ABSTRACT

Marathi is an Indo-Aryan Language and forms the official language of the state of Maharashtra. It is ranked as the 4th most spoken language in India and the 15th most spoken language in the world. When Computational linguistics is concerned, writing grammar production for a language is a bit difficult because of different gender and number forms. This is an effort to write context-free grammar for Marathi sentences. CFGs are very much suitable for expressing natural languages as well as programming languages and hence form a major part of the field of natural language processing and pattern recognition. Computational linguistics is one of the attractive research topics in the Natural Language Processing (NLP) and Artificial Intelligence domains present a particular syntactical parsing technique for Marathi texts which is one of the Indian languages. Cocke–Younger–Kasami (CYK) parsing technique has been adopted to parse Marathi sentences and identify their grammatical structure. Currently, very less NLP tools are available to parse several Indian languages. Hence, an effort has been made by us to efficiently parse the structure of the complex sentences in Marathi text using the CYK algorithm. It is a bottom-up dynamic programming approach that functions only with the grammar in Chomsky's normal form (CNF).

Keywords:

- Natural language processing
- CYK algorithm
- Syntactical parsing
- Chomsky normal form
- Production rules
- Rule-based grammar

INDEX

CHAPTER I

1. Introduction & Motivation

Introduction:

The Parser is tools which used to analysis the sentence in term of its constituent parts. A parser aims to generate automatic syntactic trees of natural language. In filed of computational linguistic, natural language processing language and artificial intelligent have two kind grammar formalism which phrase structure grammar and dependency grammar. Those two grammar mechanism are useful to develop a Parser. Today English language have phrase structure grammar formalism and dependency grammar formalism to develop parser and those two grammar formalism are provide good accuracy. When we apply those two grammar formalism to Indian languages, than we can see dependency grammar is provide good accuracy compared to phrase structure grammar. The reason is simple, English language have positional word order structure and most of the Indian languages have free word order structure and morphological rich Development of a parser is a challenging task for morphological rich and free word languages such as Indian languages.

Motivation:

A very little attempt has been made to develop a syntactical parser on Indian languages including Marathi. Marathi is a spoken as well as a written language in Maharashtra state and it is basically Used all over Maharashtra. We made an effort to develop a systematic syntax analyzer to parse all types of Marathi texts, and the same has been presented in the paper. Python programming language has been used and the NLTK tree module has also been imported to serve the purpose. NLTK tree module has been adopted to display the parser tree for every valid Marathi sentence. Syntax analysis is the process of checking the grammatical correctness of a sentence and identifying the relationship between the words in an input sentence. It is very simple to write the production rules or grammar for fixed order spoke languages compared to other free order languages.

Problem Definition:

To design a context free grammer based parser for marathi text using cyk algorithm.

Objectives:

- To Select a marathi corpus.
- Pre-processing of corpus.
- To tokenize a sentences.
- To identify parts of speech.
- To identify features, for example removing noise, special characters, emoji.
- To set lexical based grammar rules and occurrence frequency of words.

Summary I: Introduction, Motivation, Problem statement, and objectives

The Parser is tools which used to analysis the sentence in term of its constituent parts. A parser aims to generate automatic syntactic trees of natural language A very little attempt has been made to develop a syntactical parser on Indian languages including Marathi. Marathi is a spoken as well as a written language in Maharashtra state and it is basically Used all over Maharashtra. We made an effort to develop a systematic syntax analyzer to parse all types of Marathi texts, and the same has been presented in the paper.

Currently, very less NLP tools are available to parse several Indian languages. Hence, an effort has been made by us to efficiently parse the structure of the complex sentences in Marathi text using the CYK algorithm.To display the parser tree for every valid Marathi sentence NLTK tree module is used.

To develop efficient dynamic programming technique and suitable for analysing complex sentences CYK Algorithm is used.

CHAPTER II

2. Literature Survey

Syntactic Parsing in Kannada Text/Natural Language Processing

Methodology:

An input sentence is processed by a parser according to the productions of a grammar and builds one or more constituent structures that satisfy the grammar. A parser is a procedural interpretation of the grammar. It permits a grammar to be evaluated against collection of test sentences, helping linguists to find mistakes in their grammatical analysis. The CYK parser model proposed CFG grammar and converts into CNF format if required. The Kannada dataset which contains different kinds like imperative, declarative, assertive, compound sentences, and interrogative sentences has been given as input to the parser model. Each word is treated as a terminal symbol. These terminals are matched with the production rules in the CNF grammar and try to fetch the corresponding productions by RHS. This continues to identify the respective LHS of the productions recursively through the bottom-up approach until it gets the start symbol (S). Finally, if the start symbol of the grammar is not obtained, then the input sentence given is not accepted by the grammar. If the start symbol is obtained, it outputs with the NLTK parse tree. For some sentences, it may create more than one parse tree when parsing technique gets the same start symbol through different production rules.

