SCHOOL OF GRADUATE STUDIES

DEVELOPING AUTOMATIC CONSTITUENCY PARSER FOR SILTIGNA LANGUAGE USING DEEP LEARNING APPROACH

MSc. THESIS

TEKA MOHAMMED

April 21, 2024

WOLKITE, ETHIOPIA

Wolkite University

School of Graduate studies

Developing automatic Constituency Parser for Siltigna Language using Deep learning Approach

A Thesis Submitted to School of Graduate Studies, in Partial Fulfillment of the Requirement for the Degree of Master of Computer Science in Computer Science and Engineering.

Teka Mohammed

Major Advisor:- Sintayehu Hirpassa (Ph.D.)

Co_Advisor:- Abdo Ababor (M. Sc.)

April 21, 2024

Wolkite, Ethiopia

APPROVAL SHEET

WOLKITE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

We hereby certify that we have read and evaluated this Thesis titled "**Developing automatic Constituency Parser for Siltigna Language using Deep learning Approach**" prepared under our guidance by Teka Mohammed Ale. We recommend that the Thesis shall be submitted as fulfilling the requirements for the award of MSC. Degree in Computer Science and Engineering.

Sintayehu Hirpassa(Ph.D.)	Sintayehu H.	25/5/2024
Major advisor	Signature	Date
Abdo Ababor(M.Sc.)	_____	_____
Co-advisor	Signature	Date

As members of the Board of Examiners of the Master of Science Thesis open defense examination, we have read and evaluated this Thesis prepared by Teka Mohammed Ale and examined the candidate. We hereby certify that the thesis is accepted for fulfilling the requirements for the award of the degree of Master of Science (M.Sc.) in Computer Science and Engineering.

1. Kindie Biredagn (Ph.D.)		May-25-2024
Name of External Examiner	Signature	Date
2. Worku Muluye (Ass.Prof)	_____	_____
Name of Internal Examiner	Signature	Date
3. _____	_____	_____
Name of Chair Person	Signature	Date

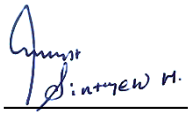
Final approval and acceptance of the Thesis is contingent upon the submission of its final copy to the Council of Postgraduate Program (CPGS) through the candidate's department or school graduate committee (DGC or SGC).

APPROVAL SHEET

WOLKITE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

We hereby certify that we have read and evaluated this Thesis titled **"Developing automatic Constituency Parser for Siltigna Language using Deep learning Approach"** prepared under our guidance by **Teka Mohammed Ale**. We recommend that the Thesis shall be submitted as fulfilling the requirements for the award of a MSc. degree in computer Science and Engineering.

Sintayehu Hirpassa (Ph.D.)		25/05/2024
Major Advisor	Signature	Date
Abdo Ababor (Msc)	_____	_____
Co-Advisor	Signature	Date

As members of the Board of Examiners of the Master of Science Thesis open defense examination, we have read and evaluated this Thesis prepared by **Teka Mohammed Ale** and examined the candidate. We hereby certify that, the thesis is accepted for fulfilling the requirements for the award of the degree of Master of Science (M.Sc.) in computer Science and Engineering.

1. Kindie Biredagn (Ph.D.)		May-25-2024
Name of extremal examiner	Signature	Date
2. Worku Muluye (Ass.Prof)	_____	_____
Name of Internal examiner	Signature	Date
3. _____	_____	_____
Name of chainman	Signature	Date

Final approval and acceptance of the Thesis is contingent upon the submission of its final copy to the Council of Postgraduate Program (CPCS), through the candidate's department or school graduate committee (DGC or SGC).

DECLARATION

In my signature below, I certify that this thesis is entirely original to me and has not previously been submitted for degree at any other university. I completed this thesis by adhering to all scholarly ethical criteria. The scholarly content of the thesis has been acknowledged throughout with citations. I certify that every source I utilized for this paper was correctly mentioned and referenced.

Name Teka Mohammed Date 16/9/2016 E.c. Signature _____

School/Department: - Software Engineering

With my approval as the university advisor, this thesis has been submitted for examination.

Name Teka Mohammed

Date 16/9/2016 E.c.

Signature _____

ACKNOWLEDGMENT

First and foremost, I would like to express my heartfelt gratitude to the Almighty God, Allah, for allowing me to fulfill my dream and providing me with the strength, determination, endurance, and wisdom to complete this study. I am grateful for His presence and guidance throughout my journey. I would like to extend my deepest appreciation to my advisor, Dr. Sintayehu Hirpasaa, for his exceptional and unwavering support. His guidance has been invaluable, and he has inspired me for my future academic endeavors. I have learned so much from him about building my academic brand, prioritizing my focus, and giving back to others. I am truly grateful for his mentorship.

I would also like to thank my co-advisor, Mr. Abdo Ababor, for his insightful guidance and constructive feedback in all aspects of my work. His contributions have been instrumental in shaping the outcome of my study.I am indebted to all those who have provided their support, contributing to the successful completion of this work. I am grateful for their assistance and encouragement. Special thanks go to Siliti'e zone education office, WRU siltigna department head ,siliti'e FM partners for servicing me during data collection time and my family and friends for their unwavering and limitless support throughout the process of writing my thesis. Their moral, material, and financial support have been indispensable, and I could not have accomplished this without them.

Lastly, I would like to express my sincere appreciation to my colleague, Mr. Getenet Degemu, for his exceptional support throughout the implementation of my work. His assistance, affection, and continued encouragement have been invaluable. Thank you all for believing in me and for being a part of my journey. I am truly grateful for your contributions.

ABBREVIATIONS AND ACRONYMS

ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
ATB	Arabic Tree Bank
AUC	Area Under The Curve
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
BILSTM	Bidirectional Long Short Term Memory
CB	Crossing Brackets
CFG	Context Free Grammar
CKY	Cocke-Kasami-Younger
CNNs	Convolutional Neural Networks
CNTK	Microsoft Cognitive Toolkit
CRF	Conditional Random Field
CPUs	Central Processing Units
DL	Deep Learning
EL	Embedding Layer
ELMo	Embedding from Language Models
GANs	Gated Adversarial Networks
GPT	Generative Pre-trained Transformer
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
GloVes	Global Vectors for Word Representation.

IDE	Integrated Development Environment
LAS	Labeled Attachment Score
LibSVM	Library Support Vector Machine
LSTM	Long Short-Term Memory
MSA	Modern Standard Arabic
NLTK	Natural Language Tool Kit
NUM_ HEAD	Number of Head
POS	Part-of- Speech
PT-XL	Pre-trained Transformer-XL
ReLU	Rectified Linear Unit
RNNs	Recurrent Neural Networks
SOV	Subject-Object-Verb
Seq2Seq	Sequence-To-Sequence
SVM	Support Vector Machine
TPU	Tensor Processing Unit
UAS	Unlabeled Attachment Score
WIC	Walta Information Center
WRU	Worabe University
Word2Vec	Word to Vector

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF APPENDICES

ABSTRACT

In our study, we focused on developing automatic constituency parsing for the Siltigna language using deep learning approaches. The Siltigna language is experiencing increased speaker numbers, and our goal was to address the language's issues and enhance its content globally. To achieve this, we employed a deep learning technique known as the transition method and the main architecture we used was a seq-to-seq auto-encoder-decoder model. This model has been widely used in natural language processing tasks. We conducted experiments using various deep learning models, including LSTM, Bi-LSTM, LSTM with attention, GRU, and transformer models. To train and evaluate these models, collected a dataset of approximately 2000 sentences and labeled them with corresponding parse trees. Before parsing the sentences into sequences, we applied preprocessing techniques such as data cleaning and tokenization and split the dataset into training and testing sets using an 80-20 split. Subsequently, we trained and tested the LSTM, Bi-LSTM, LSTM with attention, GRU, and Transformer models on the labeled parse tree data. Among these models, the transformer model achieved the best performance with 84.38% accuracy, 0.137 losses, and LAS of 0.83. This indicates that the transformer model was most effective in accurately parsing the Siltigna language. Our study highlights the importance of natural language processing with interconnected global community. By developing automatic constituency parsing for the Siltigna language, we aimed to bridge language barriers and enable effective communication across borders.

Keywords: - Automatic constituency parser, Bi-LSTM, deep learning models, GRU, LSTM, Siltigna language, Transformer.

CHAPTER ONE

1. INTRODUCTION

1.1. Background of the study

Natural language processing (NLP) plays a crucial role in our everyday lives by enabling computers to understand human languages. Constituency parsing is a fundamental task in NLP, as it involves identifying the syntactic structure of sentences. This process typically includes segmenting a sentence into words, phrases and grouping them into syntactic units. It serves as an intermediate component for various higher-level NLP applications. For example, in grammar checking, a constituency parser helps to identify and correct grammatical errors by analyzing the syntactic structure of a sentence. In question answering systems, parsing aids in understanding the structure of the question and extracting relevant information. Additionally, sentence parsing is used in word sense disambiguation, which involves determining the correct meaning of ambiguous words based on their syntactic context[1]. It involves analyzing the grammatical structure of a sentence by constructing a parse tree, which represents the syntactic relationships between words or tokens. There are two main types of parsing dependency parsing and constituency parsing. Dependency parsing focuses on establishing direct relationships between words or tokens in a sentence and it constructs a dependency parse tree that represents these dependencies, where each word or token is a node, and the relationships are represented by labeled arcs or edges between the nodes. The arcs typically indicate the grammatical relationships, such as subject, object, or modifier relationships. Whereas, Constituency parsing, on the other hand is based on the parse trees of formal grammars and it aims to identify the constituents or phrases within a sentence and their hierarchical structure. A constituency parse tree represents the sentence as a hierarchy of constituents, where each constituent is a sub tree of the parse tree[2]. Constituency parsing is a core task in natural language processing (NLP) that involves generating a structured syntactic parse tree for a given sentence. The parse tree represents the hierarchical structure of the sentence and captures the relationships between words and phrases. The goal of constituency parsing is to identify the syntactic constituents in a sentence, such as noun phrases, verb phrases, and clauses, and determine how they are hierarchically organized. This is typically done by assigning a label to each constituent and specifying the parent-child relationships between them Constituency parsing is important in NLP because it provides a means to analyze the syntax of a sentence and extract useful information

about its structure. This information can be utilized in various downstream tasks, such as relation extraction, natural language inference, and machine translation. By generating a parse tree, constituency parsing enables a more detailed understanding of the sentence's grammatical structure, which can aid in semantic analysis, information extraction, and language generation. It also serves as a useful intermediate representation for further linguistic analysis and processing. Over the years, various approaches have been developed for constituency parsing, including rule-based methods, probabilistic models, and neural network-based models. These approaches often rely on large annotated corpora to learn patterns and dependencies in the sentence structure.Constituency parsing remains an active area of research in NLP, with ongoing efforts to improve parsing accuracy, handle complex sentence structures, and develop parsing models that can generalize well across different languages and domains[3]. In the field of constituency parsing, there are two main paradigms: - transition-based methods and chart- based methods. Transition-based methods are parsing algorithms that operate by performing a sequence of transitions to build a parse tree incrementally. These methods start with an initial configuration, typically representing an empty parse tree, and apply a series of transitions to gradually construct the parse tree. Each transition represents a specific action, such as shifting a word onto the stack or combining constituents. Transition-based methods often use machine learning techniques to learn the appropriate transitions based on training data[4].

Deep learning has been successfully applied to constituency parsing, a fundamental task in natural language processing (NLP). Deep learning models for constituency parsing leverage neural networks to automatically learn hierarchical representations of sentences and capture syntactic structures[5]. These models often employ recurrent neural networks (RNNs), such as long short-term memory (LSTM) or gated recurrent unit (GRU) networks, to process sentences and encode contextual information. Additionally, advanced techniques like attention mechanisms and self-attention have been used to enhance the models' ability to capture long-range dependencies and improve parsing accuracy. Deep learning-based approaches have achieved state-of-the-art performance in constituency parsing, demonstrating the effectiveness of neural networks in capturing complex syntactic structures and improving parsing accuracy on various languages and domains. Chart-based parsers represent spans in a sentence by subtracting hidden states at the boundaries of the spans, which can result in the loss of context information within the span. This can negatively impact parsing performance, particularly for long sentences. To address this issue, n-grams can be used as a simple yet valuable source of information to fill in the missing context. By incorporating n-gram information, chart-based parsers can

improve their ability to capture the dependencies and relationships between words within spans, leading to better parsing accuracy[3]. From the above detail descriptions we perform our research on transition-based methods for constituency parsing involve applying a sequence of transitions to incrementally construct a parse tree, while chart-based methods use chart data structures and dynamic programming techniques to divide the parsing task into smaller sub problems and combine partial parse trees.

1.2. Motivation of the study

Most research on constituency parsing has primarily focused on resource-rich languages such as English, German, and Chinese. However, Siltigna language is considered an under-resourced language in terms of electronic resources and processing tools. To develop constituency parser for Siltigna language, certain steps need to be taken. Firstly, it is crucial to prepare an annotated corpus specifically for Siltigna language. An annotated corpus consists of sentences from the target language that have been manually labeled with their corresponding parse trees. This annotated data serves as the training material for the deep learning models, allowing it to learn the patterns and rules of Siltigna syntax. Additionally, it is necessary to identify and define a suitable tag set for Siltigna language. A tag set consists of a predefined set of tags that represent different grammatical categories, such as nouns, verbs, adjectives, and so on. Defining an appropriate tag set for Siltigna enables the models to assign the correct tags to words in the input sentences, facilitating the parsing process. The motivation behind researching and developing a constituency parser for Siltigna language stems from its under-resourced nature compared to other languages. By addressing the lack of electronic resources and processing tools for Siltigna language, we aim to bridge the gap and contribute to the field of natural language processing for siltigna language. This research work has the potential to improve language understanding, enable grammar checking, question answering, and other NLP applications for the language.

1.3. Statement of the problem

Siltigna language is a zonal language spoken in the central Ethiopia regional sate of Ethiopia. According to the 2007 census conducted by the Central Statistics Authority, there are approximately 750,398, of whom 364,108 are men and 386,290 women; 78,525 or 6.28% are urban inhabitants. The largest ethnic group reported in Silt'e was the Silt'e people (97.35%); all other ethnic groups made up 2.65% of the population. Silt'e is spoken as a first language by 96.95% of the population and 1.54% speak Amharic; the remaining 1.51% speak all other primary languages reported[6]. It belongs to the South-Ethio-Semitic language family and utilizes the Saba Ge'ez script (Ethiopic script). Siltigna is spoken by a significant number of people across various parts of Ethiopia. In terms of education, elementary schools in the Siltie zone teach all subjects in Siltigna from grade one to grade six. After grade six, Siltigna is taught as a single subject up to the preparatory level, and it is also included in the grade twelve national exam[7]. Despite the growing availability of digital text information, the lack of automatic syntactic parsing models for Siltigna poses challenges in accessing and utilizing the correct information. This absence of parsing models also hinders the development of robust systems for higher-level components of natural language processing (NLP), such as machine translation, grammar checker, and speech recoginition. Siltigna language falls under the category of under-resourced languages, as it lacks annotated parsed corpora (treebanks) that could be used directly as input for parsing models. Furthermore, no research has been conducted on parsing frameworks specifically for Siltigna language. While some researcher has been carried out on Siltigna language, there has been no previous research specifically focused on parsing[8].

Finally, Parsing varies across languages due to their unique grammar rules. In English, parsing involves identifying and labeling sentence constituents. German parsing requires considering case, gender, and number. Arabic parsing faces challenges with complex morphology and involves morphological and syntactic analysis. Parsing techniques can differ within languages and research communities. Each language presents its own parsing complexities and requires specific approaches for analysis. For researching automatic constituency parsing for the Siltigna language, research gaps include limited linguistic resources, handling ambiguity, complex sentence structures. Addressing these gaps the study could improve parsing accuracy and coverage for Siltigna.

1.4. Research questions

RQ1.Which vectorization technique was used for identifying siltigna language?

RQ2.Which deep learning algorithms are best for siltigna language parsing?

1.5. Objective of the Study

1.5.1. General objective

The objective of the study is to develop Automatic Constituency Parser for Siltigna Language in deep learning techniques.

1.5.2. Specific objectives

- To collect, prepare, and pre-process Siltigna language raw text data.
- To develop siltigna language constituency parser by using deep learning techniques.
- To select vectotrization techniques that helps constituency parsing models.
- To evaluate Siltigna language constituency parser performance.
- Finally, to report the finding of the study for the upcoming researchers.

1.6. Significance of the study

The study on parsing for the Siltigna language holds great significance as it provides language experts with a valuable tool. Constituency parsing, acting as a man-machine interface, allows for a better understanding of the grammatical structure of the language. This has important implications for Siltigna language learning at all levels of educational institutions, from primary and secondary schools to higher education. By utilizing a constituency parser, language experts can demonstrate how the language's modern grammatical usage operates. The parser can automatically reveal the deep syntactic structure of the language using branching tree structures, effectively organizing the syntax into its constituents. This provides a visual representation of the language's grammatical structure, aiding in comprehension and learning. Additionally, constituency parsers have specific applications that directly benefit from their use. For example, machine translation systems can leverage constituency parsing to improve the accuracy and quality of translations. Grammar checking tools can utilize the parser to identify and correct grammatical errors more effectively. Furthermore, named entity recognition, which involves identifying and classifying named entities in text, can benefit from the syntactic information provided by constituency parsing. Overall, our study on parsing for the Siltigna language offers language experts a valuable tool for understanding the language's grammatical structure, with significant implications for language learning and various applications in natural language processing.

1.7. Scope and limitation of the study

1.7.1. Scope of the study

In the scope of our study, our primary objective is to develop an automatic constituency parser specifically designed for the Siltigna language. To achieve this, we would employ deep learning techniques as we analyze a text corpus in Siltigna language. This corpus comprises a range of sentence types, encompassing both simple and complex sentences. By incorporating this variety, we enhanced the accuracy of our constituency parsing from this our focus lies in effectively understanding and parsing the constituents within Siltigna sentences, enabling us to accurately identify the syntactic structure and relationships between words. Through this study, we strive to contribute to the advancement of natural language processing by developing a robust and efficient automatic constituency parser for the Siltigna language, thereby aiding in various applications such as machine translation, information retrieval, and text summarization.

1.7.2. Limitation of the study

Our study has a few limitations that should be acknowledged. Firstly, we solely focus on constituency parsing and do not include dependency parsing or semantic parsing in our research. Our primary objective is to develop an automatic constituency parser specifically tailored for the Siltigna language. Secondly, due to limited resources within our domain, we work with sentence data instead of full documents. This decision is based on the scarcity of annotated data (Treebank) prepared by linguistic experts for parsing tasks in our study area. Despite these limitations, our commitment remains to make a contribution to the field of natural language processing. By leveraging deep learning techniques and analyzing the available text corpus, we would achieve accurate and reliable results in constituency parsing and provide valuable insights for the advancement of the field.

1.8. Thesis organization

The thesis is organized into six chapters. Chapter 1 provides an introduction to the study, including a discussion on the problem statement, the motivation behind the research, the objectives of the study, the significance of the research, and the scope and limitations of the study. In Chapter 2, a comprehensive literature review is presented, covering related works, approaches, and grammar formalisms in the field of parsing. This chapter also explores the writing structure of sentences and word classes specific to the Siltigna language. Moving on, Chapter 3 focuses on the methods employed, proposed approaches, and the corpus used for the Siltigna language. Chapter 4 delves into the architecture designs of the models and provides implementation details for the proposed Siltigna

constituency parsing. The experimental setup, along with the test results, is presented and discussed in Chapter 5. Finally, in Chapter 6, the research work concludes with a summary of findings, recommendations for future work, and contributions made by the study.

CHAPTER TWO

2. LITERATURE REVIEW

2.1. Overview

In this chapter, we provide a brief overview of parsing approaches, techniques, grammar formalisms and overview of Siltigna language, with a focus on the concept of deep learning. By utilizes neural networks to learn hierarchical representations of sentences and improve parsing accuracy. We highlighted the advantages of deep learning in capturing complex syntactic structures and achieving state-of-the-art performance.

2.2. Overview of parsing

Constituency parsing, also known as syntactic parsing, and is a fundamental process in natural language processing (NLP) that involves analyzing the words of a sentence to determine its underlying syntactic structure based on a given grammar[9]. It identifies and segments sentence constituents and establishes their hierarchical relationships. Constituency parsing is crucial in NLP as it mediates between linguistic expression and meaning, enabling deeper language understanding and supporting various downstream tasks. Researchers employ rule-based, statistical, and deep learning approaches to develop parsing algorithms and models. Advancements in constituency parsing contribute to the development of more sophisticated NLP systems capable of handling complex linguistic phenomena[10].

2.3. Approaches to parsing

2.3.1. Rule-Based Approach

It is an approach that utilizes predefined rules and grammatical constraints to analyze the syntactic structure of a sentence. These rules are typically derived from linguistic theories or formal grammars and guide the parsing process thus, rule-based parser based on lexicalized probabilistic context-free grammars. The parser employed handcrafted rules that incorporated syntactic and lexical information to generate parse trees. The approach achieved competitive parsing accuracy on standard benchmark datasets[11].

2.3.2. Stochastic Approach

Stochastic approaches for constituency parsing involve training models on annotated data to learn patterns and dependencies between words and constituents. These models use various algorithms, such as support vector machines (SVM), conditional random fields

(CRF), to make predictions based on learned features. ML-based parsers often rely on handcrafted features, such as part-of-speech tags, word embedding‟s, or syntactic features, to capture the syntactic structure of sentences. In early research, introduced a machine learning-based parser that used a probabilistic model called a maximum entropy model. The parser was trained on annotated data and learned to predict the most likely syntactic structure for a given sentence based on observed features[12].

