Dependency Based Grammar Error Detection For Low Resource Languages

O V Rupesinghe

Dependency Based Grammar Error Detection For Low Resource Langauges

Mr. Oshada Rupesinghe
Index No.: 20001525

Supervisor: Dr. B.H.R. Pushpananda

April 2025

Submitted in partial fulfillment of the requirements of the
B.Sc. in Computer Science Final Year Project (SCS4224)

Declaration

I certify that this dissertation does not incorporate, without acknowledgement, any material previously submitted for a degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, to be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

Candidate Name: O.V Rupesinghe

[illegible] 29.06.2025

Signature of Candidate Date

This is to certify that this dissertation is based on the work of Mr. O.V Rupesinghe under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Principal / Co-Supervisor's Name: Dr. B.H.R. Pushpananda

Randil

Signature of Supervisor Date 30 - 06 - 2025

Abstract

This dissertation introduces a end-to-end, data-driven framework for automated grammar error detection (GED) in Sinhala, a low-resource, free-word-order language. We begin by constructing a 400 sentence Universal Dependencies (UD) treebank via a hybrid LLM-and-expert annotation pipeline. Using custom 300 dimensional FastText embeddings, we train a graph-based UUParser with targeted data augmentation and cross-lingual transfer from Hindi. Our final parser achieves an Unlabeled Attachment Score (UAS) of 71.37% and Labeled Attachment Score (LAS) of 55.42%, and attains sentence-level parse accuracy of 82% on a standard correct corpus outperforming a leading CFG-based parser (60%) while retaining 64% accuracy on free-word-order variants . Building on this, we generate a synthetic GED corpus of 10,000 sentences covering five error types. We engineer multi-level token features—pretrained word embeddings, POS embeddings, morphological concatenations, dependency relation embeddings, and syntactic n-grams—and train a BiLSTM classifier. The combined model delivers 80% overall classification accuracy . On a 200 sentence standard evaluation set, it correctly classifies 82% versus 60% for prior CFG-based methods, and it generalizes to free-word-order GED with 64% accuracy. Our contributions include a UD treebank for Sinhala, an optimized dependency parser pipeline, the first dependency-enhanced GED classifier for Sinhala, and a synthetic error corpus. These results confirm that combining hierarchical dependency features with surface-level features significantly boosts GED in challenging low-resource, morphologically rich, free-word-order settings.

Acknowledgement

I would like to express my sincere appreciation to my supervisor, Dr. B.H.Randil Pushpananda and my co-supervisor, Mr. Chamila Liyanage, for their invaluable guidance and advice throughout my project.

Furthermore, I am deeply thankful to the Language Technology Research Laboratory at the University of Colombo School of Computing, as well as the linguistic experts who assisted with the research's annotation and evaluation work. I extend my gratitude to Dr. Ruwan Weerasinghe for his support and valuable insights.

Table of Contents

1 Introduction 1

1.1 Background to the research 1

1.2 Motivation 2

1.3 Problem Statement 2

1.4 Outline of the Dissertation 3

2 Background & Literature review 4

2.1 Sinhala Language Characteristics 4

2.2 Dependency Parsing 5

2.2.1 Dependency Parsing For Low Resource Languages 6

2.3 Grammar Error Detection 7

2.3.1 For High Resource Languages 7

2.3.2 For Low Resource Languages 7

2.4 Dependency Based Grammar Error Detection 8

2.5 Grammar Error Detection For Sinhala 8

2.6 Conclusion 9

3 Research Criteria 10

3.1 Research Gap 10

3.2 Project Aims and Objectives 10

3.2.1 Research aim 10

3.2.2 Research questions 11

3.2.3 Research objectives 11

3.3 Scope and Delimitations 12

3.3.1 In Scope 12

3.3.2 Out of Scope 12

4 Research Methodology & Design 13

4.1 Training Dependency Parser 13

4.1.1 Data Collection & Annotation 14

4.1.2 Dependency Parser Training Framework 15

4.1.3 Training Word Embedding Model 16

4.1.4 Data Augmentation 16

4.1.5 Transfer Learning (Cross Lingual Training) 18
4.2 Grammar Error Detection Model 20
4.2.1 Data Collection & Annotation 21
4.2.2 Synthetic Error Generation 22
4.2.3 Feature Engineering 24
4.2.4 Model Training 30

5 Experiments and Results **31**
5.1 Evaluation Metrics 31
5.1.1 Dependency Parsing Evaluation Metrics 31
5.1.2 Grammar Error Detection (GED) Model Evaluation Metrics . . . 32
5.2 Pleliminary Experiments 33
5.2.1 Dependency Parser Training Framework Selection 33
5.2.2 Word Embedding Model Selection 33
5.3 Dependency Parsing Experiments 35
5.3.1 Datasets 35
5.3.2 Experiment 1 : Language Selection 35
5.3.3 Experiment 2: Comparing Graph-Based vs. Transition-Based Parsing Approaches for Sinhala 37
5.3.4 Experiment 3: Applying Data Augmentation 38
5.3.5 Experiment 4: Combination Augmentation Techniques 39
5.3.6 Experiment 5: Transfer Learning 40
5.3.7 Experiment 6: Transfer Learning With Short Sentences 40
5.3.8 Experiment 7: Hierarchical Transfer Learning 42
5.3.9 Experiment 8: Transfer Learning With Data Augmentation . . . 42
5.3.10 Evaluation On The Grammar Book Dataset 43
5.3.11 Analysis of Results 44
5.3.12 Summary of Dependency Parsing Experiments 47
5.4 Grammar Error Detection Experiments 48
5.4.1 Datasets 48
5.4.2 Experiment 1 : Word Embedding Only Model 48
5.4.3 Experiment 2 : Word Embedding + POS Model 50
5.4.4 Experiment 3 : Word Embedding + Morph Model 51
5.4.5 Experiment 4 : Word Embedding + Dependency Model 52
5.4.6 Experiment 5 : Feature Combination 53
5.4.7 Evaluation on Standard Data Sets 54
5.4.8 Analysis of Results 55
5.4.9 Summary of Grammar Error Detection Experiments 57

6 Conclusion **58**
6.1 Limitations 59

6.2 Implications for Further Research 60

List of Figures

Figure 2.1 Sinhala free word order example (Liyanage et al. 2012) . 5
Figure 2.2 Sinhala dependency annotated sentence example 5

Figure 4.1 Methodology for training dependency parser 13
Figure 4.2 LLM Annotation Vs Actual Annotation 15
Figure 4.3 UD data pipeline 15
Figure 4.4 Example of Sentence Cropping 17
Figure 4.5 Example of Sentence Rotation 17
Figure 4.6 Methodology for training GE! (GE!) model 20
Figure 4.7 S-V Agreement Error Generation 24
Figure 4.8 Feature representation of Morphological Features 27
Figure 4.9 Dependency Parsed Sentence 28
Figure 4.10 Overall Model Architecture 30

Figure 5.1 Experimental setup for training word embedding model 34
Figure 5.2 Sentence Length Analysis 41
Figure 5.3 Graph vs Transition parser output 45
Figure 5.4 Confusion matrix of dependency parsing model 46

List of Tables

Table 2.1 Summary of existing Sinhala grammar modeling and correction studies . . . 9

Table 4.1 Evaluation Scores . . . 21
Table 4.2 Example of morphological analysis . . . 26

Table 5.1 Results on UD dataset with 100 sentences . . . 33
Table 5.2 Intrinsic Evaluation Scores . . . 34
Table 5.3 Extrinsic Evaluation Scores . . . 34
Table 5.4 Dataset for crosslingual training base model . . . 35
Table 5.5 Dataset for crosslingual training . . . 35
Table 5.6 Multingual Tree Bank Datasets . . . 35
Table 5.7 Comparison of Word and Subword Overlap Across Languages . . . 36
Table 5.8 Dependency Label Distribution Similarity with Sinhala . . . 36
Table 5.9 POS Tag Distribution Similarity with Sinhala . . . 36
Table 5.10 Results of baseline dependency parser . . . 38
Table 5.11 Dataset summary for data augmentation . . . 38
Table 5.12 Results of data augmentation . . . 38
Table 5.13 Results after combining dataset . . . 39
Table 5.14 Results on transfer learning . . . 40
Table 5.15 Results after aligning sentence lengths . . . 41
Table 5.16 Results on hierarchical transfer learning . . . 42
Table 5.17 Results of applying transfer learning from augmented dataset 43
Table 5.18 Results on grammar book dataset . . . 43
Table 5.19 Per class frequencies of GED dataset . . . 48
Table 5.20 Classification report of the word only model . . . 49
Table 5.21 Classification report of the word + POS model . . . 50
Table 5.22 Classification report of the Word + Morph model . . . 51
Table 5.23 Classification report of the dependency based representation 52
Table 5.24 Classification report of the combined model . . . 53
Table 5.25 Summary of Misclassified Correct Sentences by Model . . 54
Table 5.26 Results on free word order structures . . . 54

Acronyms

NLP Natural Language Processing

NMT Neural Machine Translation

GEC Grammar Error Correction

GED Grammar Error Detection

BERT Bidirectional Encoder Representations from Transformers

SMT Statistical Machine Translation

GPT Generative Pre-Trained Transformer

LLM Large Language Models

ML Machine Learning

POS Part Of Speech

SOA State Of the Art

LM Language Models

LSTM Long Short Term Memory

CFG Context-Free Grammar

UD Universal Dependencies

UAS Unlabelelled Attachment Score

LAS Labelelled Attachment Score

FWO Flexible Word Order

OOV Out Of Vocabulary

MLP Multi Layer Perceptron

Chapter 1

Introduction

1.1 Background to the research

Sinhala is an Indo-Aryan language spoken by approximately 17 million people in Sri Lanka and the global Sri Lankan diaspora. It exhibits rich morphology where nouns inflect for case, number, gender, definiteness, and animacy, while verbs carry information about tense, person, number, and aspect and permits flexible word order without altering core meaning (Liyanage et al. 2012). These linguistic traits pose significant challenges for automated grammar-error detection GED, surface cues like linear word position become unreliable when multiple valid permutations exist.

State-of-the-art GED systems for high-resource languages (e.g., English, Chinese) rely on large error-annotated corpora and deep transformer models (Devlin et al. (2019), Liu et al. (2019)), achieving F0.5-scores above 80% on benchmarks such as CoNLL-2014. In contrast, low-resource, free-word-order languages like Sinhala lack both extensive training data and off-the-shelf parsers. Early Sinhala grammar checkers adopted rule-based or CFG-based approaches (Hettige & Karunananda (2011), Liyanage et al. (2012)), offering reasonable performance on canonical SOV constructions but failing dramatically on reordered or colloquial variants.

Dependency parsing which encodes syntactic relationships between words rather than fixed positions has been shown to excel on morphologically rich, non-configurational languages by focusing on head-dependent relations (de Lhoneux, Shao, Basirat, Kiperwasser, Stymne, Goldberg & Nivre 2017). However, training high-accuracy dependency parsers typically requires large, manually annotated treebanks, of which Sinhala has only a few hundred sentences. To address this data scarcity, recent work in other low-resource settings has demonstrated the effectiveness of tree morphing (cropping, rotation) (Sahin & Steedman 2018), nonce sentence generation, and hierarchical cross-lingual transfer from typologically related languages (Ghiffari et al. 2023).

This research leverages these low-resource strategies to build the first end-to-end dependency-based GED pipeline for Sinhala. By combining selective data augmentation, transfer learning from Hindi, and multi-level feature engineering (word embeddings, POS, morphology, dependency labels), we aim to overcome both syntactic variability and data limitations. Our approach not only fills a critical gap in Sinhala NLP, but also establishes a template for deploying dependency-driven GED in other low-resource, free-word-order languages.

1.2 Motivation

The development of GED systems significantly benefit the society by enhancing educational outcomes, supporting professional communication and promoting digital inclusion. But there is a scarcity of GED systems for low resource languages. Most studies have attempted rule based approaches but they have not been developed further into usable tools since they require linguistic expertise and manual feature engineering to scale. These challenges led us into exploring the applicability of data-driven approaches for low resource GED languages. We chose Sinhala as a candidate language to conduct our study. Our findings will positively impact computational linguistics for Sinhala and other low resource languages.

1.3 Problem Statement

The primary research problem is the lack of effective grammar error detection systems for Sinhala that can accommodate the language's flexible word order and morphological complexity. Specifically:

- Existing rule-based approaches to Sinhala GED fail to adequately handle the language's flexible word order characteristics (Liyanage et al. (2012), Pabasara & Jayalal (2020)).
- The absence of a sizable grammar error annotated datasets and dependency treebanks for Sinhala has hindered progress in developing dependency based GED systems.
- There is a need to evaluate whether techniques successful in low-resource scenarios for other languages can be effectively applied to develop dependency parsing and subsequent GED systems for Sinhala.

This research problem necessitates investigating how a dependency parser can be developed with limited resources and subsequently how it can be utilizsed to create a grammar error detection system that accommodates Sinhala's distinctive linguistic properties.

1.4 Outline of the Dissertation

The remainder of this dissertation has been organized into seven chapters. This section outlines the description of each chapter :

- Chapter 1 Provides an overview of the study, covering the background, motivatio.
- Chapter 2 Surveys the theoretical foundations and reviews the state of the art related to this research.
- Chapter 3 Outlines the research gap, research questions, methodology, and delimitations
- Chapter 4 Outlines the design of the proposed solution, detailing the concepts and techniques used to achieve the research objectives.
- Chapter 5 Reports the experimental results and evaluates the effectiveness of the proposed approach.
- Chapter 6 Concludes the thesis by summarizing key contributions and suggesting directions for future work.

Chapter 2

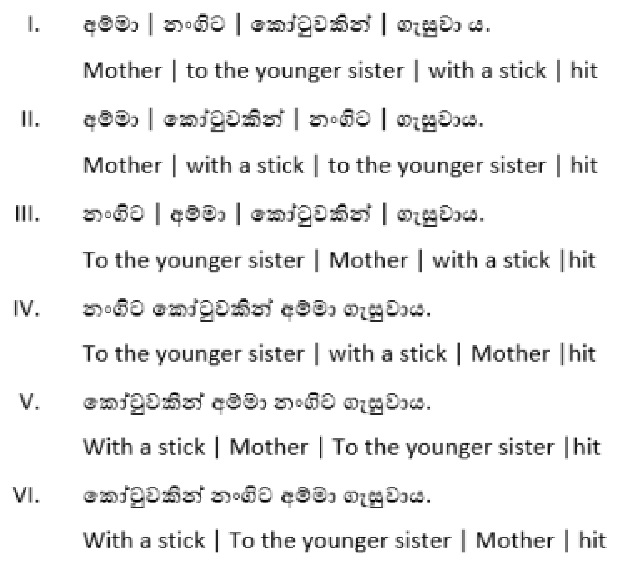
Background & Literature review

This section provides a comprehensive overview of the existing literature relevant to our research. We begin by examining the grammatical characteristics of Sinhala, highlighting key challenges specific to this language. Our study focuses on dependency-based GED. However, since dependency parsing tools are not readily available for Sinhala, we will first review the literature on dependency parsing in general before addressing grammar error detection specifically.
We'll structure this review to first establish the linguistic context of Sinhala, then explore the computational methods needed for our analysis, and finally examine current approaches to grammar error detection that inform our work.

2.1 Sinhala Language Characteristics

Sinhala is a diglossic language with two main forms: Spoken Sinhala and Written Sinhala. These forms differ in their vocabulary and sentence structure (Liyanage et al. 2012). While Sinhala typically follows SOV word order, it allows for flexible word arrangements that create different meanings in conversation. (Liyanage et al. 2012). Like most Indo-Aryan languages, Sinhala is agglutinative, where nouns can be changed to show case, number, gender, definiteness and animacy, while verbs can be changed to show tense, number, gender, person, and volition (Karuṇātilaka 2012).
Typically, a sentence consists of two parts: a verb phrase (VP) and a noun phrase (NP). In Sinhala grammar, a sentence consists of two parts: the Uktha (subject) and the Akyatha (predicate). The subject and predicate in Sinhala sentences agree on the **person, gender** and **number**.
Sinhala presents unique linguistic properties that make computational processing particularly challenging. The language features a complex morphological structure with extensive inflection patterns, where words can take numerous forms based on grammatical requirements (Karuṇātilaka 2012). Another challenge is the syntactic flexibility which we try to tackle in this study. Figure 4.9 lists every potential Sinhala sentence for the sentence the English sentence "Mother hit the younger sister with a stick."

For modeling GED tools this variability poses significant challenges since word position doesn't reliably indicate grammatical function. In order to solve this dependency grammars are used in place of context free grammars (Liyanage et al. 2012).



I. අම්මා | නංගිට | කෝටුවකින් | ගැසුවා ය.
Mother | to the younger sister | with a stick | hit

II. අම්මා | කෝටුවකින් | නංගිට | ගැසුවාය.
Mother | with a stick | to the younger sister | hit

III. නංගිට | අම්මා | කෝටුවකින් | ගැසුවාය.
To the younger sister | Mother | with a stick |hit

IV. නංගිට කෝටුවකින් අම්මා ගැසුවාය.
To the younger sister | with a stick | Mother |hit

V. කෝටුවකින් අම්මා නංගිට ගැසුවාය.
With a stick | Mother | To the younger sister |hit

VI. කෝටුවකින් නංගිට අම්මා ගැසුවාය.
With a stick | To the younger sister | Mother | hit

Figure 2.1: Sinhala free word order example (Liyanage et al. 2012)

2.2 Dependency Parsing

Dependency syntax is based on the idea that syntactic structure is formed by relationships between words in a sentence. These relationships are typically shown as binary asymmetric connections called dependencies. The figure 2.2 shows dependency relationships for the sinhala sentence “ඔහු අවධානය යොමුකර ඇත්තේ සල්ලි ගැන පමණය” with the meaning “He is focused only on money“.

