tag states from the given input sentence. The initial state also adds an artificial token called “root‟ which used determine head word for the root word of the sentence. As shown Table 3.4.

Table 3.4 - Initializing Words and POS Tags.

Word	POS tag
Root, **ካሳ, አስተማሪ, ን, ኧው, ።**	Root, PROPN, NOUN, AUX, SUBJC, PUNCT

Step4: predicting transition arc: In this step; the trained network model is loads to predict the possible score of each transition. The model uses the value of Words and its POS tags states for predicting the score. Finally, the model generates the score arc and labels for each pair of words in a sentence.

Step5: Apply greedy algorithm: in this step; the model used greedy to select the highest score of transition arc to build unlabeled dependency tree. The algorithm uses the score of each possible transition arcs for each word in a sentence as input. Then the algorithm greedily selects the highest score of incoming words as head token, and it works for each word in a sentence. If the final results form a cycle, the algorithm recalculates the edge weights going into and out of the cycle until the final result is tree [11]. Finally, the algorithm the constructed unlabeled dependency structure of the sentence as shown Table 3.5

Table 3.5 - Unlabeled Dependency of Structure of Sentence

Index	**word**	**POS tag**	**Index of head word**
1	መጽሐፍ	NOUN	4
2	ኡ	DET	1
3	ን	ACC	1
4	አስያዝ	VERB	0
5	ኧ	SUBJC	4
6	ኣት	OBJC	4
7	።	PUNCT	4

Step 6: Generate labeled dependency. In this step, the model uses predicted score label between pairs of (head, dependent, labels) and unlabeled dependency, which is result from maximum spanning algorithm as input, then the parser selects the highest score label as relationship type between pairs of words. Finally, generates the labeled dependency of the sentence as shown Table 3.6.

Table 3.6 - Labeled Dependency Structure of the Sentence Predicted by Model

Index	**Word**	**POS tag**	**Index of head word**	**Relation type**
1	ይህ	Det	3	advmod
2	አል	Neg	4	advmod
3	ይ	SUBJC	4	nsubj
4	ታበል	VERB	0	Root
5	ም	NCM	4	Discourse
6	።	PUNCT	4	Punct

3.4. Evaluation Metrics

To measure the accuracy of the system we used two metrics. These are unlabeled attachment score and labeled attachment score. Unlabeled attachment score (UAS) is used to measure the accuracy of the system for building the head-dependent relationship between entities in a sentence without considering the relationship type. Labeled attachment score (LAS) is used to measure the accuracy of the system for building a head-dependent relationship including the relationship type existing between entities in a given sentence.

3.5. Development environment

Python is easy to get NLP packages and it is powerful in text processing. We have used Python programming language to write the source code of the parser system.

Tensor flow

A free and open-source software library called Tensor Flow is used for differentiable programming and dataflow in a variety of applications. Neural networks and other machine learning applications use it. Tensor Flow is a Python numerical computation framework that provides abstraction for a variety of operations, such as backpropagation for adjusting weigh tensors, thus facilitating faster and simpler machine learning. When training the network model, we updated the weight matrix using it, among other numerical operations.

3.6. Summary

This chapter described the architectural design, development environments and evaluation metrics for the proposed Amharic dependency parser. The parser has two phases, training phase and parsing phase. In training (learning) phase two network models are trained on Amharic treebank for predicting score of arc and labels (relationship type). In the parsing phase, the system uses the trained network models to predict score of arc and relationship types for building unlabeled and labeled dependency relation for a given sentence. These score of arcs are used to build unlabeled dependency relation by using Maximum Spanning Tree (MST) algorithm. Finally, the system uses unlabeled dependency tree and predicted score of labels to find the labeled dependency relation for the given Amharic sentence.

Chapter four

4. Experimentation and Discussion

4.1. Introduction

This chapter provides a detailed description of the experimental evaluation of the suggested Transformer model for transition-based dependency parsing for the Amharic language. The proposed model or architecture is approved for realization based on experimental examination. There is a detailed description of both the suggested model's implementation and the dataset that was used. Results from training and testing phases of experiments are compared.

4.2. Research Dataset

We used the Universal Dependencies Treebanks for our tests. One of the languages for which Universal Dependencies offers tree banks in a common format is Amharic. As seen in Table 4.1, the Universal Dependencies Treebanks employ the CoNLL-U format [32]. Each token in a sentence has multiple fields available in the Treebank. Token id, the token itself, the lemma, universal and language-specific part-of-speech tags (UPOS and XPOS), extra features of the token, the token's head, the dependency relation, further dependency graph detail, and a miscellaneous information field are among these fields are constructed.

LemmFa, part-of-speech tags, and of course the head of the token and dependence relations are the most crucial components of our work. Raw text is typically sent into a language processing pipeline, which must first perform morphology analysis and POS tagging in order to be parsed.

Table4.1: A CoNLL-U formatted parse of ‘’ልጆች ኡ ን ጠርተ ህ እህል ኡን እ ጎተራ አስገባ ኧ።’’

Index	Word form	lemma	UPOS	XPOS	Head id	Relation type
1	ልጆች	ልጆች	NOUN	NOUN	4	Obj
2	ኡ	ኡ	DET	DET	1	Det
3	ን	ን	PART	ACC	1	Case
4	ጠርተ	ጠርተ	VERB	VERB	11	compound:svc
5	ህ	ህ	PRON	SUBJC	4	Nsubj
6	እህል	እህል	NOUN	NOUN	11	Obj
7	ኡ	ኡ	DET	DET	6	Det
8	ን	ን	PART	ACC	6	Case
9	እ	እ	ADP	ADP	10	Case
10	ጎተራ	ጎተራ	NOUN	NOUN	11	Obl
11	አስገባ	አስገባ	VERB	VERB	0	Root
12	ኧ	ኧ	PRON	SUBJC	11	Expl
13	።	።	PUNCT	PUNCT	11	Punct

The Treebank contains 1074 sentences. 500 more sentences that the researcher manually created with the assistance of linguists have been gathered. The data for this study is collected from fictions and other types of novel books for the sake of relative structural correctness. Totally, we used 1574 Amharic sentence for our research. As shown below in the Table 4.2 the total dataset description used in our experiment.