Parsing for Natural Language in Odia: a novel study

Methodology:

Phrases of grammar as well as in lexicon using syntactic analysis which can be described by using the term parsing.

The emerging syntactic analysis can be used as input to a system of semantic interpretation occasionally. Semantic analysis and syntactic analysis both can be included and can get the use of parsing. The parsing output is something logically equivalent to a tree in the current linguistic formalism resulting part of sentence relation between dominance and precedence. In linguistic description the similar annotations of attribute-fee equations (features) is in the form of capturing other elements. There are many approaches for representing the distinct viable linguistic formalisms resulting in many distinct approaches for parsing to represent the consequences. To anticipate a simple tree representation, an implicit in context-free grammatical formalism. All the algorithms can be defined and can be used for more effective unification based formalisms to provide the preserve of context-loose backbone but in complexity and termination the properties may be different. Tailor-made grammar can be replaced with the training grammar with the help of parsing algorithms. The parsing algorithm has to have the several important properties if it could be used for practically. If it assigns an input of sentence it should do all the analysis concerning the current grammar and lexicon then it should be complete. The algorithm must be fully efficient, inevitable the negligible of computational work accurate with satisfying both the requirements and robust. Acts in a fairly pragmatic manner when presented with a sentence that it's far impotent to fully analyse successfully. Context-free grammar is usually call them as phrases primarily based on the phrase that heads the constituent. In each production.

Indonesian Parsing Using Probabilistic Context Free Grammar and CYK

Methodology:

Parsing is a tool for understanding natural grammar patterns. The problem of structural ambiguity in identifying sentence patterns often occurs in parsing. Syntactic parsing is one approach to solving structural ambiguity problems using the Probabilistic Context-Free Grammar (PCFG) and Viterbi- Cocke Younger Kasami (Viterbi-CYK) methods. Meanwhile, a large number of Indonesian language resources are needed as machine knowledge to parse. This research build a parsing of Indonesian sentence patterns with Indonesian Tagged corpus resource then solve the ambiguity problem of Indonesian sentence pattern parsing using PCFG and Viterbi-CYK algorithms. The corpus data is processed to obtain grammar rules using the PCFG algorithm. Then, the sentence on the corpus is processed by the PCFG rule that generated and uses the Viterbi-CYK algorithm to get the parse tree taken based on the highest probability value. The results of the research produced an average value of similarity production rules which the highest values is 92.95%. This shows that the Indonesian parsing successfully parses Indonesian sentence and can solve the problem of structural ambiguity in the parsing of Indonesian sentence patterns.

Advantages of Existing Work:

1. Simplicity. Users agree with Parse's claim that it is intuitively simple to use. Queries are easy, and so is generating data cache.

2. Automation. Image cache is performed automatically. Automatic email verifications and password resets are also built in.

3. Speed. Basic functions are available instantly. Development moves fast because the Parse class for showing data in a table, PFQuery / TableViewController, is as quick as they come.

4. Freebies. You can do a lot with the options in Parse's free tier! Push notifications to one million devices per month costs a whopping zero amount of money! 30 requests per second (which equals 1800 a minute), whether that request to the app store sends or receives data, is also free.

5. Multiplatform. Parse has SDKs for almost every platform. It can handle mobile apps, desktop apps and just about any and all Internet of Things technology.

Drawbacks in the existing Literature:

1. Tricky Relationships. Users agree that the relationship between classes / tables are sometimes too complicated, or are difficult to maintain.

2. Service Limitations. At this time Parse does not support services like streaming. Parse users worldwide include Facebook (which now owns Parse), Oculus, AMC, Samsung, EBay, Cisco, MTV, Orbitz, Showtime, Gucci, and the White House.CodigoDelSur is also a proud and experienced user of Parse! If you need help developing your own app, CodigoDelSur is happy to take advantage of Parse's simplicity, automation and speed to make your JavaScript, iOS or Android product the best it can be!