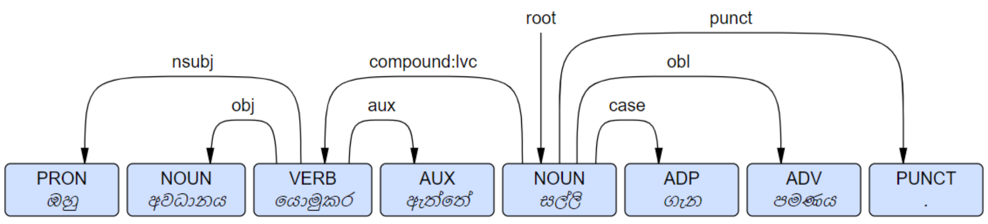


Figure 2.2: Sinhala dependency annotated sentence example

Dependency parsing avoids word order information, representing only the information that is necessary to parse. As evidence Winter (2001) showed that dependency parsers achieve better results than constituency-based approaches when parsing morphologically rich languages with flexible word order.

Based on the parsing strategy and information scope there are two dependency parsing techniques, namely:

1. Transition-based parsing : Efficient but is only suitable for simple sentence structures.
2. Graph-based parsing : Computationally in-efficient but can handle complex structures due to it's global context.

2.2.1 Dependency Parsing For Low Resource Languages

Training a high-accuracy dependency parser requires a large treebank. Sinhala does not have access to such large dependency annotated datasets. Several strategies such as transfer learning and data augmentation have been studied to tackle this problem for other languages.

Data Augmentation

One prominent data augmentation methods is dependency tree morphing, which involves transformations like cropping and rotation enabling parsers to learn syntactic flexibility, this method is particularly important for languages such as Sinhala which has flexible word order. Tree morphing has shown significant improvements in parsing performance. For instance North Sámi an extremely low resource language demonstrated a notable 9.3% increase in labeled Labelelled Attachment Score (LAS) (Vania et al. 2019).
Another approach, nonce sentence generation, replaces content words with others of similar syntactic annotations, allowing parsers to generalize better across lexical variations. This technique preserves sentence structure while introducing syntactic diversity (Feng et al. 2021).

Transfer Learning

Data augmentation is useful to a certain extent but to gain higher scores techniques such as transfer learning have shown promise. In the study done by Ghiffari et al. (2023) for Javanese, hierarchical transfer learning led to notable improvements in parsing performance. Similarly, Wang et al. (2019) applied transfer learning to Singlish, an English-based creole, by leveraging English as a source languages. They yielded Unlabelelled Attachment Score (UAS) and LAS scores of 85.57% and 79.12%, respectively, representing improvements over baseline models trained without transfer. Cross-lingual training on typologically similar languages also contributes significantly to parsing accuracy for other low-resource languages. For Sinhala, Indic languages such as Hindi, Urdu, Tamil can be considered.

Available Training Frameworks

One of the most popular training tools is the Uppsala University Parser (Smith et al. 2018) which demonstrates particular strength with its specialized feature representations for morphological information and strong performance for free word order languages. While the Stanford Neural

Dependency Parser excels with dense word representations, studies by Vania et al. (2019) indicate it requires substantial adaptation for highly inflected languages. More recent universal frameworks like UDify and UDPipe 2.0 leverage multilingual pre-trained models and multi-task learning for cross-language performance but performs poorly at capturing language-specific morphological nuances.

2.3 Grammar Error Detection

The first grammar checking tools, such as the Unix Writer's Workbench and Microsoft Word text editor used hand-crafted rules and pattern matching techniques. These systems have notable limitations, such as difficulty handling ambiguous or complex sentences and the inability to adapt to new language usages without manual updates. Since the 90s, in line with general trends in Natural Language Processing (NLP), the rule-based approach has been replaced by statistical methods with the increasing availability of annotated corpora (Wang et al. 2020).

2.3.1 For High Resource Languages

GED for high-resource languages like English have achieved State Of the Art (SOA) through deep learning approaches, particularly through fine-tuned language models like BERT and RoBERTa (Devlin et al. (2019), Liu et al. (2019)). Current SOA systems demonstrate $F_{0.5}$ scores exceeding 80% on standard benchmarks (Wang et al. 2020). These models have been trained on annotated corpora containing millions of sentences. They also perform well on semantic error detection due to attention mechanisms (Náplava & Straka 2019). Lately morphologically rich languages such as Chinese, Russian and Arabic have also applied deep learning achieving SOA in GED for each language Rozovskaya & Roth (2021).

2.3.2 For Low Resource Languages

Although deep learning approaches yield state of the art results for high resource languages their applicability in low resource settings has been in research for over a decade. Recent advancements have shown some strategies for deep learning approaches to be extended for the Grammar Error Correction (GEC) task for less resourceful languages.

Synthetic Data Generation

Several studies have shown methods for generating GED datasets by inducing synthetic errors. The study carried out by Palma Gomez et al. (2023) used spell and word based transformations to generate a dataset to train a GEC model for Ukranian language. (Ponce et al. 2023) has also used such techniques to create a Neural Machine Translation (NMT) based grammar error checker with high accuracy for Tagalog.
These methods are also applicable for morphologically rich langauges, for example Sonawane

et al. (2020) artificially created an error corpus by injecting inflectional errors into grammatically correct Hindi sentences. This approach allowed them to generate a large corpus to train a deep learning model achieving a GLEU score of 80. Since a tagged error corpus is not available for Sinhala, these approaches might allow for training deep learning based GED models for Sinhala.

Pre-trained Large Language Models (LLM)s

Pre-trained language models aim to use large unlabeled corpora to learn text representations at scale, such that the trained models can be fine-tuned on relatively smaller datasets for downstream tasks. For instance, Kaneko et al. (2020) applied BERT-based architectures to GED, achieving competitive results with minimal task-specific data. More recently, Ponce et al. (2023) utilized a tagalog BERT and RoBERTa models, achieving strong results despite the scarcity of training data. Though this technique reduce the amount of labeled ata required, still a considerable amount of gold data is required for proper finetuning.

2.4 Dependency Based Grammar Error Detection

Our work explores how dependency grammars are applicable for low resource GED. Considering the available literature, not much work has been done for this area. But some studies provide us suggestions on how we can input dependency parsed data into machine learning models. Tezcan et al. (2017) has applied dependency parsing for identifying machine translation errors. They have used syntactic n-grams to capture hierarchical dependencies which surface level n-grams such as POS fail to capture. For highly inflected languages such as Sinhala combining dependency labels with Part Of Speech (POS) and morphology would boost sentence-level error detection accuracy, because the model learns which link types rarely co-occur in correct text.

2.5 Grammar Error Detection For Sinhala

GED for sinhala has not seen significant advancements compared to other languages. The domain is heavily dominated by rule-based approaches, due to the unavailability of a large annotated grammar error corpus. But there are a few notable studies. Hettige & Karunananda (2011) have developed have developed a computational grammar for sinhala by considering the Morphological and the syntax analysis for trivial sentences. Building upon this work a feature-based Context-Free Grammar (CFG) for sinhala has been developed by Liyanage et al. (2012) which supports non-trivial sentences. But the accuracy of this computational grammar suffers due to free word order variants. They suggest that for Sinhala dependency grammars are more suitable than CFG.

Several other rule based systems have been put forward for Sinhala. Moving out of rule based approaches Pabasara & Jayalal (2020) developed a grammar checker using a hybrid approach.

This system used a rule based grammar for pattern recognition and subsequently a decision tree-based algorithm was utilized to evaluate the verb in relation to the subject, providing feedback regarding the correctness of the sentence. As future work they suggest that rule based grammars should be replaced by deep learning techniques. The table 2.1 contains a summary of the most prominent studies done in relation to GED for Sinhala.

Author	**Approach**	**Limitations**
Hettige & Karunananda (2011)	Rule Based	Supports only 3-word sentences
Liyanage et al. (2012)	Rule Based	Low accuracy (60%)
Pabasara & Jayalal (2020)	Hybrid	Limitations due to rule-based pattern detection

Table 2.1: Summary of existing Sinhala grammar modeling and correction studies

2.6 Conclusion

Considering the notable work done in the Sinhala NLP domain, initial work such as the computational grammar built by Liyanage et al. (2012) have used phrase based grammars to model grammatical rules. They also conclude that for free word order languages such as Sinhala, dependency grammars should be incorporated. Further other studies suggest to move towards data driven approaches for pattern recognition. To add to this other languages including Indic languages such as Hindi, Bangali etc. have used transformer based encoder decoder architectures to overcome structural variability, but applying these techniques for Sinhala is challenging due to it's low resource nature. But several studies show promise such as transfer learning and artifical data generation. Further incorporating features such as dependency parsing has helped GED to overcome the syntax variability. Therefore the applicability of data-driven approaches along with dependency parsing for GED has to be explored for Sinhala.

Chapter 3

Research Criteria

3.1 Research Gap

Sinhala is considered as low resource language, due to this fact work in the area of GED has mainly taken a rule based approach. But these approaches have failed to capture the word order flexibility of Sinhala. As mentioned by Liyanage et al. (2012) flexible word order problem is solved using dependency grammars. But despite its importance for Sinhala, this area has not been researched for the Sinhala language. This is due to the lack of a sizable dependency treebank for Sinhala. But with the strategies such as data augmentation and transfer learning a parser could be developed even with a small dataset. Further other languages have moved on towards more data driven approaches for GED. Studies have come up with low resource strategies that are applicable for Sinhala. Therefore a significant research gap exists on the exploration of dependency based GED for Sinhala. By addressing these research gaps, this study aims to explore and evaluate applicability of dependency parsing for GED with the aim of developing a robust GED model for Sinhala.

3.2 Project Aims and Objectives

3.2.1 Research aim

The primary objective of this research is to investigate the applicability dependency parsing compared to other features to tackle the flexible word-order nature in low-resource scenarios for developing a more comprehensive grammar error detection model specifically for the Sinhala language.

3.2.2 Research questions

To achieve the above research aim and to tackle the research problem mentioned in section 3.1 the following research questions have to be answered.

1. What low resource strategies such as data augmentation techniques and transfer learning enhance dependency parsing accuracy in a low-resource language like Sinhala ?
2. How can dependency parsing be complemented with other feature representations for detecting grammatical errors in low-resourced, free word-order languages with minimal labeled data?
3. How effective is the proposed model for grammar error detection for conventional and non-conventional sentence structures in Sinhala?

3.2.3 Research objectives

- Train and evaluate a dependency parser by evaluating the benefits of combining data augmentation techniques and transfer learning to improve dependency parsing accuracy for Sinhala. By examining each technique's impact independently and jointly, the objective is to determine an optimized model pipeline that integrates augmentation and transfer learning to boost parsing accuracy.
- Develop a pipeline that can incorporate surface-level features such as POS tags or Morphology and hierarchical-level features such as dependency relations and labels in order to compare the effectiveness of each feature in detecting grammar errors in low resource settings.
- Evaluate the effectiveness of the trained models in detecting grammar errors in Sinhala, assessing accuracy, precision, and the ability of each feature to handle free word order structures in isolation and combination.

3.3 Scope and Delimitations

3.3.1 In Scope

This research will address the following tasks:

- Collecting and manually annotating Sinhala sentences for dependency parsing.
- Producing a dataset with artificially generated grammar errors.
- Training and fine‑tuning a dependency parsing model tailored to Sinhala.
- Developing and implementing a grammar error detection model that captures syntactic error classes.
- Evaluating the parsing and error detection models using the standard metrics.

3.3.2 Out of Scope

The following items fall outside the scope of this project:

- Designing or developing a user‑friendly interface (GUI) to demonstrate the grammar error detection system.
- Implementing automatic correction of identified grammar errors.
- Pinpointing the exact token positions of each error within sentences.
- Simultaneously detecting and classifying multiple error types in a single sentence.
- Identification of semantic errors in Sinhala sentences.

Chapter 4

Research Methodology & Design

The main objective of this research is to model an accurate GED model for Sinhala using dependency parsing. In order to achieve this objective we have to conduct our research in two main phases. In the first phase we will train and evaluate a dependency parsing model for Sinhala. In the next phase a GED model will be developed incorporating dependency parsing.

4.1 Training Dependency Parser

As our first step we conduct a preliminarily experiment to identify which parsing framework will be suitable for Sinhala. Considering the results a word embedding model will be trained and evaluated that suits the configurations of the selected parser.

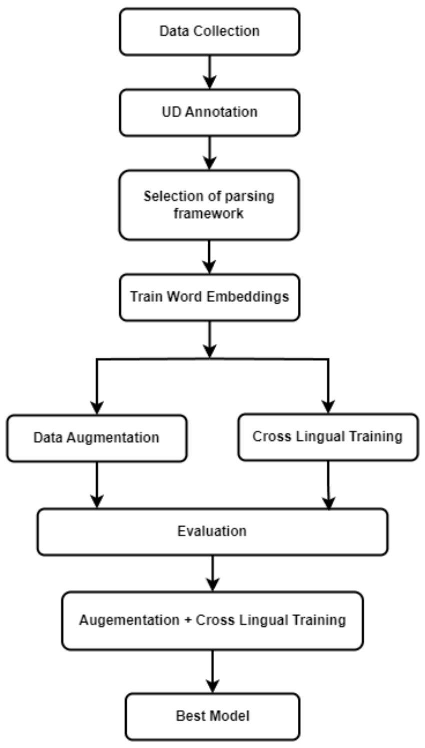


Figure 4.1: Methodology for training dependency parser

Then we train a parser on the annotated dataset and evaluate each approach low resource strategy separately and in combination based on results. Finally the best model will be used for the second phase of the study.

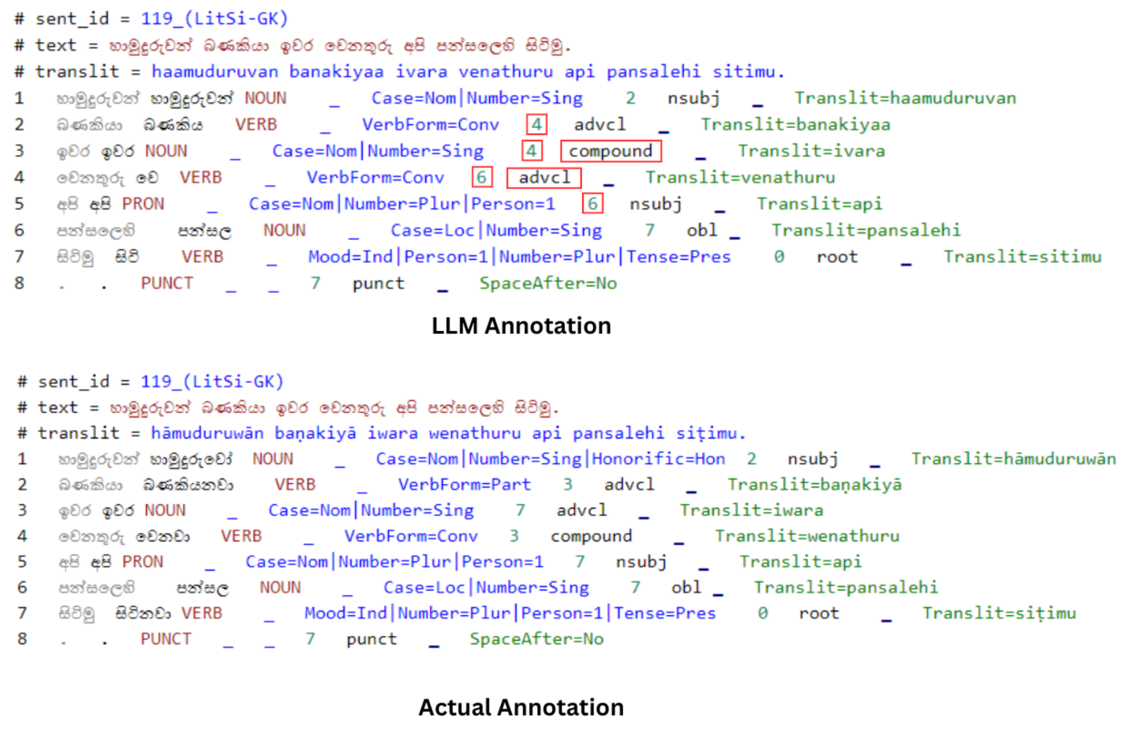
4.1.1 Data Collection & Annotation

Data Collection

The current Universal Dependencies (UD) treebank for Sinhala contains 100 sentences collected from various sources such as newspaper articles, novels and short stories. Since this dataset is insufficient we collected and annotated 300 sentences conforming to UD format. In order justify the coverage on standard sentence structures we chose Sinhala government text books and standard grammar books as sources. The sentences were extracted using an OCR tool and the dataset was cleaned to eliminate sentences in spoken form.
The sentence length was limited to 12 words in order place the scope of the research at a manageable level. This limit was selected by analyzing the 10M word corpus which contained approximately 300,000 sentences.

