SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V11I12P134

Volume 11 Issue 12, 375-385, December 2024

Original Article

Morphological and Syntactic Challenges in Malayalam: A Dependency Parsing Perspective

P.V. Ajusha[1], A.P. Ajees[2]

[1]*School of Information Science and Technology, Kannur University, Kerala, India.*
[2]*Department of Computer Science, Cochin University of Science and Technology, Kerala, India.*

[1]*Corresponding Author : ajusha321@outlook.com*

Received: 19 October 2024 Revised: 20 November 2024 Accepted: 18 December 2024 Published: 31 December 2024

Abstract – *Natural language processing is the area of study that focuses on how computers and human languages interact. Machine translation, sentiment analysis, semantic analysis, and text analysis are a few of them. The key natural language processing component is morphological analysis, which breaks words into their corresponding morphemes to determine their structure and meaning. Dependency parsing algorithms use morphological information to determine the syntactic structure of a sentence. This study evaluates the performance of various parsers, including Turbo parser, Lys-FASTPARSE, UU parser, and neural network based parser, to analyse dependency parsing methodologies used in the Malayalam language. The study evaluates the performance of these parsers in handling the difficulties and effectiveness of extensive morphological and syntactic features of Malayalam. Among these parsers, Lys-FASTPARSE performs better in LAS F1 score, MLAS score, and BLEX score, maintaining values of 56.60 and 48.58 before and after optimization. The neural network parser shows minor improvements in unlabelled attachment scores from 0.72 to 0.73 and labelled attachment scores from 0.46 to 0.47. With an LAS of 66.89% and UAS of 87.12%, the Turbo parser shows better results for baseline performance. The precision of 98.81% and recall of 88.42% in binned HEAD directions of the UU parser shows its performance in managing right direction dependencies. While lower, the parser's performance in managing left-direction and root dependencies still reflects its ability to navigate complex syntactic structures effectively. The results underscore the significance of tailored parsing techniques for morphologically rich languages like Malayalam and provide insights into optimizing parser performance for improved syntactic analysis.*

Keywords - *Neural network-based parser, Dependency parsing, Lys-FAST parser, UU parser, Transition based parsing.*

1. Introduction

Malayalam belongs to the South Dravidian language family and is an agglutinative language with rich inflectional morphology. Dependency parsing is a fundamental task in Natural Language Processing (NLP) that involves analyzing the grammatical structure of a sentence by identifying the relationships between words. In dependency parsing, the syntactic structure of a sentence is represented as a tree where each word is connected to a "head" word, forming a directed relationship known as a dependency [1]. The goal is to determine which words depend on others and the nature of these dependencies, such as subject-verb or object-verb relationships. The resulting dependency tree provides a compact and informative representation of the sentence's syntactic structure. Unlike phrase structure parsing, which represents sentence structure using nested phrases, dependency parsing focuses on binary relations between words. This makes it particularly useful for free or flexible word order languages, where dependencies between words are more informative than their linear sequence [2]. Dependency parse trees can be divided into projective and non-projective trees [3]. Figure 1 illustrates the structure of a sentence as represented by a dependency graph in projective dependency parsing. In the projective trees the edges do not cross each other and a word and its dependents can form a substring of the sentence, but in non-projective trees, there are crossing edges. Non-projective transition-based parsing has been actively explored in the last decade. Figure 2 depicts a dependency graph in non-projective dependency parsing, where syntactic dependencies between words can cross over each other, reflecting more complex sentence structures.

The success of neural networks and word embeddings for projective dependency parsing also encouraged research on neural nonprojective models 4]. In a projective dependency tree, every subtree's yield is a contiguous sentence substring. Identifying tagging issues and problems in annotation are closely connected to dependency parsing in several ways. These are crucial for improving the accuracy and efficiency of parsing systems. Dependency parsing relies heavily on accurate Part-of-Speech (POS) tags to determine the syntactic structure of sentences [5].

