Software Requirement Specification

on

Malayalam Parser for Dataset Creation

by

Fathima Jennath (U2103089)

Gautham C Sudheer (U2103092)

Godwin Gino (U2103096)

Mohammed Basil (U2103139)

Under the guidance of

Dr. Mary Priya Sebastian

Department of Computer Science and Engineering
Rajagiri School of Engineering & Technology (Autonomous)
(Parent University: APJ Abdul Kalam Technological University)
Rajagiri Valley, Kakkanad, Kochi, 682039

March 2024

Software Requirement Specification

on

Malayalam Parser for Dataset Creation

Prepared by

U2103089 Fathima Jennath

U2103092 Gautham C Sudheer

U2103096 Godwin Gino

U2103139 Mohammed Basil

Guided By: Dr. Mary Priya Sebastian

Associate Professor

Dept. of Computer Science, RSET

Table of Contents

1 Introduction **1**

1.1 Purpose 1

1.2 Product Scope 1

2 Overall Description **2**

2.1 Product Perspective 2

2.2 Product Functions 3

2.3 Operating Environment 4

2.4 Design and Implementation Constraints 5

2.5 Assumptions and Dependencies 6

3 External Interface Requirements **6**

3.1 User Interfaces 6

3.2 Hardware Interfaces 7

3.3 Software Interfaces 7

3.4 Communications Interfaces 8

4 System Features **8**

4.1 Parsing Malayalam Text 9

4.1.1 Description and Priority 9

4.1.2 Stimulus/Response Sequences 9

4.1.3 Functional Requirements 9

4.2 Lemmatization and Morphological Analysis 10

4.2.1 Description and Priority 10

4.2.2 Stimulus/Response Sequences 10

4.2.3 Functional Requirements 10

4.3 Dependency Parsing 11

4.3.1 Description and Priority 11

4.3.2 Stimulus/Response Sequences 11

4.3.3 Functional Requirements 11

4.4 Syntax Tree Generation 11
4.4.1 Description and Priority 11
4.4.2 Stimulus/Response Sequences 12
4.4.3 Functional Requirements 12
4.5 Semantic Analysis 12
4.5.1 Description and Priority 12
4.5.2 Stimulus/Response Sequences 12
4.5.3 Functional Requirements 13
4.6 Part-of-Speech tagging 13
4.6.1 Description and Priority 13
4.6.2 Stimulus/Responses Sequences 13
4.6.3 Functional Requirements 14
4.7 Performance Optimization 14
4.7.1 Description and Priority 14
4.7.2 Stimulus/Response Sequences 14
4.7.3 Functional Requirements 14

5 Other Nonfunctional Requirements 15
5.1 Performance Requirements 15
5.1.1 Parsing Malayalam Text 15
5.1.2 Lemmatization and Morphological Analysis 15
5.1.3 Dependency Parsing 15
5.1.4 Syntax Tree Generation 16
5.1.5 Semantic Analysis 16
5.1.6 Part-of-Speech tagging 16
5.1.7 Performance Optimization 16
5.2 Safety Requirements 16
5.3 Security Requirements 17
5.4 Software Quality Attributes 17

References 18

1 Introduction

1.1 Purpose

The system aims to enhance the understanding and analysis of Malayalam text data, enabling more accurate and efficient processing of information, including but not limited to retrieval, sentiment analysis, machine translation, and text summarization. By focusing on the unique linguistic characteristics of Malayalam, the project seeks to overcome the challenges posed by the language's morphology, syntax, and semantics. Through the development of innovative algorithms, data structures, and linguistic resources, the project aims to contribute to the advancement of NLP research and technology in the context of the Malayalam language.

- The primary focus of the Malayalam Parser system is to analyze and parse written text in the Malayalam language. This involves tasks such as tokenization, part-of-speech tagging, parsing, and semantic analysis.

- Initially, the system will target general-purpose parsing tasks, which are applicable to a wide range of text data. These tasks form the foundation of the parsing capabilities of the system.

- The system is designed to be extensible, allowing for the inclusion of domain-specific or specialized parsing tasks in the future. This flexibility ensures that the system can adapt to different needs and requirements over time.