Summary II: Literature Survey

After studying the Literature review, we concluded some advantages & drawbacks which helped in more specifying our project in the best way possible. A couple of papers reveal the cumulative results of syntactical parser.

CHAPTER III

3. Proposed System and Requirement Specification

3.1 Proposed System

Proposed system structure. First, parser software analyses text documents. Then, the similarity estimator method and mining process is used to identify text similarity and cluster documents.

Role of parser:

In the syntax analysis phase, a compiler verifies whether or not the tokens generated by the lexical analyser are grouped according to the syntactic rules of the language. This is done by a parser. The parser obtains a string of tokens from the lexical analyser and verifies that the string can be the grammar for the source language. It detects and reports any syntax errors and produces a parse tree from which intermediate code can be generated.

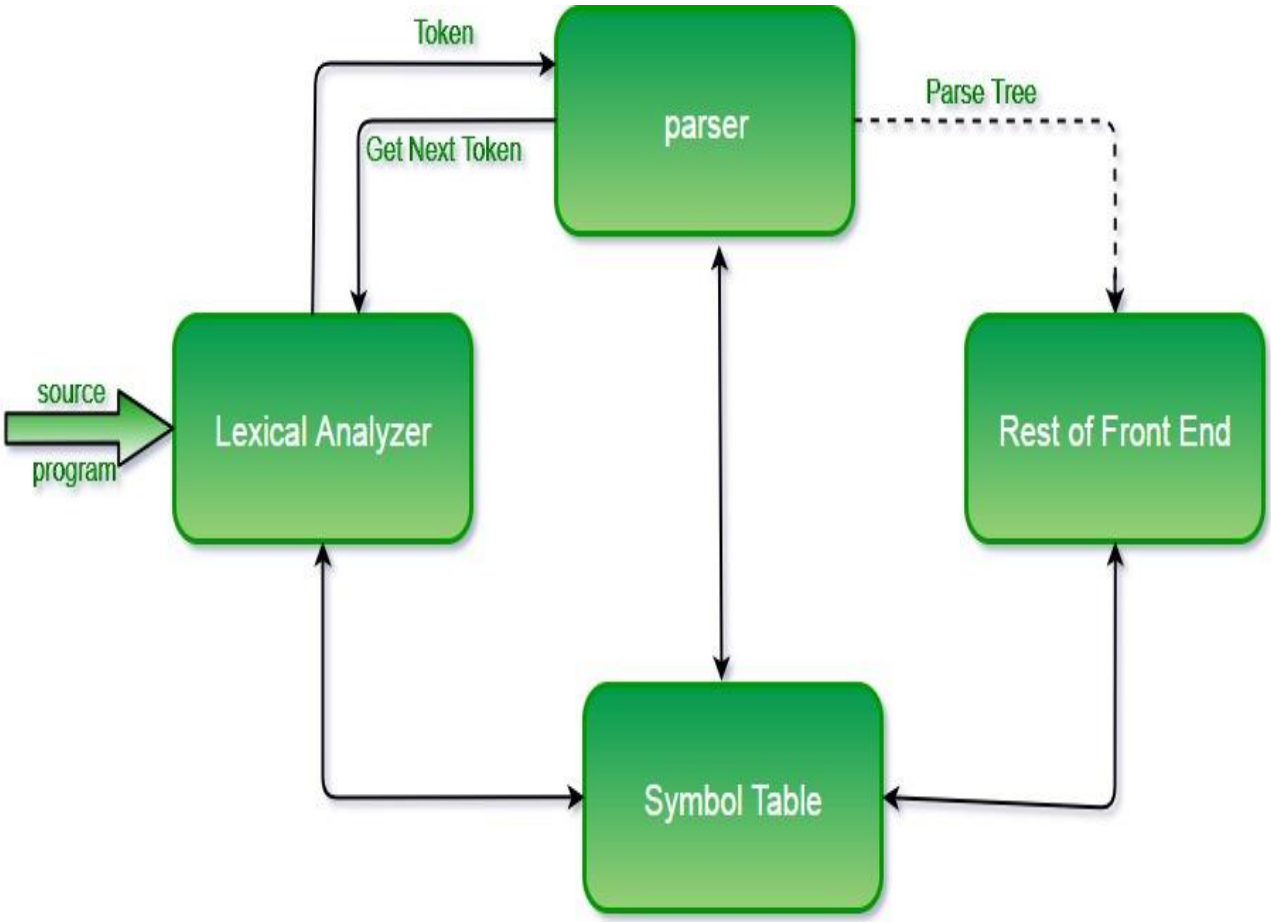


Fig. Role of parser

3.2 Methodology used:

An input sentence is processed by a parser according to the productions of a grammar and builds one or more constituent structures that satisfy the grammar. A parser is a procedural interpretation of the grammar. It permits a grammar to be evaluated against a collection of test sentences, helping linguists to find mistakes in their grammatical analysis. The CYK parser model proposed CFG grammar and converts into CNF format if required. The Maharashtrian dataset which contains different kinds like imperative, declarative, assertive, compound sentences, and interrogative sentences has been given as input to the parser model. Each word is treated as a terminal symbol. These terminals are matched with the production rules in the CNF grammar and try to fetch the corresponding productions by RHS. This continues to identify the respective LHS of the productions recursively through the bottom-up approach until it gets the start symbol(S). Finally, if the start symbol of the grammar is not obtained, then the input sentence given is not accepted by the grammar. If the start symbol is obtained, it outputs with the NLTK parse tree. For some sentences, it may create more than one parse tree when parsing technique gets the same start symbol through different production rules. The simple parsers suffer from limitations in both completeness and efficiency. In order to overcome these, dynamic programming technique has been applied here to resolve the parsing problem. Dynamic parsing algorithm accumulates the intermediate results and reuses them when relevant, achieving efficiency gains. This technique can be applied to syntactical parsing, allowing us to store partial solutions during the parsing task and then look them up as necessary in order to efficiently arrive at a complete solution. This approach to parsing is known as chart parsing. A typical sentence has been taken as an example to present the proposed technique. It is very difficult to design a grammar for the entire Marathi language. For the sake of simplicity, this paper considers writing very simple grammatical productions with CFG that will generate a subset of Marathi sentences.