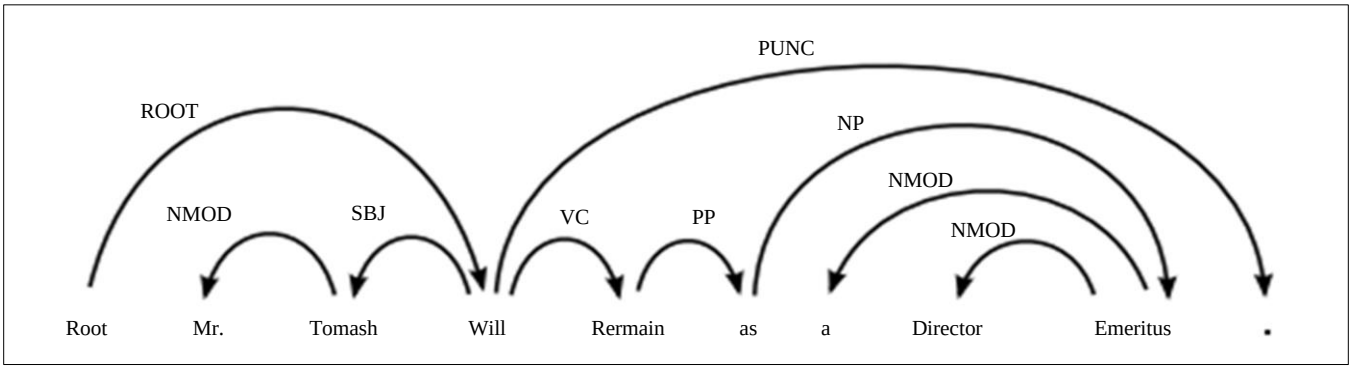

Fig. 1 Dependency graph in projective dependency parsing

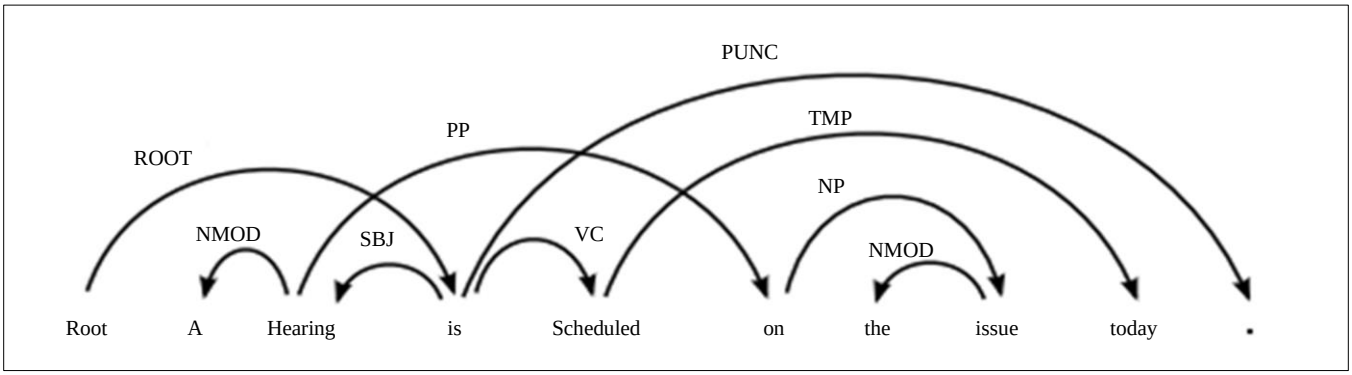

Fig. 2 Dependency graph in non-projective dependency parsing

Malayalam, a Dravidian language spoken predominantly in the Indian state of Kerala, presents unique challenges for NLP due to its rich morphology and flexible word order. Dependency parsing is crucial for many NLP applications, including sentimental analysis, machine translation, and information extraction. The relationship between words in Malayalam does not reflect the position in a sentence [6]. Dependency parsing is particularly important, enabling an accurate representation of the sentence structure by focusing on the functional relationships between words rather than their position. Malayalam's script and morphological richness necessitate robust parsing techniques for complex sentence constructions, such as compound verbs, inflected nouns, and agglutinative formations. Dependency parsing helps disambiguate these structures, clearly understanding who is doing what and to whom in a sentence. This understanding is vital for advancing NLP applications in Malayalam, enabling more effective processing, analysis, and generation of text in the language.

1.1. Historical Evolution of Parsing Techniques

Dependency parsing techniques have evolved significantly, with tailored approaches emerging for agglutinative languages due to their unique morphological characteristics. Traditional rule-based parsers were among the earliest methods, relying on handcrafted rules to parse sentences, but they struggled with scalability and linguistic diversity. Statistical parsers, such as the Maximum Entropy and Conditional Random Field-based models, improved generalization by learning from annotated corpora, yet they often fell short in handling complex agglutinations. Neural network-based approaches, particularly transition-based and graph-based dependency parsers, have shown promise by leveraging deep learning to capture intricate syntactic dependencies. For agglutinative languages like Malayalam, specific adaptations have been made, such as incorporating morphological features, suffix splitting, and subword embeddings to manage the rich morphology effectively. Hybrid methods that combine rule-based techniques with statistical or neural models have also been developed to balance linguistic insights with computational efficiency. Recent advancements include multilingual parsers, such as UDify and multilingual BERT-based parsers, which benefit from cross-linguistic data to improve performance on underrepresented languages. These advancements underline the importance of integrating linguistic features and leveraging modern machine learning techniques to tackle the challenges posed by agglutinative languages.

2. Literature Review

In order to address free word order and rich morphology, Nayana et al. [7] introduced a dependency parser approach for Malayalam. The authors used a malt parser and LIBSVM algorithm to incorporate phrase tags and parts of speech. NLP for Malayalam marks advancement using this system, but the sentence structure is more complicated and needs further

improvement.Using Paninian grammar, Aparna et al. [8] developed a Malayalam dependency parser that constructs integer programming constraints from grammatical constraints. This method introduced a linguistic parsing for Malayalam's flexible word order. Krishnakumar et al. introduced a method for identifying objects and subjects in Malayalam. The authors compared MALT and MST parsers, emphasizing the importance of extensive training data for precise parsing. While the results are promising, the study would benefit from a deeper analysis of the parsers' performance on complex sentence structures. El-Morsy [10] developed an Arabic Open Information Extraction (AOIE) system that uses dependency parsing to extract relation tuples from Arabic text, tackling the language's morphological complexity. The system is domain-independent and scalable but struggles with certain clause types, indicating room for improvement.

Al-Ghamdi et al. [11] created the first syntactically annotated corpus for Classical Arabic poetry, facilitating linguistic analysis of this complex form, and initial dependency parsing experiments showed the corpus's value. Halabi et al. [12] examined the impact of varying dependency relations on Arabic parsing performance, advocating for language-specific treebanks like I3rab. The study emphasizes empirical approaches for developing linguistic resources, though more detailed improvements from optimized dependency relations are needed. Zmigrod et al. [13] highlighted the distinction between dependency and spanning trees in dependency parsing, introducing an efficient algorithm to enforce dependency tree constraints. While the study addresses a crucial aspect of parsing, further research is needed to explore its practical implications. Zsibrita et al. [14] presented "magyarlanc," a toolkit for processing Hungarian texts, including dependency parsing. The tool is valuable for Hungarian and other morphologically rich languages, though language-specific features may limit its application. Alves et al. [15] investigated the use of typological approaches to improve Croatian dependency parsing with the Udify tool. The study found that combining Croatian with certain languages significantly improves parsing, though the limitations of typological approaches and linguistic resources pose challenges.

3. Method

The utilized dataset for this study comprises 9031 sentences from Malayalam Treebank Data_IIITH [16]. At the dependency parsing level to manually annotate data, this treebank is designed as an electronic repository to capture textual relationships. Both parts of speech and chunk level data are encoded within the treebank. Within the selected treebank, the Paninian karaka theoretical model is incorporated. Initially, SSF denotes the treebank, then undergoes CoNLL conversion. In order to satisfy the MST need, the CoNLL is transformed into MST format. This conversion makes incorporating the MST parser easier for another set of analyses. In conducting experiments with MST Parser version 0.2, the dataset was divided into 80% for training and 20% for testing. The data was formatted specifically for the parser, with each sentence laid out across three or four lines in a space-separated format. Extensive experimentation was performed to optimize the graph-based parser, which included fine-tuning various settings. The MST Parser supports two training modes, projective and non-projective and allows adjustments for the k-best parse size and feature scope.