Data Annotation

UD annotation takes requires expert linguistic knowledge and time especially for a free word order language such as Sinhala. Therefore we eased the process by partially annotating POS and Morphological features using an LLM. We tested several LLMs and Claude 3.7 Sonnet gave the best output. The LLM was reasonably accurate for POS and morphology annotations but performed poorly on dependency relations and label attachments. As depicted in Figure 4.6, although the LLM was able to assign some labels and dependencies with some accuracy, there are still a considerable amount of adjustments required especially in long sentences.



```
# sent_id = 119_(LitSi-GK)
# text = හාමුදුරුවන් බණකියා ඉවර වෙනතුරු අපි පන්සලෙහි සිටිමු.
# translit = haamuduruvan banakiyaa ivara venathuru api pansalehi sitimu.
1   හාමුදුරුවන් හාමුදුරුවන් NOUN   _   Case=Nom|Number=Sing   2   nsubj   _   Translit=haamuduruvan
2   බණකියා   බණකිය   VERB   _   VerbForm=Conv   4   advcl   _   Translit=banakiyaa
3   ඉවර ඉවර NOUN   _   Case=Nom|Number=Sing   4   compound   _   Translit=ivara
4   වෙනතුරු වෙ   VERB   _   VerbForm=Conv   6   advcl   _   Translit=venathuru
5   අපි අපි PRON   _   Case=Nom|Number=Plur|Person=1   6   nsubj   _   Translit=api
6   පන්සලෙහි   පන්සල   NOUN   _   Case=Loc|Number=Sing   7   obl _   Translit=pansalehi
7   සිටිමු   සිටි   VERB   _   Mood=Ind|Person=1|Number=Plur|Tense=Pres   0   root   _   Translit=sitimu
8   .   .   PUNCT   _   _   7   punct   _   SpaceAfter=No
```

LLM Annotation

```
# sent_id = 119_(LitSi-GK)
# text = හාමුදුරුවන් බණකියා ඉවර වෙනතුරු අපි පන්සලෙහි සිටිමු.
# translit = hāmuduruwān baṇakiyā iwara wenathuru api pansalehi siṭimu.
1   හාමුදුරුවන් හාමුදුරුවෝ   NOUN   _   Case=Nom|Number=Sing|Honorific=Hon  2   nsubj   _   Translit=hāmuduruwān
2   බණකියා   බණකියනවා   VERB   _   VerbForm=Part   3   advcl   _   Translit=baṇakiyā
3   ඉවර ඉවර NOUN   _   Case=Nom|Number=Sing   7   advcl   _   Translit=iwara
4   වෙනතුරු වෙනවා   VERB   _   VerbForm=Conv   3   compound   _   Translit=wenathuru
5   අපි අපි PRON   _   Case=Nom|Number=Plur|Person=1   7   nsubj   _   Translit=api
6   පන්සලෙහි   පන්සල   NOUN   _   Case=Loc|Number=Sing   7   obl _   Translit=pansalehi
7   සිටිමු   සිටිනවා VERB   _   Mood=Ind|Number=Plur|Person=1|Tense=Pres   0   root   _   Translit=siṭimu
8   .   .   PUNCT   _   _   7   punct   _   SpaceAfter=No
```

Actual Annotation

Figure 4.2: LLM Annotation Vs Actual Annotation

Therefore the partially annotated sentences were later reviewed by a linguistic expert. The overview of the dependency data collection pipeline is depicted below.

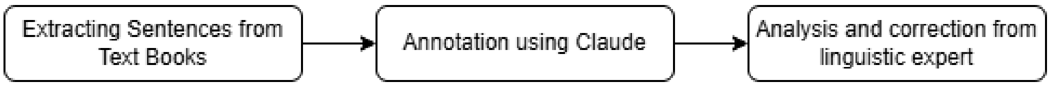

Extracting Sentences from Text Books
Annotation using Claude
Analysis and correction from linguistic expert

Figure 4.3: UD data pipeline

4.1.2 Dependency Parser Training Framework

A suitable parser for the low resource and morphologically rich nature of Sinhala was required for training an accurate dependency parser. Among possible candidates the Uppsala University Parser (UUParser) and the Stanford parser were considered based on adaptability to Sinhala. Both tools are designed to be trained on UD annotated data and additional language specific tools such as tokenizers are not required (Chen & Manning (2014), de Lhoneux, Shao, Basirat, Kiperwasser, Stymne, Goldberg & Nivre (2017)).
A preliminary experiment was conduct to identify the most suitable framework and the UUParser[1] outperformed the Stanford Parser[2]. Therefore we design our approach to facilitate the requirements of this framework. The UUParser is a versatile dependency parser developed by the Uppsala NLP group, capable of operating in both transition-based and graph-based modes.

[1]UUparser: `https://github.com/UppsalaNLP/uuparser`
[2] `https://nlp.stanford.edu/software/lex-parser.shtml`

4.1.3 Training Word Embedding Model

The UUParser can use pre-trained embeddings instead of randomly initializing vectors. Using pre-trained embeddings can assist in handling Out Of Vocabulary (OOV) words. For free word order languages, contextual embeddings can capture morphological information that signals grammatical relationships independently of word position rather than sparse representations. Instead of using publicly available embeddings for Sinhala such as fastText embeddings trained on wiki-dumps, we train our own embedding model and conduct an evaluation comparing other embedding models for Sinhala. The best candidate is used for further experiments in the study. As the training architecture fastText was chosen since it considers sub-word information therefore performs best for low-resource morphologically rich languages like Sinhala (Bojanowski et al. (2017) ; Mikolov et al. (2018)). This was also evident in the research conducted by Lakmal et al. (2020).
Only a 300 dimension embedding model was trained since it was evident in the study done by Lakmal et al. (2020) that 300 dimension embedding model is sufficient to capture rich semantic and syntactic patterns, but small enough to train efficiently. The trained model will be evaluated on the metrics used by Lakmal et al. (2020).

4.1.4 Data Augmentation

In order to address our first research question which is to analyze the augmentation techniques that can be used to improve dependency parsing in low resource settings for Sinhala. Following the survey done for low-resource dependency parsing by Vania et al. (2019), we evaluate the same strategies to find out which augmentation method suits Sinhala dependency parsing.

Dependency Tree Morphing

Dependency tree morphing is a data augmentation technique aimed at increasing the amount and variability of training data by manipulating the dependency tree structures. The assumption is that this techniques will improve generalizability for a language of free word order nature such as Sinhala. We employ the two main operations introduced by Sahin & Steedman (2018).

1. Cropping : Cropping reduces the complexity of sentences by selectively removing peripheral subtrees from the dependency tree. Specifically, this operation focuses on parts of the sentence that hold core grammatical relations such as nominal subjects (NSUBJ), indirect objects (IOBJ), direct objects (OBJ), and oblique nominals (OBL) while removing unecessary grammatical relationships.

2. Rotation : This operation, rotation, keeps all the words in the sentence but re-orders subtrees attached to the root verb, in particular those attached by NSUBJ (nominal subject), OBJ (direct object), IOBJ (indirect object), or OBL (oblique nominal) dependencies.

   To avoid creating confusing dependency structures, rotations are limited to a maximum of three per sentence. This is because Sinhala, even though it allows flexible word order,

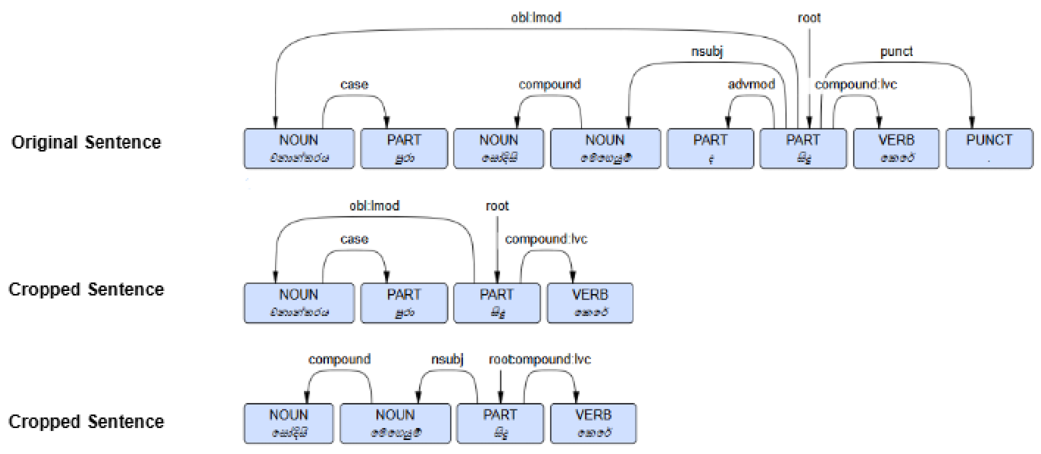


Figure 4.4: Example of Sentence Cropping

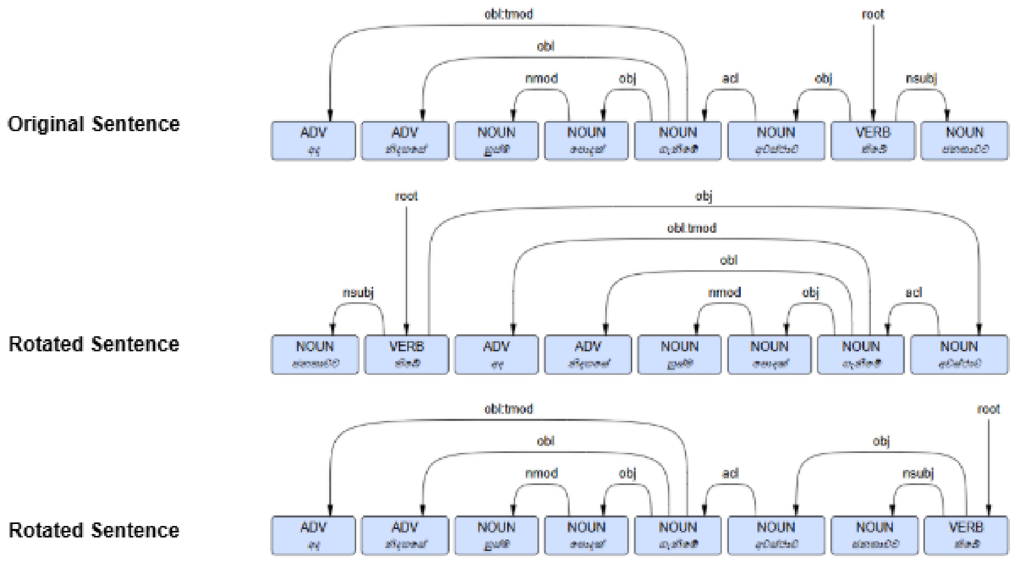


Figure 4.5: Example of Sentence Rotation

still has practical limits.

To mitigate the introduction of excessive noise into the training data, both these operations are applied selectively to only 30% of the sentences. Evaluation will be conducted on the test set which will not contained cropped or rotated sentences, this will help evaluate the effectiveness of these operations in learning free word order structures.

Nonce Sentence Generation

Since we are focusing on syntax rather than semantics we adopt the augmentation method is adapted from Gulordava et al. (2018). Nonce sentence generation works by replacing some of the words which have the same syntactic annotations. For each training sentence, we replace each content word nouns, verbs, or adjectives with an alternative word having the same universal

POS, morphological features, and dependency label. For example a sentence "ඔහු මුදල් ණයට ගත්තා." could be reconstructed as:

1. ඔහු පොත් ණයට ගත්තා.

2. ඔහු පොත් ණයට දැමුවා.

Sometimes this method produces nonsensical sentences similar to 2 above. But since we only replace words if they have the same morphological features and dependency label, this method preserves the original tree structures. Following original studies we generate five nonce sentences for each original sentence.

4.1.5 Transfer Learning (Cross Lingual Training)

To address the remaining part of the first research question, we need to explore cross-lingual training using a high-resource language similar to Sinhala. Our hypothesis is that by training a cross-lingual parser, the patterns learned from the source high-resource language can be transferred to Sinhala as the target language.

Selecting Language For Transfer Learning

Transfer learning has shown to be more effective when source and target languages are closely related. Therefore, the first step involves finding the most closely related high-resource languages to Sinhala. Therefore, the Lexical similarity, POS tag distribution and Dependency label distribution is analyzed. The analysis was carried out on 3 similarity measures,

- Lexical Similarity : Both Sinhala and the target languages were transliterated and the word overlap, Similar Words based on Edit Distance and Subword Unit Overlap was considered.

- POS Tag distribution : The distribution on POS tags such as ADJ, ADP, ADVP etc. were analyzed, the idea was that languages with similar grammatical features tend to have similar POS tag distributions.

- Dependency Label distribution : Comparing dependency label distributions reveals syntactic similarities, such as word order, grammatical structures, and typological features, helping to assess how closely related two languages are in terms of syntax.

Training Cross Lingual Parser

The multilingual setting in UUParser enables dependency parsing for multiple languages by training a single model across multiple UD treebanks. However, unlike multilingual language models UUParser requires separate word embeddings for each language. Cross-lingual transfer is achieved through shared BiLSTM parameters and treebank embeddings, allowing syntactic structures from high-resource languages (e.g., Hindi, Tamil) to improve parsing for low-resource languages such as Sinhala.

According to Vania et al. (2019), finetuning on the target language increases performance, this can be done by freezing the lower BiLSTM layers to retain cross-lingual knowledge. Higher BiLSTM layers and the Multi Layer Perceptron (MLP) classifier are updated using Sinhala-specific treebanks. This ensures that the parser adapts to Sinhala syntax while benefiting from multilingual training.

Further as suggested by Ghiffari et al. (2023) hierarchical transfer learning is conducted. The proxy language was selected on the basis of the language similarity analysis.

4.2 Grammar Error Detection Model

In this phase of our study, we address our second and third research questions concerning the incorporation of dependency parsing for GED) and its comparative performance against other features. We develop a GED model through a systematic approach. First, we collect and annotate sentences to create a synthetic error dataset specifically for Sinhala. This dataset serves as the foundation for our experiments. Next, we train separate models using different feature sets. We evaluate models trained exclusively with surface-level features (such as n-grams, POS tags, and morphological features) and compare their performance against models trained solely with dependency-based features derived from our parsing work. Based on these results, we then develop a combined feature model that integrates the most effective elements from both feature categories.

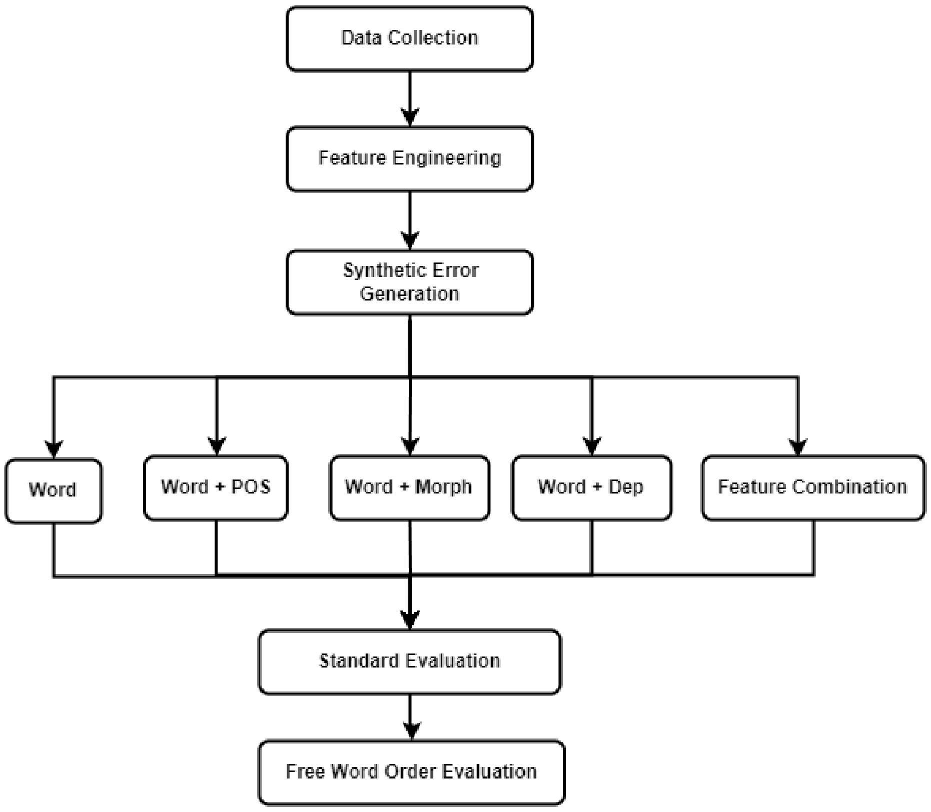


Figure 4.6: Methodology for training **GE!** model

4.2.1 Data Collection & Annotation

Data Collection

For synthetic error generation well formed grammatically correct sentences are required which are in written form since spoken Sinhala does not consider any grammatical structure. For example the sentence "මම ගමට ගියා" is correct in spoken form although should be corrected to "මම ගමට ගියෙමි" when converting to written form.

In addition to the datasets collected for the task of dependency parsing additional sentences were collected from other sources as given in table 4.1. They were extracted in the same manner as explained in section 4.1.1. Our dependency parser was trained on relatively short sentences (< 12 words). Therefore to reduce error propagation when annotating longer sentences, we remove sentences having more than 12 words.