Marathi parts of speech tag with example:

Sl. No	Category	Label	Annotation Convention **	Examples
	Top level & Subtype			
1	**Noun (नाम)**	**N**	**N**	
1.1	Common (जातीवाचक नाम)	NN	N_NN	**गाय**\N_NN **गोठ्यात**\N_NN राहते.
1.2	Proper (व्यक्तीवाचक नाम)	NNP	N_NNP	**रामाने**\N_NNP **रावणाला**\N_NNP मारले.
1.3	Nloc (स्थल-काल)	NST	N_NST	1. तो **घरी**\N_NST काम करत होता. 2. त्याने ही वस्तू **खाली**\N_NST ठेवली आहे.
2	**Pronoun (सर्वनाम)**	**PR**	**PR**	
2.1	Personal (पुरुष वाचक)	PRP	PR_PRP	**मी**\PR_PRP येतो.
2.2	Reflexive (आत्म वाचक)	PRF	PR_PRF	**मी स्वत:**\PR_PRF आलो.
2.3	Relative (संबंधी)	PRL	PR_PRL	**ज्याने**\PR_PRL हे सांगितले त्याने हे काम केले पाहिजे.
2.4	Reciprocal (पारस्परिक)	PRC	PR_PRC	**परस्पर**
2.5	Wh-word (प्रश्नार्थक)	PRQ	PR_PRQ	**कोण**\PR_PRQ येत आहे?
2.6	Indefinite (अनिश्चित)	PRI	PR_PRI	**कोणी**\PR_PRI **कोणास**\PR_PRI हासू नये. त्या पेटीत **काय**\PR_PRI आहे ते सांगा.
3	**Demonstrative (दर्शक)**	**DM**	**DM**	हे पुस्तक माझे आहे.
3.1	Deictic	DMD	DM_DMD	**तो**\DM_DMD मुलगा हुशार आहे. **हा**\DM_DMD मुलगा हुशार आहे. **ही**\DM_DMD मुलगी सुंदर आहे. **जेथे**\DM_DMD राम होता **तेथे**\DM_DMD तो होता.
3.2	Relative	DMR	DM_DMR	**हे**\DM_DMR लाल रंगाचे असते.
3.3	Wh-word	DMQ	DM_DMQ	**कोणता**\DM_DMQ मुलगा हुशार आहे?
4	**Verb (क्रियापद)**	**V**	**V**	
4.1	Main (मुख्य क्रियापद)	VM	V_VM	तो घरी **गेला**\V_VM.
4.2	Auxiliary (सहाय्यक क्रियापद	VAUX	V_VAUX	राम घरी जात **आहे**\V_VAUX.
5	**Adjective (विशेषण)**	**JJ**		**सुंदर**\JJ मुलगी
6	**Adverb (क्रियाविशेषण)**	**RB**		**हळूहळू**\RB चाल.
7	**Conjunction (उभयान्वयी अव्यय)**	CC	CC	
7.1	Coordinator	CCD	CC_CCD	तो **आणि**\ CC_CCD मी.
7.2	Subordinator	CCS	CC_CCS	**जर**\CC_CCS त्याने सांगितले असते **तर**\CC_CCS हे काम मी केले असते.
7.2.1	Quotative	UT	CC_CCS_UT	**असे**\CC_CCS_UT **म्हणून**\C_CCS_UT तो पुढे गेला.
8	**Particles**	**RP**	**RP**	
8.1	Default	RPD	RP_RPD	मी **तर**\RP_RPD खूप दमते.
8.2	Interjection (उद्गार वाचक)	INJ	RP_INJ	**अरेरे**\RP_INJ ! सचिनची विकेट झाली.
8.3	Intensifier (तीव्र वाचक)	INTF	RP_INTF	राम **खूप**\RP_INTF चांगला मुलगा आहे.
8.4	Negation (नकारात्मक)	NEG	RP_NEG	नको, न
9	**Quantifiers**	**QT**	**QT**	
9.1	General	QTF	QT_QTF	**थोडी**\QT_QTF साखर दा.
9.2	Cardinals	QTC	QT_QTC	मला **एक**\QT_QTC गोळी दे.
9.3	Ordinals	QTO	QT_QTO	माझा **पहिला**\QT_QTO क्रमांक आला.
10	**Residuals (उर्वरित)**	**RD**	**RD**	
10.1	Foreign word	RDF	RD_RDF	
10.2	Symbol	SYM	RD_SYE	$, &, *, (,),
10.3	Punctuation	PUNC	RD_PUNC	. (period), ,(comma), ;(semi-colon), !(exclamation),? (question), : (colon), etc.
10.4	Unknown	UNK	RD_UNK	Not able to identify the Tag.
10.5	Echo-words	ECH	RD_ECH	जेवण बिवण, बोके बिके