3.1. Dataset Construction and Pre-Processing

In this study, the datasets were carefully chosen for training and testing to ensure thorough coverage of the linguistic diversity of the Malayalam language. According to Universal dependency principles, the Malayalam treebank was the primary dataset. The dataset consists of 5000 tokens spread across 5000 sentences from literary works and news articles. To consider the rich agglutinative nature of Malayalam, the dataset includes compound words, syntactic notations, and inflectional forms. Sentences were tokenized and annotated with head dependent relationships, parts of speech tags and morphological features. Cleaning the noisy text and splitting the complex words into sub-words are the pre-processing steps to meet the parser's needs better.

3.2. LysFast Parser

In a transition-based parsing framework, the initial configuration consists of an initial stack, a buffer holding remaining input words, and a set of dependency arcs that are initially empty. The parser transitions from one configuration to another based on a defined set of rules, including the Shift transition, which moves the next word in the buffer to the top of the stack or Reduce, which adds a dependency arc between two words atop the stack. LysFast is based on transition-based parsing, where any sentence's dependency structure is gradually built into a series of state transitions. It is very adaptive to languages with elaborate morphology and flexible syntax, such as Malayalam. Extract a rich set of features from the current state of the stack, buffer, and partially constructed dependency tree that ensure accurate predictions. Lexical information such as word forms and lemmas are used along with part-of-speech tags, plus morphological features such as case markers and verb inflections unique to the Malayalam language. These features will enable a better decision-making process for parsing Malayalam, mainly because its syntactic variations are considerable.

3.3. Neural Network Parser

We have used encoding sentences into dense vector representations and embedding them in predicting dependency relationships between words. This approach allowed our parser to learn and generalize from large amounts of annotated data, effectively capturing local and global syntactic dependencies. The parser mapped every word in the sentence into a high-dimensional vector and captured

semantic and syntactic information based on the word's context within the large corpus. We tried some pre-trained embeddings, such as Word2Vec and FastText, but most of the learning and training happened during the parsing process in our implementation. These word embeddings helped us give continuous, dense representations of words and helped our model deal with variability and ambiguity inherent in natural language.We employed the multi-layer neural network architecture to process the sequence of word embeddings and to predict dependency relations. This allowed us to capture long-range dependencies and more complex sentence structures. We also looked into using Transformer-based architectures where self-attention mechanisms are applied to simultaneously capture the relationships between all pairs of words in a sentence. Our parser predicted the dependency relations between words after encoding the sentence in the neural network. We used a feedforward neural network, which is a multi-layer perceptron. It inputs the hidden states of word pairs and produces scores for potential dependency arcs. Our model was trained to predict each word's head and the type of dependency relation, such as subject or object. Our predictions were graph-based, where the parser scores all possible arcs and selects the highest-scoring arcs to form the dependency tree.

3.4. UU Parser

UU Parser was a rule-based approach along with statistical techniques, enabling the parser to analyze and predict dependency relations within sentences efficiently. In our study, the UU Parser highly depended on a rule-based system to perform morphological analysis, which is highly important for languages like Malayalam, where inflectional morphology is rich. We defined an entire system of linguistic rules for adequate identification and categorization of the morphological properties of words such as tense, number, and case. This is significant to ensure that the parser correctly identifies words' syntactic roles in a sentence. Besides rule-based morphological analysis, statistical models were employed to predict word dependency relations.

The UU Parser had a probabilistic model that approximated the probability of dependency arcs based on a dependency-annotated tree bank. As such, it combined output from the rule-based morphological analyzer and the predictions of the statistical model for better syntactic structure determinations of sentences. It gave us the best of two worlds: a rule-based system's accuracy and statistical methods' flexibility.

To improve the performance of the UU Parser, lexical resources in the form of an exhaustive Malayalam dictionary and part-of-speech tagger were incorporated. The extra context that such resources brought was helpful in disambiguating words that might have multiple meanings. The lexical resources were extremely useful when dealing with out-of-vocabulary words and infrequent morphological forms.

3.5. Turbo Parser

Turbo Parser uses graph-based modeling of dependency parsing. It represents sentences as graphs where nodes are words and edges are dependencies. This allows the parser to handle complicated syntactic relationships by reducing them to a maximum spanning tree problem. This paper used the approach to efficiently compute the most likely dependency structure of Malayalam sentences, known for free word order and complex syntactic features.

A maximum spanning tree algorithm at the core identifies the optimal dependency structure for any input sentence. The algorithm applies the principle of making the sum of weights as large as possible for a dependency tree based on scores assigned to potential dependency arcs. We used this algorithm so that the parser could also capture the syntactic dependencies in Malayalam, even when the dependencies were ambiguous or overlapping. The Turbo Parser has feature-based learning in its parsing mechanism. It uses a rich set of features, such as word embeddings, part-of-speech tags, and syntactic cues, for predicting dependency relations between words. We extracted and utilized these features to train the parser on a Malayalam-specific dependency-annotated corpus. This feature-based approach helps the parser distinguish between more subtle syntactic patterns and relationships while improving overall accuracy.

4. Results and Discussion

For each configuration, models were developed that significantly improved the qualitative metric of LAS, UAS, and Labeled Accuracy (LA). The percentage of correct head-dependent relation identification that is irrelevant to labels is known as UAS. This gives a metric of how effective syntactic structure is even in a complex word-building language such as agglutinative. A better UAS score indicates that the parser is doing a good job of capturing basic syntactic dependencies, which is quite important in languages with rich morphology. LAS extends the UAS score by adding the correctness of the dependency labels beside the structure. For Malayalam, where words are usually associated with many morphological variations and syntactic nuances, LAS must see how well the parser could assign the right syntactic roles and semantic labels to dependencies other than structural accuracy.