Source	**No. of Sentences**
Grade 04 Sinhala Text Book	450
Grade 05 Sinhala Text Book	115
Grade 06 Sinhala Text Book	240
Grammar Book Indika Sampath (2013)	310
Sinhala Short Sentences Dataset	2367

Table 4.1: Evaluation Scores

Data Annotation

For our work, we require annotating sentences with POS tags, morphology tags, and dependency labels and arcs. Although we were able to use an LLM to annotate POS and morphology in our previous task, this approach is infeasible for our current dataset of approximately 3,400 sentences. Therefore, we utilized publicly available tools for annotation. The annotation procedure is detailed below.

- POS : Annotated using a deep learning based POS tagger which had 90% accuracy. The format was converted to Universal POS annotation scheme in order to facilitate our dependency parser.

- Morphology : Morphological information is extracted at the word level using the deep learning based morphological analyzer developed by Ekanayaka et al. (2023) where each token is analyzed into a sequence of morphemes representing its lemma, part-of-speech category, number, gender, case, and other grammatical features.

- Dependency : After annotating sentences with POS and morphological features they were fed to the dependency parser trained in the previous stage to obtain dependency labels and arcs.

4.2.2 Synthetic Error Generation

Since an annotated grammar error dataset is not available for Sinhala and creating one would consume a substantial time and effort we thought of inducing synthetic errors into grammatically correct sentences. We selected five error classes by considering the available literature on synthetic error generation and the grammatical features of Sinhala (Sonawane et al. (2020), Ponce et al. (2023), Hossain et al. (2023)).

Verb Deletion

This process involves programmatically removing the main verb from a grammatically correct sentence to produce an ungrammatical variant. Given that verbs in Sinhala carry essential grammatical functions such as tense, aspect, modality, and agreement, their deletion results in structurally incomplete or incoherent sentences.
For example, the sentence ඔහු පාසලට ගියා (He went to school) may be transformed into ඔහු පාසලට, which lacks the predicate and therefore violates basic syntactic well-formedness.
The identification of the main verb is performed using dependency parsing, where the verb is typically labeled as the root or a core predicate (VERB tag and dependency relation such as root or ccomp). Once identified, the token is deleted while preserving the rest of the syntactic structure.

Noun Deletion

This technique involves the removal of one or more core noun arguments from a grammatically correct sentence, typically subjects or objects.
For example, the sentence "මම පොතක් කියවන්නේ" (I am reading a book) may be transformed into "මම කියවන්නේ".
In implementation, noun candidates for deletion are identified based on their universal POS tag (NOUN, PROPN, or PRON) and their dependency role (e.g., nsubj, obj, iobj, obl). A random subset of these nouns is selected for deletion under controlled stochastic rules to ensure variability across generated samples.

Word Repetition

This type of error involves repeating a content or function word within a sentence, often resulting in unnatural or ungrammatical constructions. For example, the sentence "ඔහු පාසලට ගියා" (He went to school) might be transformed into "ඔහු ඔහු පාසලට ගියා" or "පාසලට පාසලට ගියා". Such repetition errors are commonly observed in both learner generated text and spontaneous writing, often arising due to hesitation, editing artifacts, or speech-to-text noise.

Word Swapping

This method involves exchanging the positions of two or more words in a sentence, thereby altering the linear structure while keeping the lexical content unchanged. For example, the grammatical sentence "මම ඔහුට හිංසා කළෙමි" (I attacked him) may be transformed into the following forms depending on the selected word pair.

1. මම හිංසා ඔහුට කළෙමි.

2. ඔහුට මම හිංසා කළෙමි.

Since Sinhala permits some flexibility in word order due to its morphologically rich nature, such permutations may often still result in grammatically correct sentences as in 2 above. But it would be beneficial to evaluate how the model would perform in identifying incorrect word orders from correct ones.
In implementation, the system first identifies valid candidate pairs of tokens that can be swapped. These typically include combinations like subject-object, adjective-noun, or adverb-verb, based on their dependency labels and POS tags. Once a pair is selected, their positions are swapped within the sentence. To introduce diversity, the selection is stochastic, and each sentence is transformed into at most one swapped variant.

Subject-Verb Agreement Errors

Subject-verb agreement errors occur when the subject and verb in a sentence do not match in features such as number, gender, tense or person.

Ex:-
Correct: "අපි බත් කමු."
Incorrect: "අපි බත් කමි." (Number Disagreement)

In this study, the subject and verb are identified through an analysis of dependency relationships and their corresponding part-of-speech (POS) tags. A significant flaw in the previous studies on Sinhala GED is such as Pabasara & Jayalal (2020) and Fernando P. A. S. (2020) is that they have use rule based methods to identify corresponding subject and verb. But dependency parsing is the ideal candidate for this task.
Following identification, candidate words are replaced with randomly selected words that exhibit distinct morphological features. The replacement words are sourced from the UCSC morphology dataset, ensuring that the substituted terms differ significantly in their morphological characteristics from the original candidates.

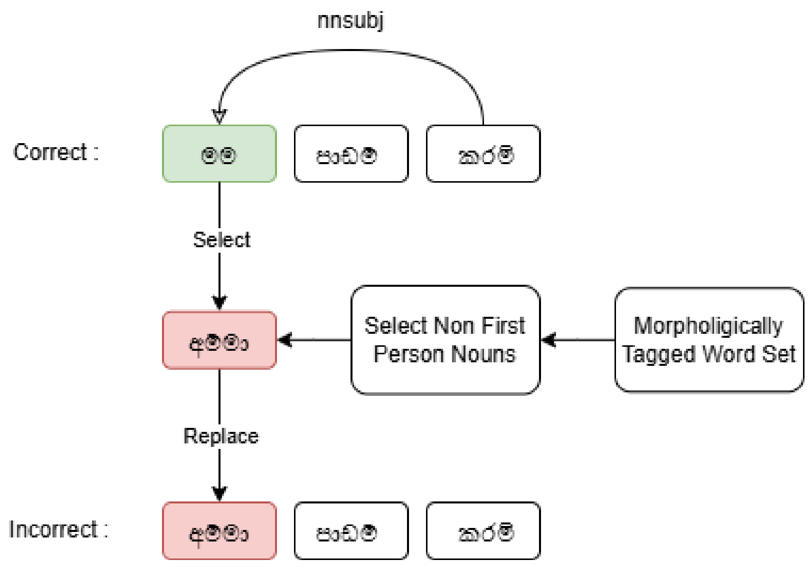


Figure 4.7: S-V Agreement Error Generation

4.2.3 Feature Engineering

This section explains how features were extracted and converted to formats suitable for machine learning tasks. This step is essential for addressing our research question on how we can complement features for optimal Grammar Error Detection (GED) in Sinhala.
In recent studies, researchers have explored various methods for transforming sentences into feature-rich sequences. Latest studies utilize neural network models that utilize word embeddings to represent contextual information.
For our work, we implemented a feature extraction pipeline that encompasses both surface-level and dependency-based features. Our approach balances traditional linguistic features with modern embedding representations to capture the complex morphosyntactic patterns of Sinhala.

Lexical(Word) Features

For this representin lexical featurse we use the Sinhala 300-dimensional embedding model trained in the prevous phase of our study. This provide rich contextual and syntactic information which will be beneficial for low resource tasks. The embeddings are integrated into the feature engineering pipeline by representing each token in the input n-grams as a 300-dimensional vector.

Algorithm 1 Extract Embedding n-grams for Grammar Error Detection

```
Require: Corpus of sentences $S$, n-gram size $n$, embedding model $E$ (300d)
Ensure: Set of embedding n-grams $\mathcal{G}$
 1: $\mathcal{G} \leftarrow \emptyset$
 2: for all sentence $s \in S$ do
 3:     Tokenize $s$ into tokens: $w_1, w_2, \ldots, w_m$
 4:     for $i \leftarrow 1$ to $m-n+1$ do
 5:         Initialize an empty list for the n-gram: $g \leftarrow []$
 6:         for $j \leftarrow i$ to $i+n-1$ do
 7:             Retrieve embedding: $v \leftarrow E(w_j)$
 8:             Append $v$ to $g$
 9:         end for
10:         Optionally, aggregate the embeddings in $g$ (e.g., via averaging or concatena-
    tion)
11:         $\mathcal{G} \leftarrow \mathcal{G} \cup \{g\}$
12:     end for
13: end for
14: return $\mathcal{G}$
```

When evaluating combined features, this representation concatenated with other linguistic features, such as part-of-speech tags, morphological attributes, and dependency relations, to form a comprehensive set of features.

Lexical + POS Features

For respresenting POS features with word embeddings for feature-based grammar error detection in Sinhala. The Lex-Pos sequences will be transformed into a feature vector using a modified version of the Word Embedding One-Hot Encoding technique from Agarwal et al. (2020). The following is an algorithmic depiction of the process. In the study done by Agarwal

Algorithm 2 Lex-Pos Sequence

```
 1: Input: Sequence
 2: Output: Lex_Pos_Seq
 3: Calculate pos-tag sequence of Sentence (Pos_Seq);
 4: Initialize the list of problematic tags (Prob_Tags);
 5: Initialize empty Lex_Pos sequence (Lex_Pos_Seq);
 6: for each Pos_tag in Pos_Seq do
 7:     if $Pos\_tag \in Prob\_Tags$ then
 8:         Append (Pos_tag + Linguistic information) to Lex_Pos_Seq;
 9:     else
10:         Append Pos_tag to Lex_Pos_Seq;
11:     end if
12: end for
```

et al. (2020) they, identified problematic POS tags that cause over generalization and added lexical Features to provide context. But in Sinhala all POS tags are problematic, since grammatical relations can be varied extensively by changing the morphological features keeping the

POS category unchanged therefore in our study we included words with corresponding pos tags for all lex-pos sequences.

Lexical + Morphological Features

Incorporating morphological information provides a deeper understanding of grammatical roles, especially in contexts where syntax alone may be ambiguous or unreliable. Prior work has shown that explicitly modeling morphology improves performance on tasks such as part-of-speech tagging, parsing, and even error detection in morphologically rich languages (Cotterell & Heigold 2017).
The table 5.4 shows how the word සංවේදීත් is analysed using the morphological analyzer. These

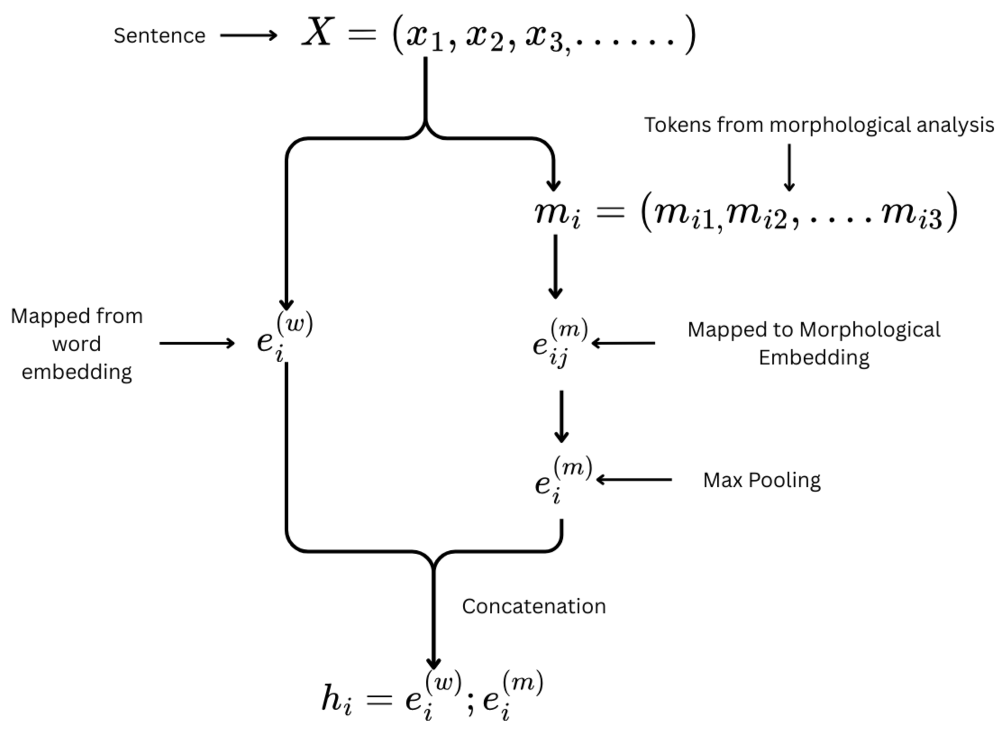
Token	Morphological Analysis
සංවේදීත්	සංවේදී (Root): `_AJ (Adjective)`
	ත්`:+CJ` (Conjunction)
	සංවේදී`_AJ +CJ`

Table 4.2: Example of morphological analysis

analyses are then transformed into structured feature representations suitable for learning algorithms.

The feature engineering pipeline includes the following steps:

- Tokenization: Sentences are first tokenized using a Sinhala-specific tokenizer capable of handling script-specific nuances, punctuation, and compound forms.
- Morphological Analysis: Each token is passed through the Sinhala morphological analyzer, which returns one or more possible analyses. Each analysis consists of a list of morphemes (e.g., root, suffixes, inflections).
- Normalization: To ensure consistency, only the first valid analysis for each word is retained. The morphemes from this analysis are then flattened into a single token by joining them with a separator (+), converting lists like ['N+RT', ' :+SG', ' :+DF'] into N+RT+ :+SG+ :+DF.
- Vocabulary Construction: A morphological vocabulary is constructed from the training corpus by counting the frequency of unique morph strings. Tokens with frequency above a threshold (e.g., min_freq = 1) are retained.
- Index Encoding: Each flattened morphological representation is mapped to a unique integer ID using the vocabulary. The result is a fixed-length integer sequence per sentence, representing the morphology-driven features of each word.



Sentence
$X = (x_1, x_2, x_{3,} \ldots\ldots)$
Tokens from morphological analysis
$m_i = (m_{i1,} m_{i2}, \ldots . m_{i3})$
Mapped from word embedding
$e_i^{(w)}$
$e_{ij}^{(m)}$
Mapped to Morphological Embedding
$e_i^{(m)}$
Max Pooling
Concatenation
$h_i = e_i^{(w)}; e_i^{(m)}$

Figure 4.8: Feature representation of Morphological Features

The figure 4.8 represents how the final feature representation is obtained after the above steps. These combined embeddings $\mathbf{h}_1, \ldots, \mathbf{h}_n$ are then used as input to the downstream model.

Lexical + Dependency Features

For dependency features, we propose a novel approach that uses dependency pair encoding combined with pre-trained word embeddings. The central hypothesis is that integrating syntactic dependency information with rich semantic representations will enhance the model's ability to detect subtle grammatical errors in Sinhala.
Dependency parsing is applied to each sentence in the corpus to extract dependency pair tuples that combine a dependency relation with the associated lexical items. For each dependency connection, we use pre-trained word embeddings to encode the lexical components.
Specifically, there are two feature engineering strategies derived from dependency pairs:

- Syntax-Aware Representation: Instead of representing each word solely by its pre-trained embedding, each word is augmented with its dependency label. This creates a syntax-aware representation that combines semantic and syntactic information.

- Syntactic N-Grams: In contrast, syntactic n-grams are constructed by grouping dependency pairs based on the structure of the dependency tree rather than their surface order. This approach captures nonadjacent but syntactically related pairs (e.g., subject–verb or verb–object relations), offering a more robust representation of grammatical structure (Vania and Lopez, 2017; Belinkov and Glass, 2017).

For example consider the following sentence:

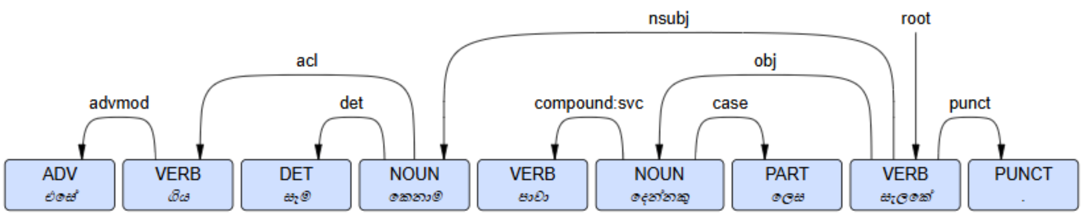


Figure 4.9: Dependency Parsed Sentence

From this parse, a syntactic 3-gram can be extracted by following a dependency path that spans three nodes. For example, consider the following path in the dependency tree:

$$\text{ගිය} \xrightarrow{\text{acl}} \text{කෙනාම} \xrightarrow{\text{nsubj}} \text{සැලකේ}$$

Here,

– ගිය (token 2) is connected to කෙනාම (token 4) with the dependency relation $r_1 =$ acl, and

– කෙනාම (token 4) is connected to සැලකේ (token 8) with the dependency relation $r_2 =$ nsubj.

We represent each word by its pre-trained embedding and each dependency relation by its own embedding. The syntactic 3-gram feature vector is then constructed by concatenating these embeddings. Formally, let:

– $\mathbf{E}(w) \in R^{d_w}$ denote the embedding of word w,

– $\mathbf{D}(r) \in R^{d_d}$ denote the embedding of dependency relation r, and

– $\oplus$ denote the concatenation operator.

The syntactic 3-gram feature vector for the path is given by:

$$\mathbf{s}_{\text{3-gram}} = \mathbf{E}(\text{ගිය}) \oplus \mathbf{D}(\text{acl}) \oplus \mathbf{E}(\text{කෙනාම}) \oplus \mathbf{D}(\text{nsubj}) \oplus \mathbf{E}(\text{සැලකේ}).$$

This representation integrates both semantic information (from the word embeddings) and syntactic structure (from the dependency label embeddings) over a hierarchical path that spans three levels of the dependency tree.