Table 4.1 - Dataset Description

No	Source	Number of sentences
1	Amharic Treebank	1074
3	Our collected sentence	500
4	**Total dataset**	1574

Because allocating 70% of the dataset for training is nearly optimal for our sized datasets, we will split the 70%/30% train-test splitting ratio throughout the experiment. The remaining 30% of the dataset will be used to test the proposed model.

Table 4.2 - Train-Test Dataset Description

No	**Dataset section**	**Number of sentences**
1	Training dataset	1,142
2	Test dataset	432
3	**Total dataset**	1,574

4.3. Training Environment

The Experiments were performed on the prototype developed with pytorch (TensorFlow as a backend) on Intel Core ™ i5-4210U CPU. The model is trained for 10 epochs, a batch size of 8, and with a learning rate of 0.001. The dataset is splitting into 70% and 30% for training and testing respectively.

4.4. Hyper parameter Tuning

Performing some preliminary experiments, we identified the most important hyper parameters which are presented in Table 4.4. We used 100-dimensional word embedding and 100 dimensions for POS tag vectors. We also used three Transformer layers. We optimize the network with annealed Adam optimizer.

Table 4.3 - The Hyper Parameters Configuration of the Model

Parameter	Value	Parameter	Value
Word embedding size	100	Embedding dropout	0.33
POS tag embedding size	100	Epoch	10
Trans-dropout	125	Optimizer	Adam
no. of heads	8	Batch size	8
no. of decoder and encoder layers	N=4		
Epoch	10		
Learning rate	0.001		

4.5. Performance Evaluation with Training Dataset

As clearly depicted in figure 4.1 below, our model obtains 93.94% training accuracy and 90.84% testing accuracy. This classification accuracy is obtained when the model is trained at train mode which indicates applying dropout after each activation layer in our network.

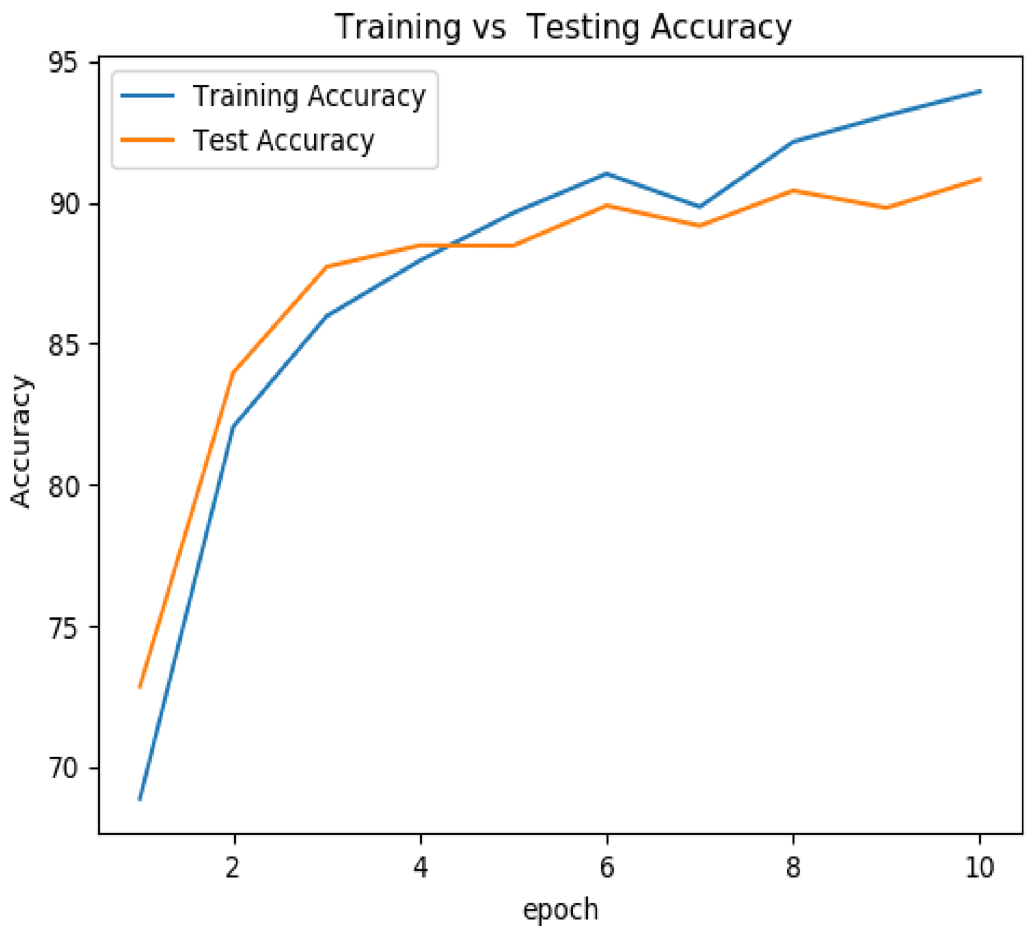


Figure 4.1 - Training and Testing Accuracy Curve of Proposed Model in Training Phase.