BLEX emphasizes lexicalized dependencies and checks how well a parser attaches the specific words correctly to other words in a sentence. The parser is not language dependent, so no special adjustments were required in language, and the best configuration for Malayalam was identified through these experiments. Efficiency is an important aspect of the design of the LysFast Parser, especially when handling large-scale text data. The parser is optimized to minimize overheads on the computation process, thereby enabling it to process sentences rapidly without losing accuracy. Table 1 illustrates the Lys-FASTPARSE Dependency Parser results for cross-validation compared with the baseline and optimisation methods.

Table 1. Parsing accuracy for Lys-FASTPARSE

Performance Parameters	Baseline	Optimization
LAS F1 Score	56.60	56.60
MLAS Score	48.58	48.58
BLEX Score	48.58	48.58

Table 2. Parsing accuracy for neural parser

Parameters	Order 1	Order 2
LAS	0.46	0.47
UAS	0.72	0.73

The metrics used to evaluate the performance include the LAS F1, MLAS Score, and BLEX scores. These scores indicate how well the parser identifies syntactic dependencies and labels morphological features. The LAS F1 Score, which stands for Labeled Attachment Score, measures the accuracy of the correct head and the dependency label assigned to each word in the sentence. In this case, the LAS F1 Score is 56.60 for both the baseline and optimization approaches, indicating that the optimization process did not improve the baseline performance in terms of syntactic dependency parsing accuracy. The MLAS Score, or Morphology-Aware Labeled Attachment Score, considers the accuracy of the syntactic dependencies while also considering the correctness of morphological features such as part-of-speech tags. Here, the MLAS Score remains constant at 48.58 for both the baseline and optimization methods. This suggests that the optimization did not enhance the parser's ability to identify and utilize morphological features in the parsing process correctly. Finally, the BLEX Score, which stands for Labeled Attachment Score, evaluates the performance by considering both the syntactic dependencies and the correctness of lemmatization (i.e., the base forms of words). The BLEX Score also remains unchanged at 48.58 for both approaches. This indicates that the optimization efforts did not improve the parser's performance in terms of accurately identifying and using lemmas along with syntactic dependencies. Table 2 compares the performance of two parsing models or configurations (Order 1 and Order 2) based on two key metrics: LAS and UAS. For Order 1, the LAS is 0.46, and the UAS is 0.72; for Order 2, the LAS is slightly higher at 0.47 and the UAS at 0.73. This indicates that Order 2 performs marginally better than Order 1 in correctly assigning syntactic heads and labels (as shown by LAS) and identifying syntactic heads regardless of the labels (as shown by UAS). The small increments suggest an improvement in parsing accuracy with the second configuration. The analysis of the parsing performance of the UU parser shown in Table 3 across different categories of words reveals variations in accuracy for several key metrics: the number of correct heads, correct dependencies, and instances where both the head and dependency were correctly identified. Out of 13,680 words, the parsing system achieved 78% accuracy in identifying the correct heads, 53% accuracy for correct dependencies, and 48% for both being correct simultaneously. Nouns and proper nouns performed well in head accuracy, achieving 83%, but showed lower accuracy in dependency assignment with 45% and 46%, respectively. Verbs demonstrated a balanced performance, with 75% accuracy for heads and 72% for dependencies, resulting in 68% accuracy for both.

Table 3. Overall accuracy over CPOSTAGs in UU parser

Category	Words	Right Head	% Head	Right Dep	% Dep	Both Right	% Both
Total	13680	10643	78%	7202	53%	6611	48%
N	6232	5186	83%	2820	45%	2571	41%
V	4599	3427	75%	3319	72%	3123	68%
PN	949	787	83%	437	46%	398	42%
CC	450	268	60%	187	42%	143	32%
AVY	423	219	52%	230	54%	187	44%
ADV	381	295	77%	5	1%	5	1%
RPD	234	169	72%	60	26%	52	22%
NST	102	69	68%	21	21%	17	17%
QTF	80	66	82%	31	39%	29	36%
V_VM_VF	53	29	55%	28	53%	28	53%
PSP	48	31	65%	13	27%	12	25%
PNQ	36	32	89%	10	28%	10	28%
NUM	26	22	85%	8	31%	7	27%
CCP	19	8	42%	8	42%	5	26%
CL	18	16	89%	16	89%	15	83%
CC_CCS	14	9	64%	3	21%	3	21%
ADJ	5	3	60%	3	60%	3	60%
INTF	3	2	67%	1	33%	1	33%
UNK	3	3	100%	0	0%	0	0%
V_VM_VNF	3	0	0%	0	0%	0	0%
DEM	1	1	100%	1	100%	1	100%

Table 4. Error rate and its distribution over CPOSTAGs in UU parser

Category	Words	Head Err	% Head Err	Dep Err	% Dep Err	Both Wrong	% Both Wrong
Total	13680	3037	22%	6478	47%	2446	18%
N	6232	1046	17%	3412	55%	797	13%
V	4599	1172	25%	1280	28%	976	21%
PN	949	162	17%	512	54%	123	13%
CC	450	182	40%	263	58%	138	31%
AVY	423	204	48%	193	46%	161	38%
ADV	381	86	23%	376	99%	86	23%
RPD	234	65	28%	174	74%	57	24%
NST	102	33	32%	81	79%	29	28%
QTF	80	14	18%	49	61%	12	15%
V_VM_VF	53	24	45%	25	47%	24	45%
PSP	48	17	35%	35	73%	16	33%
PNQ	36	4	11%	26	72%	4	11%
NUM	26	4	15%	18	69%	3	12%
CCP	19	11	58%	11	58%	8	42%
CL	18	2	11%	2	11%	1	6%
CC_CCS	14	5	36%	11	79%	5	36%
ADJ	5	2	40%	2	40%	2	40%
INTF	3	1	33%	2	67%	1	33%
UNK	3	0	0%	3	100%	0	0%
V_VM_VNF	3	3	100%	3	100%	3	100%
DEM	1	0	0%	0	0%	0	0%
V_VM_VINF	1	0	0%	0	0%	0	0%

However, categories like adverbs and conjunctions had significantly lower performance in dependency accuracy, indicating areas needing improvement. While some categories, such as unknown words and demonstratives, showed perfect accuracy in the few instances present, these results are not broadly indicative due to the small sample sizes. The error analysis for parsing is tabulated in Table 4 across various word categories, highlighting distinct challenges and variations in performance.Of 13,680 words, 22% had head errors, 47% had dependency errors, and 18% had both errors, indicating areas where the parsing system struggles. Nouns and proper nouns displayed relatively low head error rates of 17% but faced higher dependency error rates at 55% and 54%, respectively. Verbs had a higher head error rate of 25% and moderate dependency errors at 28%. Conjunctions and adverbs faced significant difficulties, with conjunctions showing 40% head errors and 58% dependency errors, while adverbs had 48% head errors and 46% dependency errors.