2.3.2.1. Supervised Learning

It involves training models using labeled training data, where both input features and corresponding output labels are known. The models learn to make predictions by generalizing from the labeled examples, aiming to accurately map input features to the correct output labels. Introduced the BERT (Bidirectional Encoder Representations from Transformers) model, which utilized supervised learning for various natural language processing tasks[13]. BERT was trained on large-scale labeled data, such as the Books Corpus and the English Wikipedia, with tasks like masked language modeling and next sentence prediction. The supervised training allowed BERT to learn contextual representations of words and achieve state-of-the-art results on tasks such as question answering and text classification.

2.3.2.2. Unsupervised Learning

It involves training models on unlabeled data without any predefined output labels. The models learn patterns, structures, or representations in the data without explicit guidance, aiming to discover meaningful information or groupings within the data.Pre-trained Transformer-XL (PT-XL) model, an unsupervised machine learning approach for language modeling and it was trained on large amounts of unlabeled text data from books and websites, utilizing self-supervised learning techniques. The model learned to predict future words given past context, capturing long-range dependencies in the text. PT-XL achieved improved language modeling performance, demonstrating the effectiveness of unsupervised learning for language understanding[14].

2.3.2.3. Neural Networks

It involves for constituency parsing that leverage neural networks to automatically learn hierarchical representations of sentences and capture their syntactic structures. These techniques typically involve the use of recurrent neural networks (RNNs), such as long short-term memory (LSTM) or gated recurrent unit (GRU) networks, to process sentences

and encode contextual information. They can learn complex patterns and dependencies from large amounts of data without the need for handcrafted features. proposed a deep learning-based parser that employed an LSTM network to encode the sentence and generate a syntactic parse tree[15].The model was trained on a large corpus of annotated data and demonstrated improved parsing accuracy compared to traditional machine learning approaches. The model employs LSTM-based encoders and pointer networks to generate parse trees in a transition-based framework. The approach achieved competitive parsing performance on multiple languages, highlighting the effectiveness of pointer networks in capturing hierarchical structures[16].

2.4. Parsing strategies

We discussed the different approaches that have been proposed to apply a set of grammar rules to the analysis of a sentence. These various approaches can be split up on a number of strategies, each strategy concerning different priorities in the derivation of a parse tree.

2.4.1. Top-down Parsing

A top-down parser starts with the S symbol and attempts to rewrite it into a sequence of terminal symbols that matches the classes of the words in the input sentence. The state of the parse at any given time can be represented as a list of symbols that are the result of operations applied so far, called the symbol list. Here is an example; the parser starts in start state (S), after applying the grammar rule S → NP VP, the symbol list would be (NP VP). It then applies the rule NP →ART N and the symbol list would be (ART N VP), and so on. The parser might recursively continue in this way until it arrives completely at terminal symbol states, and then it might check the input sentence to see if the classes of words in it matched with the rewritten sequence of terminal symbols[17] .

2.4.2. Bottom up parsing

In a bottom-up strategy, we start with the words in the sentence and use the rewrite rules backward to reduce the sequence of symbols until it consists solely of S. The left-hand side of each rule is used to rewrite the symbol on the right-hand side [17]. That is, it begins from each word and assigns its grammatical category until it reaches the start symbol. In bottom-up parsing, the operation is repeated at each state using the highest-level labels as the new string to operate on. This technique aims to group words into their respective categories based on the allowed rules. For example, a sequence like ART ADJ N can be identified as a noun phrase (NP) following the applicable rule.

2.4.3. Chart Parser

The solution to such problems is using another variant of tabular parsing algorithm called Viterbi algorithm. Viterbi algorithm is the best deep selection algorithm that generates the most possible parse of sentences (Viterbi Parse) given the grammar[18]. The Viterbi algorithm is an efficient algorithm used for selecting the most probable parse tree in the presence of ambiguity. It is called Viterbi parse selection because it employs dynamic programming techniques to compute the highest probability parse tree. By considering multiple possible interpretations of the input, the Viterbi algorithm evaluates and compares the likelihood of different parse trees. It selects the parse tree with the highest probability or likelihood as the most probable one, effectively resolving the ambiguity in parsing. Finally, depend on our researches behavior there's a notable shift towards employing bottom-up parsing strategies, particularly in the realm of constituency parsing tasks. This shift is largely propelled by the pervasive adoption of deep learning techniques, where methodologies like word embedding and vectorization, such as one-hot vectors and Word2Vec structures, play a pivotal role. These techniques enable the representation of linguistic elements in a continuous vector space, facilitating more nuanced understanding of language semantics. Transition-based algorithms, often associated with dependency parsing, have found adaptation in constituency parsing tasks, allowing for a sequential building of parse trees starting from individual words to construct the syntactic structure of sentences. This approach aligns well with the principles of deep learning, where sequential processing and hierarchical representations are central. Moreover, Seq2Seq models, a prominent architecture in the deep learning paradigm, have been leveraged to incorporate bottom-up parsing strategies effectively. These models, with their encoder-decoder architecture, provide a framework for mapping input sequences (words) to output sequences (parse trees or constituents) in a coherent manner. Consequently, the amalgamation of bottom-up parsing strategies with deep learning techniques, transition-based algorithms, and Seq2Seq models yields promising advancements in NLP, enabling more robust and accurate parsing of natural language sentences[19].

2.5. Overview of Siltinga language

Each language has its own grammar rules and sentence construction techniques. This suggests that, even when the necessary information from the provided sentences was the same, word and phrase arrangements can vary from language to language. However, Siltigna has a word order form of SOV, just as the majority of other Semitic languages (Subject-Object-verb)[20]. Silt‟e uses Ge‟ez or Ethiopic characters for writing purpose since the 1980s. The Amharic and Tigrigna languages most frequently use this system, which was originally developed for the now extinct Ge‟ez language. Although there are typically seven vowels in languages like Amharic and Tigrigna, the vowels in the Siltigna language are different from those languages. The Ge‟ez script makes distinctions among only seven vowels, so some of the short-long distinctions in Siltigna language are not marked. Since there are not many minimal pairs depending on vowel length in actual usage, understanding is probably not affected. In written Siltigna language, the seven Ethiopic vowels are mapped onto the ten Siltigna language vowelsas follows: the five short vowels are አ |a|, ኡ |u|, ኤ |e|, እ |i|, ኦ |o| and the five long vowels are ኣኣ |aa|, ኡኡ |uu|, ኢኢ |ii|, ኦኦ |oo|, ኤኤ |ee|.Only the short and long a and the short and long i have been distinguished in the alphabet. The short „a‟ is indicated by the first order (form) of the Ethiopic script and the long „a‟ by forth order (form). The short „i‟ and consonant alone are indicated by sixth order (form) and the long „i‟is indicated by the third form. The third form is also usedword finally when the word endsin „i‟ See the example below.

ä -> a,aa: አለፈ /alafa/ „he passed‟, ጋራሽ |gaaraaš| 'your (f.) house'

u-> u, uu: ሙት /mut/ „death‟,/muut/ „thing‟

i->i: ሚሪ /mari/ „friend‟, i->ii: ኢን /iin/ „eye‟

o->o, oo: ቆጮ |k'oc'e | 'tortoise', |k'ooc'e | „he cut‟

e ->ee: ኤፌ |eeffe| „he covered‟

For an Ethiopian Semitic language, the Siltigna language has a typical collection of consonants, including the letter ፕ |p|. Common ejective consonants are present, along with plain voicelessand voiced consonants and all consonants, with the exception of /h/ and /ʔ/ can be geminated, or prolonged. The symbols /p/ and /ʔ/ (glottal stop) play only a marginal role in the system because |p| appears in only a few words and |ʔ| appears in Amharic it is often omitted. Consonants in Siltigna language writing system with their vowel representation[21].

2.5.1. Siltigna language word classes

Three fundamental characteristics are typically taken into account when classifying words into their respective parts of speech. They are: the word's connotation, its physical appearance, and the context in which it appears in a sentence. These can serve as the primary criteria for classifying a specific word[22]. Because of this, traditional Siltigna grammar categorizes words into eight groups or parts of speech namely verbs, nouns, adjectives, adverbs pronouns, prepositions, conjunctions, & interjections depending on the grammar of other languages, rather than categorizing words based on Siltigna language features. This classification bases its decisions on other linguistic structures or features. The current grammars of the Siltigna language divided the words into five categories as a result of the aforementioned issues and other issues with this classification. As a result, the word classes present in the previous grammar are condensed into only five groups: noun, adjective, verb, adverb, and preposition. Pronouns have the same role and characteristics as nouns in the contemporary classification of words. As a result, the pronoun is entered in the noun. Prepositions and conjunctions serve the same purpose in Siltigna language, which means that they are joined with other words and utilized for linking. Interjections, which are words without grammatical functions, are not taken into account as parts of speech in this categorization or are omitted from the category altogether. Some linguistics classifies words into six word categories by updating the above modern classification such as major word class (i.e. noun, adjective, verb and adverb) and minor word classes (i.e. preposition and interjections). The early scholars' categorization is used in this study. The reason for this preference is because the early classification is more thorough and enables the tagger to tag terms thoroughly. The remainder of this chapter is devoted to discussing the several POS categories discussed previously, as well as the Siltigna language's Numerals and Punctuations[23].

Siltigna noun class:-Noun is the name given to most people, places, or things. Since lexical classes like nouns are defined functionally, not all words for people, places, or things are nouns. In the same way as some nouns are not words for people, places, or things. Therefore, depending on their position in the phrase, nouns are most often used as the subject or object in Siltigna language, while they are also occasionally used as adjectives. The words in the following sentences that are underlined are examples of nouns. ወራበ የገዎን ብሳወ። / Hyena is wild animal./, ከማል ገበየ ሄደ። / Kemal went to the market.

According to[22]describe the nouns based on their context and shape as flows:-If the word take postfix /-ችኦ/ , for example፡- ሰብ + ች=ሰብች/persons/, ከራብ + ች=ከራብች/oxes/ If the

word is used as subject of the sentence and for example: - ጀማል አራሺን። /Jemal is a farmer / within this sentence the word Jemal is usedas subject of the sentence and if the word is used as object with in the sentence for example: - ጀማል ከራበ ወከበ:: /Jemal bought an ox / within this sentence the underline word is used as an object of the sentence and if the word is preceded by demonstratives. For example: - ሀታይ ሚሸ/this man / within this sentence the underlined word is used as demonstratives. Then the word after the demonstratives.is used as noun. In Siltigna language, nouns are also described in singular and plural numbers. Nouns that are quantified and do not have any linguistic suffixes are known as singular nouns. For instance, ከራብ “ox”, ጣይ “sheep”, ሰብ “man/women” etc... Therefore indicate single entity or animal or person. Quantified plural nouns in Siltigna can be formed in two ways. The first way is, by preceding numerals before singular nouns. For example, ሼሸት ከራብ “three ox”, ሆሸት ጣይ “two sheep” and so on. Plural nouns are also formed through the addition of suffixes in the language. The most common plural suffix /-ችኦ/.

2.5.2. Morphology derived from noun

Table 2-1 Formation of nouns by adding suffixes

Word	**Singular**	**Plural**
ከራብ(ox)	ከራብ(ox)	ከራብች(oxen)
ጋር(house)	ጋር(house)	ጋርች(houses)

Nouns in Siltigna language also formed through derivation from other classes. For example, the following words are nouns derived from other classes: ሰብዬ <child> = ሰብዬነት<boy hood> from adjective to nounፋፍ <brave> = ፋፍነት<bravery> from adjective to noun ሰቼ <drink> = ሰቺል<drinker> from verb to noun

Proper nouns and common nouns are the two categories for nouns.Proper nouns are names of particular people or things, such as ሰይፈዲን/Seiyfedin, ሶዶ /Sodo etc. Mass nouns and count nouns are the two categories for common nouns. Counting nouns can be employed in both singular and plural forms, allowing for grammatical enumeration (ከራብ “Ox“/ከራብች “Oxen”, ጋር „House” ጋርች “houses”) as well as counting them like (ሀድ ከራብ

“one Ox“, ሆሽት ከራብ “two Ox”). Mass nouns are used to describe a homogeneous group. For instance መይ “Water” and ንፋስ “Wind”.

Siltigna pronoun class:-Pronouns are words that can be used in place of nouns or noun phrases, according to linguistic theory. It is used to make an implied reference to the sentence before it in a given text. Pronouns are also employed to convey meaning when the name of an object is not known. There are various types of pronouns, including as personal pronouns like he (ኡሀ), I(እኔ), we(እኘ), she(እሸ), you(አቱም፣አሽ፣አተ) they(ኡሁን), demonstrative pronoun (for example this(ህታይ), that(ሀእታይ/ሀታይ),those(ሀታይሱር), possessive pronoun (for example mine(የኔ), our(የኛ), your(ዩሁን፣ የቱም፣የተ፣የሽ), his(ዩሀ), her(ዩሸ)), reflexive pronoun (myself(እኔገጌ), oneself(በገግ)). As nouns, pronouns characterize based on number and gender. For instance, እሸ(She)፣ኡሀ(He)፣ኡሁን(They) are some. Let us see the following examples:- <u>እሸ</u> አሽርጌታንት። “She is a teacher.”, <u>ኡሀ</u> አሽርጌታን። “He is a teacher.”, <u>ኡሁን</u> አሽርጌታኒሙ። “They are teachers.”In the above example all the underlined words are used as pronouns.

Siltigna adjective class:-Adjective is the term that explains or describes a noun in a sentence. Its major goal is to provide a precise definition for the noun and additional details about the people, things, or animals that the noun or pronoun is meant to represent. Adjectives in Siltigna language typically characterize nouns by providing details about an object's behavior, weight, height, size, shape, color, and other characteristics. (for example ፈየ|fayya| “good” ቦዝ|booz| “bad” የሮሬ|yaroore| ቀል|qall| „small“ ጉመራ|gumara| „white“)etc.. For more clarification let us look the following example:-<u>ፈየ</u> ጋር አለይሙ ። /„They have a beautiful house.“ In this sentence the word ፈየ is used as adjective that describes ጋር „the house“. Adjectives in Siltigna sentences can be formed from original adjectives and derivative adjectives. Original adjectives are words that are not derived from any other word class, whereas derivative adjectives are words that are created by affixing various suffixes to other word classes like noun, verb, and soon. In the Siltigna language, adjectives modify a noun's behavior (structure) according to its numeric value, its gender, and its occupation.

Siltigna verb class:-The verbs are words or compound of words that expresses action, a state of being and/ or relationship between two things[22] . In the Siltigna language in its most powerful and normal position, it is found at the end of the sentence and also one verb is used as sentences in the language's nature. Also in Siltigna language the words that take

the following suffixes (/-ህኡ/፣ /-አአህአ/፣ /-ሽ /-ህኡኡ/፣ /-አአምኡ/፣ /-አአት/፤አአንአ፣…) are categorized under verb class.

Siltigna adverb class:-Adverbs are adjectives that are used to describe verbs. In Siltigna, adverbs frequently come before the verbs they modify. The Siltigna language's adverbs, adjectives (including numbers), clauses, sentences, and other parts of speech are also modified by it. Adverbs indicate the magnitude of an activity (for example, tremendously, Degree), the method of the action (for example, slowly, Manner), the location (for example, here, Directional or locative), and time of an event or an action that occurred (for example yesterday, Temporal). Similar to English, Siltigna language verb modifiers or verb phrases typically express information by providing examples of how, where, and how often an action occurs (frequency). See the sample below for further explanation. Adverbs of manner -> በሎድት (slowly) ፣ ዱምቡጠም(Accidentally) ፣ ኮሞኮሞ(fast),Eg. መዲነ ኮሞኮሞ መጣት። /medina comes fast./ Adverbs of frequency ->በሰ ገርበ(every day) ፣ አደደጊነ(sometimes) ፣ ሁለጊነ(always) ፣ በለ ወክተ

(mostof the time) Eg. ኢሄ አደደጊነ የትጩኜውቡይ አየም እረሳው።/ I sometimes forget my birthday./ Adverbs of place-

ሙለ(outside) ፣ ፈት(wide) ፣ ደር(on) ፣ ኡስጥ(in) ፣ ኮሎ (below,down)Eg. ማክደ በገበየ መጣት:: /makida come from the market./

Adverbs of time ->ታቼነ(yesterday) ፣ አጉቼነ(earlier) ፣ አኩ(now) ፣ ባለፈይ(previously) Eg. አሚነ ታቼነ መጣት::/amina cames yesterday. /.

Prepositions: - Prepositions are words that lack meaning on their own and must be used with other basic word classes like nouns, adjectives, adverbs, etc. The word category in which they function as a syntactic unit may be placed either before (preposition) or after (postposition) in constituents of structures. Their function is to express connections or associations between objects, people, events, or other entities. Prepositions are used with verbs, nouns, and pronouns in Siltigna. In this language, examples of prepositions and postpositions include:ነምኮ(like), ደር(on) አዘር(to)የሬር/በሬር(behind) በቀደ/ለቀደ(infront)ሬር(back) ስር(under) ፈሬ(without)ኮሎ (below)፣ ፎኖ(to)፣ ተ፣ ለ፣ ኮ፣…and so on.

Conjunctions:-Words that connect phrases, clauses, words, and sentences are known as conjunctions [22]. They can be divided into coordinating conjunctions and subordinating conjunctions in Siltigna language. Additionally, they could be a letter, word, phrase and prefix, or postfix. The most common conjunctions in Siltigna language are: ዋ(and)

ንገ(since)፣ የ(Ya)…ነከ(like)፣ አነግነ(or)፣ በሎነ(if..not)፣ የውንበልዳላ(even..if)፣ ሆነም ታላ(nevertheless)፣ ኢጲሚየነቀ(thus)፣ በውኖትምከ(so)፣ ሂታሚ(such as)፣ በልቀሬ፣ ሆናኔም(yet)፣ ሎነምከ(therefore)፣ ቢዮንም(but)፣ ግንመ(however)፣ እጲአሰነት(because of this)፣ በልዳላ(although)፣ ሎነከ(because) ፣ በ(ba)… See the sample below for further explanations.For example፡-ዘምዘም ዋ መዲነ ዋሽታ ማጤትኑም።/Zemzem and mdina are sisterhoods. አህመድ ቆጣሪን ዘልዘሎተኘ፤ ሆነምታላ በለ ፈረንከ ኤለይ/Ahmed is acumen nevertheless, he has no enough money. When we see the above example the letter „ዋ‟ and the word „ሆነምታላ‟are used as conjunction.

Interjections:-Interjections are words that serve certain purposes in the Siltigna language to convey emotion, sudden surprise, pleasure, sadness, and soon. Similar to other languages, Siltigna has a variety of words or expressions for emotions, surprise, joy, displeasure, and so forth. Therefore they are not included in the list of parts of speech for the Siltigna language. These Siltigna interjections can appear anywhere in a sentence or standalone by themselves outside of one. For instance, words like ሀይ! ሀበይ!(even!) አሹ!(yeah!) ኤጋሀ!(please!) ያጅባን!(amazing!) etc. are common interjections in Siltigna language.

Numeral:-All items with a quantity or amount reference are known as numerals. These numbers can be ordinal or cardinal. Counting numbers which indicate amounts and are name as cardinal numbers. For more clarification the example bellow

ሀድ => one ስድስት => six በቅል => hundred

ሆሽት => two ሰአብት => seven ኪም => thousand

ሼሽት => three ሱሙት => eight

ሀራት => four ዚጠኜ => nine

ሀምስት => five አስር => ten

On other side Ordinal number tells us the rank of things. Examples of ordinal number are:- ሀድለኜ => first, ሆሽትለኜ => second, ሼሽትለኜ => third, ሀራትለኜ => fourth, ሀምስትለኜ => fifth, ስዴስትለኜ => sixth, ሰአብትለኜ => seventh, ሱሙትለኜ => eighth, ዚጥለኜ => ninth, አስርለኜ => tenth and በቅለኜ => hundredth etc. As seen from this example Siltigna ordinal number adds suffix ለኜ atthe end of the word.

2.5.3. Siltigna tentative POS tag set and Phrase Tag set

Table 2-2 Siltigna tentative POS tag set and Phrase Tag set

Basic Category	Derived Tag	Description	Example
Noun	NN	Includes all types of nouns, invariant for number, gender and case	ስሌጤ(silte)፣ ዝሊም(rain)
	NNp	Noun with not separated Preposition	በስሌጤ(in silte)
	NNc	Noun with not separated Conjunction	ከዴርም(Kedir also)
Pronoun	Pr	Tag for all pronouns invariant for number,gender and case	ኡሀ(he),እሸ(she),የኘ(our)
	PrC	Pronouns with non-separated conjunctions	በኘም(with us also)
Verb	V	Tag for all main verbs	በሇ(eat),መጠ(come)
	Aux	Auxiliary verbs in their all Forms	ነረ(he/it was),አሇ(it/he present)
	Vi	ntransitive verb followed subject	
	Ng	Negation	ኢለይ፡የለ(not have)
	Vp	Verb with non-separated Preposition	በመጠ(after he came)
	Vc	Verb with non-separated Conjunction	በሎኒ(as soon as he eat)
Adjective	ADJ	An adjective which is not attached with othercategories	ፈየ(good),ጉመረ(white)
	ADJp	An adjective which is not separated from prepositions	ተጤም(with black)
Adverb	ADV	Tag for all types of adverb	ሁለግነ(always), አደደግነ(sometimes)
Preposition	P	preposition that are not attached with other wordcategories	በ(by),ተ(with),ለ(to)
Conjunction			
	C	Conjunctions that are not attached	ዋ(and)ሀነግነ(or),ብዮንም(eve

		with otherword categories	n if)
Numeral	Cn	Cardinal number	ሀድ(one),ሆሽት(two)…አስር(ten)
	On	Ordinal number	ሀድለኜ(first)ሀምስትለኜ(fifth),1ለኘ፣
	CnP	Cardinal number withPreposition	በሆሽ(by two
	AdjN	A numeral that functionas an adjective	ሆሽት ላም(two cows)
Punctuation	Pun	Tags for all punctuationsMarks	።, ፣,፤ ,፡ ,?, !
Interjection	Int	Interjection	ሀሹ!(wow),ኤጋሀ!(please)
Noun phrase	NP	In the sentences it comes head noun, combines with adjective	ደርሳይ ቆጣሪን/NP
Verb phrase	VP	In the sentences it comes head verb	ክነበለ ሞጡ/VP
Adjective phrase	ADJP	it comes head adjective in the sentences	አንጭሪ ሚሽ/ADJP
Adverbial Phrase	ADVP	it comes head adverb in the sentences	ኮሞ ኮሞ ነሙ/ADVP
Prepositional Phrase	PP	head preposition in the sentences	ለከምሰ/PP ጂጉ

2.5.4. Silitigna language phrase Categories

A phrase is a syntactic structure that consists of a small group of words that stand together. It is wider than a single word but smaller than a complete sentence. A phrase can be constructed by combining a head word with other words or phrases and additional elements in a phrase can serve as specifies, modifiers, or complements[24]. Specifies are words that provide specific information about the identity, location, number, possession, and other characteristics of the head word.