3.3 Software Requirements Specification-SRS

System Features

- To Parse string of symbols, according to the rules of a Marathi grammar.
- To analyse the grammatical structure of natural language.
- To separate units like subject, verb, and object and determines the relations between these units.
- when there is a need to represent input data from source code abstractly as a data structure so that it can be checked for the correct syntax

Non-Functional Requirement

- **Performance Requirement:**

The performance of the system lies in the way it is handled. Every API user and end user must be given proper guidance regarding how to use the system.

- **Software Quality Attributes**:

Reliability - The system is fault tolerant and the user can check syntax easily.
Portability - The system will be an NLP system and can be installed on any system.
Usability - The system is easy to use, understand, and highly secure.

- **Business Rules:**

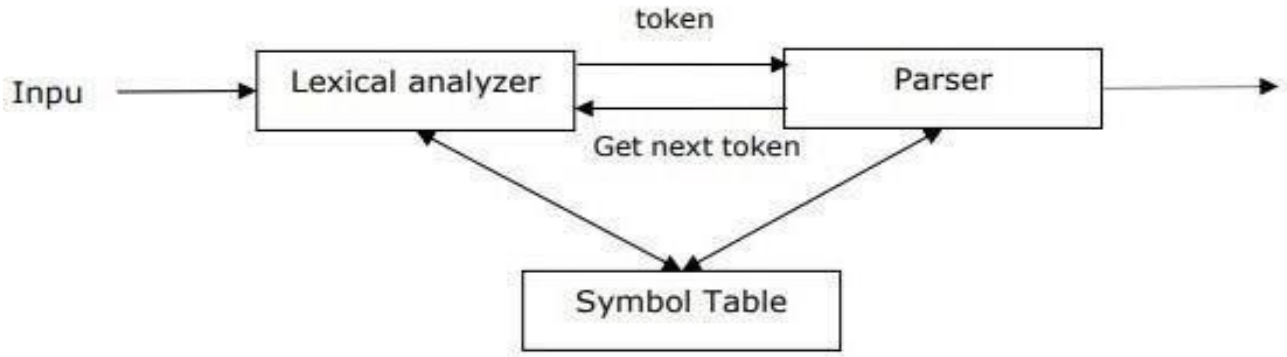
The parser module provides an interface to Python's internal parser and byte-code compiler. The primary purpose for this interface is to allow Python code to edit the parse tree of a Python expression and create executable code

3.4 Significance of the project

Syntactic analysis or parsing or syntax analysis is the third phase of NLP. The purpose of this phase is to draw exact meaning, or you can say dictionary meaning from the text. Syntax analysis checks the text for meaningfulness comparing to the rules of formal grammar. For example, the sentence like “hot ice-cream” would be rejected by semantic analyser.