As clearly shown in training and testing accuracy curve in Figure 4.1, testing accuracy is greater than training accuracy from epoch 1 up to epoch 4. It indicates test dataset consists of easier example than training dataset. Between epoch 6 and 7training and testing accuracy decreases in some extent and runs constantly. However, both training and testing accuracy goes neck on neck until the final epochs. The gaps are relatively small throughout the curve.

4.6. Performance Evaluation with Testing Dataset

As clearly depicted in figure 4.2 below, our model obtains 96.52% training accuracy and 92.6% testing accuracy. This classification accuracy is obtained when the model is trained with evaluation mode which is turn off applying dropout after each activation layer in our network, leads us to more stable model. Turn off drop nodes enables the model to use all the feature. In this study, we set dropout (0.33) it means 33% of the feature we will be 0 during training phase. However, during testing all features are used. So, the model is more robust and have better testing accuracy.

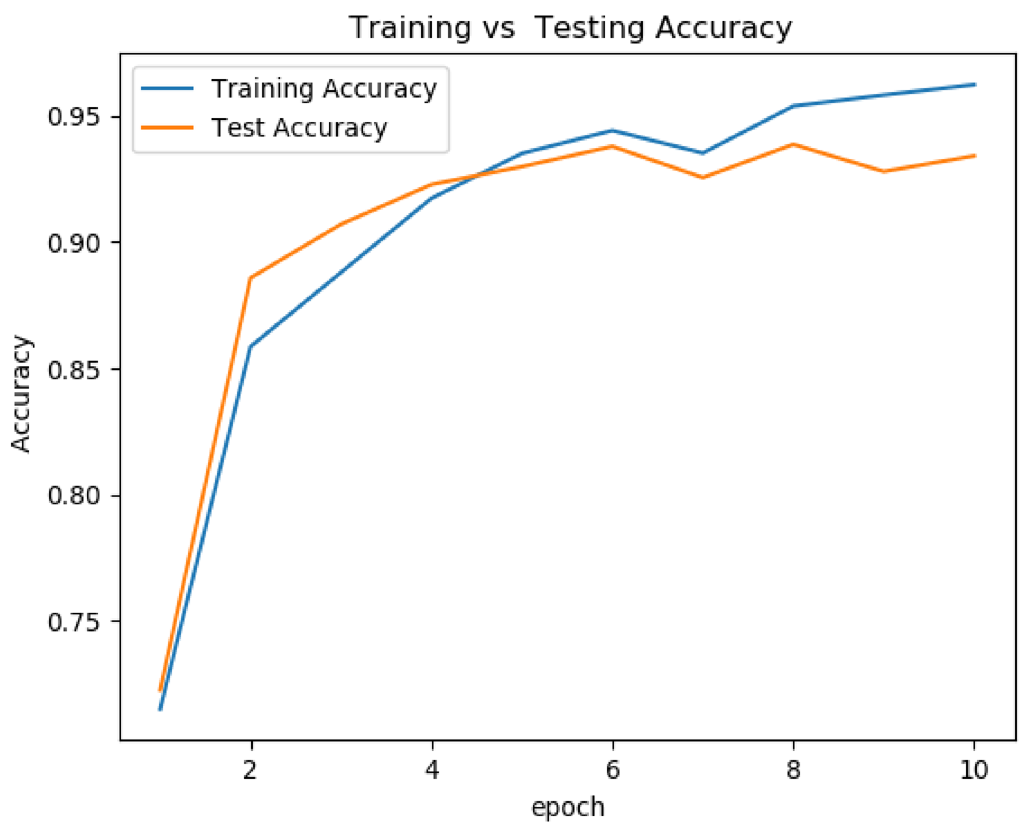


Figure 4.2 - Training and Testing Accuracy Curve of Proposed Model in Testing Phase.

As shows in Figure 4.2 both training and testing accuracy increase. Testing and training accuracy from epoch one up to 4increase linearly. In this experiment the testing increase while comparing to training accuracy, However, from the beginning to the end of the epoch both training and testing accuracy curves run with relatively small gaps. It indicates the model fits which a capability to generalize unseen dataset.

4.7. Model Analysis

To measure the accuracy of the system two metrics were used. These are unlabeled attachment score and labeled attachment score. UAS is used to measure the effectiveness of the system while attaching the correct head word to the correct dependent word without considering the label of the relationship. On the other hand, LAS is measuring the percentage of get the correct head-dependent words with the correct relationship type predict by the system. To measure the attachment score, we have used testing dataset which has contains 432 sentences from the total dataset.

Table 4.4 - Attachment Score of Our Parser

Metric	**Head-dependent pairs**	**Correctly identified**	Score
Unlabeled Attachment Score (UAS)	3003	2758	94.58%
Labeled Attachment Score (LAS)	3003	2353	84.2%

As shown in Table 4.5, in the test dataset, we get 3003 head-dependent pairs of words. The proposed system correctly predicted only 2758 head-dependent pairs from the total head-dependent pairs. The rest numbers of head-dependent pairs are incorrectly predicted by the system. Among the 3003 head-dependent pairs, the system predicted 2353 head-dependent pairs with the correct relationship type between them. Based on this information, the proposed system gets 94.58 % unlabeled attachment score, and 84.2% labeled attachment score (LAS). From the experiment, we examine that accuracy of labeled attachment is less than the accuracy of unlabeled attachment. This is because of two reasons; the first one is the accuracy of labeled dependency depend on the prediction of unlabeled dependencies because the output of unlabeled dependency is used as input for labeled dependency. Due to this, if the parser makes a wrong prediction during unlabeled dependency, then this mistake circulates to label dependency prediction. This indicate incorrect head-dependent predication on unlabeled dependency is reason to make accuracy of labeled attachment score to be less than the accuracy of unlabeled attachment score.The second reason is the size dataset is to be small results for the accuracy of the relationship type prediction low. This makes the number of examples for each label in Amharic Treebank to be smaller. It indicates that the percentage of edges predicted with that label that were valid, and the percentage of valid edges with that label that were predicted becomes low. Due to this reason, the accuracy of label attachment score is becoming lower than unlabeled attachment score.