Noun phrase:-A noun phrase (NP) is composed of a noun or pronoun as its head and other constituents. NP can be simple (when the head is a single noun or pronoun) or it can be complex (when it is formed by the combination of a noun with other word classes, including noun word category, or phrases). Noun phrases: - examples (NP(NNp ለበክት)(Vp ሁርም)(Ng ኤለይ))

Verb phrase:-A verb phrase (VP) is made up of a verb, which is the head of the phrase, and other constituents like complements, modifiers or specifies. Verbs of the verb phrase can be classified as transitive and intransitive based on the word category of the complement they take. Transitive verbs take noun phrases as their complement whereas

intransitive verbs do not; instead they take prepositional phrase. Examples (VP(Vp አልርከቦቱ)(Vc ሀርምኔ))

Adjective Phrase: - is a group of words in which the main word is an adjective and are words that describe nouns, often placed before the noun they modify. Below is an illustration of an adjective phrase. (ADJP (ADJ አንጭር) (NN ኡማር))

Adverbial phrases: - Adverbial phrases (AdvP) are constructed from one or more adverbs, as modifier, and other word categories. However, unlike other phrases AdvPs do not take complements as their constituents. Most of the time, the modifiers of adverbial phrases are PPs. For example (ADVP (ADV አኩም) (NN ዜግነተ))

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

2.5.5. The silitigna language Sentences

Simple Sentences:-It is a sentence which is composed of single Noun (N) and single Verb (V) that gives full information but it doesn‟t mean it couldn‟t contain the complements and modifiers in the verb. Examples: - „እሪ በጫል ሬበታን‟

Complex Sentences:-It is a sentence which is composed of two or more nouns and verbs. Here is an example of simple sentence „ጨረኛዋ ሱበ እቴጀባኔ ያቀርባን‟.

2.6. Grammar formalisms

In natural language processing (NLP), grammar refers to a set of rules that define the structure and composition of sentences in a particular language and provides a framework for analyzing and generating sentences, specifying how words and phrases can be combined to form grammatically correct sentences. Grammar in NLP serves two main purposes: specifying the set of grammatically correct sentences within a language and assigning a structure to each grammatical sentence[25]. Different types of grammatical formalisms have been proposed by scholars to achieve these goals. Some of the commonly used formalisms are:-

2.6.1. Context-free grammar

It is a formalism that specifies the rules for generating sentences by combining constituents (such as phrases and words) based on a set of production rules. It is widely used in syntax analysis and parsing tasks. A context-free grammar (CFG) is a formal system used to describe the syntax of a language. It consists of a set of grammar rules, where each rule has a single non-terminal symbol on the left-hand side and a sequence of symbols (terminals or non-terminals) on the right-hand side[26]. The grammar rules define how the non-terminal symbols can be replaced by sequences of symbols.

A CFG is defined by a four-tuple G = (N, Σ, R, S), where:

$$L(G) = \{w \in \Sigma^* : S \overset{*}{\Rightarrow} w\}$$

N is a finite set of non-terminal symbols representing syntactic categories.

Σ is a finite set of terminal symbols representing the vocabulary or alphabet of the language.

R is a finite set of rules, each in the form X -> Y_1 Y_2 ... Y_n, where X is a non-terminal symbol, n is the number of symbols on the right-hand side, and Y_1, Y_2, ..., Y_n are symbols from the set N U Σ. S is a distinguished start symbol from N, indicating the root of every parse tree[27]. For example, let's consider the following CFG: - S -> NP VP,NP -> Adj N and VP -> V NP.

$$R = \{S \rightarrow NP\ VP,\ NP \rightarrow Adj\ N,\ VP \rightarrow V\ NP\}$$

In this grammar, there are five non-terminal symbols: S, NP, VP, Adj, and N. The terminal symbols are not explicitly defined but would represent words in the language. The start symbol is S, indicating that every valid sentence in the language must have S as its root. The rules specify how the non-terminals can be expanded: S can be replaced by NP VP, NP can be replaced by Adj N, and VP can be replaced by V NP. This CFG can generate sentences such as "The cat eats" by expanding the non-terminals according to the production rules. It provides a formal framework for describing the syntactic structure of a language and is used in various NLP tasks, including parsing, language generation, and syntactic analysis.

2.6.2. Context Sensitive Grammar

These rules are used in a natural language to describe subject-verb agreement with respect to number, i.e., singular or plural as reflected in sentences; the student comes,

and the student comes [28]. A Context-Sensitive Grammar is a four-tuple, like that of context free grammar, G= (N, Σ, P, S) where;

N is a set of non-terminal symbols,

Σ is a set of terminal symbols,

S is the start symbol of the production and P is a finite set of production rules of the forms α1Aα2□α1βα2 (where a single non -terminal A ∈ N and α1, β, α2 ∈ (N U Σ) +).

2.6.3. Probalisitic Context Free Grammar

It extends context-free grammars by assigning probabilities to each production rule. These probabilities represent the likelihood of choosing a particular rule during a derivation (parse). The overall probability of a parse is then calculated as the product of the probabilities of the individual production rules used in that parse[26].The validity of a probabilistic grammar is limited by the context in which it is trained. The training dataset influences the probabilities assigned to the production rules, and the grammar's ability to accurately model the language is dependent on the representativeness and quality of the training data. Similar to a CFG, a probabilistic context-free grammar ***G*** can be defined by a quintuple:- Where **,M** is the set of non-terminal symbols**, T** is the set of terminal symbols**, R** is the set of production rules**, S** is the start symbol and **P** is the set of probabilities on production rules.

2.7. Related works

Numerous studies have explored various techniques and implementations for constituency parsing. For Ethiopian languages has also seen significant research efforts in constituency parsing, particularly within machine learning models, which have been documented as research endeavors. This section reviews prior works relevant to our study in the constituency parsing field. We have categorized this section into two subsections: parsing for Ethiopian languages and parsing for foreign languages.

2.7.1. Parsing for local languages

According to[1], The study titled "Automatic Sentence Parser For Amharic Language Using SVM" aimed to investigate the development of an automatic sentence parser for Amharic utilizing a machine learning approach. The researchers collected a dataset comprising 370 sample sentences sourced from various sources such as the WIC corpus, Amharic grammar books, news articles, and magazines. Each sentence was manually parsed by the researchers, with comments and suggestions provided by linguistic experts. The dataset they were split into 90% for training and 10% for testing purposes. For their experimental trials were conducted using these sets, leading to the development of a model with an impressive accuracy of 98.91%. The implementation utilized LibSVM with Weka 3.8 for the development of the prototype and the parsing algorithm. The research contributes to advancing automated parsing techniques for the Amharic language, offering promising results for further developments in natural language processing within this linguistic context. The study solely focuses on the use of Support Vector Machines (SVM) as the machine learning algorithm and does not explore the application of deep learning models. The gap lies in the lack of exploration of deep learning models for

automatic sentence parsing in Amharic and the limited adaptability of the implementation tools to other machine learning frameworks.

The research study [29] development of dependency parser for Amharic sentences that focusing on syntactic parsing, two distinct tree banks were utilized the established "Amharic Treebank" containing 1074 instances and a novel Treebank crafted specifically for the purposes of the investigation, comprising 500 instances. The study employed two classifiers for analysis a transition-Action Classifier and a Relation-Label Classifier. Both classifiers underwent testing on 30% of the dataset to assess their performance. They use the transition-Action Classifier exhibited a notably high accuracy rate, achieving an impressive 92% accuracy on the test subset. This classifier is integral to parsing algorithms as it predicts the subsequent transition or action required during the syntactic parsing process. Conversely, the Relation-Label Classifier, though still proficient, demonstrated a slightly lower accuracy rate of 81% when tested on the same dataset. This classifier is typically employed in dependency parsing to anticipate the label or type of syntactic relationship between words. For the research study, they use multiple hyper-parameter tunings were conducted to optimize the classifier models' efficiency. These tunings included adjustments to parameters such as batch size, number of epochs, activation function, and optimizer function. The classifiers were implemented and trained utilizing the TensorFlow-based Keras deep learning framework, known for its flexibility and ease of use in building neural network models. Through systematic experimentation with different hyper-parameter configurations, the study aimed to identify the most effective combination that would enhance the classifiers' performance in their respective tasks. In the research implementation, LSTM (Long Short-Term Memory) deep learning models were utilized, along with the Arc-eager transition system. The models were trained using the Adam optimizer with a fixed learning rate of 0.01 over a span of 30 epochs. This setup aimed to leverage the memory-enhancing capabilities of LSTM networks while utilizing the Arc- eager transition system for syntactic parsing tasks. The choice of Adam optimizer and specific learning rate settings reflects a deliberate effort to balance model convergence and accuracy during training. The results of the study underscored the efficacy of the Transition-Action Classifier, which outperformed the Relation-Label Classifier in terms of accuracy. This finding suggests the potential superiority of transition-based approaches in syntactic parsing tasks, particularly in contexts like the Amharic language. Further exploration and refinement of such classifiers may yield valuable insights into syntactic analysis and contribute to the development of more accurate parsing algorithms.

2.7.2. Parsing for foreign Languages

According to [30] their objectives that focus on parsing Arabic using deep learning technology and they used ATB corpus is comprised of data extracted from linguistic sources written in both standard and modern Arabic language. It encompasses 599 texts sourced from the Lebanese newspaper "An Nahar." These texts are either non-vowel or partially vowelized and segmented. Each word within the corpus is annotated with various pieces of information, including morphological traits, parts of speech, and English translations. Additionally, the corpus contains syntax trees for each sentence. During the training phase, numerous machine learning algorithms require a vector representation as input. In this particular study, they utilized embedding resources, such as those generated by neural systems. In their work, they are interested in the creation of a model for each syntactic level. This model has an important objective, is to determinate the different constituents of a sentence and the different relations between them. Long Short-Term Memory Network (LSTM) presents a building unit for the layers of a recurrent neural network (RNN). An RNN made up of LSTM units is often referred to as an LSTM network. Finally, they use BILSTMs increase the amount of information available on the network and improve the context available for the algorithm.

According to [9]their focus on deep learning techniques applied to constituency parsing of German ,from this the development and evaluation of a constituent parser are outlined based on a dataset comprising 104,787 sentences, split into training, dev and test sets of sizes 84,787, 10,000, and 10,000, respectively. They used python programming in Python 3.7.4, the complex neural network model required significant parameter tuning, taking approximately one day to train on a GeForce GTX 1050 Ti GPU. Modifications were made to Encoder and ELMo modules to align with project requirements. Evaluation relied on comparing predicted syntax trees to golden trees using the F1 score, necessitating sub tree matching. Remarkably, a simple model configuration with one layer and one attention head achieved nearly 90% test accuracy, underscoring the effectiveness of self-attention mechanisms in capturing sentence context. The self-attention matrix elucidated the influence of words on each other, vital for the CKY decoder to predict syntactic structures. With multi-layered self-attention networks, the constituent parsing model attained an impressive F1 score of 93.68%, with higher-layer heads discerning abstract patterns. These findings underscore the importance of dataset quality, model architecture, and GPU efficiency in effectively training and evaluating such intricate models.

Table 2-3 Related works

No	Authors	Title	Tools	Dataset	Results	Research gaps
1	B.Achamyeleh, B. Hailu, and G. Mamo(2021)	Automatic SentenceParser For Amharic Language UsingSvm.	LIBSVM package, Weka 3.8 tools	370 sample sentences from WIC corpus, Amharic grammar book, news article, Magazines	Accuracy of the model was 98.913	Use simple sentences of limited data, their work not implement on other ML models and DL models. Does not include latest DL tools like tensor flow.
2	Rahma Maalej Nabil Khoufi and Chafik Aloulou (2021)	Parsing Arabic using deep learning technology.	LSTM, GRU, BI-LSTM	ATB of 1650 sentences one for learning 70% and one for evaluation 30%.	F-score=83.24% accuracy.	For their implementation not use transformer models to increase train and test accuracy.
3	Kandhasamy Rajasekaran(2020)	Deep Learning techniques applied to Constituency parsing of German	Pytorch, LSTM	The dataset consists of 104,787 sen tences.	F1 score of 93.68% for Noun and verb	Not use multi-layered self-attention networks and other deep learning models BiLSTM,GRUand transformer models, only use LSTM model.
4	Mizanu Zelalem Degu, Worku Birhanie Gebeyehu(2022)	Development of dependency parser for Amharic sentences.	Uses transition -action classifier method, LSTM	Use 1574 annotated sentences collected from universal-dependency Amharic Treebank.	91.54% unlabeled and86%labeled attachment scores	For implementation of their dataset does not use other deep learning techniques BILSTM,,GRU,transformer.

The study on German and Arabic languages, their studies dataset accuracy applies only NP and VP from all the datasets they used and the implementation models and tools are not include latest deep learning tools like keras as not shows models predicted results.

2.7.3. Summary of related work

While prior studies have focused on parsing in different languages, there has been a lackof research specifically on constituency parsing in Siltigna language we review existing work in the field, noting a strong endorsement of deep learning algorithms in recent years, particularly for parsing tasks. To implement constituency parsing, we used experiment with various deep learning models, including LSTM, LSTM with attention, BI-LSTM, GRU, and transformer models. We evaluate the performance of these models using accuracy, label attachment scores metrics, comparing the system's output with manually parsed reference sentences. We also employ preprocessing techniques and optimization methods to reduce complexity and enhance parsing efficiency, thereby improving training speed. Our study focuses on implementing constituency parsing using deep learning techniques using transition based algorithms this means it supports sequence to sequence methods and we highlighted the lack of previous research in this specific area and review the use of deep learning algorithms in related studies. They experiment with different models and evaluate their performance using established metrics. Additionally, we employed on preprocessing and optimization techniques to enhance parsing efficiency.

CHAPTER THREE

3. MATERIALS AND METHODS

3.1. Introduction

In this chapter we provide a comprehensive overview of the detailed approaches, methods, tools, and techniques employed in the research on the constituency parsing model. It encompasses the research approach used in the study, which reflects the proposed method for constituency parsing, as well as the materials, tools, and techniques utilized in the thesis. The research followed pre-processed research methods to accomplish both the general and specific objectives of the study by undertaking a series of steps. These steps included data collection, conducting a literature review from various journals, employing specific methods, preparing a parse trees corpus, preprocessing the data, training and testing the proposed framework, developing a prototype, and validating the model. In addition, the chapter provides an in-depth analysis of the techniques and algorithms that were reviewed and implemented to construct the parsing model. Furthermore, it investigates and identifies the linguistic behavior of the Silitigna language.

3.2. Proposed Approach

The general research approach adopted in this study would be discussed as a list of steps:

- Prepare parse trees dataset from various sources of Silitigna languages.
- Conduct a comprehensive review of existing literature on constituency parsing to gather insights from previous studies.
- Align the collected dataset using software tools to ensure the corresponding sentences in the constituency parsing corpora.
- Apply several data preprocessing techniques, including data cleaning, tokenization, word embedding, and padding, to prepare the dataset for training.
- Convert the preprocessed data into vector representations suitable for deep learning models.

- Design and implement a constituency parsing model capable of converting input sentences into the target labels.
- Train the proposed model using the prepared dataset, allowing it to learn the parsing patterns and improve its performance.
- Train the model with different hyper parameters, exploring variations in train rate, embedding dimension, batch size, number of epochs, and train/test percentages.
- Evaluate the performance of the model by assessing its learning parameters and selecting the best-performing configuration for constituency parsing.
- Finally, summarize the findings and write the thesis report, documenting the research methodology, results, and conclusions.

3.3. Corpus of the language

In order to conduct the research in Siltigna language, we collected dataset of 2000 sentences from various sources and manually construct sentences with corresponding parsed trees by using their pos taggs. These sources include the Silitie Zone Education Office, the Siltigna Department of Werabe University, Silitie FM, and grade 1-12 textbooks. The purpose of these data collection was to gather a diverse range of language samples for analysis and study. By obtaining data from these different sources, the research aims to capture the linguistic characteristics and patterns specific to the Siltigna language. These dataset would serve as a valuable resource for further investigation and development of the Siltigna language.

3.4. Materials and tools

Hardware tools: - We utilized an Acer Core i7 processor (2.6 GHz) renowned for high performance. Backed by 6GB RAM for seamless multitasking. Storage handled by a 32GB flash disk for swift data access. This setup empowers efficient handling of complex computations and data management.

Software tools:-Our study encompass a variety of frameworks, libraries, and platforms designed to facilitate the development, training, and deployment of deep learning models for implementing parsing tasks. These tools provide researchers and practitioners with the necessary infrastructure and functionality to explore, experiment, and innovate in the field of deep learning.

Google Colab: -it is a cloud-based platform provided by Google that enables collaborative Python coding and execution, particularly for deep learning tasks. It offers free access to GPU and TPU resources, integrates with Jupyter Notebooks, supports popular deep learning libraries like TensorFlow and PyTorch, facilitates collaboration and sharing, and provides an environment suitable for deep learning experimentation and development.

TensorFlow: -it is an open-source deep learning framework known for its flexibility and scalability. It provides a comprehensive ecosystem for building various types of neural network models, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs) for implementing parsing tensor flow was necessary libraries.

Keras: it is a high-level neural networks API written in Python and capable of running on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK). It provides a user-friendly interface for building and training deep learning models with minimal code.

Jupyter Notebooks: are interactive computing environments that allow researchers to create and share documents containing live code, equations, visualizations, and narrative text. They are commonly used in deep learning research for prototyping, experimentation, and collaboration.

Visio: -it is a drawing concept that refers to a popular software tool developed by Microsoft, primarily used for creating diagrams, flowcharts, and other visual representations. It provides a user-friendly and intuitive interface for designing and editing various types of visual diagrams. It offers a wide range of pre-defined shapes, symbols, and templates that users can utilize to construct professional-looking drawings. It supports the creation of diverse visualizations, including organizational charts, network diagrams, process flows, floor plans, and more. Users can easily drag and drop shapes onto the canvas, connect them with lines or arrows, and add text or labels to customize the diagram.

Table 3-1 Materials and tools

Materials and tools	**Description's**
Python integrated anaconda	To implement data
Google colab	To compile and run python codes
Mendely	For making document citation
Notepad	Used for preparing raw text data
Google drive	Used to store the dataset
Acer laptop,HDD 500GB,6GB RAM	Used for processing computer tasks
Microsoft office	used to write document and make power point
USB flash disk	Used to store data externally

3.5. Data pre-processing

Data pre-processing is an essential step in research as it involves cleaning, transforming, and preparing raw data for analysis. It encompasses various techniques that aim to improve the quality of the data and make it suitable for further processing and modeling. To begin with, missing values are addressed through imputation methods such as mean, median, or regression imputation[31]. Outliers, which are extreme values that can skew the analysis, are identified and handled either by removing them. Additionally, data transformation techniques such as logarithmic or square root transformation, as well as normalization methods like z-score normalization, are used to normalize or scale the data. This helps to reduce the influence of extreme values and ensure that the data is comparable across different variables. In addition to cleaning and transforming the data, feature selection and extraction techniques are employed to identify the most relevant and informative features for analysis. This process helps to reduce the dimensionality of the data, improve model performance, and mitigate the risk of over fitting. Categorical variables are encoded into numerical values using techniques like one-hot encoding or label encoding, enabling them to be used in machine learning algorithms. Furthermore, imbalanced datasets, where the distribution of classes is highly skewed, are addressed by employing techniques such as oversampling or under sampling to balance the dataset and prevent biased model performance.

3.5.1. Data cleaning

Data cleaning, or data cleansing, is an important step in data pre-processing where errors and inconsistencies in the dataset are addressed. It involves tasks like handling missing values by imputing them, dealing with outliers to remove extreme values, correcting inconsistent data formats, removing duplicate entries, and standardizing data formats[32]. By performing data cleaning, researchers can improve the quality and reliability of the dataset, ensuring that it is accurate, complete, and ready for analysis. This process helps to enhance the integrity of the data and prevents biases or errors that could affect the outcomes of subsequent analyses and modeling.

Algorithms 1 Data cleaning algorithms

Input: Dataset

Read the text from the local disc drive for each word in the text

Check whether the word is <title>, <body>, <PUNC>, or <dateline place> If the word is oneof the above

Leave or omit the word

Else

For each character in the word

Check the presence of “<”, “>”, “̈”, “̈:”, “̈:̈” Presence

e of the above is detected Leave or omit the character

Write the characters of the word in the file(write the word)

End If End For

Output: cleaned text

3.5.2. Data Tokenization

Data tokenization is the process of breaking down raw data, particularly text, into smaller units called tokens. It is commonly used in natural language processing tasks to convert sentences or paragraphs into individual words or sub words. Tokenization helps computers understand and analyze human language more effectively by providing a structured representation of the data. This process enables further analysis, such as text classification or sentiment analysis, and allows for the application of various techniques like stemming, lemmatization, or encoding tokens for machine learning algorithms. By tokenizing data, it becomes more manageable and suitable for processing in NLP tasks[33].