In this sense, syntactic analysis or parsing may be defined as the process of analysing the strings of symbols in natural language conforming to the rules of formal grammar. The origin of the word **‘parsing’** is from Latin word **‘pars’** which means **‘part’**.

It is used to implement the task of parsing. It may be defined as the software component designed for taking input data (text) and giving structural representation of the input after checking for correct syntax as per formal grammar. It also builds a data structure generally in the form of parse tree or abstract syntax tree or other hierarchical structure.



The main roles of the parse include − •

To report any syntax error.

- To recover from commonly occurring error so that the processing of the remainder of program can be continued.
- To create parse tree.
- To create symbol table.
- To produce intermediate representations (IR).

3.5 Scope of Project

- To Select a marathi corpus.
- Pre-processing of corpus.
- To tokenize a sentences.
- To identify parts of speech.
- To identify features, for example removing noise, special characters, emoji.
- To set lexical based grammar rules and occurrence frequency of words.

3.6 Deployment Requirements

Software Requirement

- **Python**

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming.

- **Jupyter Notebook**

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. Its uses include data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning, and much more.

- **NLP**

Natural language processing is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyse large amounts of natural language data.

- **Kaggle Notebook**

Kaggle Notebook is a cloud computational environment which enables reproducible and collaborative analysis. Notebooks, previously known as kernels, help in exploring and running machine learning codes

- **NLTK**

The **Natural Language Toolkit**, or more commonly **NLTK**, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania.[4] NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit, plus a cookbook.

Hardware Requirement

- Windows 7 or higher
- I3 processor system or higher
- 4 GB RAM or higher
- 100 GB ROM or higher

3.7 Project Cost Estimate

The project cost estimate is null as of now but it would be altered if funds are utilised during the implementation of the later phases of the project

Summary III: Proposed System & Requirement Specification

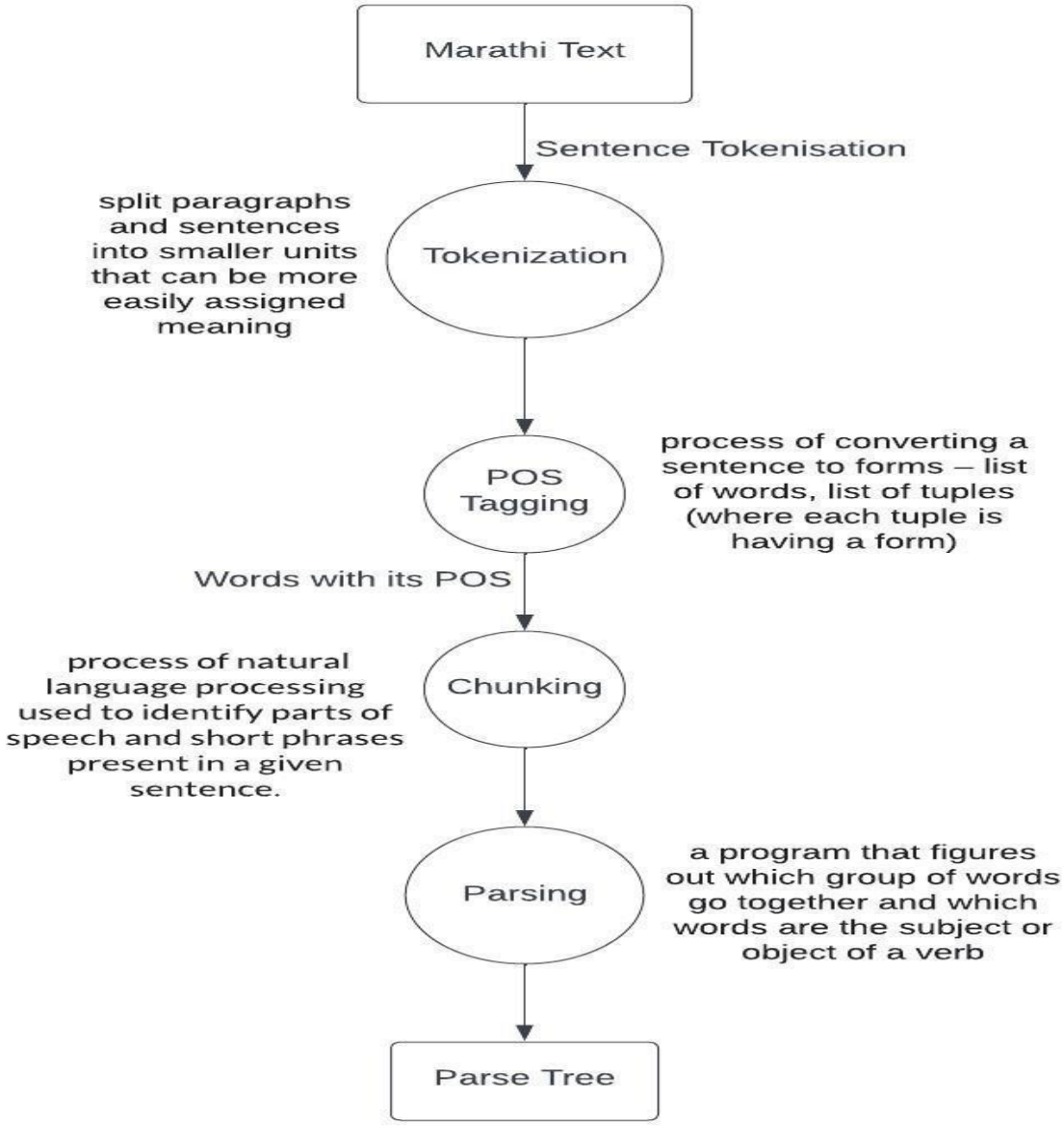
Proposed system structure. First, parser software analyses text documents. Then, the similarity estimator method and mining process is used to identify text similarity and cluster documents. An input sentence is processed by a parser according to the productions of a grammar and builds one or more constituent structures that satisfy the grammar. A parser is a procedural interpretation of the grammar. It permits a grammar to be evaluated against a collection of test sentences, helping linguists to find mistakes in their grammatical analysis**.**