- In addition to algorithms and data structures, the project also involves the development of linguistic resources that are crucial for parsing Malayalam text. These resources may include lexicons, grammars, and annotated corpora.

1.2 Product Scope

The Malayalam Parser is a smart tool that understands Malayalam text. It breaks down sentences to figure out their grammar, structure, and meaning using advanced language processing techniques. It helps with tasks like identifying different parts of speech, understanding how sentences are put together, and analyzing the overall meaning of text.

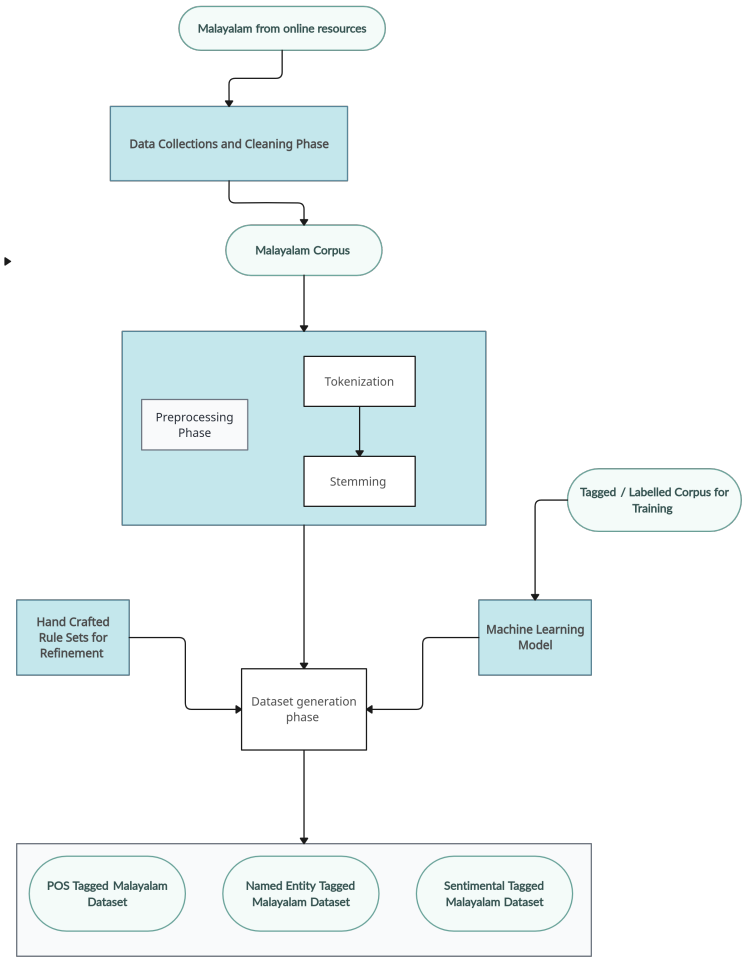
- **Informed Decision-Making:** Implementation of parsing tasks including tokenization, part-of-speech tagging, parsing, and semantic analysis for Malayalam text. Extracting insights from Malayalam text aids in data-driven decision-making, enhancing strategic planning and execution.

- **Innovation:** Developing a Malayalam parser for dataset creation is significant as it encompasses crucial linguistic tasks such as tokenization, part-of-speech tagging, and named entity recognition. This effort is essential for advancing natural language processing capabilities in Malayalam, enabling the development of robust models for sentiment analysis, machine translation, question answering, and domain-specific parsing. The creation of annotated datasets is vital to train and evaluate these models, contributing to the overall improvement of Malayalam language processing technologies.

2 Overall Description

2.1 Product Perspective

This software represents a new, standalone product developed to fulfill the crucial task of parsing and analyzing Malayalam language text. Unlike existing systems or components, this software is designed from the ground up to specialize in the intricacies of Malayalam language parsing, catering to the specific needs of users requiring accurate linguistic analysis in Malayalam.
The Malayalam Parser software does not belong to a product family but serves as an independent solution, tailored specifically for parsing Malayalam text. It is not intended as a replacement for existing systems but rather as a complementary tool to enhance language processing capabilities in Malayalam. While it operates autonomously, it may interface with other systems or applications, such as text input sources or linguistic databases, to facilitate seamless integration within broader linguistic analysis frameworks.