These two feature representations will be used in combination providing a hybrid representation that has both local and global context.

Feature Combinations

Finally, we will train a model that combines the most effective features as determined by our experimental results. Rather than pre-determining which features to include, we will let empirical performance guide our feature selection. This approach will assist us in addressing our second research question comprehensively by identifying which combination of features optimizes GED performance for Sinhala.

To implement this combined model, we will design a flexible multi-input neural architecture that can accommodate varying feature sets. Each token in a sentence will be represented by parallel feature streams that may include: pre-trained FastText embeddings (300 dimensions), POS embeddings, morphological embeddings (from a trainable embedding layer), and dependency relation embeddings. These token-level vectors can be concatenated to form a composite representation based on our experimental findings.

The specific combination of features and architectural details will be finalized based on the performance of our isolated feature models, ensuring we build upon empirically validated components rather than predetermined assumptions.

4.2.4 Model Training

We adopt a Bidirectional Long Short-Term Memory (BiLSTM) network since they are well suited for capturing context-dependent information in natural language, as they process input sequences in both forward and backward directions, effectively modeling long-range dependencies. Before transformers, BiLSTMs demonstrated SOA results on sequence classification tasks (Vaswani et al. (2017), Devlin et al. (2019)).
Further to support our decision, Sinhala has long range dependencies such as subject verb agreement therefore more contextual information could benefit greatly. This can also complement dependency features therefore providing rich context. A max-pooling or final hidden state is then used to generate a sentence-level representation, which is passed through a dense softmax layer for classification over six grammatical error types. The model is trained using categorical cross-entropy loss with the Adam optimizer, and early stopping is applied to prevent overfitting. The figure 4.10 represents the overall architecture of the deep learning model for GED. Slight changes will occuron on the basis of the features used, but the underlying framework remains the same.

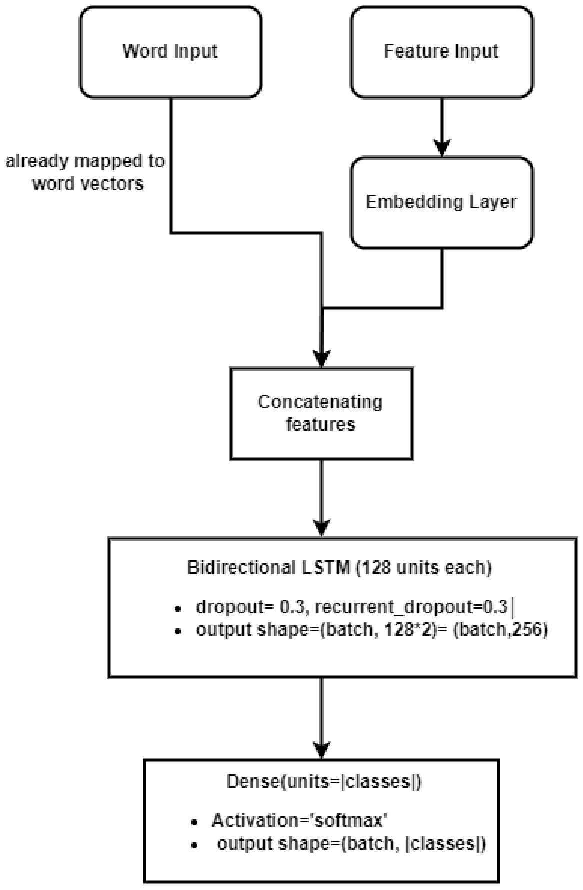

Word Input
Feature Input
already mapped to word vectors
Embedding Layer
Concatenating features
Bidirectional LSTM (128 units each)
• dropout= 0.3, recurrent_dropout=0.3
• output shape=(batch, 128*2)= (batch,256)
Dense(units=|classes|)
• Activation='softmax'
• output shape=(batch, |classes|)

Figure 4.10: Overall Model Architecture

Chapter 5

Experiments and Results

This chapter will discuss the results of the experimentation done in this research. First section of the chapter will introduce the evaluation metrics used throughout the research and the second section will discuss the results obtained from the experimentation and the analysis on them

5.1 Evaluation Metrics

5.1.1 Dependency Parsing Evaluation Metrics

This section discusses the primary metrics used to evaluate dependency parsing systems. Considering existing studies conducted on dependency parsing two main evaluation metrics can be highlighted.

Unlabeled Attachment Score (UAS)

Unlabeled Attachment Score represents the fundamental measure of parsing accuracy. UAS calculates the percentage of tokens that are assigned the correct head or parent node, irrespective of the specific dependency label Nivre (2003). The metric is computed as:

$$\text{UAS} = \left(\frac{\text{Number of correctly attached tokens}}{\text{Total number of tokens}} \right) \times 100\% \tag{5.1}$$

Labeled Attachment Score (LAS)

Labeled Attachment Score represents a more nuanced evaluation metric. LAS not only assesses the correct head assignment but also requires the precise dependency relation label to be correct (Manning & Schutze 1999). The calculation is:

$$\text{LAS} = \left(\frac{\text{Number of tokens with correct head and correct dependency label}}{\text{Total number of tokens}} \right) \times 100\% \tag{5.2}$$

LAS offers a more stringent assessment of parsing performance, capturing both structural and semantic precision.

5.1.2 GED Model Evaluation Metrics

To evaluate the performance of the proposed model, three commonly used classification metrics are used: Precision, recall, and F1 score. These metrics provide insights into the model's accuracy in correctly identifying instances, particularly in tasks involving imbalanced data sets or multi-label classification.
Let:

- TP = True Positives (correctly predicted positive instances)
- FP = False Positives (incorrectly predicted positive instances)
- FN = False Negatives (incorrectly predicted negative instances)

Precision

Precision measures the proportion of correctly predicted positive observations to the total predicted positives. It is a measure of a classifier's exactness.

$$\text{Precision} = \frac{TP}{TP+FP}$$

Recall

Recall (also known as Sensitivity or True Positive Rate) measures the proportion of correctly predicted positive observations to all actual positives. Reflects the completeness of a classifier.

$$\text{Recall} = \frac{TP}{TP+FN}$$

F1-Score

The F1-score is the harmonic mean of precision and recall. It balances the two metrics, especially when there is an uneven class distribution.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

5.2 Pleliminary Experiments

This section outlines the experiments conducted to drive the research approach forward.

5.2.1 Dependency Parser Training Framework Selection

For selecting a suitable parser the two most popular available parsers which were the Standford Parser and the uuparser were evaluated.

Experimental Setup

Both parsers optionally required word embedding models but since it is a comparison the vectors were allowed to randomly initialized. Both the parsers were given the UD dataset and allowed to train for 30 epochs. The graph based setting of the uuparser was selected since the underlying architecture of the Stanford parser is similar to a graph based parser. The best models were compared.

Results and Conclusion

The UAS scores after the training were as below

Evaluation Metric	uuparser	Standford Parser
UAS	**53.85**	24.6

Table 5.1: Results on UD dataset with 100 sentences

As shown in table 5.1 it is clear that the uuparser performs best for Sinhala and this parser has features for training multi-lingual models and other customizable featuers which makes it ideal for further experiments. Also it was noted that the superior performs of the uuparser is due it's surpport for non-projective dependency relationships which is a common occurrence in free word order languages (de Lhoneux, Stymne & Nivre 2017).

5.2.2 Word Embedding Model Selection

The objective of this experiment is to train a word embedding model on a modern dataset and through comparative evaluation select a model that is suitable for further experiments.

Experimental Setup

The following figure 5.1 illustrates the steps for training the embedding model. For training the embedding model the dataset from HuggingFace was used. This dataset contained approximately 403 million tokens. In the pre-processing step it was thought not to remove stop words, since stop words carry important information when it comes to syntax.

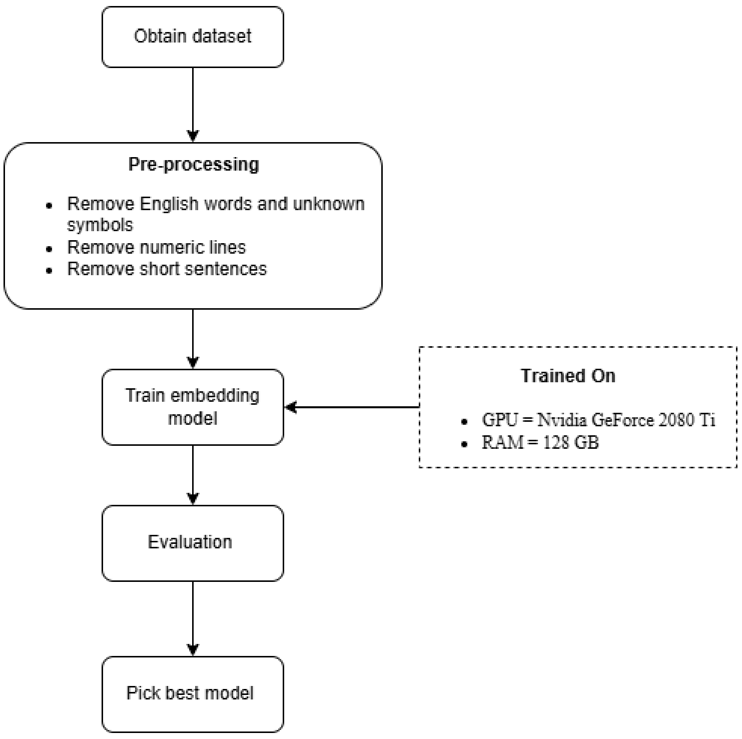


Figure 5.1: Experimental setup for training word embedding model

Results and Conclusion

For evaluation the intrinsic and extrinsic evaluation framework provided by Lakmal et al. (2020) was used. The intrinsic evaluation task consisted of using evaluating analogies for Sinhala such as encyclopedic, derivational and inflectional while the extrinsic evaluation was conducted using WordSim 353 which is a relatedness evaluation metric. The following tables depict the results.

Category	Our model	Lakmal et al. (2020)	Pre-Trained FastText
Derivational	**36.7**	28.7	20.6
Encyclopedic	**32.1**	17.1	6.8
Inflectional	**47.0**	34.7	20.9

Table 5.2: Intrinsic Evaluation Scores

Category	Our model	Lakmal et al. (2020)	Pre-Trained FastText
Relatedness	0.4534	**0.644**	0.611

Table 5.3: Extrinsic Evaluation Scores

As seen in the tables our model performed well on the analogy tasks but performed sub-optimally for the relatedness task. But when provided words, our model provide fairly accurate synonyms. The low score for relatedness is possibly due to the under representation of the words used for the similarity measure in our dataset. Since the model provided fairly better results than the existing models, we used it for further experiments.

5.3 Dependency Parsing Experiments

5.3.1 Datasets

The evaluation was carried out on two datasets. The details of the datasets are mentioned in table 5.4. An evaluation on the grammar book data alone was required since the GED dataset

Dataset	No. of sentences	Sources
Grammar Book	298	Sinhala Text Books & Grammar Books
Combined	398	Grammar Book + UD Sentences

Table 5.4: Dataset for crosslingual training base model

contains mostly sentences from grammar text books while the UD sentences were from various sources such as newspapers, articles and novels. Therefore we had to find out how well the model would fit to our task of grammar error detection.
For the task of direct and hierarchal transfer learning the following datasets available through the UD framework were used primarily.

Dataset	No. of sentences
Hindi	16649 sentences
Tamil	398 sentences

Table 5.5: Dataset for crosslingual training

5.3.2 Experiment 1 : Language Selection

Transfer learning is more effective if the source and target language pairs are similar. Therefore this experiment will provide insights into what languages are suitable for conducting transfer learning experiments.

Experimental Setup

The similarity measure will consider Lexical, POS tag distribution and dependency label distribution similarity. The following languages were considered based on their region of origin and the availability of UD resources. All the experiments were conducted on UD datasets of

Language	No. of Sentences
Hindi	16649
Gujarati	187
Marathi	466
Tamil	600
Telugu	1328
Urdu	5130

Table 5.6: Multingual Tree Bank Datasets

the respective languages, the metrics mentioned above were analyzed by comparing respective UD datasets. For Lexical similarity measure transliterated texts were considered which were already available in the UD annotation framework to overcome differences in writing script.

Results and Discussion

This section represents the results of the analysis conducted to figure out the most suitable language to apply transfer learning for dependency parsing.

Language	Word Overlap (%)	Similar Words	Subword Unit Overlap (%)
Hindi	**46 words, 9.20%**	**13639 pairs**	**317 units, 61.20%**
Gujarati	1 word, 0.20%	1219 pairs	0 units, 0.00%
Marathi	-	2023 pairs	124 units, 23.94%
Tamil	2 words, 0.40%	1667 pairs	108 units, 20.85%
Telugu	2 words, 0.40%	1815 pairs	147 units, 28.38%
Urdu	3 words, 0.60%	9473 pairs	43 units, 8.30%

Table 5.7: Comparison of Word and Subword Overlap Across Languages

Language	Cosine Similarity with Sinhala
Telugu	**0.9892**
Marathi	0.9845
Tamil	0.9821
Gujarati	0.9751
Hindi	0.9736
Urdu	0.9684

Table 5.8: Dependency Label Distribution Similarity with Sinhala

Language	Cosine Similarity with Sinhala
Telugu	0.**8988**
Tamil	0.8930
Marathi	0.8747
Urdu	0.8568
Hindi	0.8568
Gujarati	0.8493

Table 5.9: POS Tag Distribution Similarity with Sinhala

By analyzing the above metrics, it can be seen that Hindi is the best candidate considering its lexical similarity and the size of the dataset, but Telugu shows better similarity in terms of dependency label distribution and POS Tag distribution, but since the Hindi dataset was approximately 15 times larger, Hindi was considered for the preliminary experiments for transfer learning.

5.3.3 Experiment 2: Comparing Graph-Based vs. Transition-Based Parsing Approaches for Sinhala

There are two dependency parsing strategies namely,

1. Graph Based Parsing
2. Transition Based Parsing

Each strategy has it's own strengths and weaknesses. To asses which method is suitable for Sinhala we train a model in each mode separately on the combined dataset. This model will also act as our base model for further evaluations.

Experimental Setup

The model was trained on both graph-based and transition-based settings of the uuparser provided by de Lhoneux, Shao, Basirat, Kiperwasser, Stymne, Goldberg & Nivre (2017). The training split of the dataset is as follows.

- Train : 275 sentences (70%)
- Test : 47 sentences (15%)
- Validation : 47 sentences (15%)

The following settings and options were used for this and further experiments :

- Learning Rate : 0.001 (default)
- Epochs : 30 (default)
- Batch Size : 32 (default)
- Embeddings Dimension : 300 (Aligned with embedding model we trained)
- Other options : Non projective, Non filtered vectors

The non-projective setting allows overlapping arcs which can be seen in free word order languages. Further non filtered vectors in order to handle OOV tokens.

Model Type	UAS	LAS
Graph-Based	**67.74**	**52.35**
Transition-Based	66.67	50.99

Table 5.10: Results of baseline dependency parser

Results & Conclusion

Table 5.10 shows the obtained results from the baseline model on the combined dataset. The graph-based model outperforms the transition-based model by 1.07 points on UAS and 1.36 points on LAS. This indicates that the graph-based approach has a slight edge for parsing Sinhala. While both parsers utilize contextual information through their underlying neural architectures, the graph-based parser's advantage likely stems from its ability to optimize the entire dependency tree globally. This global optimization approach considers all possible dependency arcs simultaneously, which may better capture the long-distance dependencies common in Sinhala's flexible word order. In contrast, the transition-based parser makes sequential decisions based on parser state, and despite the rich contextual representations provided by its BiLSTM component. Following these results we use the graph-based model for further experiments.

5.3.4 Experiment 3: Applying Data Augmentation

In this experiment we evaluate the effectiveness of various data augmentation techniques to find out which method suits the best for training a dependency parser for Sinhala.

Experimental Setup

The training splits for each augmentation techniques is depicted in table 5.11. A consistent data set size was maintained for fair evaluation.

Augmentation Type	Train (Sentences)	Test & Validation (Sentences)
Rotating	904	47
Cropping	887	47
Nonce	960	47

Table 5.11: Dataset summary for data augmentation

Results & Conclusion

The results of tree morphing and nonce sentence generation are depicted in the table 5.15. Sen-

Morph Type	UAS	LAS
Rotating	**68.80**	52.78
Cropping	64.43	51.44
Nonce	68.59	**53.42**

Table 5.12: Results of data augmentation

tence rotation has provided best results in terms of UAS scores, because Sinhala is a free word order, SOV language, "rotating" sentences exposes the parser to many valid word order variants. That boosts its ability to recover the correct structure (UAS), though it only modestly improves label accuracy (LAS).
Cropping removes chunks of a sentence, breaking long distance dependencies and erasing critical context. In Sinhala's richly inflected syntax, this loss of context hurts both attachment (UAS) and labeling (LAS).
Nonce augmentation replaces low frequency tokens with synthetic placeholders while preserving full sentence structure. This diversifies the parser's vocabulary exposure without disturbing the dependency structure - producing nearly the same UAS as Rotating but the highest LAS, because the model becomes better at assigning correct dependency labels to varied lexical items.

5.3.5 Experiment 4: Combination Augmentation Techniques

Experimental Setup

From the conclusion of Experiment 3 we can see that Rotating and Cropping techniques provide better results than the baseline model. Therefore to find out their performance on combination we joined the two datasets and trained a single model. Our hypothesis was that since Rotating boosts UAS and Nonce sentences improve LAS, the combination would improve both scores.