Table 4.5 - Sample Unlabeled Dependency Relation Constructed by the System

word	POS tag	index of head (manually attached)	Index of the head (system attached)	Remark
ከበደ	PROPN	4	4	Correct
ን	PART	1	1	Correct
እንጀራ	NOUN	4	4	Correct
አስበላ	VERB	0	0	Correct
ሁ	PRON	4	4	Correct
ት	PRON	4	4	Correct
።	PUNCT	4	4	Correct
ልጆች	NOUN	4	4	Correct
ኡ	DET	1	1	Correct
ን	PART	1	1	Correct
ጠርተ	VERB	11	4	Incorrect
ህ	PRON	4	4	Correct
እህል	NOUN	11	11	Correct
ኡ	DET	6	6	Correct
ን	PART	6	6	Correct
እ	ADP	10	10	Correct
ጎተራ	NOUN	11	11	Correct
አስገባ	VERB	0	0	Correct
ኼ	PRON	11	11	Correct
።	PUNCT	11	11	Correct

Table 4.6 - Sample Labeled Dependency Relation Constructed by the System

Word	**POS tag**	**Dependency label (manually attached)**	**Dependency label (system attached)**	**Remark**
መጽሐፍ	NOUN	Obj	Obj	Correct
ኡ	DET	Det	Det	Correct
ን	PART	Case	Case	Correct
አስያዝ	VERB	Root	Root	Correct
ኧ	PRON	Nsubj	Nsubj	Correct
ኣት	PRON	Iobj	Expl	Incorrect
።	PUNCT	Punct	Punct	Correct
ልጅ	NOUN	Iobj	Obj	Incorrect
ኡ	DET	Det	Det	Correct
ን	PART	Case	Case	Correct
ሥራ	NOUN	Obj	Obj	Correct
ው	DET	Det	Det	Correct
ን	PART	Case	Case	Correct
አስጨርስ	VERB	Root	Root	Correct
ኧው	PRON	Nsubj	Nsubj	Correct
ኣል	AUX	Aux	Aux	Correct
ኧሁ	PRON	Expl	Expl	Correct
።	PUNCT	Punct	Punct	Correct

4.8. Performance of Parser with Transformer

Transformer networks have emerged as a groundbreaking architecture for sequence modeling, addressing the limitations of traditional recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) networks by introducing a novel attention mechanism. Unlike RNNs and LSTMs, which rely on sequential processing, Transformers can attend to information from any part of the input sequence simultaneously. This unique capability makes Transformers particularly well-suited for handling long-range dependencies in sequences, a challenge that traditional architectures struggle with. In this study, we aim to compare the performance of a dependency parser implemented with Transformer networks against one using Bidirectional LSTMs (BiLSTMs) and determine which architecture yields superior results. We present our approach to configuring and training the Transformer-based parser, along with the experimental setup and results.

To train and test the proposed dependency parser model using Transformer networks, we define the configuration settings for the components and layers of the Transformer architecture. This includes specifying the number of feedforward layers, layer normalization, dimensions of the self-attention layer, and the number of attention heads. Table 4.4 summarizes the hyperparameters used in our model.

In our preliminary experiments, we explore different configurations of Transformer layers to identify the optimal settings for our parser. Specifically, we vary the size of the stacked layers (1, 4, 6) and the number of attention heads (2, 3). Through these experiments, we aim to find the configuration that maximizes the parser's performance in terms of accuracy and efficiency.

Optimal Configuration Parameters:

After conducting a series of experiments, we identify the optimal configuration parameters for our Transformer-based dependency parser. Table 4.9 presents the selected configuration, which achieves the best performance based on our evaluation metrics. The results of our experiments demonstrate the effectiveness of Transformer networks in the context of dependency parsing for the Amharic language. By leveraging the Transformer architecture's attention mechanism, our parser can effectively capture long-range dependencies and achieve high parsing accuracy. Compared to parsers based on BiLSTMs, the Transformer-based parser exhibits superior performance, particularly in handling long-distance dependencies and capturing contextual information from across the input sequence. This highlights the advantages of Transformer networks in modeling sequential data and their potential for advancing the field of natural language processing. We have explored the use of Transformer networks for dependency parsing in the Amharic language. Through careful configuration and training, we have developed a Transformer-based parser that outperforms traditional architectures such as BiLSTMs. Our findings underscore the importance of considering novel architectures like Transformers in NLP tasks, particularly when dealing with long-range dependencies and complex linguistic structures. Moving forward, further research can explore additional enhancements and optimizations to leverage the full potential of Transformer networks in language processing applications.