Notably, categories like adverbs and particles exhibited nearly complete dependency errors, with 99% and 73%, respectively, underscoring the parsing system's challenges with these words. Smaller categories, such as unknown words and demonstratives, had perfect head accuracy but high dependency error rates, though these results are based on very few instances and may not be statistically significant. The performance metrics for the parsing system are tabulated in Table 5 and, based on direction-specific dependencies, reveal varying levels of accuracy and precision. Figure 3 illustrates the precision and recall metrics for binned HEAD directions, highlighting the parser's accuracy in predicting syntactic dependencies based on the direction of the HEAD to its dependents. For dependencies directed "to_root," the system correctly identified 1,148 out of 1,872 instances, achieving a recall of 61.32% and a precision of 61.65%.

This indicates a moderate ability to identify root dependencies correctly. For dependencies going "left," the system showed a high recall of 90.76% by correctly identifying 953 out of 1,050 instances, but with a precision of 43.50%, indicating a significant number of false positives. Conversely, for "right" dependencies, the system performed very well, with a recall of 88.42% and a precision of 98.81%, correctly identifying 9,512 out of 10,758 instances and showing very few false positives. No "self" dependencies made recall and precision metrics not applicable (NaN).

These results suggest that while the system identifies rightward dependencies with high precision, it struggles with leftward dependencies, as evidenced by the lower precision in that category. Figure 4 displays the precision and recall metrics for binned HEAD distances, demonstrating the parser's performance in predicting dependencies based on the varying distances between the HEAD and its dependents. When considering dependencies at a distance of 1, the system excelled, achieving a high recall of 92.24% and precision of 93.14%, correctly identifying 6,691 out of 7,254 instances, as shown in Table 6.

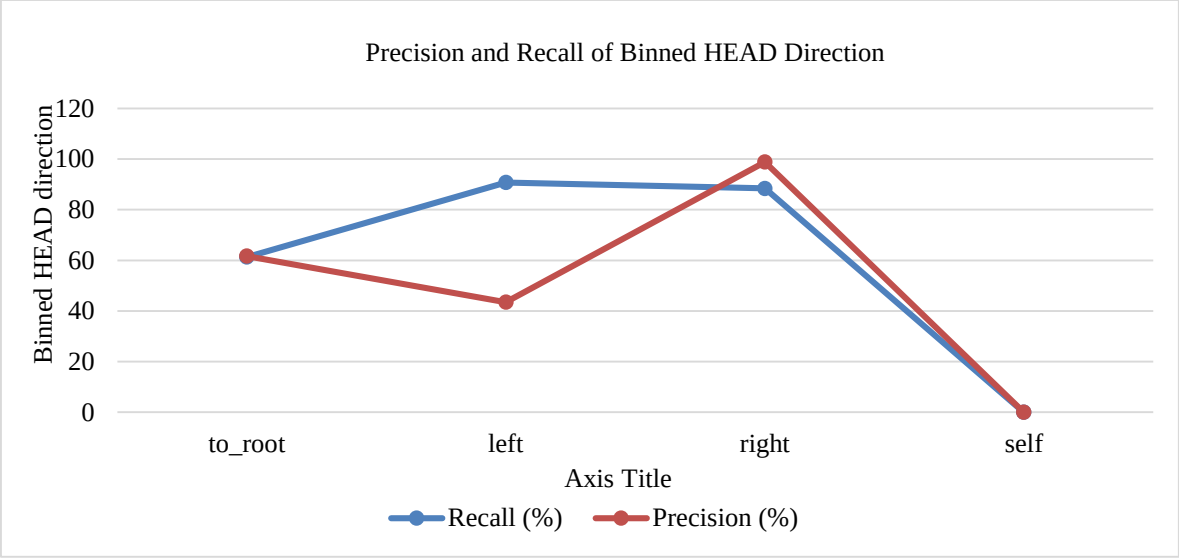

Fig. 3 Precision and recall of binned HEAD direction

Table 5. Precision and recall of binned HEAD direction

Direction	Gold	Correct	System	Recall (%)	Precision (%)
to_root	1872	1148	1862	61.32	61.65
left	1050	953	2191	90.76	43.50
right	10758	9512	9627	88.42	98.81
self	0	0	0	NaN	NaN

Table 6. Precision and recall of binned HEAD distance

Distance	Gold	Correct	System	Recall (%)	Precision (%)
to_root	1872	1148	1862	61.32	61.65
1	7254	6691	7184	92.24	93.14
2	2370	1808	2218	76.29	81.51
3-6	1872	1246	2008	66.56	62.05
7-...	312	66	408	21.15	16.18