2.2 Product Functions

Major Functions of the Malayalam Parser:

- Parse and analyze Malayalam language text to identify linguistic components such as words, phrases, and sentences.
- Determine grammatical structure, syntax, and semantics of Malayalam sentences to facilitate accurate linguistic analysis.
- Provide functionality for part-of-speech tagging, syntactic parsing, and semantic analysis tailored for the Malayalam language.
- Support for handling compound words, inflections, and variations in word forms commonly found in Malayalam text.

- Generation of a part-of-speech tagged dataset, named entity dataset, and sentimental tagged dataset, contributing to the advancement of language processing technologies in Malayalam

2.3 Operating Environment

For creating the Malayalam Parser, the operating environment encompasses the following specifications:

Hardware Platform

- The software should be compatible with standard computing hardware commonly used for software development and deployment.
- Minimum hardware requirements include a modern processor (e.g., Intel Core i5 or equivalent), sufficient RAM (at least 4GB), and available storage space for software installation and data processing.

Operating System and Versions

- The software should be compatible with popular operating systems used for software development and deployment, including Windows, MacOS, and Linux distributions.
- Specific operating system versions supported include:

  Windows 10 or later

  MacOS 10.13 High Sierra or later

  Ubuntu 18.04 LTS or later

Software Components and Applications

- The Malayalam Parser may require integration with various software components and applications for development, testing, and deployment purposes.
- Development tools such as Python (version 3.6 or later), Java Development Kit (JDK), or other programming languages and frameworks suitable for NLP development may be necessary.

- Libraries and frameworks for natural language processing, such as NLTK (Natural Language Toolkit), spacey or TensorFlow, may be utilized.

- Database systems such as SQLite, PostgreSQL, or MongoDB may be employed for data storage and retrieval if necessary.

2.4 Design and Implementation Constraints

The development of the Malayalam Parser is subject to various constraints and limitations, including:

- **Hardware Limitations:** The software must be optimized to operate within hardware limitations, including timing and memory constraints. Performance optimizations are crucial to ensure efficient operation across different hardware configuration.

- **Integration with External Systems:** The parser must support standard interfaces and data formats for interoperability with other language processing tools or databases. Ensuring smooth integration is a technical requirement.

- **Language Requirements:** The Malayalam Parser must be designed and implemented to meet the linguistic complexities of the Malayalam language. This includes handling unique grammar rules, character encoding, and text processing requirements inherent to Malayalam.

- **Character Encoding and Localization:** Proper handling of character encoding specific to Malayalam and localization considerations are crucial. The parser should accurately interpret and process Malayalam characters.

- **Security Considerations:** Security measures such as encryption, access control, and data integrity must be incorporated into the design and implementation of the Malayalam Parser to safeguard against security threats and vulnerabilities.

- **Design Conventions and Programming Standards:** Adherence to design conventions, programming standards, and coding guidelines is paramount. Developers must follow established best practices and organizational standards to ensure maintainability, readability, and consistency of the codebase.

2.5 Assumptions and Dependencies

1. **Third-Party or Commercial Components:** It is assumed that the development of the Malayalam Parser will rely on third-party or commercial components. The project is intended to be developed from existing models, without significant dependencies on pre-existing software solutions or proprietary libraries.

2. **Development Environment:** The assumption is made that the development environment for the Malayalam Parser will be properly configured and equipped with necessary tools, including programming languages, development frameworks, and version control systems.

3. **Operating Environment:** It is assumed that the operating environment for testing and deployment of the Malayalam Parser will meet the specified requirements, including compatibility with supported operating systems and hardware configurations.

4. **Constraints and Limitations:** The project assumes that constraints and limitations identified in the SRS, such as corporate policies, regulatory requirements, and hardware limitations, will be properly addressed and accounted for during the development process.

5. **External Dependencies:** The project does anticipate reliance on external dependencies or reused software components from other projects. Any dependencies arising during development will be clearly documented and managed accordingly.

6. **Availability of Resources:** It is assumed that the necessary resources, including human resources, budget allocation, and development timelines, will be available and allocated appropriately to support the successful development and completion of the Malayalam Parser project.