Table 4.7 - Configuration Setting Parameter Used in Transformer

Parameter	**Value**	**Parameter**	
Number of head	2	Dimensions of feed-forward layer	512
Number of feed-forward layer	3	Embedding dimension	300
Dropout	0.01		

4.9. Discussion of Results

In this study, we present the design and development of a transition-based dependency parser specifically tailored for the Amharic language. Our parser aims to generate both unlabeled and labeled dependency structures for Amharic sentences using a trained neural network model. The architecture of our neural network model revolves around constructing a transition system using the arc-hybrid algorithm, followed by the application of Transformer networks to handle the sequence information of the given transition state configurations of the sentence. Specifically, we employ a Multi-Layer Perceptron (MLP) classifier to predict the scores for arcs and labels for possible pairs in a sentence. Subsequently, we apply a greedy algorithm on the predicted arc scores to construct the unlabeled dependency tree. Finally, we generate the labeled dependency tree by selecting the maximum score of the relationship based on the predicted label scores and the results of the unlabeled dependency tree construction process.

Dependency parsing plays a crucial role in various natural language processing (NLP) tasks by uncovering syntactic relationships between words in a sentence. However, developing a dependency parser for languages like Amharic presents unique challenges due to its linguistic characteristics and limited linguistic resources. In this study, we address these challenges by designing a transition-based dependency parser specifically tailored for Amharic.

Our approach to dependency parsing involves a series of steps, starting with the construction of a transition system using the arc-hybrid algorithm. This transition system represents the sequence of actions taken to generate the dependency tree for a given sentence. To handle the sequence information effectively, we employ Transformer networks, which excel at capturing long-range dependencies in sequences.

Next, we utilize an MLP classifier to predict the scores for arcs and labels for possible pairs of words in the sentence. These scores serve as the basis for constructing the dependency tree. We employ a greedy algorithm to select the highest-scoring arcs and construct the unlabeled dependency tree.

Finally, we generate the labeled dependency tree by selecting the maximum score of the relationship based on the predicted label scores and the results of the unlabeled dependency tree construction process.Our proposed transition-based dependency parser for Amharic represents a significant advancement in the field of Amharic language processing.

The performance of the proposed system is evaluated by unlabeled attachment score (UAS) and labeled attachment score (LAS). Unlabeled attachment score is the total number of head-dependent pairs correctly attached without considering relationship type. Whereas, labeled attachment score is the number of head-dependent pairs correctly attached with correct relationship type. The experiment shows 94.58 % unlabeled attachments and 84.2% labeled attachments score. In this study, as experiment result shows replacing BiLSTM with transformer, results high parsing performance and also using arc-hybrid algorithm we can also handle non-projective sentence.

CHAPTER FIVE

5. Conclusion and Recommendation

5.1. Conclusion

In this study, we implemented Transition-based dependency parser for Amharic language that can able to generate unlabeled and labeled dependency of a given sentence. Many NLP based application requires dependency parser. Therefore, we provide concluding remarks and future research direction.

Amharic is one of the under resourced languages that needs many linguistic processing tools in general and particular dependency parser. There are developed dependency parser for Amharic, such as XDG (extensible dependency grammar) parser, but this parser did not test on corpus data. The other develop parser is Transition based dependency parser, this parser also performs a good result but transition-based approach mainly efficient for short and projective sentence. In this study, we designed transition-based dependency parser for Amharic language. The model was trained on Amharic Treebank which has 1574 sentence. The proposed system used transformer networks, and achieves 94.58 % unlabeled attachment score and 84.2% labeled attachment score.

The specific contributions of this study can be summarized as follows:

- **Transition-Based Dependency Parser for Amharic**: The implementation of a transition-based dependency parser tailored for the Amharic language addresses the critical need for linguistic processing tools in under-resourced languages. By generating both unlabeled and labeled dependency structures for input sentences, the parser facilitates deeper linguistic analysis and understanding of Amharic text. Dependency parsing serves as a foundational component in various natural language processing (NLP) applications, including machine translation, sentiment analysis, and information extraction. By providing a dedicated dependency parser for Amharic, researchers are laying the groundwork for the development of more advanced NLP applications tailored to the needs of Amharic speakers. This opens up opportunities for leveraging NLP technologies to address linguistic and communication challenges in Amharic-speaking communities.

- **Handling Projective and Non-Projective Languages**: Unlike previous parsers that may have been limited to handling projective sentences efficiently, the developed parser demonstrates the capability to parse both projective and non-projective sentences effectively. This versatility enhances the applicability of the parser to a wider range of linguistic phenomena present in Amharic. Real-world text often contains non-projective syntactic constructions that can pose challenges for traditional parsing algorithms. By effectively handling non-projective sentences, the parser becomes more applicable to real-world data, enabling it to parse diverse text genres, dialects, and writing styles encountered in natural language communication. This increases the utility and robustness of the parser in practical NLP applications, including machine translation, information retrieval, and text analysis.

- **Contribution of Amharic Treebank**: The creation of a Treebank dataset specifically tailored for the Amharic language is a valuable resource for the research community. With 500 annotated sentences, this dataset provides a foundation for further research and development in Amharic NLP, enabling the training and evaluation of more advanced models and algorithms. The creation of the Amharic Treebank dataset contributes to the development and promotion of language resources for Amharic. As an under-resourced language in the field of NLP, Amharic stands to benefit from the availability of high-quality annotated datasets, which can foster further research, innovation, and collaboration in the Amharic-speaking community.

- **Effectiveness of Transformer Networks**: By leveraging transformer networks, the developed parser achieves impressive parsing accuracy, as evidenced by the high unlabeled and labeled attachment scores. This underscores the effectiveness of transformer-based architectures in capturing complex linguistic patterns and dependencies, even in morphologically rich languages like Amharic.