Results & Conclusion

The results of the model trained by combining both Nonce and Rotated datasets is mentioned in table 5.13. The combination seems to degrade performance significantly. Combining augmentations might lead to an overly noisy training set. Although each technique individually improves robustness, too much variability can confuse the model during training, resulting in lower overall structural accuracy (UAS).

Augemented Types	**UAS**	**LAS**
Nonce + Rotating	66.24	53.63

Table 5.13: Results after combining dataset

5.3.6 Experiment 5: Transfer Learning

Following the conclusion of experiment 1, the Hindi language was selected for transfer learning. The evaluation was conducted on the Sinhala test dataset.

Experimental Setup

The Hindi-HDTB dataset was used along with the Sinhala combined dataset. The cross-lingual parser was trained using the multilingual setting in the uuparser. The multilingual setting requires both source and target embedding models. For Hindi the publicly available FastText embedding model was selected. After the training cross lingual model fine tuning was applied on the sinhala dataset. The dataset splits are as below.

- Train : 13306 sentences (Hindi)
- Test & Validation : 47 sentences (Sinhala)

Results & Conclusion

Model Type	UAS	LAS
Hindi-Sinhala	68.59	52.14
Hindi-Sinhala + Finetuning	68.80	51.71

Table 5.14: Results on transfer learning

Fine-tuning did not improve performance as expected, this might be due to the fact that the model learned more generalizable patterns from hindi sentences. Another concern about the result is that the performance did not improve compared to the baseline.

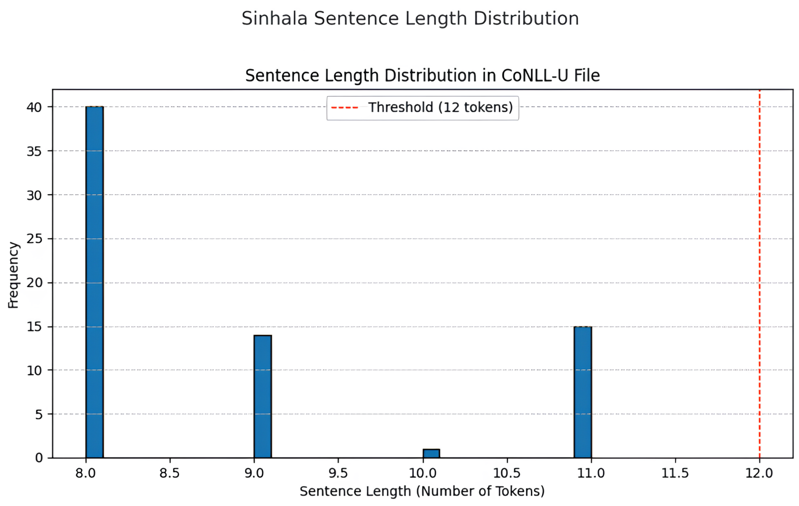
5.3.7 Experiment 6: Transfer Learning With Short Sentences

Since experiment 7 did not improve the LAS scores as expected we analyzed the sentence lengths in the Hindi dataset. Comparing The average sentence length of the Hindi dataset was 21 words while the Sinhala dataset had a maximum sentence length of 12 words. The vast difference between sentence lengths in the source and target datasets might have effected performance. Therefore a separate experiment was carried out by removing long sentences from the hindi dataset.

Experimental Setup

Removing sentences having length greater than 12 would significantly reduce amount of training examples adversely effecting the transfer learning process. Therefore to retain balance between high resource nature and sentence length similarity all sentences having length 21 or below were retained for this experiment. The training split is as follows

- Train : 7290 sentences (Hindi)



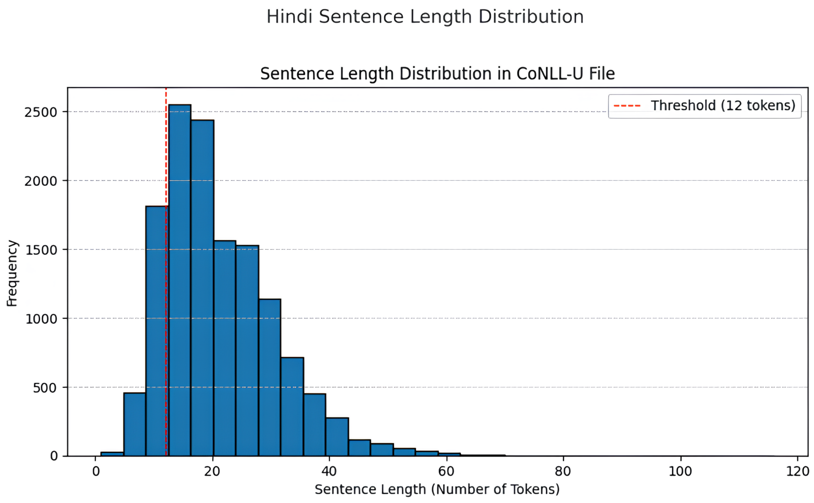


Figure 5.2: Sentence Length Analysis

- Test & Validation : 47 sentences (Sinhala)

Results & Conclusion

The results are shown below. This experiment resulted in substantial improvement in both

Model Type	**UAS**	**LAS**
Hindi - Sinhala	68.59	52.14
Hindi (Short Sentences) - Sinhala	**70.40**	**54.75**

Table 5.15: Results after aligning sentence lengths

scores. Since the Hindi short sentences dataset was more aligned with the Sinhala dataset the model might have avoided unwanted patterns.

5.3.8 Experiment 7: Hierarchical Transfer Learning

Further experiments were carried out to test the effectiveness of proxy languages in improving language disparity.

Experimental Setup

Telugu was chosen as the proxy language based on the results on experiment 1. Since Telugu is also closely related to Hindi, the hypothesis was that this would be the ideal language to reduce language dissimilarity. The multilingual setting in the uuparser allows for training more than 2 languages at once. Similar to Hindi the publicly available FastText word embedding model for Telugu was used here. For Hindi the short sentences dataset was considered since it provided better results in experiment 6. The training splits are as follows.

- Train : 7290 sentences (Hindi) , 1051 (Telugu), 275 (Sinhala)
- Test & Validation : 47 sentences (Sinhala)

Results & Conclusion

The results are shown in table 5.16. The performance has dropped significantly. This shows

Model Type	**UAS**	**LAS**
Hindi - Telugu - Sinhala	63.03	44.44

Table 5.16: Results on hierarchical transfer learning

that the choice of Telugu as a proxy language has added more ambiguity to the dataset making it difficult to generalize across the 3 languages.

5.3.9 Experiment 8: Transfer Learning With Data Augmentation

To test the effectiveness of data augmentation for transfer learning, Nonce sentence dataset was used since it provided the best average results in both scores in experiment 3.

Experiment Setup

The model was trained using the Hindi dataset containing short sentences. The training splits are as follows.

- Train : 7290 sentences (Hindi) , 960 sentences (Sinhala)
- Test & Validation : 47 sentences (Sinhala)

Model Type	**UAS**	**LAS**
Nonce + Transfer	71.37	55.42

Table 5.17: Results of applying transfer learning from augmented dataset

Results & Conclusion

The results are mentioned in table 5.13. There is a slight improvement in both UAS and LAS scores compared to the cross-lingual model trained without augmentation.

5.3.10 Evaluation On The Grammar Book Dataset

The best performing model was which the cross-lingual model trained by applying Nonce augmentation along with Short Hindi sentences with an average dev score of 63.4 was the tested against the Grammar Book Dataset to justify the use of the dependency parser for our grammar error detection task. The results are mentioned in table 5.18. The model seems to perform exceptionally for well formed Sinhala sentences in written form. This could be due to the class imbalance of Sentences from Grammar books vs sentences from News Sources. Further the UD dataset contained sentences with varied structures compared to the simple structures in the grammar book dataset.

Model Type	**UAS**	**LAS**
Best Model	92.71	88.34

Table 5.18: Results on grammar book dataset

5.3.11 Analysis of Results

Graph vs Transition

In our test set of 47 sentences (1,382 total tokens), the graph based parser attains a slightly higher overall score (UAS 67.74 vs. 66.67; LAS 52.35 vs. 50.99). Breaking down by dependency distance reveals that both parsers excel at short range attachments (token–head distance ≤2), yet performance degrades substantially for long range arcs:

- Short range dependencies (308 tokens)
  - Graph based UAS/LAS = 75.65% / 61.04%;
  - Transition based UAS/LAS = 74.03% / 59.09%.
  - Here, both parsers correctly capture most local relations (e.g., determiners, nominal modifiers, local adjuncts) with a small edge to the graph model.
- Long range dependencies (114 tokens; distance > 2)
  - Graph based UAS/LAS = 48.25% / 35.96%;
  - Transition□based UAS/LAS = 45.61% / 32.46%.
  - Long distance modifiers such as temporal or locative adjuncts and non projective links are notably harder. The graph parser's global scoring enables it to recover about 3 pp more of those arcs than the transition model.

The above scenario can be described using an example in the test dataset. Consider the sentence "වයස අවුරුදු හයේ දී ඇයට ගමෙන් නගරයට එන්නට සිදු විය". The output of both models is depicted in the diagram 5.3. The correct green ticks indicate the correct long range dependencies that the graph based parser has predicted compared to the transition based parser. These results suggest that for free word order Sinhala, models benefit from global inference (as in graph-based algorithms) to better handle non-local dependencies.

Transfer Learning (Hindi - Sinhala)

The initial cross-lingual model (Hindi-Sinhala) achieved UAS and LAS scores of 68.59% and 52.14% respectively, as shown in Table 5.7. When fine-tuning was applied, the UAS marginally improved to 68.90%, but the LAS unexpectedly decreased to 51.71%. This counterintuitive outcome suggests that the model may have learned more generalizable patterns from Hindi sentences that did not transfer effectively to Sinhala during fine-tuning. Further literature suggests that transfer learning usually enhances LAS score though in our scenario it did not result in a significant gain. When investigatint this effect, we found that the performance increased when long sentences were removed. The substantial difference in syntactic complexity likely contributed to the poor transfer learning performance initially.
This approach yielded significant improvements, with the Hindi (Short Sentences)-Sinhala model achieving UAS and LAS scores of 70.40% and 54.75% respectively, as detailed in Table 5.8.

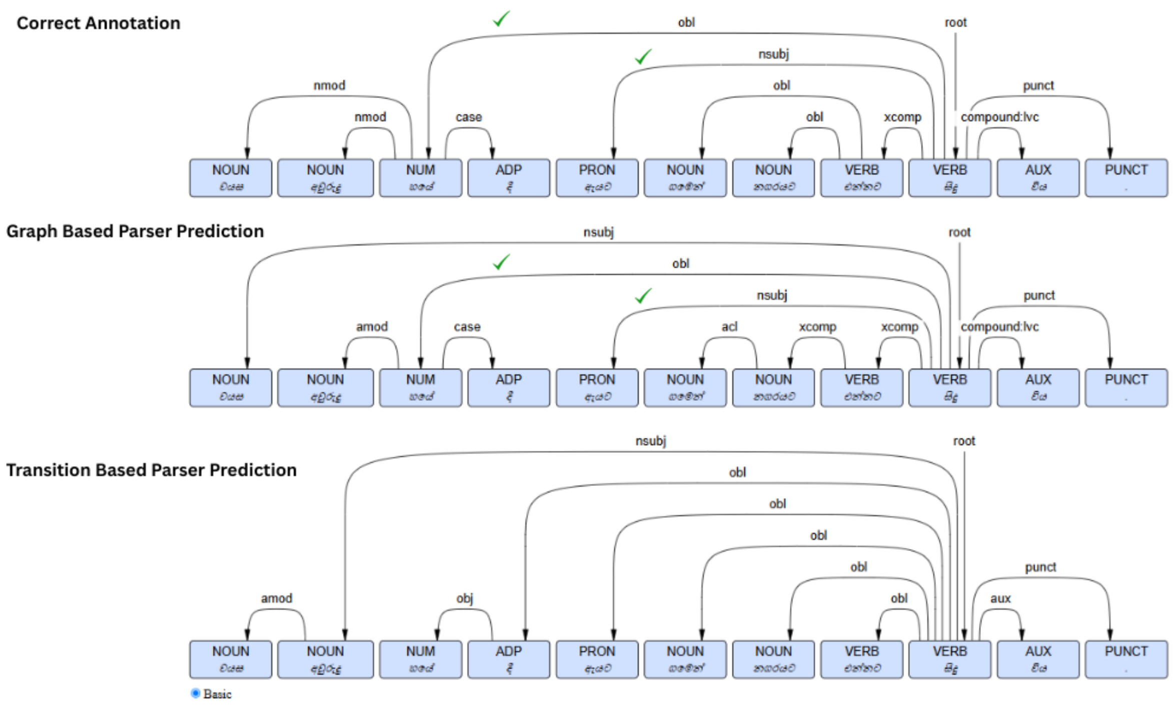


Figure 5.3: Graph vs Transition parser output

This represents a substantial improvement of 1.81 percentage points in UAS and 2.61 percentage points in LAS compared to the original transfer learning model.

Proxy Language Experiments

We further explored the use of proxy languages to potentially bridge the language disparity between Hindi and Sinhala. Specifically, Telugu was introduced as an intermediary language given its linguistic proximity to Sinhala. However, contrary to our expectations, this approach resulted in a significant performance drop, with UAS and LAS scores of 63.03% and 44.44% respectively (Table 5.9).
This substantial decline indicates that introducing Telugu as a proxy language added more ambiguity to the model, complicating its ability to generalize across the three languages. The increased complexity of managing three different linguistic structures appears to have overwhelmed the model's capacity to identify consistent cross-lingual patterns.

Data Augmentation

In our data augmentation experiments for Sinhala dependency parsing, sentence rotation achieved the highest UAS (68.80%) by leveraging Sinhala's free word order characteristics, while nonce sentence generation produced the best LAS (53.42%) by maintaining intact dependency structures while diversifying vocabulary. Sentence cropping performed poorest (UAS: 64.43%, LAS: 51.44%) as it eliminated critical contextual information necessary for parsing Sinhala's richly inflected grammar. Interestingly, combining augmentation techniques (nonce and rotation) resulted in decreased UAS (66.24%) compared to individual approaches while slightly improving LAS (53.63%), suggesting that excessive data variation may introduce noise during training. These findings indicate that while individual augmentation strategies enhance parser robustness in specific ways, their combination creates too much variability, potentially confusing the model and resulting in suboptimal performance for low-resource dependency parsing of Sinhala.

Analysis on best model

To gain deeper insight into the parser's performance, a confusion matrix was generated comparing the predicted dependency labels against the gold annotations for the best model which is the cross-lingual model trained with nonce sentences according to the evaluation in section. The confusion matrix 5.4 reveals that while a strong diagonal trend indicates generally accurate label predictions, several off-diagonal regions suggest consistent misclassifications between specific syntactic roles. Key Observations:

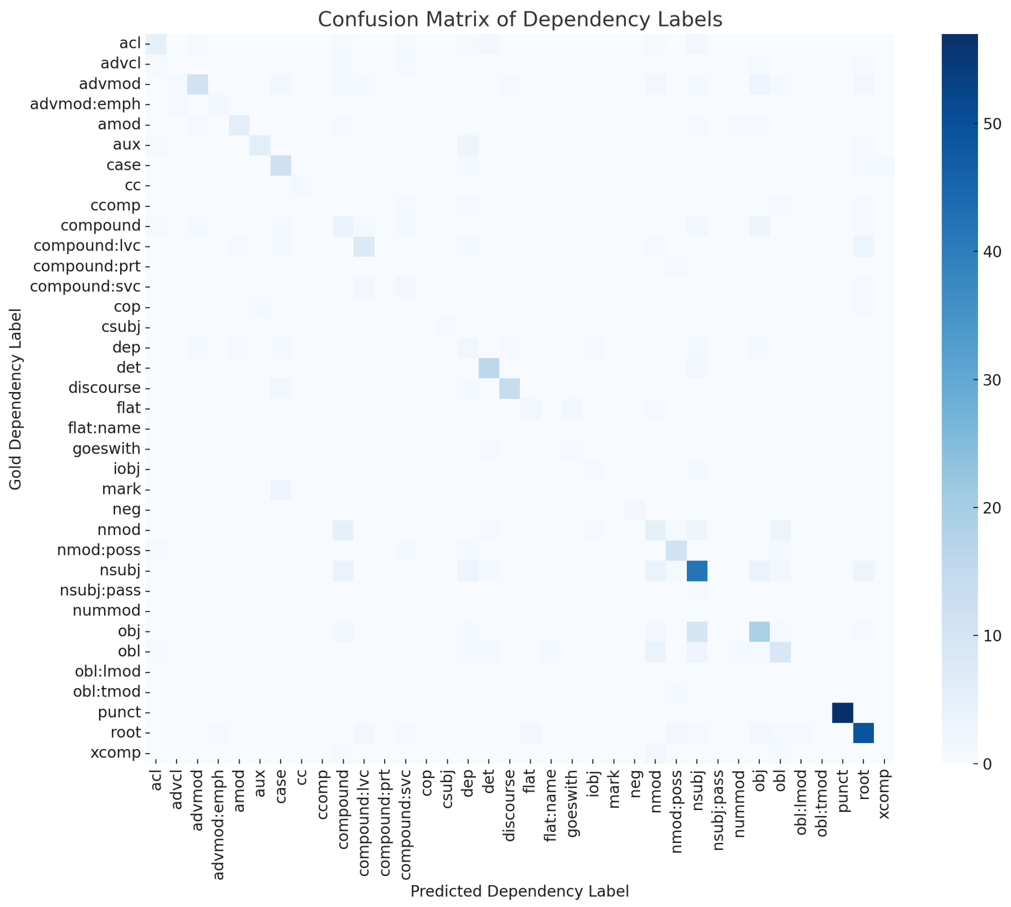


Figure 5.4: Confusion matrix of dependency parsing model

- Modifier Confusions: A significant portion of the confusion involves nmod, obl, and advmod relations. This is not unexpected, as these labels often represent similar or overlapping syntactic functions especially in morphologically rich and free word order languages like Sinhala. The boundaries between nominal modifiers (nmod), oblique arguments (obl), and adverbial modifiers (advmod) can be subtle, and even linguists may disagree in edge cases.