3 External Interface Requirements

3.1 User Interfaces

The project has the potential to be utilized in various future applications, particularly in educational tools designed for teaching purposes. Additionally, it can serve as a foundation for developing

apps focused on language learning, text analysis, and speech recognition. This versatility opens up opportunities for creating interactive and innovative tools that cater to different learning styles and linguistic analysis needs. By leveraging the project's capabilities, we can create user-friendly applications that enhance the learning and analysis experience for users. These applications can play a significant role in promoting the Malayalam language and its usage in various contexts.

3.2 Hardware Interfaces

The software interacts with the hardware components of the system through various interfaces, each with specific communication protocols. It communicates with the CPU to execute algorithms, process data, and manage computational tasks, utilizing the CPU's instruction set architecture. The software uses RAM for temporary storage of linguistic data, parsing results, and intermediate processing steps, managing memory efficiently for complex linguistic analyses. For long-term storage, the software reads and writes data to the storage device, employing file systems for dataset management and data integrity. Optionally, the software can access external datasets, online resources, or collaboration tools through a network interface, using standard networking protocols such as HTTP or FTP for data exchange. User interaction and data visualization are facilitated through input/output devices such as keyboards, mouse, and monitors, utilizing standard interfaces for user input and output display.

3.3 Software Interfaces

The Malayalam parser integrates with various software components, including databases such as SQLite, PostgreSQL, and MongoDB for data storage and retrieval. It is designed to run on popular operating systems like Windows 10 or later, MacOS 10.13 High Sierra or later, and Ubuntu 18.04 LTS or later. Development and NLP tasks are facilitated by Python (version 3.6 or later) along with libraries like NLTK, Spacy, and TensorFlow. These tools aid in implementing NLP algorithms and processing linguistic data. The system may also include integrated commercial components for specific functionalities, each with its own data formats, configurations, and communication protocols. To ensure consistent data sharing, the system may implement global data areas or standardized data formats, depending on the software components involved. Detailed application

programming interface (API) protocols should be referenced for precise communication details.

3.4 Communications Interfaces

The software requires communication functions for fetching inputs from websites, email, and network server communications. It needs to support standard email protocols like SMTP for sending emails, following MIME standards for message formatting. For web scraping, it should interact with web servers using HTTP or HTTPS for secure communication, and handle HTML parsing for extracting data from web pages. Network server communication may require protocols like FTP or SFTP for file transfers, adhering to communication standards for compatibility and security. Secure protocols such as SMTPS, SFTP, or TLS should be used for data transmission, with defined data transfer rates and synchronization mechanisms for efficiency.

4 System Features

The system features for the Malayalam Parser are organized to highlight the major services provided by the product:

1. **Text Parsing:** The parser breaks down Malayalam language text into its constituent components such as words, phrases, and sentences.

2. **Lemmatization and Morphological analysis:** Identifying the base forms of words and analyzing word forms to determine grammatical properties in Malayalam text .

3. **Dependency Parsing:** Identifying the syntactic dependencies between words in a Malayalam sentence.

4. **Semantic Analysis:** The system determines the meaning and interpretation of Malayalam text, capturing semantic relationships between words and phrases.

5. **Part-of-Speech Tagging:** Each word in a Malayalam sentence is assigned appropriate part-of-speech tags, indicating its grammatical function.

6. **Syntactic Parsing:** It analyzes the syntactic structure of Malayalam sentences, identifying constituents and their hierarchical relationships.

7. **Error Handling:** Mechanisms are in place to detect and handle errors or inconsistencies in input Malayalam text, ensuring parsing reliability.

8. **Customization and Extension:** Users can customize and extend the parser with custom linguistic rules, dictionaries, or algorithms for domain-specific parsing tasks.

4.1 Parsing Malayalam Text

4.1.1 Description and Priority

This feature involves analyzing and understanding the linguistic structure of Malayalam text, including sentence segmentation, word tokenization, and part-of-speech tagging. It is of high priority as it forms the core functionality of the system.

4.1.2 Stimulus/Response Sequences

Stimulus: User inputs a Malayalam text.
Response: The system parses the text, segments sentences, tokenizes words, and tags parts of speech.