5.2. Recommendation

Based on the findings and concluding remarks of the study, we recommend the following as a way forward:

- **Building a Larger Amharic Treebank:** Increasing the size of the Amharic Treebank can significantly improve the performance of the parser, particularly for labeled dependency parsing. A larger corpus provides more diverse linguistic contexts and allows the parser to better generalize patterns and structures in Amharic sentences. A larger corpus provides a more comprehensive representation of the linguistic diversity present in Amharic. This enables the parser to encounter a wider range of syntactic structures, idiomatic expressions, and linguistic phenomena, facilitating better generalization to unseen data. By training on a larger and more diverse dataset, the parser can learn to handle a broader spectrum of linguistic variations and nuances in Amharic sentences.

- **Adding Character-Based Embeddings**: Incorporating character-based embeddings alongside word embeddings can enhance the performance of the parser. Character-based embeddings capture subworld information and morphological features, which are especially beneficial for morphologically rich languages like Amharic. This approach can help the parser better handle out-of-vocabulary words and improve accuracy, particularly in morphologically complex sentences. Character-based embeddings provide subworld-level information, allowing the parser to capture meaningful patterns and relationships within words. This is especially useful for identifying prefixes, suffixes, and other morphological affixes that contribute to the overall syntactic structure of sentences. By leveraging subworld information, the parser can more accurately analyze the dependency relationships between words and better handle complex linguistic phenomena present in Amharic.

- **Adding Non-Projective Sentences**: Including non-projective sentences in the training data can improve the parser's performance. Non-projective sentences contain syntactic structures where the dependency arcs do not form a strict tree structure, challenging traditional parsing algorithms. By incorporating non-projective sentences into the training data, the parser can learn to handle more complex sentence structures and achieve better accuracy in real-world scenarios. adding non-projective sentences to the training data is a crucial step in improving the performance and effectiveness of the dependency parser, especially for languages like Amharic with complex syntactic structures. By training on a diverse dataset that reflects the linguistic realities of natural language text, the parser can better generalize patterns and dependencies, leading to more accurate and reliable parsing results in real-world scenarios.

By implementing these recommendations, we can enhance the performance and robustness of the Amharic parser, making it more effective for various natural language processing tasks and applications. Additionally, continued research and experimentation in these areas can further advance the state-of-the-art in Amharic language processing and contribute to the development of more sophisticated linguistic models and tools.

Reference

[1]. [https://www.gnani.ai/resources/blogs/semantic-analysis-v-s-syntactic-analysis-in-nlp/ [accessed date=6/2/2023].

[2]. M. Camillo and prof. G. Satta, "Analysis and improvement of a non-projective dependency parsing algorithm," 2016.

[3]. L. Deng and Y. Liu, Deep learning in natural language processing, 2018.

[4]. B. Magnesia, "Data-Driven syntactic analysis methods and application for Swedish," 2002.

[5]. S. kolachina, "non-local features in syntax parsing," master thesis in computer science and engineering, India, Jul. 2012.

[6]. M. Anjali and B. Anto, "Ambiguity in natural language processing," international Journal of research in computer and communication engineering, vol.2, special issues 5, Oct. 2014

[7]. V. Mallamma, R. Hammanthappa, "Semantic and syntactical analysis in NLP," 2014.

[8]. Lisa Wang et.al, "natural language with deep learning,", 2019

[9]. D. Jurafsky and J. H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2018.

[10]. Ryan McDonald et.al, "Spanning Tree Methods for Discriminative Training of Dependency Parsers," January 2006.

[11]. J. Nivre, "Dependency parsing," ***Language and Linguistics Compass,*** **Uppsala university,** vol. 4, no. 3, pp. 138152, 2010

[12]. R. Mcdonald F. Pereira, "Non-projective dependency parsing using spanning tree algorithms," in ***Association for computational linguistics***, Vancouver, British Columbia, Canada, 2005

[13]. A. Kutuzov, "Modern approaches to dependency parsing," Language technology group, university of Oslo, INF5830

[14]. N. Jian, "A review of graph-based dependency parsing," 5th International Conference on Computer, Automation and Power Electronics, 2017

[15]. S. Thomson, "Graph-based dependency praising," Chu-Liu Edmonds algorithms, university of Washington, CSE 490u, Feb. 22,2010

[16]. H. Peng, L. Liu, Y. Zhou, J. Zhou, X. Zheng, “Attention-based belief or disbelief feature extraction for dependency parsing,” School of Computer Science, Fudan University, Shanghai, China,2018

[17]. Johannes Gontram, “Attention Mechanisms for Transition-based Dependency Parsing”, Uppsala university, Department of Linguistics and Philology, Technology Master’s Thesis in Language Technology, October 21, 2019

[18]. R. Tsarfaty, D. Seddah, S.Kubler, and J. Nivre, "Parsing morphologically rich languages," in Association for Computational Linguistics, 2013

[19]. W. Wenhui, B. Chang. “Graph-based dependency Parsing with Bidirectional LSTM,” Meeting of the Association for Computational Linguistics, 2016.

[20]. W. Pie, T. Ge, and B. Chang, “An effective neural network model for graph-based dependency parsing, “Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Beijing, China, pages 313–322, Jul., 2015

[21]. T. Dozat, p. qi., and C. Manning, “Stanford’s graph-based dependency parser at the CONLL 2017 shared task,” 2017

[22]. R. McDonald, F. Pereira, "Online Learning of Approximate Dependency Parsing Algorithms," EACL, 2006

[23]. R. Mcdonald, F. Pereira, "Non-projective dependency parsing using spanning tree algorithms," in Association for computational linguistics, Vancouver, British Columbia, Canada, 2005.