Algorithms 2 Data Tokenization algorithm

Input: parse trees dataset

Scan the parse trees dataset and check the presence of a whitespace for each character in the sentence

Check whether the character is white space or not If the character is white space

Print the characters that are before the white space (the word) Else

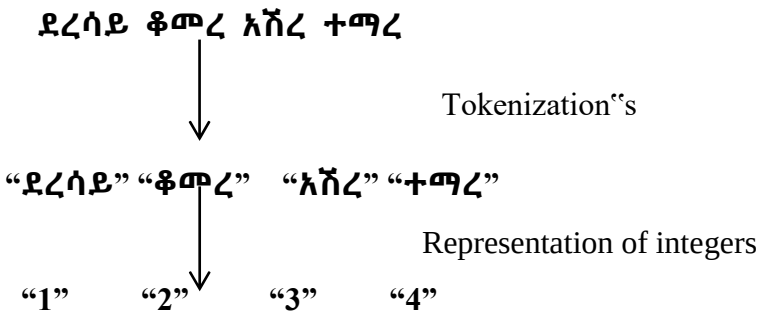
Append the character to a string variable (the variable that will be printed when the if condition is satisfied)

End If End For

Output: separated words of the sentence

In the algorithm described above, we began by exploring and accessing the corpus of data to assign each word a unique number. This process of assigning a special number to a word occurs as soon as the word is encountered, and the words are then represented by these unique numbers. The most recent term in the vocabulary is assigned the last number, with the size of the vocabulary determining the range of these numbers. This process enhances data preparation for training by converting the source dataset into integer number representation form. Once the data is converted to integer form, it is referred to as tokenized data. This tokenization establishes the vocabulary size and preprocesses the raw data for further analysis. The vocabulary is indexed, with the last word having an index of 1 and the first word an index of 0[34].For the dataset used for Silitigna languages in our corpus, we have a total of 12,000 different vocabularies. For example, the phrase "ደረሳይ ቆመረ አሸረ ተማረ" would be tokenized as individual words: "ደረሳይ", "ቆመረ", "አሸረ",

"ተማረ". Furthermore, to delineate the boundaries of sentences, we've employed special tokens. The beginning of a new sentence is indicated by "Bos" (Beginning of Sentence), while the end of a sentence is marked by "Eos" (End of Sentence). This notation aids in the structuring and segmentation of the text into coherent sentences.



In our work, we follow a process to enable machines to understand text data. This involves breaking down sentences into lists of words and representing these words in integer form. Each word is assigned a unique integer during the vocabulary creation process. Once the text is tokenized, further processing steps may include removing punctuation and unnecessary symbols. Tokenization facilitates the transformation of text into a format suitable for input into various natural language processing (NLP) models. In our approach, we tokenize text by splitting it into words using whitespace as the delimiter. This process allows NLP models to extract meaningful information and perform desired analyses on the text data.

3.6. One hot vector representation

One-hot vector representation is a technique used to represent categorical variables or features in a binary format. It is commonly employed in various fields, including machine learning and data analysis. The concept involves creating a binary vector where each element corresponds to a unique category within the variable. If a data point belongs to a particular category, the corresponding element in the vector is set to 1, while all other elements are set to 0[35]. This representation allows categorical data to be encoded in a numerical format that can be easily understood and processed by machine learning algorithms. One-hot vector representation is useful for tasks such as classification, where categorical variables need to be transformed into a numerical representation for analysis and modeling. It helps capture the distinct categories and their relationships within the data, enabling researchers to work with categorical variables effectively in their research studies.

Integer “1”	Integer “2”	Integer “3”
[1, 0, 0, 0]	[0, 1, 0, 0]	[0, 0, 1, 0]

3.7. Padding

Padding refers to the process of adding extra elements or values to a dataset or sequence to make it conform to a desired length or size. Padding is commonly used in various fields, such as natural language processing and machine learning, where sequences of varying lengths need to be processed or analyzed using algorithms that require fixed-size inputs. For example, in text analysis, when working with sentences of different lengths, padding can be applied by adding special tokens or symbols (e.g., zeros) to the end of shorter sentences to make them equal in length to the longest sentence in the dataset. This ensures uniformity and compatibility in the input data, allowing researchers to apply consistent analysis techniques and models. Padding is an important step to handle variable-length sequences and ensure that datasets can be effectively processed and analyzed in research tasks involving sequential data[29].

Integer “4”	Integer “5”	Integer “6”
[0, 0, 0, 4]	[0, 0, 0, 5]	[0, 0, 0, 6]

3.8. Word Embedding

Word embedding refers to a technique used to represent words or textual data in a numerical format that captures semantic and contextual information. It is commonly employed in natural language processing and text analysis tasks[36]. The concept involves mapping words from a vocabulary to dense vector representations in a continuous vector space. Each word is assigned a vector, and the values within the vector encode semantic relationships and contextual similarities between words. Word embedding allows computers to understand the meaning and relationships between words based on their numerical representations. This representation enables researchers to leverage mathematical operations and algorithms on words, such as measuring similarity or performing calculations on word vectors. Word embedding is widely used in various research areas, including sentiment analysis, document classification, machine translation, and information retrieval, to enhance the understanding and processing of textual data.

3.9. Encoder

An encoder refers to a component or model that transforms input data into a different representation or encoding. It is commonly used in various fields such as machine learning and deep learning. The concept involves taking raw data, such as images, text, or sequences, and mapping them to a compressed or latent representation that captures essential features or information. Encoders are designed to extract meaningful patterns or representations from the input data, enabling researchers to effectively analyze, classify, or generate new data based on the learned encodings. Encoders can be trained using techniques like auto encoders, recurrent neural networks (RNNs), or convolutional neural networks (CNNs), among others. They play a crucial role in research tasks such as image recognition, text generation, anomaly detection, and dimensionality reduction. By employing encoders, researchers can transform complex data into more compact and informative representations, facilitating subsequent analysis and modeling processes[37].

3.10. Decoder

A decoder refers to a component or model that takes encoded or compressed representations of data and reconstructs it back into its original form or generates new outputs based on the learned representations. The concept is commonly used in various fields such as machine learning and deep learning. The decoder complements the encoder by reversing the process and converting the encoded or latent representations back into a format that is understandable or usable. Decoders are often used in tasks such as image generation, text generation, or sequence reconstruction. They are trained using techniques like generative models, recurrent neural networks (RNNs), or generative adversarial

networks (GANs), among others. Decoders play a crucial role in research by enabling researchers to generate new data or reconstruct missing or incomplete data based on the learned representations[38]. They contribute to tasks such as image synthesis, language modeling, and data completion, enhancing the capabilities of research models and systems.

3.11. Attention mechanism

It refers to a computational technique that allows models to focus on specific parts or elements of input data when making predictions or performing tasks. It is commonly used in fields such as natural language processing and computer vision. The concept is inspired by human attention, where we selectively pay attention to relevant information while ignoring irrelevant details. In machine learning, attention mechanisms enable models to learn to assign different weights or importance to different parts of the input data. This allows the model to focus on the most relevant or informative elements during processing[39]. Attention mechanisms are often used in tasks such as machine translation, image captioning, or sentiment analysis, where understanding the contextual relationships or identifying salient features is crucial. By incorporating attention mechanisms, researchers can improve the performance and interpretability of models by allowing them to dynamically attend to important information in the input data.

3.12. Performance measurement methods

Performance measurement methods refer to techniques used to evaluate and assess the effectiveness or quality of models, algorithms, or systems. These methods are commonly employed in various fields, including machine learning, data analysis, and experimental research. The goal is to quantitatively or qualitatively measure how well a system or approach performs in achieving its intended objectives. Performance measurement methods can include metrics such as accuracy, precision, recall, F1 score, mean squared error, or area under the curve (AUC)[40]. These metrics provide numerical assessments of model performance across various tasks, such as classification, regression, or clustering. Additionally, qualitative methods such as user surveys, expert evaluations, or qualitative analysis of outputs can be used to assess the performance of systems that involve subjective aspects. By employing performance measurement methods, researchers can objectively evaluate the strengths, weaknesses, and limitations of their models or systems, allowing them to make informed decisions and improvements based on the obtained results.

3.12.1. Accuracy

Accuracy is a commonly used performance metric to evaluate the effectiveness of parsing models. Constituency parsing involves analyzing the syntactic structure of a sentence by assigning hierarchical labels to its constituent parts, such as phrases and clauses. Accuracy, in this context, measures the percentage of correctly predicted constituents compared to the total number of constituents in the parsed sentences. It provides an overall assessment of how well the model is able to identify and label the different structural elements of a sentence. Higher accuracy values indicate that the model is successfully capturing the syntactic relationships within the text. Researchers can use accuracy as a benchmark to compare different parsing models, optimize hyper parameters, or assess the impact of various architectural choices on the performance of the constituency parsing task[41].

Parsing Accuracy = (1/N) * Σ_{i=1}^N 1[P(s_i) = T(s_i)]

3.12.2. Loss function

A loss function is a crucial component used to measure the dissimilarity between the predicted output of a parsing model and the ground truth annotations. The loss function quantifies the discrepancy between the predicted constituency structure and the correct structure in the training data. It serves as a guide for the model to update its internal parameters and optimize its performance during the training process. Commonly used loss functions for constituency parsing include cross-entropy loss or negative log-likelihood loss. These functions calculate the difference between the predicted probabilities of different constituents and the true labels, encouraging the model to minimize errors and improve its ability to accurately predict the syntactic structure of sentences. By minimizing the loss function, researchers aim to train the constituency parsing model to generate output that aligns as closely as possible with the ground truth annotations, enabling it to generalize well to unseen data and produce accurate parsing results[42].

3.12.3. Cross-Entropy

Cross-entropy is a commonly used loss function to measure the dissimilarity between the predicted output of a parsing model and the ground truth annotations. Cross-entropy calculates the average negative log-likelihood of the predicted probabilities assigned to each constituent label compared to the true labels. It quantifies the difference between the predicted and actual distributions of labels, penalizing the model for incorrect predictions. By minimizing the cross-entropy loss, the model is encouraged to produce 0,1 higher probabilities for the correct labels and lower probabilities for incorrect labels. This helps

to guide the training process and improve the model's ability to accurately predict the hierarchical structure of sentences. Researchers leverage cross-entropy as an optimization objective to update the model's parameters iteratively, aiming to minimize the loss and enhance the performance of the constituency parsing task[43].

3.13. Fine Tuning

Fine-tuning refers to the process of adjusting a pre-trained parsing model on a specific task or dataset to improve its performance. Fine-tuning allows researchers to leverage the knowledge and representations learned from a pre-trained model, typically trained on a large dataset or a related task, and adapt it to a more specific parsing task or a smaller dataset. During fine-tuning, the pre-trained model's parameters are updated using the target dataset, often with a smaller learning rate than the initial training. This process helps the model to specialize and adapt its representations to the specific syntactic structures and linguistic characteristics of the target constituency parsing task. Fine-tuning can significantly improve the parsing performance by utilizing the pre-existing knowledge encoded in the pre-trained model, leading to more accurate and robust constituency parsing results. It is a valuable technique in deep learning research for constituency parsing, enabling researchers to achieve state-of-the-art performance even with limited labeled data.

3.13.1. Dropout

Dropout is a regularization technique commonly used to prevent over fitting and improve the generalization ability of parsing models. Dropout randomly deactivates a fraction of the neurons or connections in a neural network during training. This means that during each training iteration, a subset of neurons is temporarily ignored, and their contributions to the model's predictions are removed. By doing so, dropout forces the model to rely on a more robust and diverse set of features and prevents the network from becoming overly dependent on specific neurons or combinations of neurons. This regularization technique helps to reduce over fitting by discouraging the model from memorizing the training data and encourages it to learn more generalized representations that can better handle unseen or noisy inputs. In the context of constituency parsing, dropout can be applied to the layers or connections within the parsing model to improve its ability to generalize and accurately predict the hierarchical structure of sentences, leading to better parsing performance on unseen data[44].

3.13.2. Early stopping

Early stopping is a technique used to prevent over fitting and determine the optimal stopping point during the training process. Over fitting occurs when a parsing model becomes too specialized to the training data, resulting in poor generalization to new, unseen data. Early stopping helps mitigate this issue by monitoring the model's performance on a validation set during training. The validation set consists of data that is not used for training but is representative of the overall distribution. The training process is halted when the performance on the validation set starts to deteriorate or reaches a plateau. This indicates that further training may lead to over fitting. By stopping the training early, researchers can prevent the model from becoming overly complex and ensure it retains the ability to generalize well to unseen constituency parsing tasks or datasets. Early stopping is an effective technique to find the right balance between model complexity and generalization, ultimately improving the parsing performance and avoiding over fitting in deep learning research for constituency parsing[14].

3.14. Evaluation metrics

The evaluation metrics are used to assess the performance of parsing models. These metrics provide quantitative measures of how well the model predicts the syntactic structure of sentences. One commonly used metric is labeled attachment score (LAS), which calculates the percentage of correctly predicted labeled dependencies between constituents in the parsed sentences. LAS take into account both the correct attachment of constituents and the correct labeling of the dependencies[45]. Another commonly used metric is the unlabeled attachment score (UAS), which measures the percentage of correctly predicted unlabeled dependencies between constituents. UAS only considers the correct attachment of constituents without considering the specific labels assigned to the dependencies. Precision and recall are also important evaluation metrics. Precision measures the proportion of correctly predicted constituents out of all predicted constituents, while recall measures the proportion of correctly predicted constituents out of all true constituents. F1 score, which is the harmonic mean of precision and recall, provides a single metric that balances both precision and recall[40]. In addition to these metrics, researchers may also consider other measures such as sentence-level accuracy, which evaluates the percentage of completely correctly parsed sentences, or the average crossing brackets (CB) score, which quantifies the number of crossed brackets in the predicted constituency trees. By employing these evaluation metrics, researchers can objectively assess the performance of parsing models, compare different approaches, and identify areas for improvement in deep learning research for constituency parsing.

Finally, we use label attachment score for evaluating our experimental studies for all the models we have used for implementing constituency parsing tasks, this evaluation techniques expressed in equations as follow:- by break down the evaluation of constituency parsing using labeled attachment scores (LAS) into a series of equations: the first parts are started in define Variables as **NN**: Total number of words in the dataset, **CC**: Total number of correct attachments (where predicted labels match the ground truth labels), **TT**: Total number of words in the dataset for which we make predictions.

The second is calculating correct attachments as follows for each word in the dataset, we compare the predicted label with the ground truth label, if they match, we increment **CC** by 1. And C=∑i=1Nδ(groundTruthi,predictedi) Where: δ(x,y)δ(x,y) is the Kronecker delta function, which equals 1 if x=yx=y and 0 otherwise the ground Truth i ground Truth i represents the ground truth label for word ii, by predicted i predicted i represents the predicted label for word ii. Finally, calculate Labeled Attachment Score (LAS) as LAS is the ratio of correct attachments to the total number of words where predictions are made for LAS=CTLAS=TC Whereas: **TT** is the total number of words in the dataset for which predictions are made. This series of equations defines how to evaluate constituency parsing using labeled attachment scores (LAS) mathematically. You can implement these equations in your code to compute LAS for your constituency parsing model.

$$LAS = (1/N) * \Sigma_{i=1}^{N} [(t_i == p_i) * (h_i == q_i)]$$

CHAPTER FOUR

4. RESEARCH DESIGN

4.1. Introduction

In this chapter, we would discuss the design of general architectures for constituency parsing in the Siltigna language. By using the encoder-decoder architecture, which has become a key technology in machine learning and deep learning, it was predominantly employed in applications involving parsing and other natural language processing tasks.

The first step in this architecture is the encoder, which carries out preprocessing on the input data. The input is read from the specified file path, and after the preprocessing step, the dataset is converted into a vectorized format. Various techniques such as one-hot encoding, padding, and word embedding like Word2Vec and GloVes are applied in the vectorization process. These techniques help in representing the data in lower-dimensional vectors using Tensor flows Keras embedding techniques. The encoder-decoder model is a popular architecture for constituency parsing. It consists of two main parts: the encoder and the decoder. The encoder takes an input sequence of Siltigna parse tree sentences and encodes it into a fixed-length vector. It processes the input sequence token by token, updating its hidden state at each time step. In the encoder-decoder architecture for constituency parsing, the encoder plays a crucial role in capturing the relevant information from the input sentence. It achieves this by iteratively computing a hidden state vector for each word in the sentence. This computation is performed using both the word itself and the previous hidden state of the recurrent neural network (RNN)[37].

As the RNN processes each word, it updates its hidden state, incorporating the current word's information and the knowledge acquired from the previous words. This iterative process continues until the entire sentence has been analyzed. At this point, the last hidden state of the RNN, often referred to as the context or thought vector, encapsulates the relevant information of the input sentence. The decoder, on the other hand, is responsible for generating the output in the original representation space, word by word. It utilizes the information contained in the context vector and the previously decoded words to make accurate predictions for the next word in the target sequence. The specific implementation of the architecture can vary depending on several factors. These include the choice of the RNN cell type, such as a genuine RNN cell, a LSTM cell, or a GRU cell. Additionally, the number of cells per layer and the number of hidden layers are among the parameters that can influence the architecture's performance and behavior. These variations allow for

flexibility in designing the architecture based on the specific requirements and characteristics of the constituency parsing task[37].

On the other hand, the decoder takes the encoded vector and generates a target sequence of constituency parse tree labels one token at a time. At each time step, the decoder takes as input the previously generated target token and the previous hidden state. The hidden state is updated based on the previous state and the previous target token, and it is used to generate the next target token. This process continues until the decoder generates an end-of-sequence token, indicating that the entire target sequence has been generated. The encoding and decoding processes are jointly trained using a Seq2Seq loss function, such as maximum cross-entropy loss. During training, the input sequence is supplied to the encoder, and the decoder generates the target sequence token by token. The final hidden layer's context vector from the encoder and the target embedding sequence are fed into the decoder model to predict the target output. Thus, the context vector is passed through the decoder to generate the output sequence.

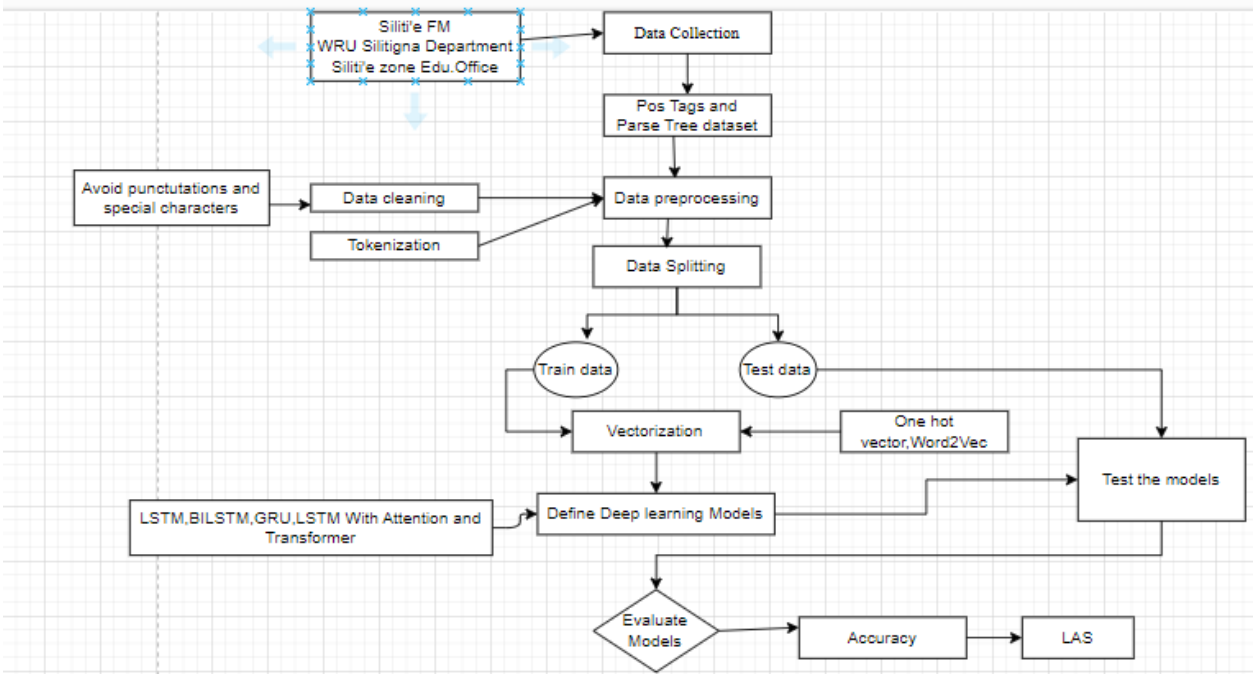


Figure 4-1 Proposed Architecture

4.2. Data Collection and Preparation

In our research study, we carefully assessed several areas to gather the dataset for the Siltigna language. The diagram provided earlier was just an example of the dataset source areas we explored. For the Siltigna language dataset, we collected data from various sources. Firstly, we obtained data from hard and soft copies of Silti'e FM broadcasting texts. These texts provided valuable linguistic content in the Siltigna language. Secondly, we collected text books from the Silti'e Zone Education Office, specifically focusing on the curriculum books of the WRU Siltigna Department. These books contained relevant linguistic material that was important for our research.