4.1.3 Functional Requirements

1. The system shall be able to segment sentences in Malayalam text based on punctuation marks and grammatical rules.

2. The system shall tokenize words in Malayalam text, considering compound words and affixes.

3. The system shall tag each word in Malayalam text with its respective part of speech (e.g., noun, verb, adjective).

4. The system shall handle variations in spelling and word forms commonly found in Malayalam text.

5. The system shall provide error messages for invalid inputs or unsupported characters.
6. The system shall process text efficiently, aiming for a parsing speed of at least X characters per second.
7. The system shall be able to handle text inputs of up to Y characters in length.

4.2 Lemmatization and Morphological Analysis

4.2.1 Description and Priority

This feature involves performing lemmatization to identify the base form of words and morphological analysis to understand the grammatical properties of words. It is of high priority as it enhances the accuracy of language analysis.

4.2.2 Stimulus/Response Sequences

Stimulus: User inputs a Malayalam word
Response: The system lemmatizes the word and performs morphological analysis to determine its grammatical properties.

4.2.3 Functional Requirements

1. The system shall lemmatize Malayalam words to identify their base forms.
2. The system shall analyze the morphology of Malayalam words to determine their grammatical properties.
3. The system shall handle variations in word forms and inflections commonly found in Malayalam.
4. The system shall provide error messages for invalid inputs or unsupported characters.
5. The system shall process words efficiently, aiming for a lemmatization speed of at least X words per second.
6. The system shall be able to handle word inputs of up to Y characters in length.

4.3 Dependency Parsing

4.3.1 Description and Priority

Dependency parsing involves identifying the syntactic dependencies between words in a sentence, which is crucial for understanding the relationships within the sentence. It is of high priority as it provides valuable linguistic information for further analysis.

4.3.2 Stimulus/Response Sequences

Stimulus: User inputs a Malayalam sentence.
Response:The system parses the sentence to identify syntactic dependencies between words.

4.3.3 Functional Requirements

1. The system shall parse Malayalam sentences to identify syntactic dependencies between words.
2. The system shall generate dependency trees that represent the syntactic structure of parsed sentences.
3. The system shall handle complex sentence structures and dependencies between words.
4. The system shall provide error messages for invalid inputs or unsupported characters.
5. The system shall process sentences efficiently, aiming for a parsing speed of at least X sentences per second.
6. The system shall be able to handle sentence inputs of up to Y characters in length.

4.4 Syntax Tree Generation

4.4.1 Description and Priority

Syntax tree generation involves creating syntax trees that represent the grammatical structure of sentences. It is of high priority as it provides a visual representation of the sentence's structure, aiding in linguistic analysis.

4.4.2 Stimulus/Response Sequences

Stimulus: User inputs a Malayalam sentence.

Response: The system analyzes the sentence's grammatical structure and generates a syntax tree.

4.4.3 Functional Requirements

1. The system shall generate syntax trees for Malayalam sentences based on their grammatical structure.
2. The system shall use standard tree representation formats (e.g., Penn Treebank format) for generated syntax trees.
3. The system shall handle complex sentence structures, including coordination and subordination.
4. The system shall provide options for visualizing syntax trees, such as tree diagrams or textual representations.
5. The system shall process sentences efficiently, aiming for a tree generation speed of at least X trees per second.

4.5 Semantic Analysis

4.5.1 Description and Priority

Semantic Analysis is a critical feature of the system, with a high priority. It involves determining the meaning and interpretation of Malayalam text, capturing semantic relationships between words and phrases.

4.5.2 Stimulus/Response Sequences

Stimulus: User inputs Malayalam text for analysis.

Response: The system processes the text to extract meaning and semantic relationships.

4.5.3 Functional Requirements

1. The system shall identify subject-verb-object relationships in Malayalam sentences.
2. It shall detect and interpret negation, conjunctions, and other semantic features in the text.
3. The system shall extract key concepts and entities from the text.
4. It shall handle ambiguous phrases and provide contextually appropriate interpretations.
5. The system shall support the identification of sentiment and emotions expressed in the text.
6. It shall provide an API or interface for developers to access the semantic analysis results.
7. The system shall handle errors gracefully, providing informative messages for incorrect inputs.
8. It shall process text efficiently, aiming for a response time of less than 1 second for short to medium-length texts.
9. It shall be able to process text with a wide range of vocabulary and language complexity levels, including formal and informal language variants.