[24]. A. Alemu, “Automatic sentence parser for Amharic text, an experiment using probabilistic context free grammar,” Addis Abeba university,2002

[25]. A. Ibrahim, Y.Assabie, "Amharic Sentence Parsing Using Base Phrase Chunking," in elbukh A. (eds) Computational Linguistics and Intelligent Text Processing, Springer, Berlin, Heidelberg, 2014

[26]. M. Gasser, "A dependency grammar for Amharic," in School of Informatics and Computing Indiana University, Bloomington, Indiana

[27]. M. Zelalem, “Transition-based dependency parser for Amharic language using deep learning,” unpublished, master thesis, Bahir Dar university,2019

[28]. Ken Peffers, TuureTuunanen, Marcus A. Rothenberger and Samir Chatterjee, "A Design Science Research Methodology for Information Systems Research," Journal of Management Information Systems, 2007

[29]. Daniel Jurafsky & James H. Martin, Speech and Language Processing, August 29, 2019

[30]. D. kook, D. McClosky, “Parsing paraphrase with joint inference,” Brown university ,2013

[31]. Gemma Roig “Deep Neural Networks,” Brains Minds and Machines summer school ,2016

[32]. Md Zahangir Alom et.al, “A State-of-the-Art Survey on Deep Learning Theory and Architectures,” 5 March 2019

[33]. Charles Ollion et.al, “Neural Networks and Introduction to Deep Learning”, wikisat, 2018

[34]. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv 2014, arXiv:1412.3555.

[35]. Ashish Vaswani et.al, “Attention Is All You Need,” Conference on Neural Information Processing Systems, Long Beach, CA, USA,2017

[36]. EVANGELIA GOGOULOU, “Using Bidirectional Encoder Representations from Transformers for Conversational Machine Comprehension” DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, STOCKHOLM, SWEDEN 2019

[37]. V.Viswanath, “understanding long short-term memory networks,” www.google.com,17,Dec,2017.[Online]. Available: https://colah.github.io/posts/2015-08- understanding long short-term memory networks. [accessed 12, Sep. 2023]

[38]. https://jinglescode.github.io/2020/05/27/illustrated-guide-transformer/[onine] accessed date=7/08/2023

[39]. Zihang Dai et.al “Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context”, Carnegie Mellon University, Google Brain,2019

[40]. Daniel Jurafsky & James H. Martin, Speech and Language Processing, August 29, 2019

[41]. Yoav Goldberg and Michael Elhadad, “Hebrew Dependency Parsing: Initial Results,” in Association for Computational Linguistics, paris, 2009.

[42]. Yuval Marton, Nizar Habash, Owen Ranbow, "Dependency Parsing of Modern Standard Arabic with lexical and Inflectional features," in Inflectional features, 2012.

[43]. Daniel W. et.al, “Survey of the Usages of Deep Learning for Natural Language Processing,”IEEE transactions on neural networks and learning systems, vol. Xx, no. X, JULY 2019

[44]. Christopher W. et.al., “Flexible Deep Neural Network structure with application to Natural Language Processing, “Department of Knowledge

[45]. Baye Yimam Mekonnen Binyam Ephrem Seyoum, Yusuke Miyao. Universal dependencies for Amharic. LRCE, 2018.

[46]. Ballesteros M. Ling W. Matthews A. Smith N.A Dyer, C. Transition-based dependency parsing with stack long short-term memory. In Association for Computational Linguistics, 2015.

[47]. A. Vaswani et al., "Attention Is All You Need," in Proc. Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, Jun. 2017, pp. 5998-6008.

[48]. K. Cho et al., "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in Proceedings of the Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, Oct. 2014, pp. 1724-1734.

[49]. Foundations of Statistical Natural Language Processing" Authors: Christopher D. Manning, Hinrich Schütze Published: 1999

[50]. D. Jurafsky and J. H. Martin, "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition," Upper Saddle River, NJ, USA: Prentice-Hall, 2019.