To ensure the accuracy and reliability of the data, we consulted with language experts who validated the collected dataset. Their expertise and insights helped to verify the quality and authenticity of the data. Based on the information mentioned above, we prepared a dataset for part-of-speech (POS) tagging and parse trees. This dataset was created by chunking the POS tags, which involves grouping together consecutive words with the same POS tag. By combining data from multiple sources and involving language experts in the validation process, we aimed to create a comprehensive and reliable dataset for the Siltigna language. This dataset would serve as a valuable resource for our research study on constituency parsing and other related natural language processing tasks. From the below graph our data preparation was in all the constituents are labeled in braches notation by using the ATB structures for dataset construction mechanisms that are used for constituency parsing models to predict the correct labels during training the dataset uploads the codes give below constituency trees we show as an example below figure.

```python
def read_parse_tree_from_file(file_path):
    with open(file_path, 'r') as file:
        parse_tree_string = file.read()
    parse_tree = Tree.fromstring(parse_tree_string)
    return parse_tree

# Visualize a parse tree
def visualize_parse_tree(parse_tree):
    parse_tree.pretty_print()
    parse_tree.pretty_print(unicodelines=True)

# Example usage
file_path = '/content/drive/MyDrive/new parse.txt'
parse_tree = read_parse_tree_from_file(file_path)
visualize_parse_tree(parse_tree)
```

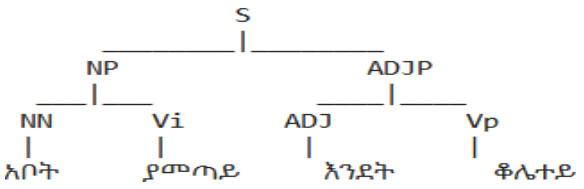


Figure 4-2 Sample parse trees preparation code screenshot

4.3. Pre-processing

In a research study, data pre-processing involves several important steps. One of the initial steps is data cleaning, where the raw data is carefully examined and processed to remove any inconsistencies, errors, or missing values. This ensures that the data is accurate and reliable for analysis. Additionally, tokenization is performed to break down textual data into smaller units, such as words or sentences. This step aids in further text analysis and feature extraction. Finally, data splitting is conducted to divide the dataset into separate subsets, typically training, validation, and test sets. The training set is used to train the models, the validation set is utilized for tuning model parameters, and the test set is employed to evaluate the final model's performance. This division helps ensure unbiased evaluation and prevents over fitting. By performing these pre-processing steps, we can work with clean and properly organized data, facilitating subsequent analysis and modeling tasks[31].

4.4. Data Splitting

For our research study, we prepared a dataset containing 2000 sentences. To ensure reliable results, we divided the dataset into a training set and a testing set. Before the split, we shuffled the data. Shuffling the dataset was important to eliminate any biases or patterns that might have been present in the original order of the sentences. By randomizing the sequence of sentences, we aimed to achieve a more diverse and representative distribution of data points across the sets. Additionally, shuffling the dataset helped prevent over fitting during training. Over fitting occurs when a model becomes too specialized in capturing specific patterns from the training data and performs poorly on unseen data. Shuffling disrupted any potential sequence-based patterns, reducing the risk of over fitting and promoting better generalization.

Table 4-1 Data splitting and usage

Total datasets	**For training used**	**For testing used**
The datasets in percentages (%)	80%	20%
	85%	15%

4.5. Vectorization

In the realm of parsing tasks, vectorization plays a pivotal role as it allows words and sentences to be represented as numerical vectors that can be processed by deep learning models. In this study, various approaches for vectorization were employed, including one-hot vector encoding and word embedding techniques such as word2vec and Glove. From the behavior for our dataset we prepared sentences by corresponding parse trees we use the architecture auto-encoder for seq2seq model implementation for implementing these we applied both one-hot vector and word2vec for representing words in dense vector of vocabulary size determination of the dataset of parse trees. Word embedding, specifically, enables the creation of dense vectors within a continuous vector space, thereby capturing semantic relationships between words. Overall, these methods facilitate the transformation of textual data into a format suitable for analysis and modeling[15].

4.6. DL Model Selection

4.6.1. Long Short-term Memory Model

It is a specialized type of recurrent neural network (RNN) architecture designed to address the vanishing gradient problem encountered in traditional RNNs. This problem arises when gradients diminish rapidly during back propagation through time, hindering the ability of the network to learn long-range dependencies in sequential data. The LSTM architecture includes a memory cell that maintains information over time intervals, enabling it to learn and remember patterns across longer sequences. At its core, an LSTM cell consists of several key components: the forget gate, which determines what information should be discarded from the cell state; the input gate, which decides what new information should be stored in the cell state; the cell state update mechanism, which combines information from the forget gate and the input gate to update the cell state; and the output gate, which controls what information should be output from the cell state. Together, these components empower LSTM networks to effectively capture and utilize long-term dependencies in sequential data, making them well-suited for tasks such as natural language processing, speech recognition, and time series prediction. LSTM models enhance RNNs by extending memory capabilities, allowing them to learn and maintain dependencies over long sequences. This is achieved through gated cells that can read, write, and delete data from memory. Three main gates - forget, input, and output - regulate the flow of information within the LSTM[46]. The forget gate determines whether to keep or discard existing information, while the input gate identifies critical features from the input. During training, weight values control the significance of information, enabling the model to learn what to retain and what to discard.

The sequence-to-sequence (Seq2Seq) architecture involves an encoder LSTM that processes input sequences and generates a context vector. This vector is then used by a decoder LSTM to produce output sequences one step at a time. During training, cross-entropy loss is minimized to improve predictions. The model is optimized through back propagation, adjusting weights based on the gradient of the loss function. Input and output sequences are transformed using embedding layers, and LSTM layers handle sequence processing. During inference, the encoder generates the context vector from input sequences, which is then used by the decoder to generate output sequences[47].The above Encoder LSTM architectures are defined in the following input and out equations for encoding Siltigna language data analysis parse tree datasets like the first Input sequence: $x=(x1,x2,...,xn)$,secondly embedding by: $e(x)$,the learning encoder LSTM : $hn=LSTM(e(x))$and get the context vector: $c=hn$ after encoding the dataset next decoder lstm equations as input context vector and previous output token: $(c,yt-1)$by decoder

LSTM: ht=LSTM(yt−1,ht−1,c) and output token: yt=softmax(Ws∗ht).The training section use loss function as cross-entropy loss and use update rule for back propagation through time use the inference for encode input sequence: c=LSTM(e(x)) and finally described in to generate output sequence yt=LSTM(yt−1,ht−1,c)[46].

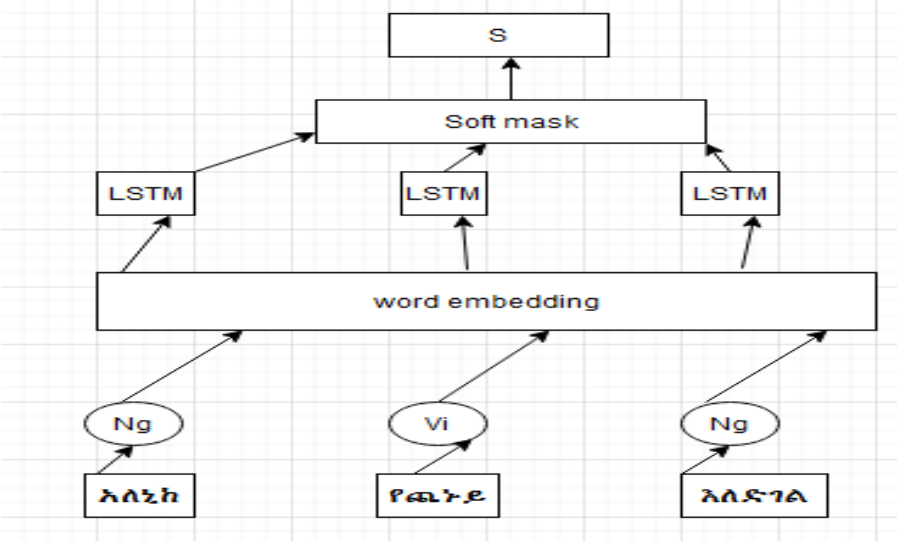


Figure 4-3 LSTM Architecture

4.6.2. LSTM with Attention architecture

In the context of constituency parsing for languages such as Siltinga, employing LSTM attention involves leveraging Long Short-Term Memory (LSTM) networks with attention mechanisms to enhance the parsing process. Constituency parsing aims to break down sentences into their syntactic components, like phrases and clauses, based on formal grammar rules. LSTM networks, known for their ability to capture long-range dependencies in sequential data, serve as the backbone of this approach. In the initial stage of constituency parsing with LSTM attention, the input sentence undergoes encoding, typically into fixed-size vector representations. This encoding step often utilizes techniques like word embedding‟s or character embedding‟s, transforming each word into a numerical representation while preserving its contextual meaning within the sentence[48]. As the LSTM network processes the encoded input sequence, it sequentially analyzes each word, incorporating its contextual information and position within the sentence. Through this process, the LSTM generates hidden representations at each step, effectively capturingthe underlying syntactic structure of the sentence. One of the critical enhancements introduced by LSTM attention is the incorporation of attention mechanisms. These mechanisms dynamically weigh the relevance of different words or phrases in the input sequence at each step of the LSTM's processing. By focusing attention on salient parts of the sentence, the model can better discern syntactic relationships and dependencies, thus improving parsing accuracy. With the LSTM's output and the attention weights in hand,the parsing process culminates in the generation of parse trees. These trees represent the hierarchical structure of the input sentence, delineating constituent phrases and their interrelationships. By effectively integrating LSTM attention mechanisms, constituency parsing models can yield more nuanced and accurate syntactic analyses, particularly beneficial for languages like Silitinga where complex structural patterns may exist. An attention mechanisms have emerged as powerful tools in machine learning, with attentive neural networks providing a structured framework for their implementation across various tasks. The transition functions within these networks define how attention is applied and integrated into the model's decision-making process[49].

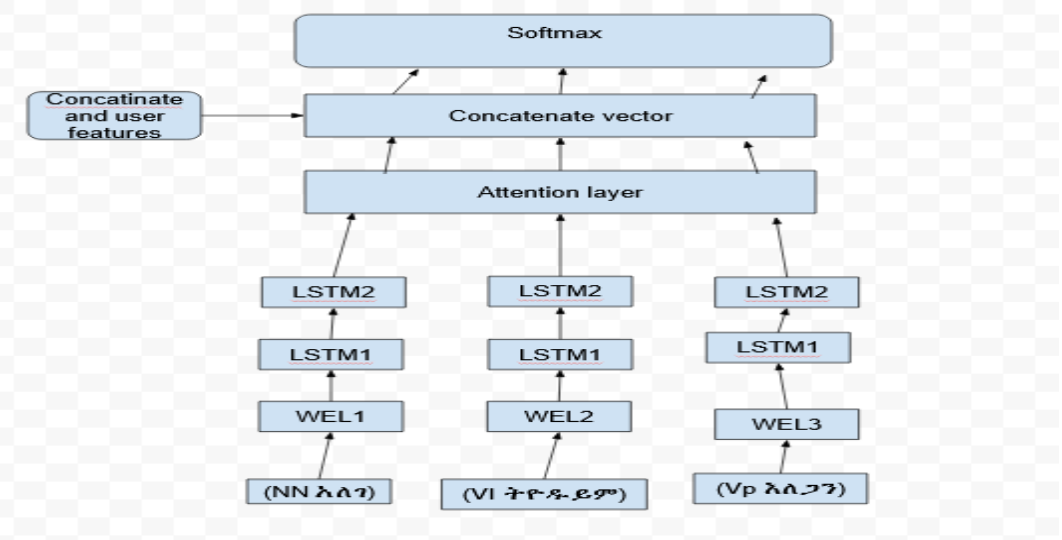


Figure 4-4 LSTM Attention Architecture

4.6.3. Bi-directional Long Short-term Memory Model

Bidirectional Long Short-Term Memory (BiLSTM) models are designed to enhance classification performance by leveraging bidirectional information flow. Unlike unidirectional LSTMs where information flows strictly from past to future, BiLSTMs process input sequences in both directions, from past to future and from future to past, using separate hidden states. This bidirectional processing allows the model to capture dependencies from both the preceding and succeeding contexts simultaneously. By incorporating information from both directions, BiLSTMs gain a more comprehensive understanding of the input sequence, thereby improving the learning capacity of LSTM networks. In essence, BiLSTM models utilize forward and backward hidden states to enhance classification performance by capturing a richer representation of sequence patterns and dependencies[50]. It is an extension of the Long Short-Term Memory (LSTM) architecture that enhances sequence modeling by processing input sequences in both forward and backward directions. In a BiLSTM, the input sequence is fed into two separate LSTM layers: one processes the sequence from the beginning to the end (forward LSTM), while the other processes it in reverse (backward LSTM). The outputs of both LSTM layers are then concatenated at each time step, allowing the model to capture information from both past and future contexts simultaneously. BiLSTM is particularly effective for tasks requiring a comprehensive understanding of sequential data, such as natural language processing and time series prediction, as it can capture long-range dependencies in both directions[51]. For the analysis of the dataset by BiLSTM architectures use firstly forward LSTM equestions use an input sequence: x=(x1,x2,...,xn),forward LSTM: $h\rightarrow t=LSTM(xt,h\rightarrow t-1)$ and the Backward LSTM follow as the equestions input sequence: x=(x1,x2,...,xn), the backward LSTM as: $h\leftarrow t=LSTM(xt,h\leftarrow t+1)$, after these use the concatenate forward and backward hidden

states: ht=[h→t,h←t] for the training:loss function: Cross-entropy loss to update rule: back propagation through time and finally use the inference was as forward pass through both LSTM directions and concatenate hidden states for each time step to obtain the final representation[30].

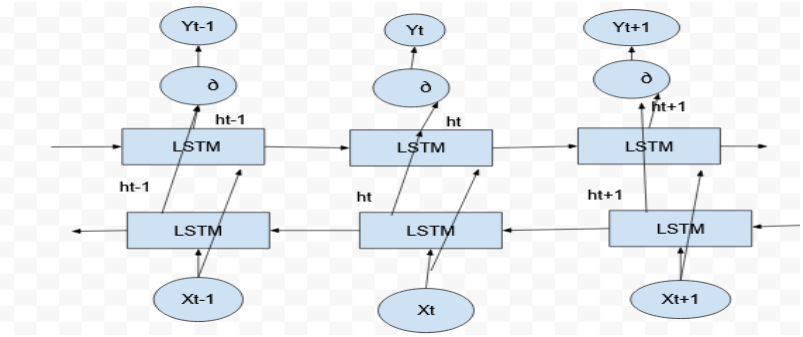


Figure 4-5 BiLSTM architecture

4.6.4. Gated Recurrent Unit Model

It is a type of recurrent neural network (RNN) architecture that has been applied to various natural language processing tasks, including constituency parsing. GRU is similar to LSTM but has a simpler architecture with fewer parameters. It consists of a set of gates that control the flow of information within the network, including an update gate and a reset gate. These gates enable GRU to selectively update its hidden state based on the input and the previous hidden state. For constituency parsing, GRU can be used as part of a sequence-to-sequence model, where the input sequence is a sentence and the output sequence is its syntactic parse tree. The GRU layers process the input sentence and encode its information into a fixed-size vector representation. This representation is then decoded by another set of GRU layers to generate the parse tree. GRU-based models for constituency parsing have been shown to achieve competitive performance compared to other approaches, offering a balance between computational efficiency and effectiveness in capturing long-range dependencies in sentences. However, the specific implementation details and performance may vary depending on the dataset and model architecture used[52]. In research findings indicate that among all single models tested, the Gated Recurrent Unit (GRU) consistently outperformed others in the context of the task at hand. However, the performance of the GRU model was further enhanced when it was incorporated into an ensemble model. Ensemble methods typically combine multiple models to produce superior results compared to any individual model, leveraging diverse perspectives and learning from different sources of information[52]. By integrating the strengths of the GRU model with other complementary models, the ensemble approach effectively boosted the overall performance, demonstrating the benefits of combining multiple models for improved results in the given task.

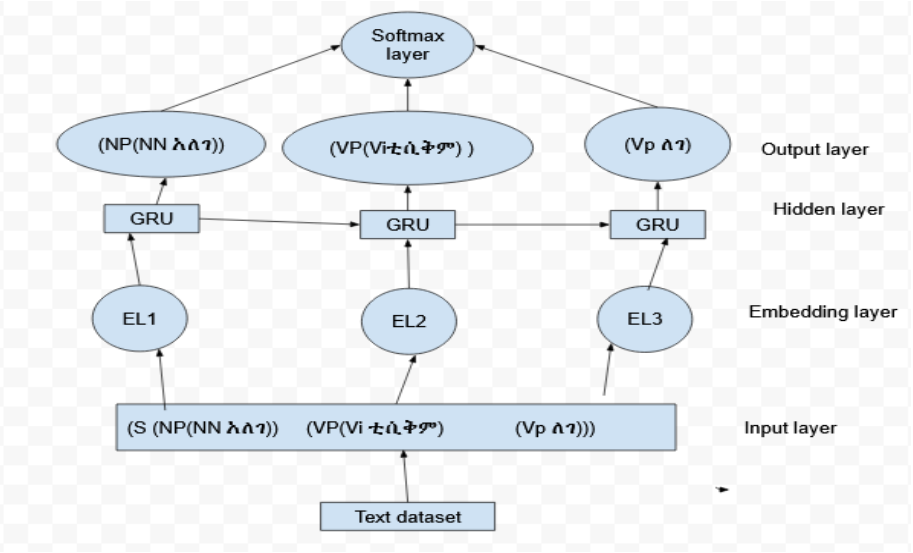


Figure 4-6 GRU architectures

4.6.5. Transformer Model

The Transformer model architecture is a type of deep learning architecture introduced in the paper [53]. It's primarily used in natural language processing (NLP) tasks and has become a foundational architecture for various NLP applications due to its effectiveness and scalability. We frequently build upon existing Transformer architectures or develop new variations to improve performance on specific tasks. For constituency parsing using the Transformer architecture, we typically adapt existing Transformer models, such as the original Transformer introduced by the researchers. or variations like BERT (Bidirectional Encoder Representations from Transformers) or GPT (Generative Pre-trained Transformer), to the task of constituency parsing. Here's a brief overview of the key components of the Transformer model architecture:

Self-Attention Mechanism: This mechanism allows the model to weigh the importance of different words in a sentence when encoding or deco ding. It computes attention scores between all pairs of words in a sequence, capturing dependencies regardless of their positions in the input sequence.

Encoder-Decoder Architecture: The Transformer model typically consists of an encoder and a decoder. The encoder processes the input sequence, while the decoder generates the output sequence. Each encoder and decoder layer in the Transformer contains multiple self-attention layers followed by feed-forward neural networks.

Positional Encoding: Since the Transformer model doesn't inherently possess information about the order of tokens in a sequence, positional encoding is added to the input embedding‟s to provide positional information to the model.

Feed-Forward Neural Networks: Both the encoder and the decoder have feed-forward neural networks, which are applied to each position separately and identically.

Layer Normalization and Residual Connections: Layer normalization is applied after each sub-layer, and residual connections are employed around each sub-layer to aid in training deeper networks. When using the Transformer architecture for constituency parsing, you would typically adapt it to predict constituency labels for each word or subsequence in a sentence, producing a parse tree representation. The model would be trained on a dataset of parse trees, learning to map input sentences to their corresponding parse trees. Overall, the Transformer's self-attention mechanism allows it to capture dependencies across the entire input sequence efficiently, making it well-suited for tasks like constituency parsing, which require understanding the hierarchical structure of language[54].

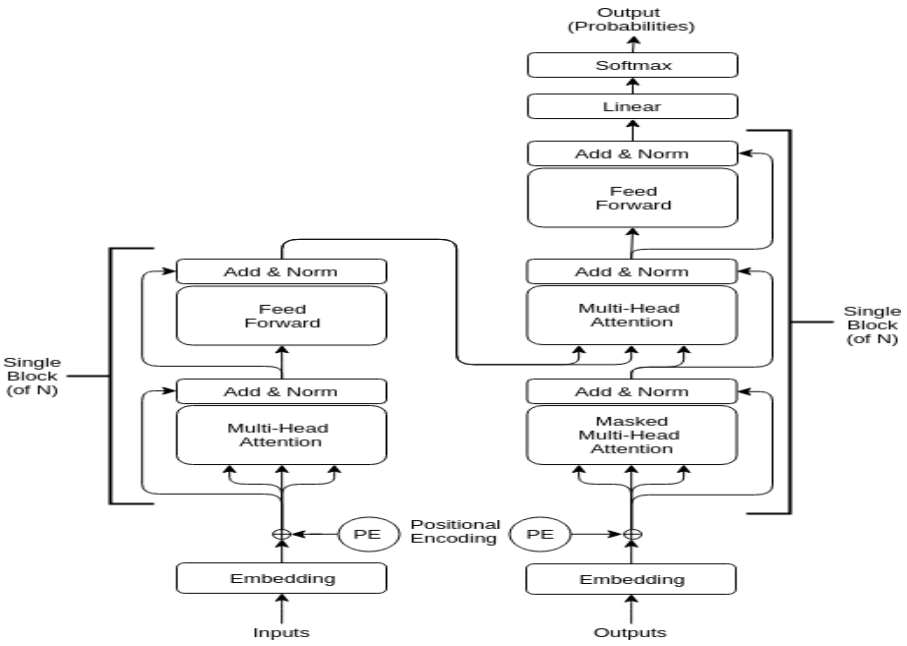


Figure 4-7 Transformer architecture[10]

CHAPTER FIVE

5. EXPERMENTATION AND RESULTS

5.1. Introduction

We begin by providing an overview of the architecture designed for the task, as outlined in the preceding chapter. This architecture serves as the framework for the experiments conducted in the thesis, establishing the baseline for subsequent discussions. Following the architecture overview, the section delves into the details of the corpus source utilized in the experiments. It includes information on the origin, size, composition, and any preprocessing steps applied to the dataset. This context is essential for understanding the data used in the study. Next, the experimental implementation is described, outlining the software tools, libraries, and hardware resources employed for the implementation of the experiments. This provides clarity on the technical infrastructure utilized to conduct the research. A significant aspect covered in this section is the preprocessing of the dataset. The text explains the preprocessing steps undertaken, such as tokenization, normalization, cleaning, and feature extraction. These steps are crucial for preparing the data for subsequent model training and evaluation. The section elaborates on each conducted model experiment, detailing the variations in architecture, hyper parameters, and training methodologies. This may involve comparing different neural network architectures, optimization algorithms, or regularization techniques to assess their impact on performance. Furthermore, the process of parameter selection is discussed, encompassing the methods used to determine optimal values for model parameters such as learning rates, batch sizes, and network sizes.