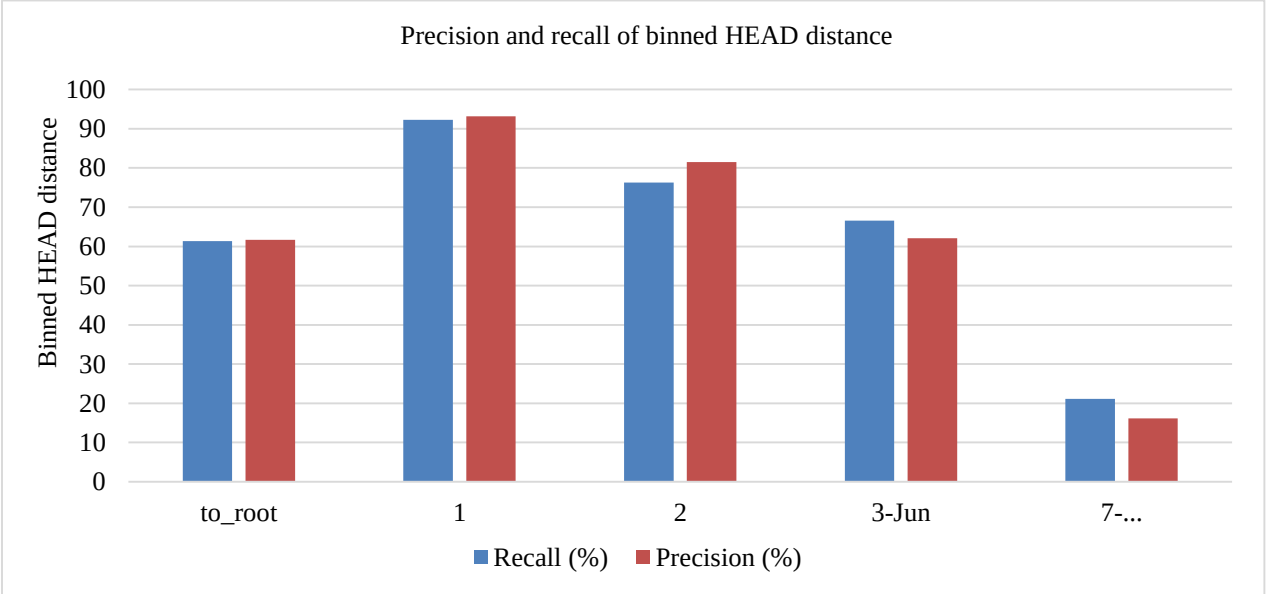

Fig. 4 Precision and recall of binned HEAD distance

Performance decreased at a distance of 2, with a recall of 76.29% and precision of 81.51%, correctly identifying 1,808 out of 2,370 instances. The system's effectiveness dropped further for dependencies at distances between 3 and 6, where it correctly identified 1,246 out of 1,872 instances, resulting in a recall of 66.56% and a precision of 62.05%. The system struggled significantly with long-distance dependencies (7 or more), achieving a recall of only 21.15% and a precision of 16.18%, with just 66 correct identifications out of 312 instances. This analysis highlights the system's strong performance in handling short-distance dependencies but considerable challenges with longer dependencies.

Figure 5 illustrates the error words identified by the UU Parser, highlighting instances where the parser struggled to predict the dependencies correctly, likely due to the complex morphological features of the Malayalam language. Table 7 highlights the focus words where most parsing errors occur. The word "NULL / CC" shows dependency errors, and 92 are both errors. Lastly, the adverb "തന്നെ" shows 14 head errors, 52 dependency errors, and 13 both errors. These focus words contribute significantly to the overall parsing errors, indicating areas where the parsing system might need improvement.

Table 7. The most frequent error words in the UU parser

	any	head	dep	both
NULL / CC	307	182	263	138
ആണ് / V	209	201	199	191
എന്ന് / AVY	109	96	105	92
തന്നെ / RPD	53	14	52	13
ആയി / V	45	41	36	32

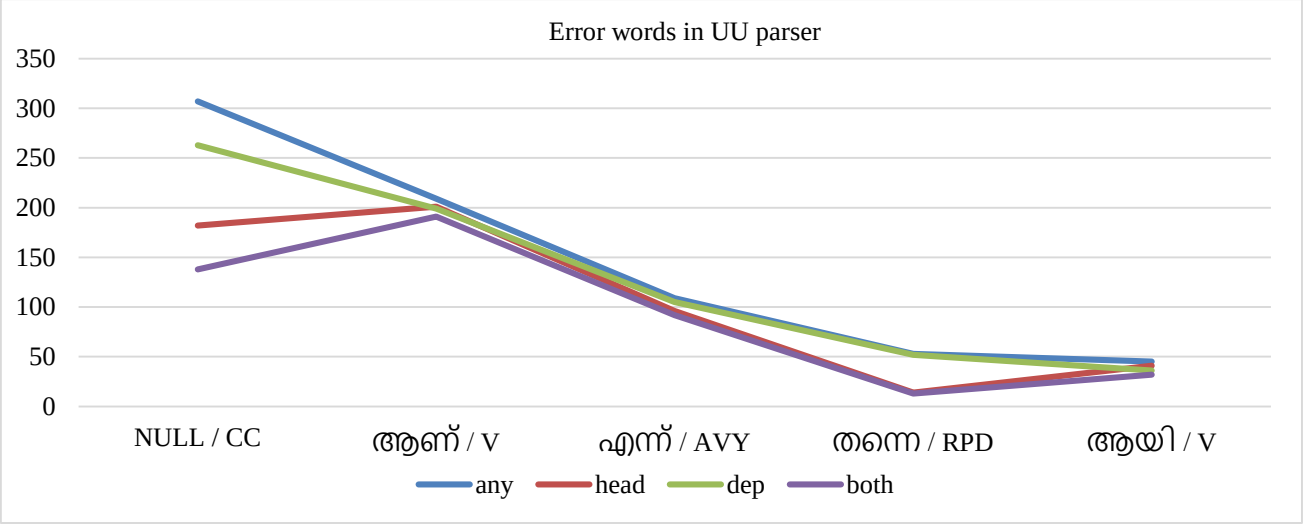

Fig. 5 Error words in the UU parser

The errors identified in the dependency analysis are primarily related to incorrect dependency relationships between words, particularly after the focus word (the word of interest in each error). Here is a concise explanation for each of the top 10 errors.

- Dependency "k1" instead of "k2"

This error occurs when the system incorrectly assigns a dependency relationship ("k1") instead of the correct dependency ("k2") after the focus word, and it has happened 289 times. It often involves nouns (N) or verbs (V), indicating a systematic misinterpretation of syntactic relationships.

- Dependency "k2" instead of "k1"

Conversely, this error reflects instances where the system incorrectly assigns "k2" instead of "k1" after the focus word, typically with nouns (N) or verbs (V). During testing, this error occurred 276 times.

- Dependency "pof" instead of "k1"

Here, the system erroneously assigns "pof" (postposition) instead of "k1" (first case marker) after the focus word, and this happened 272 times.