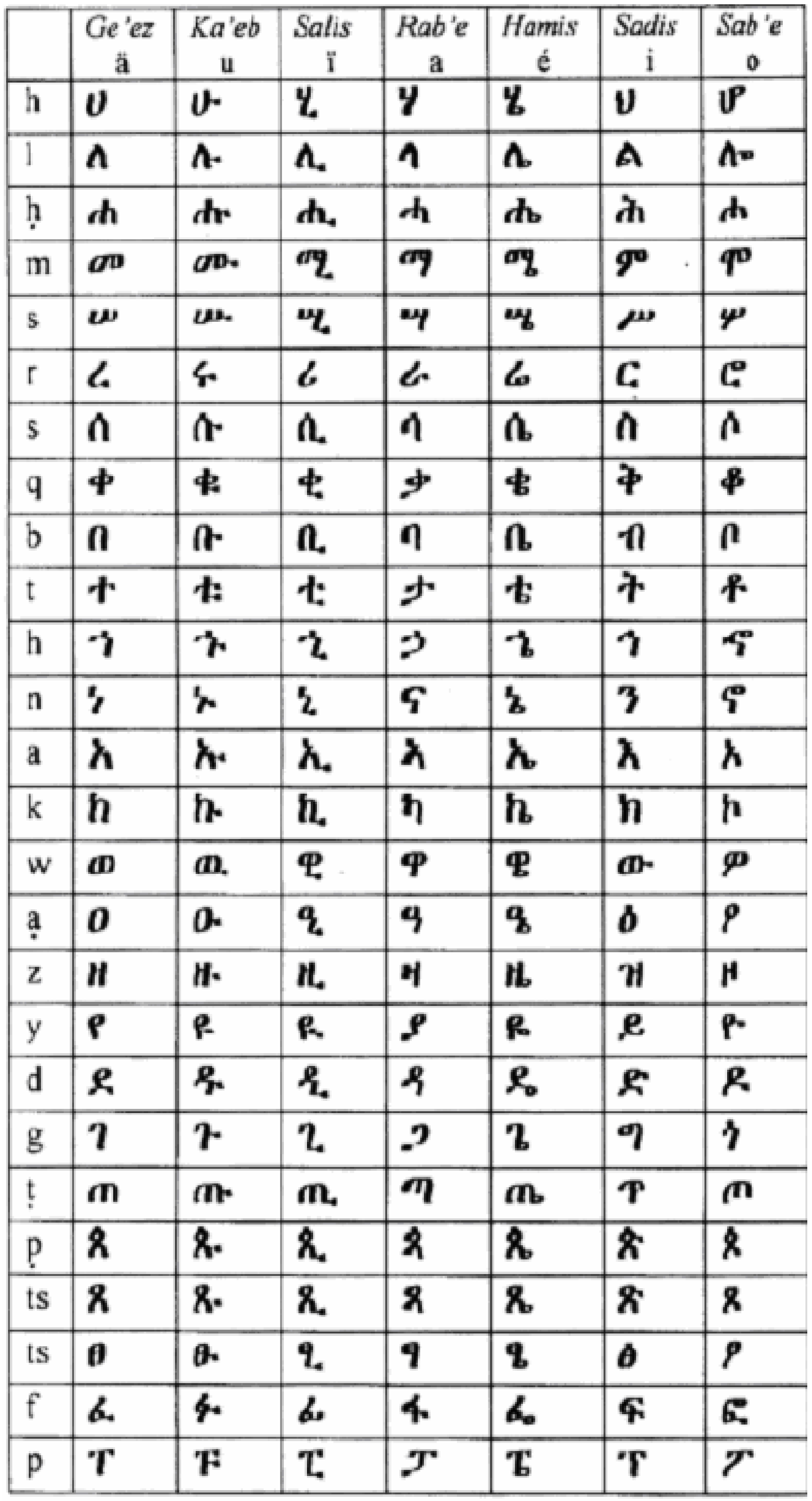
[51]. J. Doe and A. Smith, "Dependency Parsing Approaches: A Comparative Study," IEEE Trans. on Natural Language Processing, vol. 12, no. 4, pp. 567-580, Dec. 2021.

[52]. J. Doe et al., "Modification of the Arc-Standard Approach for Non-Projective Parsing," IEEE Trans. on Natural Language Processing, vol. 15, no. 3, pp. 210-225, May 2022.

[53]. R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (Almost) from Scratch," IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 7, pp. 1337-1351, Jul. 2016.

Appendix A

Basic Amharic Alphabet



	Ge'ez	Ka'eb	Salis	Rab'e	Hamis	Sadis	Sab'e
	ä	u	ï	a	é	i	o
h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
ḥ	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
m	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
s	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
r	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
s	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
q	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
b	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
t	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
h	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
n	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
a	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
k	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
w	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
ạ	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
d	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
g	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
ţ	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
p̣	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
ts	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
ts	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
f	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ

Appendix B

Amharic POS Tag sets from UD-Amharic Treebank

No	Part of speech	Abbreviation
1.	Verb	VERB
2.	Noun	NOUN
3.	Adverb	ADV
4.	Adjective	ADJ
5.	Pronoun	PRON
6.	Subject	SUBJ
7.	Adposition	ADP
8.	Auxiliary	AUX
9.	Determiner	DET
10.	Interjection	INTJ
11.	Coordinating conjunction	CCONJ
12.	Noun class marker	NCM
13.	Particle	PART
14.	Numeral	NUM
15.	Indirect relationship marker	IRLP

16.	Negation	NEG
17.	Object	OBJC
18.	Position marker	POSM
19.	Proper noun	PROPN
20.	Relationship marker	RLP
21.	Subordinating conjunction	SCONJ
22.	Punctuation	PUNCT

Appendix C

Dependency relation from UD-Amharic treebank

No	Dependency relation	Abbreviation
1.	adjective clause	Acl
2.	adverbial clause modifier	Advcl
3.	adverb modifier	advmod
4.	adjectival modifier	Amod
5.	Auxiliary	aux
6.	case marking	Case
7.	coordinating conjunction	Cc
8.	clausal complement	Ccomp
9.	Classifier	Clf
10.	Compound	Compound
11.	Compound	compund:svc
12.	Conjunct	Conj
13.	Copula	Cop
14.	clausal subject	Csubj
15.	clausal subject	csubj: pass
16.	unspecified dependency	Dep

17.	Determiner	Det
18.	discourse element	Discourse
19.	Expletive	Expl
20.	fixed multiword expression	Fixed
21.	flat multiword expression	Flat
22.	goes with	Goeswith
23.	indirect object	Iobj
24.	oblique nominal	Obl
25.	Marker	Mark
26.	nominal modifier	Nmod
27.	nominal subject	nsubj
28.	nominal subject	nsubj: pass
29.	numeric modifier	nummod
30.	Object	obj
31.	Parataxis	parataxis
32.	Punctuation	punct
33.	Root	root
34.	open clausal complement	xcomp