5.2. Data Collection and Preparation

We collected 2000 sentences and prepared corresponding parse tree. These sentences were gathered from various sources, including Siliti'e FM, the Siliti'e zone education office, the WRU Siltigna department, and textbooks spanning from grade 1 to grade 12. Specific details about the sources of the data used in the study are provided below a tabulated format. This breakdown allows for clear delineation of the contributions from each portion of the dataset. The inclusion of sentences from Siliti'e FM suggests that real-world language usage was considered, likely reflecting contemporary spoken Siliti'e. Additionally, data sourced from educational institutions and textbooks may offer insights into formal or standardized language usage, particularly relevant for linguistic analysis and educational applications.

Table 5-1 Data Collection and Preparation

Source of Data	**Expressed In Numbers**
Siliti'e FM	800
WRU siltigna department	300
Siliti'e zone educ.office	200
Text books	700
Total in numbers	2000

5.2.1. System Environment

In order to implement the proposed study's model, selecting a suitable programming language and setting up the necessary environment is essential. Python was chosen as the primary programming language due to its extensive support for freely available libraries. Python 3.11, along with Jupyter Notebook, was utilized for coding, with the Keras, TensorFlow, and NumPy libraries being key components. Google Colab was selected as the main integrated development environment (IDE) due to its provision of cloud resources, including GPUs with 24.5GB of RAM, which significantly expedite training times compared to CPUs. While desktop and laptop systems were utilized for corpus preparation, preprocessing, and report writing, Google Colab with GPU support served as the primary environment for implementing and training the proposed model. The experimental workflow involved storing the dataset as .txt files, uploading them to Google

Drive, mounting the drive in Google Colab, and then performing preprocessing steps within the Colab environment to prepare the data for training.

5.2.2. Data Cleaning

The initial step in data preprocessing involves cleaning the dataset, which is fundamental for preparing data for further processing. In this process we begins by loading the dataset using the open() function, with each line of the text files containing a sentence separated by spaces.

```
    # Remove any non-Silitigna characters
    text = re.sub(r'[^ʼhɯθʁɸʔ0\s]', '', text)
    # Remove extra whitespaces
    text = ' '.join(text.split())
    return text

def clean_silitigna_dataset(dataset_path, output_path):
    cleaned_lines = []
    with open(dataset_path, 'r', encoding='utf-8') as file:
        for line in file:
            cleaned_line = clean_silitigna_text(line)
            cleaned_lines.append(cleaned_line)
    with open(output_path, 'w', encoding='utf-8') as output_file:
        for cleaned_line in cleaned_lines:
            output_file.write(cleaned_line + '\n')
input_silitigna_dataset_path = '/content/drive/MyDrive/dataparser.txt'
output_cleaned_silitigna_dataset_path = '/content/drive/MyDrive/dataparser.txt'
clean_silitigna_dataset(input_silitigna_dataset_path, output_cleaned_silitigna_dataset_path)
```

Figure 5-1 Sample data cleaning screenshot code

5.2.3. Tokenization

In the tokenization process, plain text is transformed into a sequence of tokens. This involves two main operations carried out sequentially[55]. Firstly, words are mapped to integers using the Tokenizer class from the Keras library, which is essential for modeling purposes. Additionally, parse trees are tokenized. To train the tokenizer on a list of phrases, a function named create_tokenizer() is utilized, leveraging the built-in function tf.keras.preprocessing.text with a whitespace delimiter. The Tokenizer(filters=' ') library, derived from the Tokenizer class, facilitates this tokenization process.

```
def tokenize_silitigna_sentences_from_file(file_path):
    with open(file_path, 'r', encoding='utf-8') as file:
        sentences = [line.strip() for line in file]

    tokenizer = Tokenizer(filters='')
    tokenizer.fit_on_texts(sentences)
    tokenized_sentences = tokenizer.texts_to_sequences(sentences)
    return tokenized_sentences, tokenizer
file_path = '/content/drive/MyDrive/dataparser.txt'
tokenized_sentences, tokenizer = tokenize_silitigna_sentences_from_file(file_path)

print("Tokenized Sentences:")
for sentence, tokenized_sentence in zip(sentences, tokenized_sentences):
    print("Sentence:", sentence)
    print("Tokenized:", tokenized_sentence)
```

Figure 5-2 Sample data tokenization screen shot code

```
Sample 1 tokens: ['(S', '(ADJP(ADJ', '1ወ)(VIm', 'ጠወረ))(NP(NN', 'መዬ)(Vp', 'እሰቻን)))']
Sample 2 tokens: ['(S', '(NP(NNp', 'ለበክት)(Vp', 'ሁርም)(Ng', 'ኤለይ)))']
Sample 3 tokens: ['(S', '(NP(NNp', 'ለበክት)(ADV', 'ኡዛረ)(Vp', 'እላሱኒ)))']
Sample 4 tokens: ['(S', '(NP(NNp', 'ለኩጦ)(NN', 'መዪም)(Vp', 'ድባያንከ)))']
Sample 5 tokens: ['(S', '(NP(NN', 'አፈ)(ADJ', 'ነከ))(NP(NN', 'ደለ)(Vp', 'አልጄሳ)))']
```

Figure 5-3 Sample example output tokens from the model

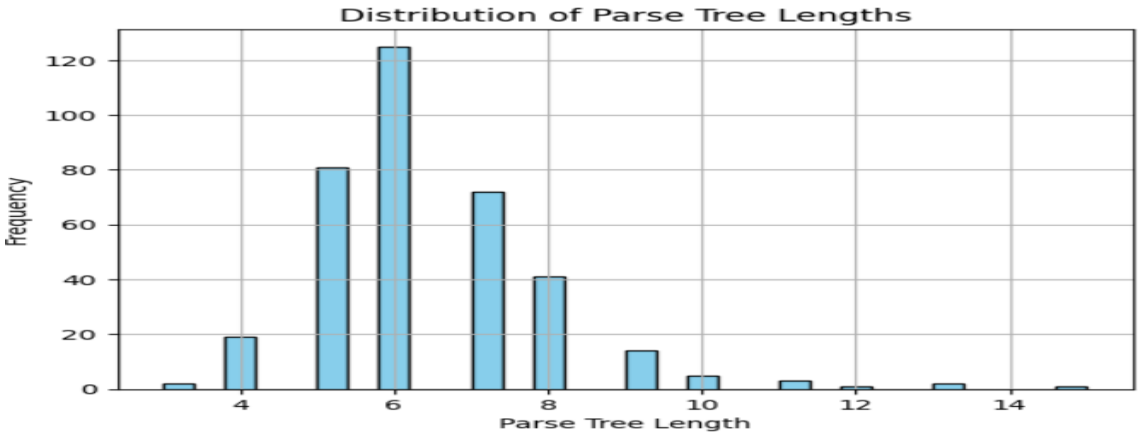


Figure 5-4 Frequency distribution of parse tree length

5.2.4. Parameter Selection

This section outlines the parameters used in the experimental study to train and test the selected model. Various experiments were conducted on these parameters using the training and testing data. The process began by selecting the embedding size dimensions, with options of 512 and 256 for latent dimensions and 128 and 256 for embedding dimensions. The choice of dimensionality depends on the selected model's requirements. A batch size of 64 and 32 was chosen, indicating the number of samples examined before updating the internal model parameters. A learning rate of 0.01, and 0.001 was selected to minimize training loss. Additionally, our experiments were conducted with epochs multiplied by 30, representing the number of iterations to learn the model across the entire dataset, where each epoch represents a single pass through the dataset.

Table 5-2 All model parameters

No	**Used hyper Parameters**	**All Train and Test Deep Learning Model Types**				
		LSTM	LSTM attention	Bi-LSTM	GRU	Transformer
1	Embedding Dimension	128	256	128	256	128
2	Latent Dimension(units)	256	256	256	256	256
3	Learning rate	0.01	0.01	0.01	0.01	0.001
4	Batch size	64	32	64	64	32
5	Loss function	Categorical cross entropy	Categorical cross entropy	Categorical cross entropy	Categorical cross entropy	Categorical cross entropy
6	Optimization	Adam	Adam	Adam	Adam	Adam
7	Num_ Head	No	No	No	No	8
8	Epoch	30	30	30	30	30
9	Dropout rate	0.2	0.1	0.2	0.2	0.2

5.3. RESULT AND DISCUSSION

We analyzed by LSTM,LSTM with attention mechanism, Bi-LSTM, GRU, and the Transformer model that are deep learning algorithms used for sequence modeling in NLP. LSTM and GRU address the vanishing gradient problem and capture long-term dependencies. Bi-LSTM processes sequences in both directions for a more comprehensive understanding. The Transformer model employs self-attention mechanisms to capture global dependencies, achieving state-of-the-art results. When analyzing datasets, consider data characteristics and task requirements. LSTM, Bi-LSTM, and GRU excel in long-term dependencies, while the Transformer captures contextual information effectively. Experimentation and evaluation are crucial to determine the best algorithm for specific needs.

5.3.1. Experimental Result

Experiment 1 LSTM models

For this model we built, several hyper parameters we used for training and optimization. The batch size was set to 64, which determines the number of samples processed in each training iteration. The embedding dimension was set to 128, representing the size of the vector representation for each word or token in the input sequence. The latent dimension (or hidden state size) was set to 256, which determines the number of units in the LSTM layer. The learning rate was set to 0.01, controlling the step size during gradient descent optimization. The chosen loss function was categorical cross entropy, which is commonly used for multi-class classification tasks. The model was trained for 30 epochs, indicating the number of times the entire dataset was passed through during training. A dropout rate of 0.2 was applied, which helps regularize the model and prevent over fitting. The accuracy was 81%, and the loss was 2.280, representing the model's performance on the training data and the evaluated labeled attachment score was again 0.5.

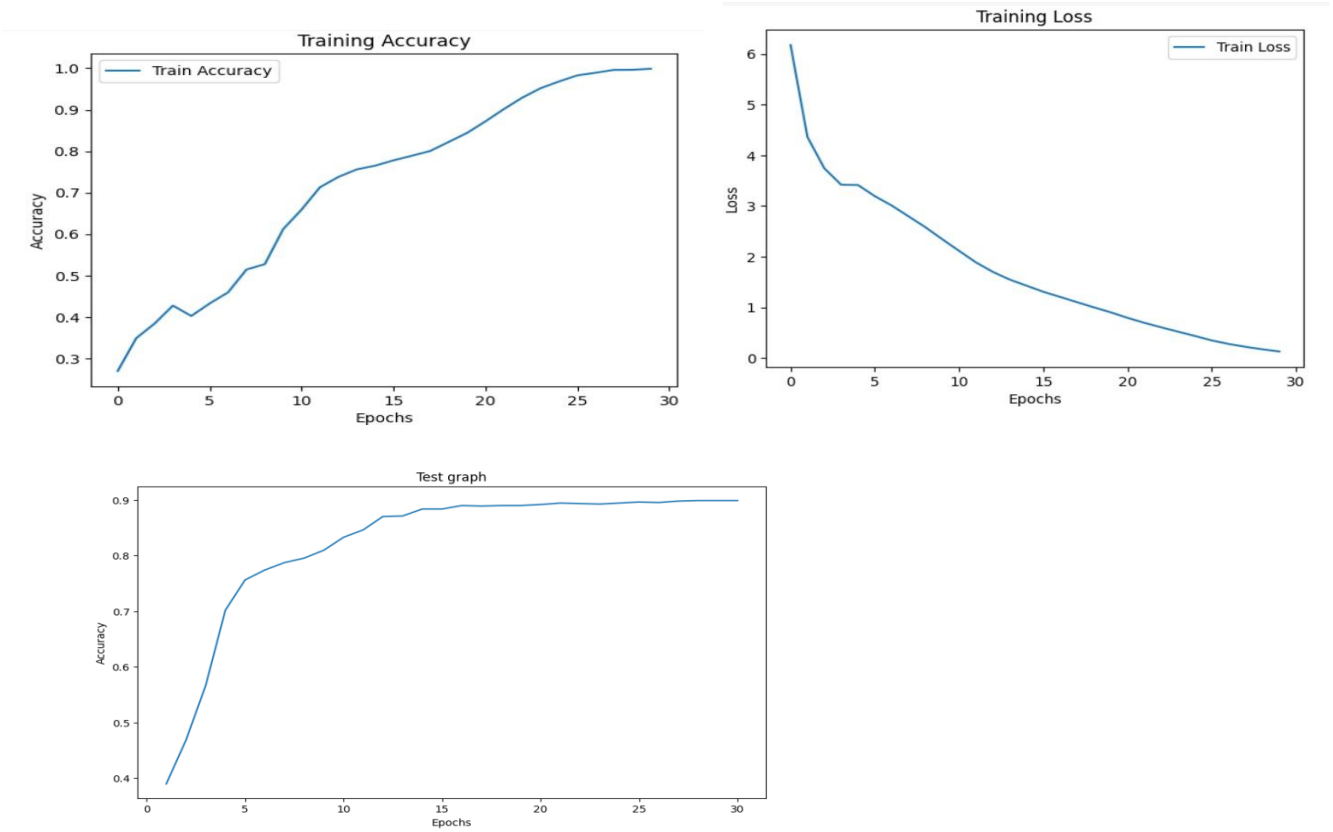


Figure 5-5 LSTM Train and test graph

Experiment 2 LSTM with attention models

We conducted on the LSTM with attention model was constructed with careful consideration of its hyper parameters. A batch size of 32 was chosen, indicating that 32 samples were processed in each training iteration. This choice balances computational efficiency and model update frequency during training. The embedding dimension, set to 256, defines the size of the vector representation for each word or token in the input sequence. This dimensionality is critical for capturing semantic information effectively. The latent dimension, or hidden state size, of the LSTM with attention layer was configured to 256 units, determining the complexity and capacity of the model to learn intricate patterns within the data. A learning rate of 0.01 was selected to control the step size during gradient descent optimization, influencing the rate at which the model parameters are updated. For the loss function, categorical cross-entropy was employed, well-suited for multi-class classification tasks, measuring the discrepancy between predicted and true label distributions. Training was conducted over 30 epochs, allowing the model to iteratively learn from the dataset, with each epoch representing a pass through the entire dataset. To prevent over fitting, a dropout rate of 0.1 was applied, randomly deactivating a fraction of input units during training to promote generalization. The reported metrics indicate a loss of 3.18 and an accuracy of 80.04%and LAS result was 0.4, offering insights into the model's performance on the training or validation data and

highlighting areas for potential improvement through further experimentation and fine-tuning of hyper parameters.

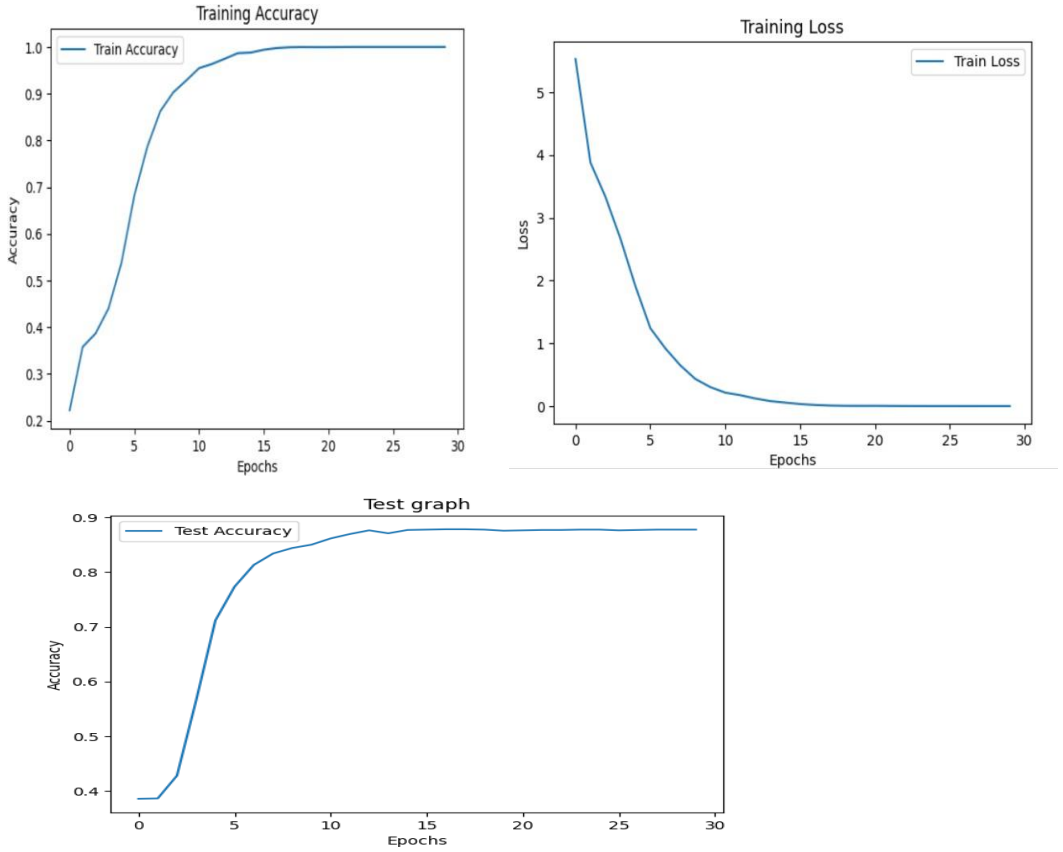


Figure 5-6 LSTM with attention train and test graph

Experiment 3 BiLSTM models

In the BiLSTM model we built, the analysis was conducted on hyper parameters for model construction and training. The batch size was set to 64, determining the number of samples processed in each training iteration. The embedding dimension was set to 128, representing the size of the vector representation for each word or token in the input sequence. The latent dimension (or hidden state size) was set to 256, determining the number of units in the BiLSTM layer. The learning rate was set to 0.01, controlling the step size during gradient descent optimization. The chosen loss function was categorical cross entropy, which is commonly used for multi-class classification tasks. The model was trained for 30 epochs, indicating the number of times the entire dataset was passed through during training. A dropout rate of 0.2 was applied to regularize the model and mitigate over fitting. The model had 256 units in total. The reported loss was 2.88, and the accuracy was 80.9% and LAS result was 0.5.

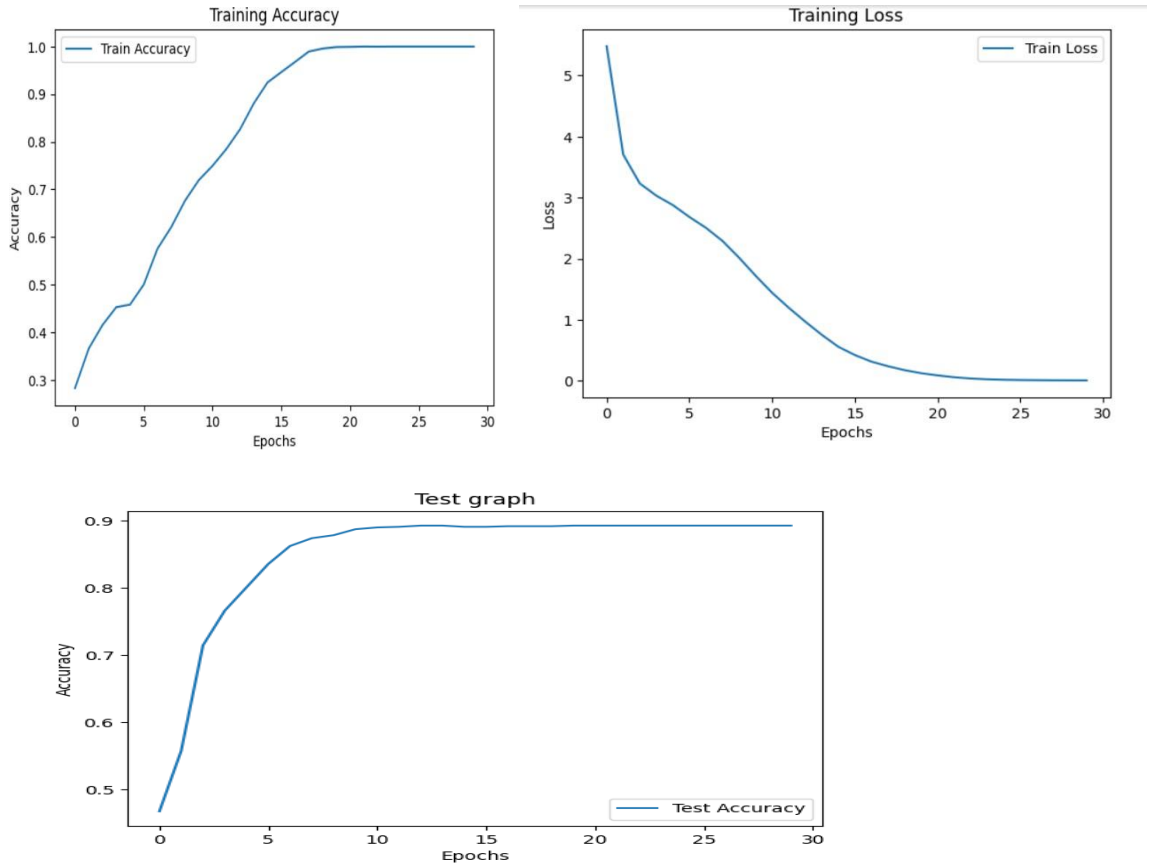


Figure 5-7 BiLSTM Train and test graph

Experiment 4 GRU models

In our GRU model, we carefully considered the hyper parameters and architecture. The embedding dimension was set to 256, allowing the model to effectively capture the semantic meaning of words. We chose a latent dimension of 256, which represents the hidden state size of the GRU cells, enabling the model to learn complex patterns in the data. To optimize the model, we utilized a learning rate of 0.01, determining the step size for parameter updates during training. A batch size of 32 was selected, balancing computational efficiency with the quality of updates. Categorical cross-entropy served as the loss function, measuring the dissimilarity between predicted class probabilities and true labels. Training the model for 30 epochs involved iterating over the entire dataset multiple times, allowing the model to learn and adjust its parameters. Incorporating a dropout rate of 0.2 helped prevent over fitting by randomly deactivating a fraction of input units during training. Our efforts resulted in loss is 2.04 accuracy of 84.3%, indicating the model's ability to correctly predict the training data. However, it's important to evaluate the model's generalization performance on a separate test dataset to assess its effectiveness in real-world scenarios and LAS result was 0.6.