- Subject vs. Object Confusion: There are also noticeable confusions between nsubj and obj. This points to the difficulty of the model in distinguishing the role of the core argument in complex or less canonical sentence structures, especially when case markers or positional cues are ambiguous or missing.
- Compound Constructions: Errors were also frequent between compound:lvc and compound:svc, indicating challenges in handling Sinhala's verb compounding phenomena. These distinctions rely on subtle lexical cues and syntactic conventions that may be underrepresented in the training data.
- Low-Resource Effects:Labels with fewer instances (e.g., iobj, ccomp, xcomp) had more scattered confusion, which is expected given the data sparsity. This again highlights the importance of balanced annotation coverage in the dataset.

These patterns suggest that, while the model is generally effective in identifying coarse-grained syntactic relations, it struggles with fine-grained distinctions between similar labels, particularly in cases involving long-distance dependencies, syntactic ambiguity, or complex compound structures.

5.3.12 Summary of Dependency Parsing Experiments

The results suggest that graph-based parsers outperformed transition-based approaches, achieving better accuracy on both unlabeled (UAS) and labeled (LAS) attachment scores by more effectively capturing the complex, non-projective dependencies characteristic of Sinhala syntax. To address data scarcity, we tested several augmentation strategies with mixed results. Sentence rotation proved most effective for overall structure recognition, while nonce sentence generation excelled at improving label accuracy by exposing the parser to diverse lexical contexts.
The most significant breakthrough came from cross-lingual transfer learning using Hindi as a source language. The key insight was matching sentence length distributions between languages when Hindi training data was filtered to shorter sentences that better matched Sinhala's typical length, performance jumped substantially. Combining the optimized Hindi transfer with nonce augmentation yielded the best overall model.
The final system achieved impressive results on held-out test data, with accuracy scores reaching over 90% on well-formed written Sinhala text.

5.4 Grammar Error Detection Experiments

5.4.1 Datasets

Synthetic GED Dataset

As training data for the grammar error classification model the GED dataset generated synthetically using strategies discussed in section 4.2.2 will be used. Per class distribution is depicted in table 5.19.

Table 5.19: Per class frequencies of GED dataset

Class	**No.of Sentences**
Correct	3949
SV Agreement	2794
Noun Deletion	1219
Verb Deletion	1548
Word Repetition	416
Word Swapping	765

Evaluation Datasets

The models will be evaluated upon free-word order structures to identify which model effectively captures the structural flexibility of Sinhala.

- Dataset used by Liyanage et al. (2012) : This dataset contains 200 correct sentences covering sentence various structures of Sinhala.
- Free word order dataset : Contains 4 sentences and possible free word order patterns, totaling 25 sentences. For example the sentence "මිනිසුන් පැමිණිය හොත් අපි රැස්වීම පවත්වමු." will have the following corresponding free word order structures.
  - අපි මිනිසුන් පැමිණිය හොත් රැස්වීම පවත්වමු.
  - අපි රැස්වීම මිනිසුන් පැමිණිය හොත් පවත්වමු.
  - රැස්වීම මිනිසුන් පැමිණිය හොත් අපි පවත්වමු.

5.4.2 Experiment 1 : Word Embedding Only Model

The purpose of this experiment is to establish a baseline to evaluate which surface level features will complement with dependency features for the final architecture of the dependency based GED model.

Experimental Setup

As explained in our research design section we use a BiLSTM model to train our GED model. This model was trained using only lexical features obtained using a word embedding model.

The model was trained for 10 epochs with a batch size of 32. Dropout was set to 0.3 to avoid overfitting. The same settings were used for further experiments for consistent evaluation. The training split was 70:15:15

Results & Conclusion

The performance of the baseline model is summarized in 5.20. The overall classification accuracy achieved by the model was 0.75, demonstrating a reasonable capacity to distinguish between grammatical and ungrammatical constructs using only lexical representations.

Table 5.20: Classification report of the word only model

Label	**Precision**	**Recall**	**F1-score**	**Support**
SVAgreement	0.83	0.82	0.82	563
correct	0.67	0.86	0.75	769
noun_deletion	0.88	0.80	0.84	261
verb_deletion	0.80	0.75	0.78	301
word_repetition	0.86	0.07	0.13	87
word_swapping	0.80	0.32	0.46	158
Overall Accuracy	0.75 (on 2139 instances)			

The results show that while the word embedding-only model provides a strong starting point, particularly for more structurally grounded errors such as subject-verb agreement and deletions, it struggles with more nuanced or surface-level errors such as repetition and swapping. These findings motivate the incorporation of additional syntactic features or contextual modeling (e.g., POS tags, dependency relations, or sequence models) to improve detection sensitivity for subtle grammatical anomalies.

5.4.3 Experiment 2 : Word Embedding + POS Model

To improve upon the baseline model, part-of-speech (POS) tag features were integrated alongside word embeddings.

Experimental Setup

The settings for the BiLSTM model used in experiment 1 was used here as well. The POS features are converted to embeddings instead of one hot encoding for efficiency. The POS Embedding dimension was set to 32. The features are fed to the model in terms of lexical and POS sequences.

Results & Conclusion

The classification report for this model is shown in 5.21. The inclusion of POS tags provides syntactic context that complements the lexical information from word embeddings.
The overall classification accuracy improved to 0.78, a +3% gain compared to the word-only model, demonstrating that syntactic cues significantly improve the model's ability to detect grammatical patterns and errors.

Table 5.21: Classification report of the word + POS model

Label	**Precision**	**Recall**	**F1-score**	**Support**
SVAgreement	0.89	0.80	0.84	563
correct	0.69	0.83	0.75	769
noun_deletion	0.89	0.87	0.88	261
verb_deletion	0.86	0.86	0.86	301
word_repetition	0.44	0.17	0.25	87
word_swapping	0.74	0.51	0.60	158
Overall Accuracy	0.78 (on 2139 instances)			

The inclusion of POS features led to consistent improvements across nearly all error types, with notable gains in detecting deletion and word-order-related errors. The results highlight the importance of syntactic signals in grammar error detection, especially for morphologically rich languages. While classes like word repetition remain challenging, the upward trend in recall demonstrates progress.

5.4.4 Experiment 3 : Word Embedding + Morph Model

This experiment incorporated morphological features alongside word embeddings to assess whether fine-grained lexical information (such as case, number, gender, tense, etc.) would enhance the model's ability to detect grammatical errors.

Experimental Setup

First a morphological vocabulary will be created by turning each word's sequence of morphemes into a single string (e.g. ['root','+','suffix'] to "root+suffix"), then counts how often each unique string appears, and then assigns each one a unique integer ID. Then during the data preprocessing stage the morphological features of the word will be mapped to this vocabulary and the unique ID will be fed to the embedding layer of 32 dimensions.

Results & Conclusion

Surprisingly, despite the addition of morphology-based signals, the model's overall accuracy dropped slightly to 0.74, suggesting that morphology alone does not consistently enhance performance when used in isolation from syntactic features like POS tags.

Table 5.22: Classification report of the Word + Morph model

Label	**Precision**	**Recall**	**F1-score**	**Support**
SVAgreement	0.87	0.81	0.84	563
correct	0.64	0.80	0.71	769
noun_deletion	0.89	0.80	0.85	261
verb_deletion	0.78	0.70	0.74	301
word_repetition	0.33	0.08	0.13	87
word_swapping	0.59	0.46	0.51	158
Overall Accuracy	0.74 (on 2139 instances)			

Based on the results it seems that the morphological features were not able improve performance compared to the word only model. This is possibly because embedding models such as the fastText model trained by us already provide rich morphological context.

5.4.5 Experiment 4 : Word Embedding + Dependency Model

In this experiment we test our primary objective, which is to develop a GED model using dependency parsing. In this experiment only lexical and dependency features are considered.

Experimental Setup

The dependency relations are fed in terms of syntax-aware and syntactic n-gram representations as suggested by Tezcan et al. (2017). Each token's dependency relation label (e.g. "nsubj", "obj") and built a small vocabulary that maps every unique relation to an integer index. At model-time, those relation indices are fed into a embedding layer, which learns a dense vector for each relation type.
For the n-gram component, we slide a small window of 'dependency path' of size n through the parse tree of each sentence. Because a sentence can yield multiple such paths, we take the element-wise average of all valid path vectors to form one fixed-length n-gram feature per sentence. The dimension of the dependency embedding layer was set to 50 and the n-gram embedding was 1000.

Results & Conclusion

Table 5.23 presents the results of the model that integrates word embeddings with dependency-based representations. The model achieved an overall accuracy of 0.74. Notably, it yielded a strong F1-score of 0.85 for the Subject-Verb Agreement (SVA) class, and similarly high performance for noun deletion (0.83) and correct sentences (0.72).

Table 5.23: Classification report of the dependency based representation

Label	**Precision**	**Recall**	**F1-score**	**Support**
SVAgreement	0.87	0.78	0.82	563
correct	0.65	0.79	0.71	769
noun_deletion	0.84	0.84	0.84	261
verb_deletion	0.82	0.74	0.78	301
word_repetition	0.40	0.11	0.18	87
word_swapping	0.53	0.49	0.51	158
Overall Accuracy	0.74 (on 2139 instances)			

These results suggest that dependency structures gives a solid foundation for catching missing or mislinked elements (nouns, verbs, agreement), but alone it struggles with purely lexical or sequential errors like repetition. Combining dependency features with POS and lexical signals might yield the better coverage.

5.4.6 Experiment 5 : Feature Combination

Our research aims to develop a robust GED model that leverages dependency features. Experiment 4 hinted that combining dependency features with other surface-level feature would yield optimal results on the synthetic error dataset. Based on previous experimental findings, we found that only POS features improve performance on the baseline. Therefore POS features were combined with dependency features (along with word embeddings).

Experimental Setup

Consistent with our previous experimental design, we incorporate each feature set into the BiLSTM model as separate embedding layers: word embeddings, POS embeddings, dependency embeddings, and dependency n-gram embeddings. The model is trained for 30 epochs using a batch size of 32 and a learning rate of 0.001.

Results & Conclusion

Overall, the chosen combination of features strikes a strong balance between all types of error and yields robust performance. Compared to all other experiments this version improves upon almost all error classes.

Table 5.24: Classification report of the combined model

Label	**Precision**	**Recall**	**F1-score**	**Support**
SVAgreement	0.92	0.92	0.92	563
correct	0.71	0.81	0.76	769
noun_deletion	0.85	0.86	0.86	261
verb_deletion	0.84	0.84	0.84	301
word_repetition	0.35	0.07	0.11	87
word_swapping	0.64	0.46	0.55	158
Overall Accuracy	0.80 (on 2139 instances)			

5.4.7 Evaluation on Standard Data Sets

To evaluate the parsing performance each model thay are tested against 200 correct sentences used to evaluate the rule based parser developed by Liyanage et al. (2012). The results are as below.

Table 5.25: Summary of Misclassified Correct Sentences by Model

Model	Most Misclassified Class	Accuracy
Word Only	SVAgreement	0.80
Word + POS	verb_deletion	0.82
Word + Morph Based	word_swapping	80
Word + Dependency	SVAgreement	**0.89**
Feature Combination	verb_deletion	0.82
Liyanage et al. (2012)	-	0.60

As expected the dependency based representation seems to perform best, this is due to capability of dependency features to handle flexible word order. All the models perform better that the rule based parser developed by Liyanage et al. (2012) which was only able to parse 118 sentences out of the 200 sentences.
To evaluate the effectiveness of dependency parsing in capturing free word order structures, we evaluated 3 models namely the models trained with Word + POS, Word + Dependency and the Combined representations on the free-word order sentence dataset.

Table 5.26: Results on free word order structures

Model	Most Misclassified Class	Accuracy
Word + POS	SV Agreement	0.36
Word + Dependency	SV Agreement	**0.64**
Feature Combination	SV Agreement	**0.64**

Both the dependency based and combined representation model perform well against flexible structures while the word-pos model performs poorly although having exceptional performance on the synthetic error dataset. This shows how only surface level features are not capable of facilitating flexible syntax structures.

5.4.8 Analysis of Results

Performance on general error types

The experimental results reveal clear distinctions between the strengths of dependency-based features and those of surface-level features such as POS tags and morphology, particularly when addressing different types of grammatical errors.

Surface-level representations namely word embeddings combined with POS tags proved effective for detecting lexical and syntactic patterns that are sequentially or positionally predictable. The Word + POS model delivered high performance on classes such as noun deletion and verb deletion, where syntactic roles are often tightly linked to positional cues and POS transitions. It also achieved the highest overall accuracy (0.78) among the isolated feature models, confirming that POS tags provide strong local context for classifying well-structured grammatical errors.

In contrast, dependency-based models which encode hierarchical relationships between words irrespective of word order demonstrated superior robustness across varied sentence structures. The dependency-based models performed better on the evaluation dataset of 200 correct sentences, where the Word + Dependency model achieved 0.89 accuracy, the highest among all models. Its ability to correctly parse a diverse set of sentence structures many of which do not conform to conventional structures depicts its generalizability.

Nevertheless, dependency features alone showed limitations when handling surface level anomalies such as word repetition and word swapping. These errors often leave dependency structures intact and are better detected through surface patterns or token-level statistics. As such, while dependency representations are essential for modeling flexible or hierarchical grammar, they are insufficient on their own for capturing all error types.

The combined feature model, which integrates both dependency and surface-level features, achieved the best overall performance (0.80 accuracy on test data, 0.64 on free word order). It preserved the syntactic flexibility of dependency parsing while compensating for its limitations in lexical pattern recognition through embeddings and POS features.

Performance on free word order structures

The Lex-POS model, despite its high accuracy on standard test data, failed to correctly classify majority of the FWO sentences. All 15 misclassified outputs from this model were predicted as Subject-Verb Agreement (SVAgreement) errors. This suggests that the model heavily relies on familiar surface-level word orders learned during training and cannot handle syntactic reordering.

The Dependency model exhibited improved generalization. It correctly classified 16 out of 25 FWO sentences, achieving an accuracy of 64%. Its ability to capture grammatical relations beyond word position enabled it to correctly interpret several reordered variants. However, it still produced errors in structurally valid sentences, including misclassifications such as word swapping and SV Agreement, suggesting sensitivity to ambiguous structures or parsing limitations in more complex re-orderings. The misclassified sentences are as follows.

- මිනිසුන් පැමිණිය හොත් අපි රැස්වීම පවත්වමු . : SV Agreement
- අපි මිනිසුන් පැමිණිය හොත් රැස්වීම පවත්වමු . : SV Agreement
- රැස්වීම මිනිසුන් පැමිණිය හොත් අපි පවත්වමු . : SV Agreement
- ළමයකු බසයට නගිද්දී කොන්දොස්තර කෑ ගැසී ය . : SV Agreement
- කොන්දොස්තර ළමයකු බසයට නගිද්දී කෑ ගැසී ය . : Word Repetition
- බසයට ළමයකු නගිද්දී කොන්දොස්තර කෑ ගැසී ය . : SV Agreement
- රැස්වීමට මිනිස්සු බොහෝ දෙනෙක් පැමිණියහ . : Word Swapping
- රැස්වීමට බොහෝ දෙනෙක් මිනිස්සු පැමිණියහ . : Word Swapping
- නළුවෙක් වේදිකාවට ආවේ නිළියක හා ය . : Word Swapping

Looking into these misclassifications a clear pattern can be seen. Whenever the subject and object are misplaced the model classifies them as subject verb agreement errors. Further structures such as "රැස්වීමට බොහෝ දෙනෙක් මිනිස්සු පැමිණියහ" are uncommon and is not often used in written or spoken form therefore the model reasonably classified them as word misplacement errors. The Combined Feature model achieved similar performance. This model only seems to struggle with FOW structures of the sentences "මිනිසුන් පැමිණිය හොත් අපි රැස්වීම පවත්වමු" and "ළමයකු බසයට නගිද්දී කොන්දොස්තර කෑ ගැසී ය" and correctly classified all the other FOW structures. A special scenario here is that the combined model only misclassified them into SV Agreement errors, suggesting the surface level features such as POS and Morphology might have failed to capture the subject verb dependencies due to the deviation from SOV order.
These results highlight several key points:

- Models relying solely on surface-level features such as POS tags or Morphology fail to generalize under word order variation.
- Dependency-based features enable structural generalization and perform substantially better on flexible syntax.
- The Combined model achieves the best balance, correctly interpreting a wide range of free word order constructions while performing well against conventional structures.

In conclusion, this evaluation underscores the necessity of syntactic representations, particularly dependency parsing, in grammar error detection tasks for morphologically rich and flexible word order languages like Sinhala.