4.6 Part-of-Speech tagging

4.6.1 Description and Priority

Part-of-Speech Tagging is a critical feature of the system, with a high priority. It involves assigning grammatical tags to words in Malayalam sentences, indicating their syntactic role and category.

4.6.2 Stimulus/Responses Sequences

Stimulus: User inputs Malayalam text for analysis.
Response: The system processes the text and assigns appropriate part-of-speech tags to each word.

4.6.3 Functional Requirements

1. The system shall identify the part of speech for each word in the input Malayalam text.
2. It shall handle compound words and inflected forms, assigning appropriate tags based on their usage in context.
3. The system shall support standard part-of-speech tag sets for Malayalam, ensuring compatibility with existing linguistic resources.
4. It shall provide an option to output the tagged text in a readable format for users.
5. The system shall be able to handle texts with varying levels of complexity and vocabulary.
6. It shall provide an API or interface for developers to access the tagged text and integrate it into their applications.
7. The system shall handle errors gracefully, providing informative messages for incorrect inputs.

4.7 Performance Optimization

4.7.1 Description and Priority

Performance optimization involves improving the efficiency and speed of the system. It is of low priority as it is desirable but not critical for basic functionality.

4.7.2 Stimulus/Response Sequences

Stimulus: System performance falls below specified thresholds.
Response: The system implements optimization techniques to improve performance.

4.7.3 Functional Requirements

1. The system shall monitor performance metrics, such as processing speed and resource usage.
2. The system shall identify performance bottlenecks and areas for improvement.

3. The system shall implement optimizations, such as algorithmic improvements or resource management strategies, to enhance performance.

4. The system shall provide configuration options for adjusting performance settings based on user needs

5 Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 Parsing Malayalam Text

Rationale: Parsing text should be fast to provide timely analysis for user interactions or batch processing.
Requirement: The system should be able to parse Malayalam text at a rate of at least 1000 characters per second.

5.1.2 Lemmatization and Morphological Analysis

Rationale: Lemmatization and morphological analysis are core linguistic processes that should not introduce significant delays.
Requirement: The system should be able to perform lemmatization and morphological analysis for at least 100 words per second.

5.1.3 Dependency Parsing

Rationale: Dependency parsing is computationally intensive but crucial for understanding sentence structures.
Requirement: The system should be able to perform dependency parsing for at least 50 sentences per second.

5.1.4 Syntax Tree Generation

Rationale: Syntax tree generation requires complex computations and should be efficient for real-time applications.
Requirement: The system should be able to generate syntax trees for at least 20 sentences per second.

5.1.5 Semantic Analysis

Rationale: Semantic analysis involves processing text to understand its meaning, requiring significant computational resources. Efficient performance is crucial for real-time or near real-time applications, such as chatbots or text summarization systems.
Requirement: The system should be able to perform semantic analysis for a minimum of 1000 words per second, ensuring timely and responsive processing for various use cases.

5.1.6 Part-of-Speech tagging

Rationale: Part-of-speech tagging is a fundamental task in natural language processing, providing essential grammatical information about words in a sentence. Efficient tagging is necessary for various downstream tasks such as parsing, machine translation, and information extraction.
Requirement: The system should be able to perform part-of-speech tagging for a minimum of 1000 words per second, ensuring fast and responsive tagging for different text lengths and complexities.

5.1.7 Performance Optimization

Rationale: Performance optimization ensures that the system meets its performance requirements under varying loads and conditions.
Requirement: The system should implement optimization strategies to maintain performance levels even under heavy loads, ensuring that response times do not exceed specified thresholds.

5.2 Safety Requirements

Loss, Damage, or Harm Concerns: The product must ensure user and property safety, preventing physical harm or damage. It must also safeguard sensitive data from unauthorized access.

Actions to be Taken: The product should provide clear instructions and warnings to prevent misuse or accidental damage. Regular maintenance and updates are crucial for safe operation. Actions to be Prevented: Unauthorized modifications and operations that could lead to data loss or corruption should be prevented.

External Policies or Regulations: Compliance with safety standards and regulations is essential, including guidelines from regulatory bodies and industry standards organizations.