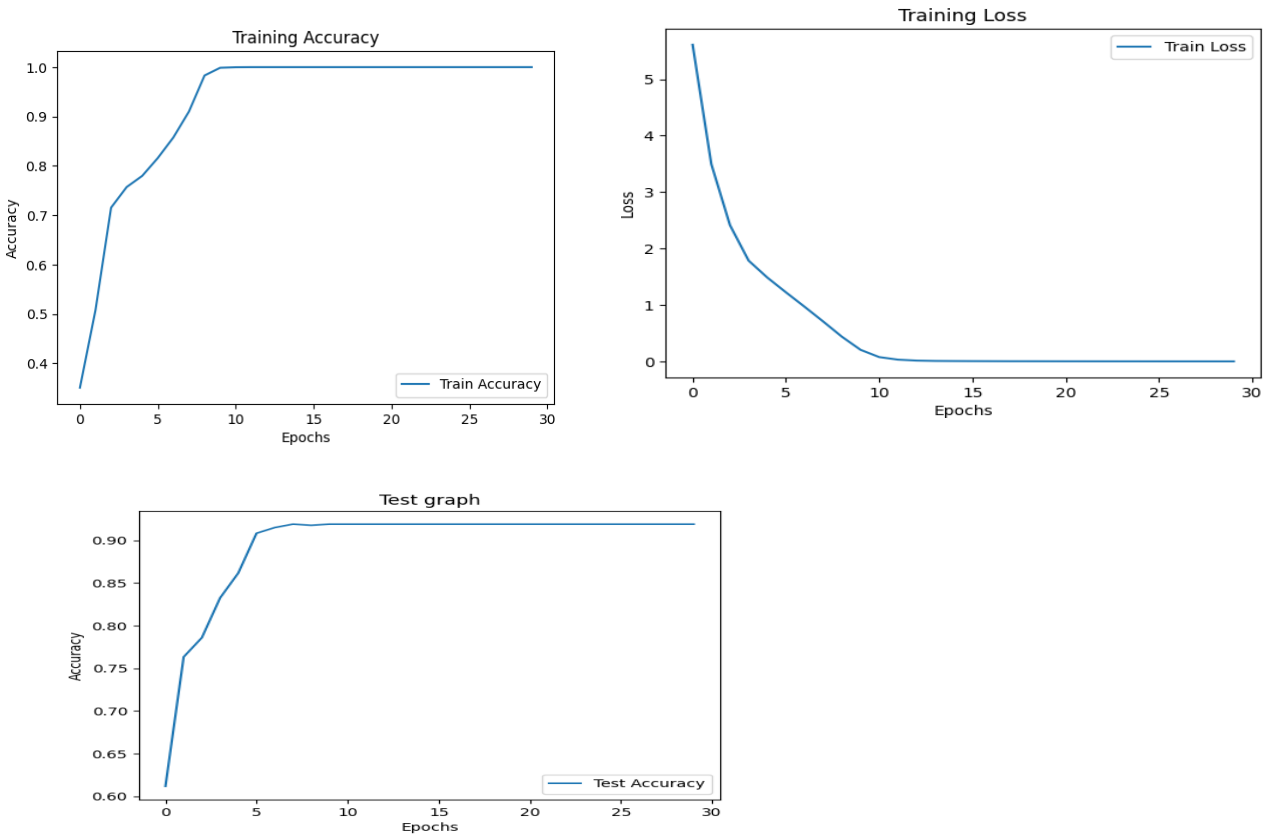


Figure 5-8 GRU Train and test graph

Experiment 5 Transformer models

When we build our transformer model, we conducted thorough analysis of various factors such as batch size, embedding dimension, epoch, loss function, learning rate, and dropout hyper parameter. This allowed us to make informed decisions and construct an effective model. In the transformer architecture, we utilized the ReLU activation function in the feed forward layer. This function helps activate the neurons, introducing non-linearity and enhancing the model's ability to capture complex patterns in the data. For the specific model configuration, we chose an embedding dimension of 128, which determines the size of the input word embedding. Additionally, we set the latent dimension to 256, indicating the hidden state size of the transformer model. These choices were made to strike a balance between model complexity and computational efficiency. To optimize the model's training, we set the learning rate to 0.001, controlling the step size for parameter updates during the optimization process. The batch size was set to 32, considering the trade-off between computational resources and the quality of parameter updates. Categorical cross-entropy was selected as the loss function, which is commonly used for multi-class classification tasks. This loss function quantifies the dissimilarity between the predicted class probabilities and the true class labels. Training the model for 30 epochs allowed it to learn and adjust its parameters over multiple passes through the dataset. This iterative process enables the model to capture underlying patterns and

improve its performance. To prevent over fitting, we incorporated a dropout rate of 0.2. This technique randomly deactivates a fraction of input units during training, encouraging the model to rely on multiple features and reducing its dependence on specific ones. In our training process, the transformer model achieved an accuracy of 84.38% and loss of 1.37. However, it is important to evaluate the model's performance on a separate test dataset to assess its generalization capabilities and their labeled attachment score was 0.83.

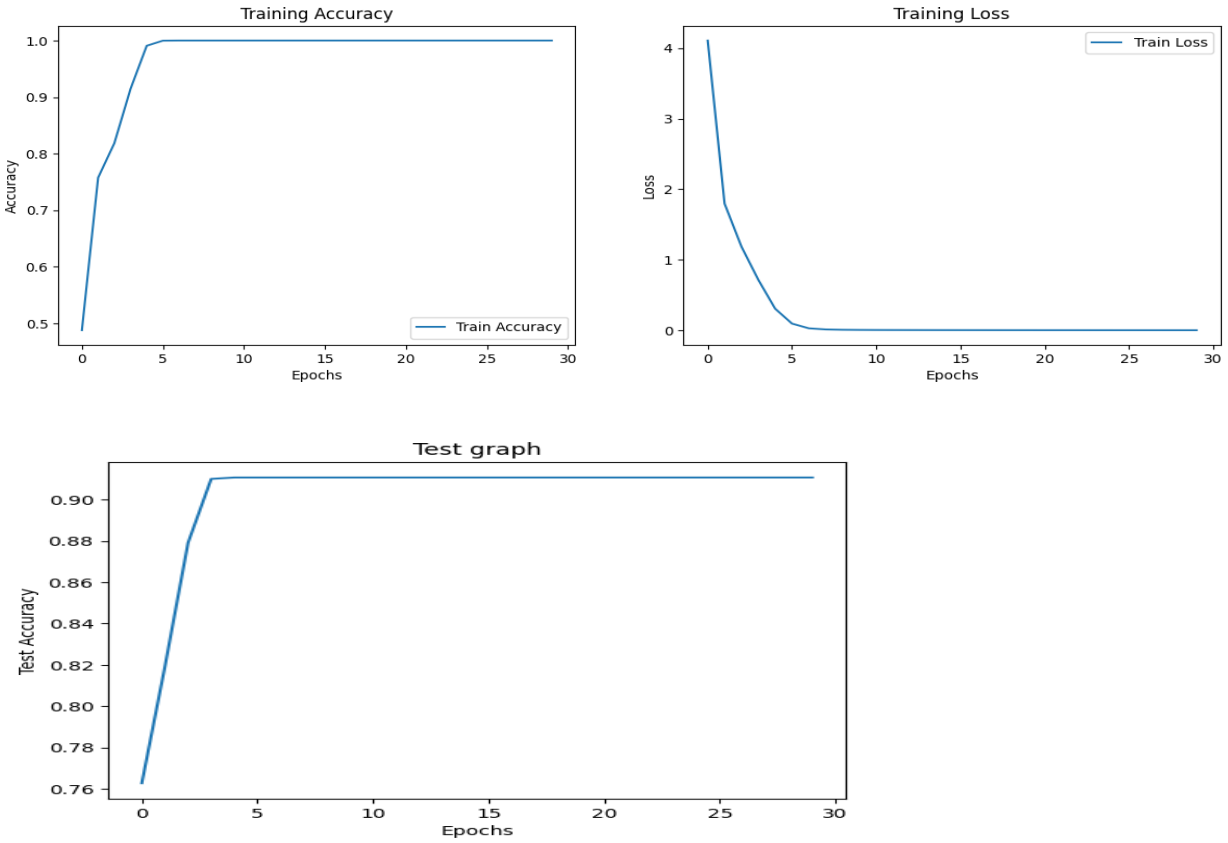


Figure 5-9 Transformer Train and test graph

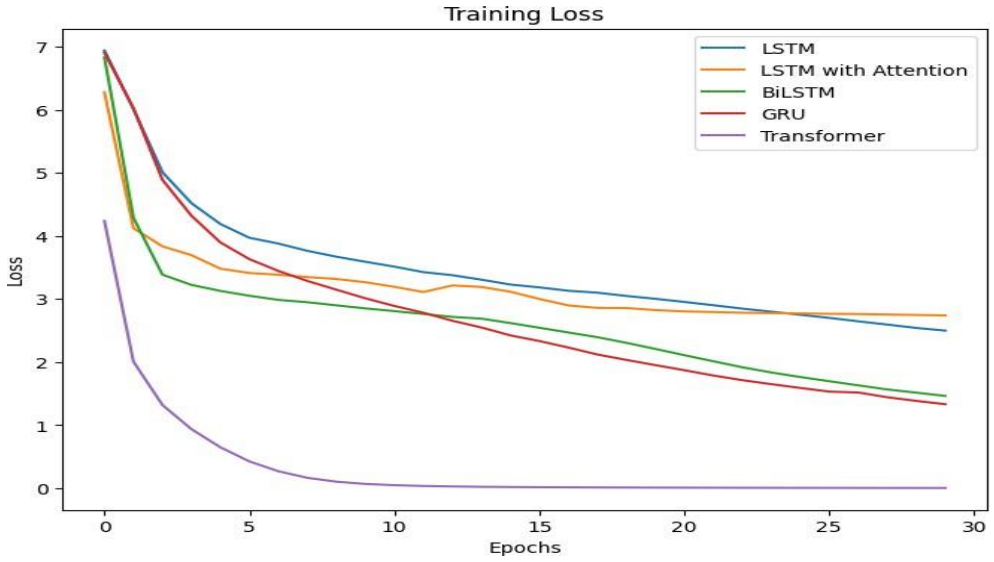


Figure 5-10 Evaluation for models in LAS in train loss

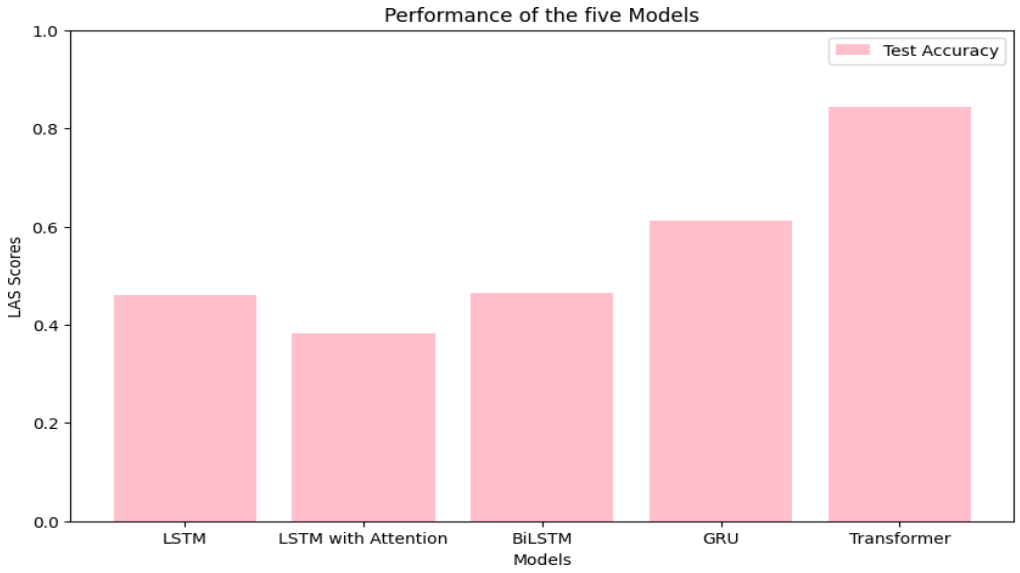


Figure 5-11 Models performance evaluations LAS

5.3.2. Discussions of the results

In our study, we performed various preprocessing steps and built deep learning models with different architectures, including LSTM, LSTM with attention, BILSTM, GRU, and transformers. Each model was evaluated separately, and we obtained different experimental results for each. The general architecture we used involves a sequential auto-encoder that reads the input sentences and converts them into a single real-valued vector. This vector is then passed to the decoder, which generates the predicted parse trees. We found that as the length of a sentence increases, the interdependency between words at the start and end of the sentence becomes less strong. However, the architecture performed well for both simple and complex sentences. To handle the interdependency of words within a sentence, we used an attention mechanism in combination with the Auto-encoder-Decoder architecture. This helped address the issue of identifying words by specific indices as they are encountered in the data. Then each consecutive model increases the performance of constituency parsing tasks. Based on our experimental findings, we concluded that the Transformer model outperformed LSTM, LSTM attention, BILSTM, and GRU models. However, due to the limited amount of data used in the study, it was challenging to develop a parsing model that was highly effective. Despite this limitation, we achieved LAS (Labeled Attachment Score) of 0.83 by adjusting different hyper parameters with the limited dataset and we observed that deep learning has become a successful technique for parsing due to increased processing capacity. During analyzing Auto-encoder-Decoder architecture allowed for accurate parsing of sentences. Overall, the experiments led to improved LAS for predicting parsing results.

Table 5-3 Discussion of experimental results by LAS

Experiments	**Evaluated Models**	**Analyzing by LAS results**
Experiment 1	LSTM	0.5
Experiment 2	LSTM with Attention	0.4
Experiment 3	BiLSTM	0.5
Experiment 4	GRU	0.64
Experiment 5	Transformers	0.83

5.3.3. Summary

From the above experimental results reveal that the transformer model outperforms other architectures such as LSTM, LSTM with attention mechanism, BiLSTM, and GRU in terms of performance on both the training and test datasets. The transformer model achieved a labeled attachment score of 0.83, indicating its superior ability to capture dependency relations. Additionally, the total loss during training was reported as 1.37, suggesting effective learning and optimization during the training process. Furthermore, the model demonstrated an impressive accuracy of 84.38%, affirming its robust performance across the dataset. Overall, the transformer model emerges as the top performer among the evaluated architectures, showcasing its effectiveness in handling the given task.

5.3.4. Answering Research Questions

In our work, we gather data from various parts of the Siliti'e Zone and WRU Siltigna Departments, encompassing text data in both simple and complex Siltigna languages.This data is in raw form initially and undergoes a thorough cleaning and preprocessing phase utilizing algorithms tailored for analysis and implementation cases. . During data collection and annotation phase we guided by two language experts the first expert name Kedir Awel(B.Sc. In siltigna language and literature) he guided by annotate pos tags and the second expert was name Tewfic Dilebo(M.Sc. in literature and he is head department of siltigna language and literature at WRU) he guided by phrase tags of the language. During this process, we label the data with part-of-speech tags and construct parse trees using the expertise of the language to ensure clarity and eliminate impurities from the raw text our additional resources include textbooks, Siliti'e FM broadcasts, and materials from the Siliti'e Zone Education Office in both soft and hard copy formats. All data undergoes scrutiny and approval by language experts who possess a deep understanding of Siltigna grammar. Their involvement ensures the accuracy and integrity of the labeled data. Overall, our methodology involves a meticulous approach to collecting, cleaning, preprocessing, and labeling text data in Siltigna languages, with the ultimate goal of creating high-quality resources for language understanding and analysis in various domains.

RQ1. Which Vectorization technique was used for identifying siltigna language?

In our study, we utilized various techniques for language understanding and parsing tasks, with a focus on deep learning models and word embedding methods. Word embedding, particularly techniques like one-hot vector encoding and Word2Vec, played a central role in our approach. These methods transform words into dense vector representations, capturing rich semantic and syntactic relationships within a high-dimensional space. Our experiments involved implementing constituency parsing using LSTM, LSTM with attention mechanisms, BILSTM, GRU, and transformer models. These deep learning architectures leverage word embedding to enhance language understanding and enable

more accurate parsing and labeling tasks. By employing deep learning techniques and word embedding, we were able to revolutionize language analysis in our domain, achieving remarkable accuracy in constituency parsing and labeling tasks, particularly through seq2seq systems. We organized our methodologies and architectures into architectural diagrams, as outlined in Chapter 4 of our study. Ultimately, our experiments in Chapter 5 demonstrated the effectiveness of Word2Vec techniques, particularly in sequentially processing text data, highlighting its superiority in our parsing and labeling tasks.

RQ2. Which deep learning algorithm was best for siltigna language parsing?

In Chapter 5 of our experimental study, we conducted analyses on our dataset using five different deep learning model algorithms: LSTM, LSTM with attention, BILSTM, GRU, and transformer models. Our evaluation primarily focused on assessing the performance of these models in constituency parsing tasks, utilizing the labeled attachment score evaluation metric. Among these algorithms, the transformer model, particularly when leveraging pre-trained language models like BERT, demonstrated superior understanding and accuracy in handling our dataset. The transformer model outperformed other algorithms in terms of its ability to comprehend sequential structures and support transition-based methods. The obtained results suggest that transformer models, especially when incorporating pre trained language models like BERT, offer state-of-the-art performance in constituency parsing tasks, making them potentially valuable for current language understanding applications.

CHAPTER SIX

6. CONCLUSION AND RECOMMENDATION

6.1. Conclusion

In conclusion, this research focused on the development of constituency parsing models for Siltigna languages using deep learning techniques. We utilized various deep neural network algorithms, including LSTM, LSTM with attention, Bi-LSTM, GRU, and transformer models, to parse sentences with corresponding parse trees. Our study was conducted on a dataset comprising 2000 parse tree datasets prepared specifically for this purpose. Through experimentation and analysis, we found that the transformer model outperformed other models in terms of parsing accuracy and efficiency. Specifically, the transformer model achieved a significant improvement in labeled attachment scores, increasing parsing quality by 0.23 compared to other models. Furthermore, the transformer model demonstrated superior time efficiency in constituency parsing for Siltigna sentences. Overall, our study highlights the effectiveness of deep learning models, particularly the transformer model, in constituency parsing tasks for Siltigna languages. These findings contribute to advancements in natural language processing and pave the way for further research in language understanding and parsing in similar linguistic contexts.

6.2. Recommendation

We recommended that for analyzing advanced constituency parsing for the Siltigna language has encountered several challenges. One major obstacle has been the difficulty in accessing linguistic experts and obtaining accurate data sources. This issue is exacerbated by the limited availability of online resources and linguistically annotated datasets (Treebank) specifically tailored for Siltigna. As a result, significant time and effort have been invested in collecting and preparing experimental data for analysis. To address these challenges and progress in constituency parsing for Siltigna, there is a critical need to collaborate with linguistic experts. Their expertise is indispensable in creating a corpus that is linguistically annotated and tailored to the unique characteristics of the Siltigna language. However, the process of engaging these experts and developing such a corpus requires time and resources. The delays in the advancement of constituency parsing for Siltigna can also be attributed to the low quality of the available training data. This data may suffer from inaccuracies and inconsistencies, making it challenging to train robust parsing models effectively. Furthermore, the lexically and syntactically ambiguous nature of the Siltigna language adds another layer of complexity to the parsing task. Another

significant challenge is the absence of predefined models and algorithms for implementing Siltigna in terms of lemma and morphology. This lack of existing frameworks means that efforts must be directed towards developing custom models and algorithms tailored specifically to the linguistic characteristics of Siltigna. This process involves collaboration between computational linguists and domain experts familiar with the intricacies of the language.

6.3. Contribution

Our research contribution addresses the challenges inherent in conducting research in Siltigna language, particularly in the realm of parsing, whether through constituency or dependency methods. We have tackled these obstacles by developing sentences along with their corresponding parse trees, utilizing tentative part-of-speech (POS) tags and phrase tags. This process was facilitated through collaboration with domain experts, enabling us to overcome the linguistic complexities specific to Siltigna. Our contribution lies in the comprehensive assessment of all Siltigna language sentences, utilizing the language's word classes, phrase categories, and grammatical structures. Through rigorous analysis and parsing approaches, we have systematically evaluated and parsed Siltigna sentences, thereby advancing our understanding of the language's linguistic features and facilitating the development of parsing models tailored to Siltigna. By addressing these challenges and conducting thorough linguistic assessments, our research significantly contributes to the advancement of parsing techniques for the Silitigna language. It provides valuable insights into the language's syntactic structures and lays the groundwork for future research in natural language processing and computational linguistics within the Siltigna linguistic domain.

6.4. Future Work

From the results of the study, we recommend that following future research directions: - from our efforts for the thesis we conducted it was the first study in the Siltigna language domain. We have developed sentences with corresponding parse trees by using tentative pos tags and phrase tags for using our work done from these we developed 2000 sentences with corresponding parse trees datasets.

• For future work we recommend that interested researcher in Siltigna language use the prepared data set can be used for other NLP applications tasks like spell checking, questions answering and grammar checker.

• As the goal of our study is to implement constituency parsing only using text parsing, we suggest that additional studies conduct on document parsing and semantic parsing.

• Finally, we suggest that other researchers extend our work by adding dataset and use manually annotated dataset (Treebank).

7. REFERENCES

[1] B. Achamyeleh, B. Hailu, and G. Mamo, "Automatic Sentence Parser For Amharic Language Using SVM," vol. 11, no. 2, pp. 72–75, 2021, doi: 10.46360/cosmos.et.620212013.

[2] S. Jaf and C. Calder, "Deep Learning for Natural Language Parsing," *IEEE Access*, vol. 7, pp. 131363–131373, 2019, doi: 10.1109/ACCESS.2019.2939687.