- Dependency "pof" instead of "k2"

This error mirrors the previous one but involves "pof" instead of "k2" after the focus word, indicating an incorrect assignment of postpositional relationships.

- Head = 0 instead of after the focus word

This error occurs when the system incorrectly places the head dependency as zero (root) instead of correctly identifying

the relationship after the focus word, often observed with nouns (N) or verbs (V).

- Dependency "k1" instead of "k1s"

Here, the system assigns "k1" instead of the correct "k1s" (noun sub-case marker) after the focus word, indicating a specific error related to noun case marking.

- Dependency "k2" instead of "pof"

This error involves the incorrect assignment of "k2" instead of "pof" after the focus word, usually seen with nouns (N) or verbs (V) 129 times.

- Dependency "pof" instead of "k1s"

This error indicates the incorrect assignment of "pof" instead of "k1s" after the focus word, particularly in cases involving noun sub-case markers, which happened 126 times.

- Dependency "k7" instead of "k7p"

This error reflects cases where the system assigns "k7" instead of "k7p" after the focus word, suggesting a misinterpretation of syntactic relationships, often with nouns (N) or verbs (V).

- Dependency "k7" instead of "k4" (101 times)

Lastly, this error occurs when "k7" is incorrectly assigned instead of "k4" after the focus word, indicating a systematic error in identifying syntactic dependencies, particularly with nouns (N). Such errors were reported in 101 times. These errors highlight common challenges in accurately parsing dependency relationships, especially in languages with rich morphosyntactic features like Malayalam. Improving the precision of dependency parsing algorithms would require refining the rules and features to identify and classify these relationships based on linguistic context and syntactic structures. Figure 6 shows the parsing accuracy of the Turbo Parser, presenting the performance metrics. Lys-FASTPARSE, while generally stable in performance, struggled with complex compound words and agglutinative forms typical of Malayalam. These forms often involve the concatenation of multiple morphemes, leading to difficulties in correctly identifying head-dependent relationships. Additionally, the parser faced challenges with rightward dependencies, where the syntactic structure was not captured as effectively as leftward or root dependencies. Errors often involved misidentifying the syntactic head in constructions with heavy inflectional morphology. The Neural Network-based parser showed improvements in parsing accuracy, but it still faced errors related to morphological ambiguities. In particular, it struggled with parsing sentences involving nominal compounds and verb conjugations. This parser, which depends on feature learning, also had difficulty correctly predicting the relationships between words in sentences with non-canonical word orders, as is common in Malayalam. While it performed better on simpler structures, its accuracy dropped significantly in complex syntactic constructions, such as postpositions or relative clauses.The UU parser performed well in simpler syntactic structures but encountered challenges with long-range dependencies and sentence structures involving multiple subclauses, which are prevalent in Malayalam. In particular, the parser often fails to capture the relationships between the verb and its arguments in passive voice constructions, a common syntactic feature in Malayalam. Errors in handling subject-object agreement were also frequent in sentences with complex subject-predicate relations. The Turbo Parser demonstrated the best overall performance, but it still encountered challenges with parsing intricate syntactic constructions typical of Malayalam, such as complex relative clauses, multi-word compounds, and the flexibility of word order. Additionally, it occasionally misidentified the syntactic head in long-distance dependencies, especially when dealing with nested clauses or when multiple compound words were involved. Moreover, the parser showed difficulty in accurately labeling the dependencies in sentences with elliptical constructions, which are frequent in conversational Malayalam.

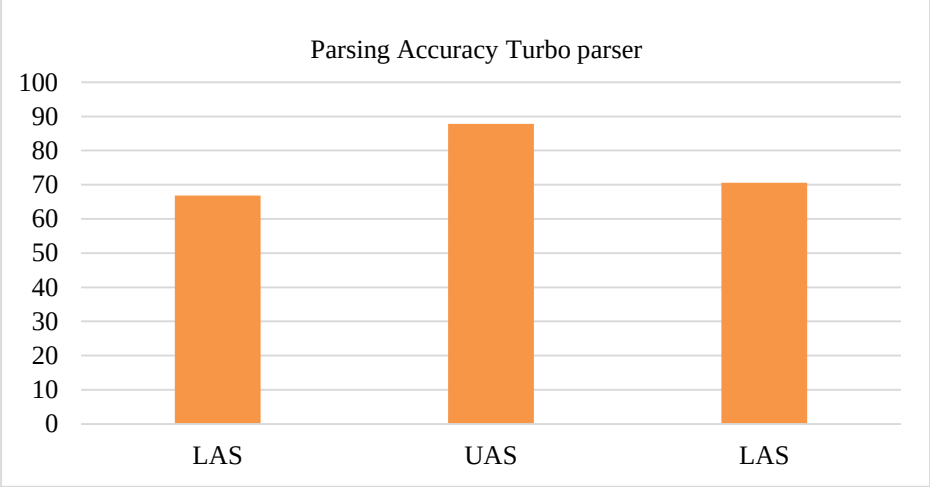

Fig. 6 Parsing accuracy of turbo parser

Table 8. Parsing results of Turbo parser

Performance parameters	Baseline	Optimization
Labeled attachment score	66.89 %	66.88 %
Unlabeled attachment score	87.12 %	87.83 %
Label accuracy score	70.66 %	70.58 %

Table 9. Comparative performance of the parsers

Parser Comparison	UAS p-Value	LAS P-Value	Statistical Significance	Conclusion
Lys-FASTPARSE vs. Neural Network	> 0.05	> 0.05	Not significant	No significant difference in performance.
Lys-FASTPARSE vs. UU Parser	< 0.05	< 0.05	Significant	UU Parser outperforms Lys-FASTPARSE.
Lys-FASTPARSE vs. Turbo Parser	< 0.01	< 0.01	Significant	Turbo Parser significantly outperforms Lys-FASTPARSE.
Neural Network vs. Turbo Parser	< 0.01	< 0.01	Significant	Turbo Parser significantly outperforms Neural Network-based parser.
UU Parser vs. Turbo Parser	> 0.05	> 0.05	Not significant	No significant difference in performance.