Safety Certifications: Obtaining safety certifications demonstrates compliance with standards and builds user trust in the product's safety.

5.3 Security Requirements

The system must prioritize the security and privacy of the data used or created by the system. It should implement robust measures to prevent unauthorized access, disclosure, or modification of sensitive information, such as annotated dataset files. User identity authentication should be incorporated to verify the identity of users accessing the system, ensuring that only authorized users can access or modify the dataset files. The system must comply with external policies and regulations regarding security and privacy issues, adhering to guidelines set by regulatory bodies or industry standards organizations. Obtaining security and privacy certifications is essential to demonstrate compliance with these standards, reassuring users of the parser's commitment to data security and privacy protection.

5.4 Software Quality Attributes

The Malayalam parser for dataset creation must prioritize correctness and reliability, ensuring accurate parsing and consistent results. Maintainability is crucial for easy updates, while usability requires an intuitive interface. Portability enables the parser to run on different platforms, and interoperability allows integration with other NLP components. Testability ensures thorough testing, and adaptability allows for evolving requirements. These characteristics collectively ensure the parser's efficiency, effectiveness, and adaptability.

References

1. Asopa, S., & Sharma, N. (2021) A Hybrid Parser Model for Hindi Language. Indian Journal of Computer Science and Engineering (IJCSE), Vol. 12(1).

2. Chen, D., & Manning, C. D. (2014). A Fast and Accurate Dependency Parser using Neural Networks. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).

3. Nair, L. R. (2013). Language Parsing and Syntax of Malayalam Language. 2nd International Symposium on Computer, Communication, Control and Automation (3CA 2013).

4. Berger, A. L., Della Pietra, V. J., & Della Pietra, S. A. (1996). A Maximum Entropy Approach to Natural Language Processing. Association for Computational Linguistics, Vol 22(1).

5. Mestry, A., Shende, S., Mahadik, A., & Virnodkar, S. (2014). A Parser: Simple English Sentence Detector and Correction. International Journal of Engineering Research and Technology (IJERT).

6. Sethi, N., Agrawal, P., Madaan, V., & Singh, S. K. (2016). A Novel Approach to Paraphrase Hindi Sentences using Natural Language Processing. Indian Journal of Science and Technology, Vol 9(28).

7. Smith, D. A., & Eisner, J. (2008). Dependency Parsing by Belief Propagation. Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, Page 145-156.

8. Bharati, A., Kulkarni, A., & Chaudhury, S. (2007). English Parsers: Some Information-based Observations.

9. Jayan, J. P., & R, R. (2009). A Morphological Analyzer for Malayalam - A Comparison of Different Approaches. International Journal of Computer Science and Information Technology. Vol 2(2), Page 155-160.

10. Vaidya, A., Choi, J. D., Palmer, M., & Narasimhan, B. (2011). Analysis of the Hindi Proposition Bank using Dependency Structure. Proceedings of the Fifth Law Workshop (LAW V), Page 21-29.

11. Rajan, M., T.S, R., & Bhojane, V. (2014). Information Retrieval in Malayalam Using Natural Language Processing. International Journal of Scientific & Engineering Research, Vol 5(6)

12. Rajan, M., Thirumalai, R., & Kumar, V. (2006). Development of a Tamil Parser using Natural Language Processing Techniques. A survey of the state of the art in tamil language technology Vol 6(10).

13. Venkatesh, R., Kumar, S., & Arumugam, P. (2014). Building a Lexical Analyzer for Tamil Texts using NLP Approaches. 2014 International Conference on Advances in ICT for Emerging Regions (ICTer).

14. Thavareesan, S., & Mahesan, S. (2019). Sentiment Analysis in Tamil Texts: A Study on Machine Learning Techniques and Feature Representation.2019 IEEE 14th Conference on Industrial and Information Systems (ICIIS).

15. Pai, T. V., Devi, J. A., & Aithal, P. S. (2020). A Systematic Literature Review of Lexical Analyzer Implementation Techniques in Compiler Design. International Journal of Applied Engineering and Management Letters (IJAEML), Vol 4(2), Page 285-301.

16. Simmons, R. F., & Burger, J. F. (1968). A Semantic Analyzer for English Sentences. Mechanical Translation and Computational Linguistics, Vol 11.