5.4.9 Summary of Grammar Error Detection Experiments

The experiments show that while a simple BiLSTM using only pre-trained word embeddings can already catch many agreement and deletion errors, it struggles with surface anomalies like word repetition and swapping. Adding POS embeddings yields consistent gains across most error types, confirming that part-of-speech information helps the model understand local grammatical structure. In contrast, explicitly learned morphological embeddings overlap too much with what FastText already encodes and do not boost performance. Introducing dependency-based features and syntactic n-grams enables the model to handle Sinhala's flexible word order more effectively, but these hierarchical cues alone are not enough to resolve lexical oddities. Ultimately, the strongest and most robust error detector combines word, POS, and dependency representations: it achieves the highest overall accuracy, excels at subject-verb agreement and deletion errors, and—unlike the surface-only models—still performs respectably on reordered sentences. This hybrid approach therefore offers the best balance for grammar error detection in a free-word-order, morphologically rich language with limited annotated data.

Chapter 6

Conclusion

This dissertation is about evaluating the effectiveness of dependency parsing for detecting grammatical errors in Sinhala sentences under low resource settings. This study was divided into two main phases, in the first phase experiments were conducted to identify which low resource strategies will be effective in building an accurate dependency parser while the second phase evaluated the effectiveness of dependency parsing compared to other syntactic features in handling variant word orders of Sinhala. The research was conducted under three major research questions :

RQ 1 : What low resource strategies such as data augmentation techniques and transfer learning enhance dependency parsing accuracy in a low-resource language like Sinhala ?

RQ 2 : How can dependency parsing be complemented with other feature representations for detecting grammatical errors in low-resourced, free word-order languages with minimal labeled data ?

RQ 3 : How effective is the proposed model for grammar error detection for conventional and non-conventional sentence structures in Sinhala ?

To address RQ1 we collected and annotated 400 sentences on dependency relations and applied various augmentation strategies. While each augmentation technique provided improvements individually the best performance boost was gained from nonce sentences. Varying words while keeping syntactic structure intact provided best results. Further we found that graph based models perform best for Sinhala due to its contextual knowledge (considers global state) compared to transition based parsers. Out of all strategies transfer learning improvement LAS scores the most. Due to the lexical similarity of Hindi to Sinhala the cross-lingual model was exposed more training examples improving labeling accuracy. An important discovery was that reducing the sentence length disparity between source and target datasets provide better generalization.

The RQ2 explored the effectiveness of dependency parsing compared to other feature representations for grammar error detection. The model trained using lexical and pos features performed best in these experiments. The combined model incorporating both surface and hierarchical level features exhibited the best f1 score across most error categories. This concludes that dependency features alone are not sufficient for detecting grammatical nuances, they should be complemented with other features to be effective.
To answer RQ3, the dependency based model was evaluated on a standard dataset (evaluation dataset used by (Liyanage et al. 2012)). The dependency based model was able to parse majority of the sentences with an accuracy of 89% compared to other models. Further to evaluate the capability of parsing free word order structures a dataset containing 25 vague free word order structures were used. Here as expected the dependency parsing model and the combined model both were able to handle free word order structures with equal accuracy.
The model trained by combining features showed the best balance between classifying grammar errors while handling FOW.

6.1 Limitations

Looking at the conducted study and the design decisions taken some key limitations can be observed:

- Both the dependency parser and the GED model was trained and evaluated on sentences with average lengths (< 12 words). The effectiveness of the models in handling long structures was not explored.
- The dependency parser was tested on a small dataset, a more comprehensive evaluation is required to understand the models capabilities against unseen data.
- The GED models was trained on synthetic data with only 5 error classes which does not represent actual errors made by Sinhala learners accurately. Further this model is not able to classify sentences with multiple errors at once.

6.2 Implications for Further Research

Building on our findings, several directions stand out for strengthening Sinhala grammar tools. First, enriching dependency parsers with explicit morphological information promises clearer syntactic analyses: for Turkish, injecting morphological tags into deep parser networks yielded significant UAS/LAS gains, as the model could leverage case and affix cues to resolve attachments in a Flexible Word Order (FWO) context Özcan & Hakkani-Tür (2021). Likewise, studies have shown that incrementally adding morphological features and lexicalization into data-driven parsers consistently enhances performance, confirming that suffixal and agreement markers offer vital signals for disambiguation Smith & Lee (2020).
Transformer-based transfer learning offers powerful contextual encodings that can be fine-tuned for low-resource languages. Experiments on Marathi demonstrated that mBERT and XLM-R outperform traditional parsers when adapted to a related target, achieving several points of UAS/LAS improvement over non-pretrained baselines Patil et al. (2022), Gupta et al. (2021). Evaluating dedicated Sinhala BERT variants or multilingual models in our dependency and error-detection pipelines could similarly lift performance, especially on rare constructions and long-distance dependencies.
Moving from sentence-level tagging to token-level error localization is critical for practical correction tools. Neural sequence labeling approaches—such as bidirectional LSTMs with subtoken representations—have successfully flagged individual error tokens across multiple languages, allowing fine-grained detection and span identification Chen & Toutanova (2019). Joint models that learn sentences and token levels together further enhance both classification accuracy and interpretability by regularizing token predictions with overall grammaticality ques Li & Sun (2020).
Expanding and curating error-annotated corpora via semi-automatic generation and active learning can vastly improve model generalization. Early work has shown that injecting synthetic errors guided by linguistic patterns and then validating them manually yields larger, more diverse corpora for training robust detectors Rana et al. (2018). Finally, integrating these models into interactive writing assistants demands user-centric evaluations. Studies of tools like Grammarly reveal that while immediate feedback and ease of use are valued, users also encounter false positives and limitations in contextual understanding. By pursuing these avenues future work can build comprehensive, accurate, and user-friendly grammar correction systems for Sinhala.

Bibliography

Agarwal, N., Wani, M. A. & Bours, P. (2020), `Lex-pos feature-based grammar error detection system for the english language', Electronics **9**(10), 1686.

Bojanowski, P., Grave, E., Joulin, A. & Mikolov, T. (2017), `Enriching word vectors with subword information'.

Chen, D. & Manning, C. D. (2014), A fast and accurate dependency parser using neural networks, in `Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)', Association for Computational Linguistics, pp. 740--750.

Chen, W. & Toutanova, K. (2019), Token□level neural sequence labeling for grammar error detection, in `Proceedings of the CoNLL 2019 Shared Task: Multilingual Grammatical Error Detection', Association for Computational Linguistics, pp. 112--122.

Cotterell, R. & Heigold, G. (2017), Cross-lingual character-level neural morphological tagging, in M. Palmer, R. Hwa & S. Riedel, eds, `Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing', Association for Computational Linguistics, Copenhagen, Denmark, pp. 748--759.
URL: https://aclanthology.org/D17-1078/

de Lhoneux, M., Shao, Y., Basirat, A., Kiperwasser, E., Stymne, S., Goldberg, Y. & Nivre, J. (2017), From raw text to Universal Dependencies - look, no tags!, in J. Hajic□ & D. Zeman, eds, `Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies', Association for Computational Linguistics, Vancouver, Canada, pp. 207--217.
URL: https://aclanthology.org/K17-3022/

de Lhoneux, M., Stymne, S. & Nivre, J. (2017), Arc-hybrid non-projective dependency parsing with a static-dynamic oracle.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2019), `Bert: Pre-training of deep bidirectional transformers for language understanding', arXiv preprint arXiv:1810.04805 .
URL: https://www.arxiv.org/pdf/1810.04805v1.pdf

Ekanayaka, Y., Pushpananda, R., Welgama, V. & Liyanage, C. (2023), `Applying deep learning for morphological analysis in the sinhala language', International Journal on Advances in ICT for Emerging Regions (ICTer) **16**, 1 -- 10.

Feng, S., Gangal, V., Wei, J. J., Sibanda, T.-H., Ruder, S., Neubig, G., Shu, R., Jiang, H., Ng, H. & Luo, Z. (2021), `A survey of data augmentation approaches for nlp', arXiv preprint arXiv:2105.03075 .

Fernando P. A. S., A. T. (2020), `Sinhala grammar checker using parts of speech tagging'.
URL: http://repo.lib.jfn.ac.lk/ujrr/handle/123456789/2134

Ghiffari, F. A. A., Alfina, I. & Azizah, K. (2023), Cross-lingual transfer learning for Javanese dependency parsing, in D. Li, R. Mahendra, Z. P. Tang, H. Jang, Y. Murawaki & D. F. Wong, eds, `Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics: Student Research Workshop', Association for Computational Linguistics, Nusa Dua, Bali, pp. 1--9.
URL: https://aclanthology.org/2023.ijcnlp-srw.1/

Gulordava, K., Bojanowski, P., Grave, E., Linzen, T. & Baroni, M. (2018), Colorless green recurrent networks dream hierarchically, in M. Walker, H. Ji & A. Stent, eds, `Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)', Association for Computational Linguistics, New Orleans, Louisiana, pp. 1195--1205.
URL: https://aclanthology.org/N18-1108/

Gupta, N., Singh, A. & Verma, P. (2021), Evaluating xlm□r for dependency parsing of low□resource languages, in `Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)', Association for Computational Linguistics, pp. 4561--4572.

Hettige, B. & Karunananda, A. S. (2011), `Computational model of grammar for english to sinhala machine translation', IEEE Xplore .

Hossain, N., Bijoy, M. H., Islam, S. & Shatabda, S. (2023), `Panini: A transformer based grammatical error correction method for bangla', Neural Computing and Applications .

Indika Sampath, B. (2013), වියරණ විවරණ 2 (පද විරාම ලක්ෂණ ඇතුළු ව්‍යාකරණාංග).

Kaneko, M., Mita, M., Kiyono, S., Suzuki, J. & Inui, K. (2020), Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction, in D. Jurafsky, J. Chai, N. Schluter & J. Tetreault, eds, `Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics', Association for Computational Linguistics, Online, pp. 4248--4254.
URL: https://aclanthology.org/2020.acl-main.391/

Karu□a□tilaka, □. E. (2012), සිංහල භාෂා ව්‍යාකරණය.

Lakmal, D., Ranathunga, S., Peramuna, S. & Herath, I. (2020), Word embedding evaluation for sinhala, Katubedda 10400, Sri Lanka.

Li, H. & Sun, M. (2020), Joint sentence and token level modeling for grammatical error detection, in `Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)', Association for Computational Linguistics, pp. 678--687.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. & Stoyanov, V. (2019), `Roberta: A robustly optimized bert pretraining approach', arXiv preprint arXiv:1907.11692 .
URL: https://arxiv.org/pdf/1907.11692

Liyanage, C., Pushpananda, R., Herath, D. L. & Weerasinghe, R. (2012), A computational grammar of sinhala, in `International Conference on Intelligent Text Processing and Computational Linguistics', Springer, pp. 188--200.

Manning, C. & Schutze, H. (1999), Foundations of Statistical Natural Language Processing, Foundations of Statistical Natural Language Processing, MIT Press.
URL: https://books.google.lk/books?id=YiFDxbEX3SUC

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C. & Joulin, A. (2018), Advances in pre-training distributed word representations, in `Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)'.

Náplava, J. & Straka, M. (2019), Grammatical error correction in low-resource scenarios, in W. Xu, A. Ritter, T. Baldwin & A. Rahimi, eds, `Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)', Association for Computational Linguistics, Hong Kong, China, pp. 346--356.
URL: https://aclanthology.org/D19-5545

Nivre, J. (2003), An efficient algorithm for projective dependency parsing, in `Proceedings of the 8th International Workshop on Parsing Technologies', Association for Computational Linguistics, pp. 149--160.

Özcan, A. & Hakkani-Tür, D. (2021), Incorporating morphological tags into deep dependency parser networks, in `Proceedings of the 16th International Conference on Parsing Technologies (IWPT)', Association for Computational Linguistics, pp. 123--134.

Pabasara, H. M. U. & Jayalal, S. (2020), Grammatical error detection and correction model for sinhala language sentences, in `2020 International Research Conference on Smart Computing and Systems Engineering (SCSE)', pp. 17--24.

Palma Gomez, F., Rozovskaya, A. & Roth, D. (2023), A low-resource approach to the grammatical error correction of Ukrainian, in M. Romanyshyn, ed., `Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP)', Association for Computational Linguistics, Dubrovnik, Croatia, pp. 114--120.
URL: https://aclanthology.org/2023.unlp-1.14

Patil, R., Kulkarni, M. & Sharma, P. (2022), Finetuning multilingual bert for marathi dependency parsing in lowresource settings, in `Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)', Association for Computational Linguistics, pp. 789--799.

Ponce, A. D. H., Jadie, J. S. A., Espiritu, P. E. A. & Cheng, C. (2023), Balarila: Deep learning for semantic grammar error correction in low-resource settings, in D. Wijaya, A. F. Aji, C. Vania, G. I. Winata & A. Purwarianti, eds, `Proceedings of the First Workshop in South East Asian Language Processing', Association for Computational Linguistics, Nusa Dua, Bali, Indonesia, pp. 21--29.
URL: https://aclanthology.org/2023.sealp-1.3

Rana, S., Joshi, A. & Patel, R. (2018), Synthetic error injection for semiautomatic corpus expansion in grammar correction, in `Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)', European Language Resources Association, pp. 3450--3457.

Rozovskaya, A. & Roth, D. (2021), How good (really) are grammatical error correction systems?, in P. Merlo, J. Tiedemann & R. Tsarfaty, eds, `Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume', Association for Computational Linguistics, Online, pp. 2686--2698.
URL: https://aclanthology.org/2021.eacl-main.231

Sahin, G. G. & Steedman, M. (2018), Data augmentation via dependency tree morphing for low-resource languages, in E. Riloff, D. Chiang, J. Hockenmaier & J. Tsujii, eds, `Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing', Association for Computational Linguistics, Brussels, Belgium, pp. 5004--5009.
URL: https://aclanthology.org/D18-1545

Smith, A., de Lhoneux, M., Nivre, J., Agi, . & Plank, B. (2018), An investigation of the interactions between pre-trained word embeddings, character models and pos tags in dependency parsing, in `Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing', Association for Computational Linguistics, pp. 2711--2720.

Smith, J. & Lee, K. (2020), `Datadriven lexicalization strategies for enhanced dependency parsing', Journal of Computational Linguistics **46**(2), 245--267.

Sonawane, A., Vishwakarma, S. K., Srivastava, B. & Kumar Singh, A. (2020), Generating inflectional errors for grammatical error correction in Hindi, in B. Shmueli & Y. J. Huang, eds, `Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop', Association for Computational Linguistics, Suzhou, China, pp. 165--171.
URL: https://aclanthology.org/2020.aacl-srw.24

Tezcan, A., Hoste, V. & Macken, L. (2017), `A neural network architecture for detecting grammatical errors in statistical machine translation', Prague Bulletin of Mathematical Linguistics **108**, 133--145.

Vania, C., Kementchedjhieva, Y., Søgaard, A. & Lopez, A. (2019), A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages, in K. Inui, J. Jiang, V. Ng & X. Wan, eds, `Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)', Association for Computational Linguistics, Hong Kong, China, pp. 1105--1116.
URL: https://aclanthology.org/D19-1102

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017), Attention is all you need, in `Neural Information Processing Systems'.
URL: https://api.semanticscholar.org/CorpusID:13756489

Wang, H., Yang, J. & Zhang, Y. (2019), `From genesis to creole language: Transfer learning for singlish universal dependencies parsing and pos tagging', ACM Transactions on Asian Low-Resource Language Information Processing **19**(1), 1--29.

Wang, Y., Wang, Y., Liu, J. & Liu, Z. (2020), `A comprehensive survey of grammar error correction'.

Winter, Y. (2001), `Review of "computing meaning: volume 1" by harry bunt and reinhard muskens. kluwer academic publishers 1999', Computational Linguistics - COLI .

Adjustment of Uppsala Parser For Fine-Tuning

Freezing Lower BiLSTM Layers

```python
def __init__(self,in_dim,out_dim,model,dropout_rate=None, freeze=False):
    self.dropout_rate = dropout_rate
    self.freeze = freeze
    self.surfaceBuilders = [dy.VanillaLSTMBuilder(1, in_dim, out_dim, model
   ),
                            dy.VanillaLSTMBuilder(1, in_dim, out_dim, model
   )]
    if freeze:
        # Disable gradient updates for frozen layers
        for builder in self.surfaceBuilders:
            builder.disable_updates()
```

Listing 1: Uuparser Modification

Final Combined Feature Grammar Error Detection Model Implementation

```python
    # Token-level inputs
    word_inputs = Input(shape=(max_seq_length,), dtype='int32', name='
   word_inputs')

    pos_inputs = Input(shape=(max_seq_length, pos_vocab_size), dtype='
   float32', name='pos_inputs')

    dep_inputs = Input(shape=(max_seq_length, dep_vocab_size), dtype='
   float32', name='dep_inputs')

    synth_inputs = Input(shape=(synth_vocab_size,), dtype='float32', name='
   synth_inputs')

    token_features = Concatenate(axis=-1)([word_embeddings, pos_inputs,
   dep_inputs])

    # BiLSTM encoder
    x = Bidirectional(LSTM(128, return_sequences=True))(token_features)
    x = Dropout(0.5)(x)
    x = Bidirectional(LSTM(64, return_sequences=False))(x)
    x = Dropout(0.5)(x)

    # Syntactic branch
    synth_branch = Dense(64, activation='relu', name='synth_dense')(
   synth_inputs)
```