Table 8 presents the performance metrics of a dependency parsing model, comparing the baseline configuration with an optimized configuration. LAS, which measures the accuracy of both the head and the dependency label, shows a slight decrease from 66.89% in the baseline to 66.88% in the optimized version. The UAS, which measures only the accuracy of the head, improved from 87.12% to 87.83% after optimization. Finally, the LAS, indicating the correctness of the dependency labels alone, shows a minor decrease from 70.66% to 70.58%. These results suggest that while the optimization improved the accuracy of the head attachments, it had a negligible impact on the labeled attachment accuracy and a slight negative impact on the label accuracy.

5. Conclusion

This study provides an in-depth exploration of dependency parsing methodologies applied to the Malayalam language, with a focus on evaluating the performance of various parsers, including Lys-FASTPARSE, a Neural Network-based parser, UU parser and the Turbo Parser. The findings underscore the importance of morphological analysis in enhancing the accuracy of dependency parsing, particularly for Malayalam, which is characterized by rich morphological and syntactic complexity. The analysis reveals that while the Lys-FASTPARSE maintained stable performance metrics across various optimization attempts, the Neural Network parser demonstrated a modest yet significant improvement in capturing sentence dependencies, as evidenced by slight gains in UAS and LAS scores. The Turbo Parser emerged as the most robust among the evaluated parsers, exhibiting high UAS and LAS scores and demonstrating a balanced approach between accuracy and computational efficiency. The detailed examination of precision and recall in handling binned HEAD directions, particularly in right-direction dependencies, highlights the parser's strength in navigating intricate syntactic structures, though challenges remain in accurately parsing left-direction and root dependencies. These findings highlight the necessity of customized parsing techniques for languages with intricate morphologies like Malayalam and emphasize the potential of combining rule-based and statistical approaches to achieve higher parsing accuracy. This study advances the understanding of dependency parsing for underrepresented languages and provides practical insights that can be applied to improve natural language processing systems, making them more adaptable and effective in processing diverse linguistic inputs.

Acknowledgments

I want to express my sincere gratitude to all those who contributed to completing this research paper. I extend my heartfelt thanks to my supervisor, family, colleagues and fellow researchers for their encouragement and understanding during the demanding phases of this work.

References

[1] Sandra Kübler, Ryan McDonald, and Joakim Nivre, *Dependency Parsing*, Synthesis Lectures on Human Language Technologies, Springer, pp. 11-20, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[2] Carlos Gómez-Rodríguez, John Carroll, and David Weir, "Dependency Parsing Schemata and Mildly Non-Projective Dependency Parsing," *Computational Linguistics*, vol. 37, no. 3, pp. 541-586, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[3] David Vilares, and Carlos Gómez-Rodríguez, "A Non-Projective Greedy Dependency Parser with Bidirectional Lstm," *arXiv*, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[4] Daniel Hershcovich, *Universal Semantic Parsing with Neural Networks*, Ph.D. Thesis, Hebrew University, pp. 1-288, 2019. [Google Scholar] [Publisher Link]

[5] Dat Quoc Nguyen, and Karin Verspoor, "From POS Tagging to Dependency Parsing for Biomedical Event Extraction," *BMC Bioinformatics*, vol. 20, pp. 1-13, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] B. Premjith et al., "Embedding Linguistic Features in Word Embedding for Preposition Sense Disambiguation in English—Malayalam Machine Translation Context," *Recent Advances in Computational Intelligence*," Springer, pp. 341-370, 2019.[CrossRef] [Google Scholar] [Publisher Link]

[7] R.M. Nayana, R.R. Rajeev, and C. Naseer, "Dependency Parser for Malayalam: A Machine Learning Approach," *2018 International CET Conference on Control, Communication, and Computing (IC4)*, Thiruvananthapuram, India, pp. 394-398, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[8] T. Aparnna, P.G. Raji, and K.P. Soman, "Integer Linear Programming Approach to Dependency Parsing for Malayalam," *2010 International Conference on Recent Trends in Information, Telecommunication and Computing*, Kerala, India, pp. 324-326, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[9] K. Krishnakumar, S. Rajendran, and N. Rajendran, "Subject and Object Identification in Malayalam Using Machine Learning Approach," *IJDL International Journal of Dravidian Linguistics*, vol. 42, no. 1, pp. 36-66, 2013. [Google Scholar] [Publisher Link]

[10] Sally Mohamed Ali El-Morsy, Mahmoud Hussein, and Hamdy M. Mousa, "Arabic Open Information Extraction System Using Dependency Parsing," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 1, pp. 541-551, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[11] Sharefah Al-Ghamdi, Hend Al-Khalifa, and Abdulmalik Al-Salman, "A Dependency Treebank for Classical Arabic Poetry," *Proceedings of the Sixth International Conference on Dependency Linguistics, (Depling, SyntaxFest 2021)*, pp. 1-9, 2021. [Google Scholar] [Publisher Link]

[12] HDana Halabi, Arafat Awajan, and Ebaa Fayyoumi, "Improving Arabic Dependency Parsers by using Dependency Relations," *2020 21st International Arab Conference on Information Technology (ACIT)*, Giza, Egypt, pp. 1-7, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[13] Ran Zmigrod, Tim Vieira, and, Ryan Cotterell, "Please Mind the Root: Decoding Arborescences for Dependency Parsing," *arXiv*, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[14] Janos Zsibrita, Veronika Vincze, and Richard Farkas, "Magyarlanc: A Tool for Morphological and Dependency Parsing of Hungarian," *Proceedings of Recent Advances in Natural Language Processing*, Hissar, Bulgaria, pp. 763-771, 2013. [Google Scholar] [Publisher Link]

[15] Diego Alves, Boke Bekavac, and Marko Tadić, "Typological Approach to Improve Dependency Parsing for Croatian Language," *Proceedings of the 20th International Workshop on Treebanks and Linguistic Theories*, pp. 1-11, 2021. [Google Scholar] [Publisher Link]

[16] Malayalam Treebank Data_IIITH, TDIL, GoTranslate Web Localizer, 2018. [Online]. Available: https://tdil-dc.in/index.php?option=com_download&task=showresourceDetails&toolid=1980&lang=en