[3] Y. Tian, Y. Song, F. Xia, and T. Zhang, "Improving Constituency Parsing with Span Attention," Researchgate, Association for Computational Linguistics, January, 2020.

[4] Z. Chen, L. Zhang, A. Imankulova, and M. Komachi, "Neural Combinatory Constituency Parsing," *Find. Assoc. Comput. Linguist. ACL-IJCNLP 2021*, pp. 2199–2213, 2021, doi: 10.18653/v1/2021.findings-acl.194.

[5] L. Liu, M. Zhu, and S. Shi, "Improving sequence-to-sequence constituency parsing," *32nd AAAI Conf. Artif. Intell. AAAI 2018*, pp. 4873–4880, 2018, doi: 10.1609/aaai.v32i1.11917.

[6] Central Statistical Agency of Ethiopia, "Population and Housing Census 2007 Report," *Cent. Stat. Agency, Addis Ababa*, p. 385, 2007, [Online]. Available: http://www.csa.gov.et/surveys/Population and Housing census/ETH-pop-2007/survey0/data/Doc/Reports/National_Statistical.pdf

[7] A. Johar, "Text information retrieval system for silti'e language," ASTU,M.Sc. thesis ,September, 2017.

[8] H.Gereziher,"Design and Development of Automatic Unlexicalized Statistical Constituency Parser for Tigrigna,"ASTU, M.Sc thesis, 2017.

[9] Kandhasamy Rajasekaran, "Deep Learning techniques applied to Constituency parsing of German," Koblenz Landau ,Msc.thesis,January,2020.

[10] J. Park, "A note on constituent parsing for Korean," *Nat. Lang. Eng.*, vol. 28, no. 2, pp. 199–222, 2022, doi: 10.1017/S1351324920000479.

[11] D. Klein and C. D. Manning, "A Generative Constituent-Context Model for Improved Grammar Induction," Stanford University, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July, 2002.

[12] M. Collins,"Head-Driven Statistical Models for Natural Language Parsing." Association for Computational Linguistics, Volume 29, Number 4 , 2003.

[13] M. C. Kenton, L. Kristina, and J. Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Association for Computational Linguistics, no. Mlm, June, 2019.

[14] C. M. Greco and A. Tagarelli, *language models for artificial intelligence and law*, no. 0123456789. Springer Netherlands, 2023. doi: 10.1007/s10506-023-09374-7.

[15] D. W. Otter, J. R. Medina, and J. K. Kalita, "Natural Language Processing," IEEE Transactions On Neural Networks And Learning Systems, Vol. Xx, No. X, July 2019.

[16] T. Watanabe and E. Sumita, "Transition-based Neural Constituent Parsing," Association for Computational Linguistics,pp. 1169–1179, 2015.

[17] Yanshet.M, "Automatic Parser for Tigrigna Sentences Using Bottom-Up Probabilistic Chart Parser," AAU,Msc.thesis,2017.

[18] Kitaev, NikitaCao, Steven Klein, Dan., "Multilingual Constituency Parsing with Self-Attention and Pre-Training," Researchgate,Vol.9, June, 2017.

[19] L. M. da Costa, "Bottom-up parsing approach to modelling local coherence effects," Retrieved from https://escholarship.org/uc/item/4v3265ds.,June, 2021.

[20] J.Zuber, "Silti'e Text Information Retrieval System Based On Vector Space Model," ASTU,Msc.thesis,September, 2016.

[21] M. Seyfedin, "Desing and develop Automatic Speech Tagger for Silitigna Language," WSU, M.Sc. Thesis ,Septemper ,2023.

[22] H.Mohammed,"የአማርኛ - ስልጥኛ መዝገበ ቃላት የአማርኛ - ስልጥኛ ቃሙስ" Siltigna Grammar book,2007.

[23] M. Guwta, "Sili'e Text Information Retrieval System Using Vector Space Model," BDU,Msc.thesis,2021.

[24] Muzeyn Kedir, "Designinig A Stemming Algorithm For Silt " E Texts,"

SEMANTIC SCHOLAR,June,2012.

[25] Abdurehim Dawud, "A Top-Down Chart Parser for Amharic Sentences," AAU, Msc.thesis, April, 2015.

[26] Y. Sakakibara, "Probabilistic Context-Free Grammars," *Encycl. Mach. Learn.*, pp. 802–805, 2011, doi: 10.1007/978-0-387-30164-8_669.

[27] J. C. O. P. Y and A. R. Stevens, "eLECTE Di,"Book, PP.88-89,March, 1988.

[28] T. G. Ayana, “Afaan Oromo Parser using Hybrid Approach,” JU repository,November, 2017.

[29] M. Z. Degu and W. B. Gebeyehu, “Development of dependency parser for Amharic sentences,”JU repository, 2022.

[30] R. Maalej, N. Khoufi, and C. Aloulou, “Parsing Arabic using deep learning technology,” Tunisian Algerian Conference on Applied Computing ,no. Tacc, December,2021.

[31] E. M. Sibarani, M. Nadial, E. Panggabean, and S. Meryana, “A study of parsing process on natural language processing in bahasa Indonesia,” *Proc. - 16th IEEE Int. Conf. Comput. Sci. Eng. CSE 2013*,December 2013, pp. 309–316, doi: 10.1109/CSE.2013.56.

[32] R. R. Deshmukh, “Data Cleaning: Current Approaches and Issues,” Semantic Scholar,Vol.3,June, 2015.

[33] R. Friedman, “Tokenization in the Theory of Knowledge,” ResearchGate,Vol.11,pp. 380–386, 2023.

[34] P. Qi, T. Dozat, Y. Zhang, and C. D. Manning, “Universal dependency parsing from scratch,” *CoNLL 2018 - SIGNLL Conf. Comput. Nat. Lang. Learn. Proc. CoNLL 2018 Shar. Task Multiling. Parsing from Raw Text to Univers. Depend.*, pp. 160–170, 2018, doi: 10.18653/v1/K18-2016.

[35] J. I. Samuels, “One-Hot Encoding and Two-Hot Encoding : An Introduction,”Semantic Scholar, January, 2024, doi: 10.13140/RG.2.2.21459.76327.

[36] D. Suresh, A. Naresh, K. Nagwani, and P. Singh, *Impact of word embedding models on text analytics in deep learning environment : a review*, vol. 56, no. 9. Springer Netherlands, 2023. doi: 10.1007/s10462-023-10419-1.

[37] M. R. Costa-Jussà, Á. Nuez, and C. Segura, “Experimental research on encoder-decoder architectures with attention for chatbots,” *Comput. y Sist.*, vol. 22, no. 4, pp. 1233–1239, 2018, doi: 10.13053/CyS-22-4-3060.

[38] D. Vilares, M. Strzyz, A. Søgaard, and C. Gómez-Rodríguez, “Parsing as pretraining,”

AAAI 2020 - 34th AAAI Conf. Artif. Intell., no. ii, pp. 9114–9121, 2020, doi: 10.1609/aaai.v34i05.6446.

[39] Y. Tian, Y. Song, F. Xia, and T. Zhang, "Improving constituency parsing with span attention," *Find. Assoc. Comput. Linguist. Find. ACL EMNLP 2020*, pp. 1691–1703, 2020, doi: 10.18653/v1/2020.findings-emnlp.153.

[40] T. Bladier, "RRGparbank : A Parallel Role and Reference Grammar Treebank," IEEE,Vol.11,no. June, pp. 4833–4841, 2022.

[41] X. Gu, Y. Shen, J. Shen, J. Shang, and J. Han, "Phrase-aware Unsupervised Constituency Parsing," *Proc. Annu. Meet. Assoc. Comput. Linguist.*, vol. 1, no. Mlm, pp. 6406–6415, 2022, doi: 10.18653/v1/2022.acl-long.444.

[42] P. Barham, "TensorFlow : A system for large-scale machine learning,"Google scholar,Vol.9, June 2021.

[43] N. Kitaev and D. Klein, "Constituency parsing with a self-attentive encoder," *ACL 2018 - 56th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf. (Long Pap.*, vol. 1, pp. 2676–2686, 2018, doi: 10.18653/v1/p18-1249.

[44] Z. Zeng and D. Xiong, "Unsupervised and Few-Shot Parsing from Pretrained Language Models (Extended Abstract)," *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2023-Augus, pp. 6995–7000, 2023, doi: 10.24963/ijcai.2023/797.

[45] L. M. da Costa, F. Bond, and R. V. P. Winder, "The Tembusu Treebank: An English Learner Treebank," *2022 Lang. Resour. Eval. Conf. Lr*, pp. 4817–4826.

[46] E. Ekinci, "A comparative study of LSTM-ED architectures in forecasting day-ahead solar photovoltaic energy using Weather Data," *Computing*, no. 0123456789, 2024, doi: 10.1007/s00607-024-01266-1.

[47] T. T. Nguyen, X. P. Nguyen, S. Joty, and X. Li, "A conditional splitting framework for efficient constituency parsing," *ACL-IJCNLP 2021 - 59th Annu. Meet. Assoc. Comput. Linguist. 11th Int. Jt. Conf. Nat. Lang. Process. Proc. Conf.*, pp. 5795–5807, 2021, doi: 10.18653/v1/2021.acl-long.450.

[48] K. Mrini, F. Dernoncourt, Q. Tran, T. Bui, W. Chang, and N. Nakashole, "Rethinking

self-attention: Towards interpretability in neural parsing," *Find. Assoc. Comput. Linguist. Find. ACL EMNLP 2020*, vol. 1, no. 2017, pp. 731–742, 2020, doi: 10.18653/v1/2020.findings-emnlp.65.

[49] X. Ran, Z. Shan, Y. Fang, and C. Lin, "An LSTM-based method with attention mechanism for travel time prediction," *Sensors (Switzerland)*, vol. 19, no. 4, pp. 1–22, 2019, doi: 10.3390/s19040861.

[50] S. Bhanumathi and S. N. Chandrashekara, "Deep Learning Based BiLSTM Architecture for Lung Cancer Classification," *Int. J. Adv. Res. Eng. a Technol. nd*, vol. 12, no. 1, p. 503, 2021, doi: 10.34218/IJARET.12.1.2020.045.

[51] Y. Yu, D. Qiu, and R. Yan, "A Quantum Language-Inspired Tree Structural Text Representation for Semantic Analysis," *Mathematics*, vol. 10, no. 6, pp. 1–21, 2022, doi: 10.3390/math10060914.

[52] M. Zulqarnain, R. Ghazali, M. G. Ghouse, and M. F. Mushtaq, "Efficient processing of GRU based on word embedding for text classification," *Int. J. Informatics Vis.*, vol. 3, no. 4, pp. 377–383, 2019, doi: 10.30630/joiv.3.4.289.

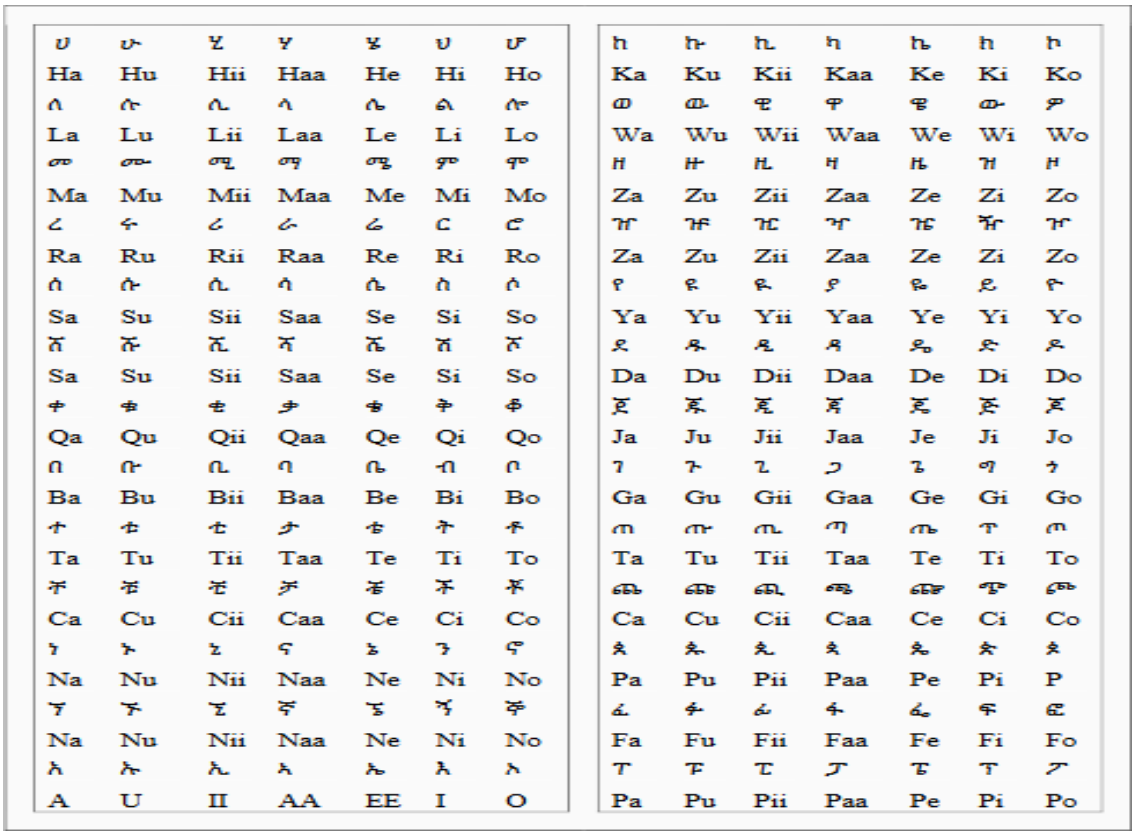
[53] I. Sucholutsky, A. Narayan, and M. Schonlau, "Pay attention and you won " t lose it : a deep learning approach to sequence imputation," pp. 1–17, 2019, doi: 10.7717/peerj-cs.210.

[54] Y. Shen, Y. Tay, C. Zheng, D. Bahri, D. Metzler, and A. Courville, "StructFormer: Joint unsupervised induction of dependency and constituency structure from masked language modeling," *ACL-IJCNLP 2021 - 59th Annu. Meet. Assoc. Comput. Linguist. 11th Int. Jt. Conf. Nat. Lang. Process. Proc. Conf.*, pp. 7196–7209, 2021, doi: 10.18653/v1/2021.acl-long.559.

[55] A. Mulu and V. Goyal, "Amharic Phrase Chunking with Conditional Random Fields,"Google scholar, vol.64, no. 1, pp. 253–259, 2020.

8. APPENDICES

8.1. Appendix A: Siltigna language alphabet



ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
Ha	Hu	Hii	Haa	He	Hi	Ho
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
La	Lu	Lii	Laa	Le	Li	Lo
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
Ma	Mu	Mii	Maa	Me	Mi	Mo
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
Ra	Ru	Rii	Raa	Re	Ri	Ro
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
Sa	Su	Sii	Saa	Se	Si	So
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
Sa	Su	Sii	Saa	Se	Si	So
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
Qa	Qu	Qii	Qaa	Qe	Qi	Qo
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
Ba	Bu	Bii	Baa	Be	Bi	Bo
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
Ta	Tu	Tii	Taa	Te	Ti	To
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
Ca	Cu	Cii	Caa	Ce	Ci	Co
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
Na	Nu	Nii	Naa	Ne	Ni	No
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
Na	Nu	Nii	Naa	Ne	Ni	No
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
A	U	II	AA	EE	I	O

ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
Ka	Ku	Kii	Kaa	Ke	Ki	Ko
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
Wa	Wu	Wii	Waa	We	Wi	Wo
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
Za	Zu	Zii	Zaa	Ze	Zi	Zo
ዠ	ዥ	ዢ	ዣ	ዤ	ዥ	ዦ
Za	Zu	Zii	Zaa	Ze	Zi	Zo
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
Ya	Yu	Yii	Yaa	Ye	Yi	Yo
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
Da	Du	Dii	Daa	De	Di	Do
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
Ja	Ju	Jii	Jaa	Je	Ji	Jo
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
Ga	Gu	Gii	Gaa	Ge	Gi	Go
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
Ta	Tu	Tii	Taa	Te	Ti	To
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
Ca	Cu	Cii	Caa	Ce	Ci	Co
ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
Pa	Pu	Pii	Paa	Pe	Pi	P
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
Fa	Fu	Fii	Faa	Fe	Fi	Fo
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
Pa	Pu	Pii	Paa	Pe	Pi	Po

Appendix figure 8-1 Siltigna language alphabet

8.2. Appendix B: Siltigna punctuation marks

Appendix table 8-1 Siltigna punctuation marks

፡	ሆሽት ጩቁት
/	እዘባር
?	የሱል መልከት
፣	ኑጥል ብጭረ
፤	ዱርብ ብጭረ
« »	የቂሰ መልከት
( )	ቁንፍ
።	አራት ጩቁት
!	አትጣሞቃት መልከት
¡	የዛቀነ መልከት
·	ሀድ ጩቁት
…	ረባረቦ
፡-	ሆሽት ጩቁት ቱንስ ብጭረ

8.3. Appendix C: Sample parse trees data in the dataset

(S (ADJP(ADJ ገወ)(VIm ጡወረ))(NP(NN ሞዬ)(Vp እሰቻን)))
(S (NP(NNp ለበክት)(Vp ሁርም)(Ng ኤለይ)))
(S (NP(NNp ለበክት)(ADV ኡዝረ)(Vp እላሱኒ)))
(S (NP(NNp ለቡጦ)(NN ሞዪም)(Vp ድባያንከ)))
(S (NP(NN አፈ)(ADJ ነኮ))(NP(NN ደለ)(Vp አልጀጎ)))
(S (NP(NN አለበጆ))(VP(Vp ገባት)(ADV ጂንጆ)))
(S (VP(Ng አለኒከ)(Vi የጩኑይ)(Ng እለድገል)))
(S (NP(NNp በሱብር)(Vi የጄሩይ)(Ng እለበስል)))
(S (NP(NNDf አለንገ)(NNp ለፈረዝ))(NP(NN ጮርቄ)(ADJp ለሰነፍ)))

(S (NP(NNp አለደለምከ))(NP(NN ደለ)(Vp ታጄ)))

(S (NP(NN አለገ))(VP(Vi ቲሲቅም)(Vp ለገ)))

(S (NP(NN አለገ))(VP(Vi ትዮዱይም)(Vp እለጋን)))

(S (ADJP(ADJ አሌዘዞ))(NP(NN ገበየ)(Vp ብዞ)))
(S (VP(Vp አልሶጃን)(Aux ባለ))(VP(Vp ባሊ)(NNCo ኡሪ)))
(S (VP(Vp አልርከቦቴ)(Vc ሀርምኔ))(NP(Aux ባለ)(NNCo ወራባይ)))

(S (ADJP(ADJDf አልቄሮ))(ADJP(Pr ገማኮ)(ADJCo ራንጂን)))

(S (ADJP(ADJDf አልቆጮ))(NP(Pr ገማኮ)(NNCo መሶን)))

(S (NP(NN አልቦ)(ADJDf ገንበዘቴ))(NP(NN ጊቾ)(ADJCo ደንገዘቴ)))

(S (ADJP(ADJ አልተርፎ)(NNCo ሎልዴ))(ADJP(ADJ ማሪ)(Vp ለገጌ)))

(S (ADJP(ADJDf አልትሬገጦ))(NP(NNp በገልበ)(Vp ወጦ)))

(S (NP(Np አልቻሎት))(NP(NNCo ታቦት–ወልድም)(Vp እላያን)))

(S (NP(NIfp አልቻሎት))(NP(NN ጦንቆ)(Vi ያበላን)))

(S (NP (NN ክምዳሬ)) (VP ቆማሪንት (NN ደረሰ)))

(S (NP ያየርንባረት (NN ባሎት)) (VP ምንባሎትን))

8.4. Appendix D: Sample correctly predicted output by the models

['s', 'np', 'nn', 'ፎል', 'aux', 'ባለ', 'vp', 'nn', 'ፎለ', 'v', 'እበሌን']

['s', 'np', 'nn', 'አቦት', 'v', 'ያመጣይ', 'vp', 'adj', 'እንደት', 'v', 'ቆሌተይ']

['s', 'np', 'nn', 'አካሮት', 'np', 'nn', 'ተውቆት', 'adj', 'ቅጦን']

['s', 'adjp', 'adjdf', 'አልቆጮ', 'adjp', 'pr', 'ገማምከ', 'adjco', 'መሰን']

['s', 'np', 'nn', 'ጡቢ', 'vim', 'ኦሽትን']

['s', 'np', 'nnp', 'ለበክት', 'advp', 'adv', 'ኡዝረ', 'vp', 'እላሱኒ']

['s', 'np', 'nnp', 'ለቡጦ', 'np', 'nn', 'መዪም', 'vp', 'ድባያንከ']

['s', 'np', 'nndf', 'ጋረ', 'vim', 'መኚሎ', 'np', 'nn', 'ጮርቄ', 'vp', 'ያትንቃቅፋን']

['s', 'vp', 'pr', 'ገገ', 'vi', 'ናዶት', 'vp', 'pr', 'ገገከ', 'vp', 'ታጎትን']

['s', 'advp', 'adv', 'ጡፈ', 'vp', 'ጥፍን']

['s', 'np', 'nndf', 'ለሞተይ', 'vp', 'ቲበኪም', 'np', 'nndf', 'የቀተለይ', 'vi', 'አምለገጠ']

['s', 'adjp', 'adj', 'ፈሪቶ', 'nn', 'በርተ', 'vpr', 'በሬደዳን']

['s', 'adjp', 'adjdf', 'ለኑጉራም', 'nn', 'ገሌ', 'vp', 'vp', 'ባት', 'nnco', 'አደንቴ']

['s', 'adjp', 'adjp', 'ለቦዝ', 'adv', 'አመል', 'np', 'nif', 'ዱንየ', 'adjco', 'ዶንዳንከ']

['s', 'np', 'dt', 'ሁልም', 'nn', 'ጋር', 'vp', 'vp', 'እንጥፍላን']