Scope of project only include analyse the sequence of tokens to determine their grammatical structure.To Compute the parse tree corresponding to the input.

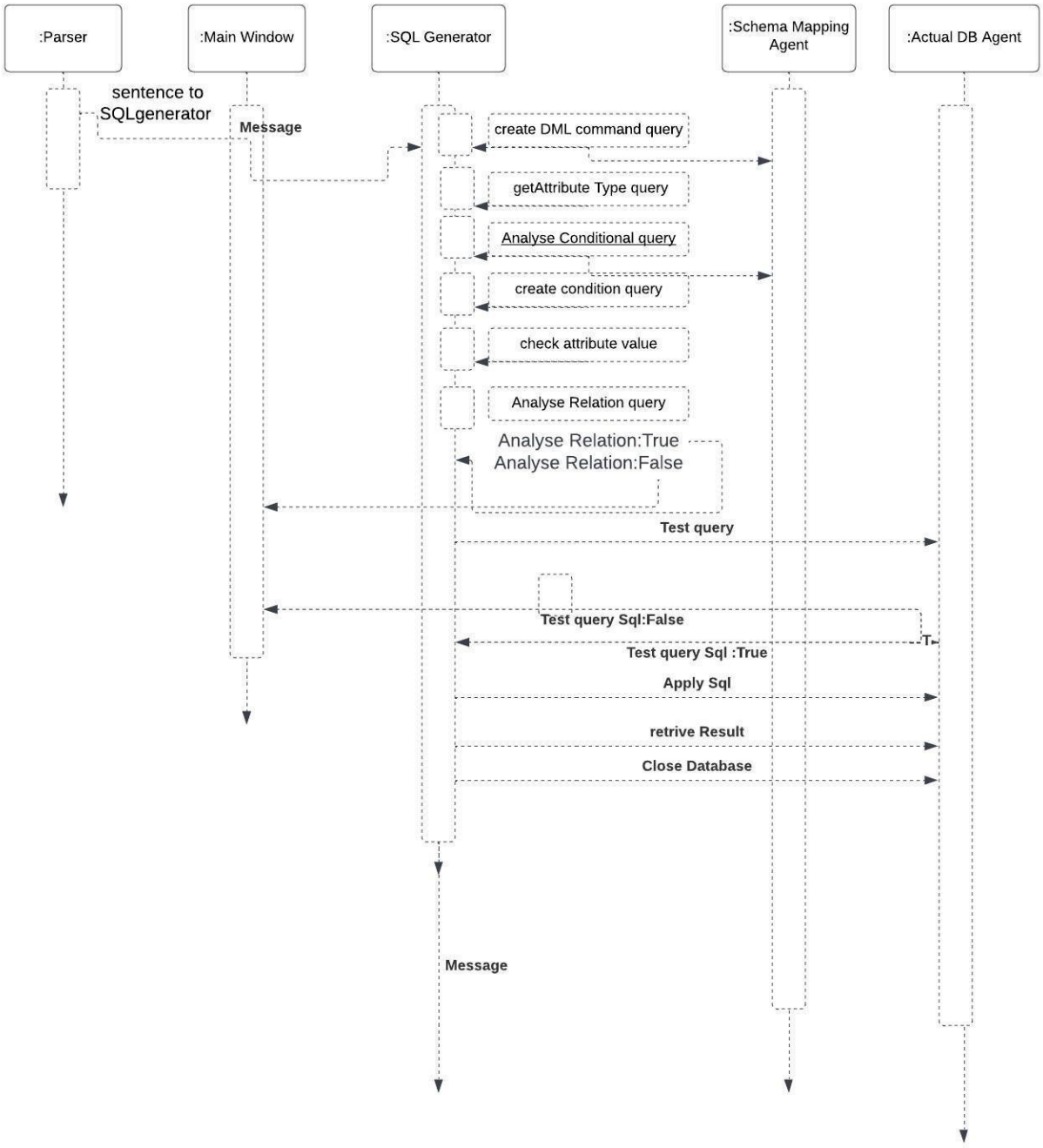
CHAPTER IV

4. Design

Data Flow Diagram



Sequence Diagram



Summary IV: Design

Here above diagram shows data flow diagram and sequence diagram.

CHAPTER V

5. Development/Implementation Details

Technologies used:

Artificial Intelligence:

Artificial intelligence (**AI**) is intelligence - perceiving, synthesizing and inferring information - demonstrated by machines, as opposed to intelligence displayed by animals and humans. OED (OUP) defines artificial intelligence as:

the theory and development of computer systems able to perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages. Several important sub-fields of AI research (as opposed to AI itself) have used working definitions of the intelligent agents field of study, which refers to any system that perceives its environment using an AI-component and takes actions (using procedural / hard-coded components) that maximize its chance of achieving its goals.

Machine Learning:

Machine learning (**ML**) is a field of inquiry devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, agriculture, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks. A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers, but not all machine learning is statistical learning.

NLTK:

The **Natural Language Toolkit**, or more commonly **NLTK**, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania. NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit, plus a cookbook.

NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning.

NLP:

Natural language processing (**NLP**) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyse large amounts of natural language data. The goal is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them. The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves.

Python:

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.

Tools Used

Python:

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.

Jupyter Notebook:

Project Jupyter is a project with goals to develop open-source software, open standards, and services for interactive computing across multiple programming languages. It was spun off from IPython in 2014 by Fernando Pérez and Brian Granger. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R. Its name and logo are an homage to Galileo's discovery of the moons of Jupiter, as documented in notebooks attributed to Galileo. Project Jupyter has developed and supported the interactive computing products Jupyter Notebook, JupyterHub, and JupyterLab. Jupyter is financially sponsored by NumFOCUS.

Summary V: Development/Implementation Details

Technology used are artificial intelligence, machine learning, nltk, nlp, python. Tools used are jupyter notebook and python.

CHAPTER VI

6. Results & Discussion

- The result of a parser function is a discriminated union that represents either success or failure. In both cases need the position field to indicate where in the input stream we are. This is field is defined in the base interface.
- When parsing succeeds we return the the Ok<T> object. It contains the result returned by the parser. Type parameter T indicates its type.
- When parsing fails we return the Fail object. It does not have a result, but information about the input found and input expected.

Summary VI: Results & Discussion

Syntactic parser includes the result parser function is a discriminated union that represents either success or failure.

CHAPTER VII

7. Conclusion & Future Work

The Syntactic Parser model on marathi language using CYK algorithm has been developed and presented in this report .It is an efficient dynamic programming technique and suitable for analysing complex marathi sentences . CYK technique follows bottomup approach and it is less prone to errors.

Future Work

The subsequent chapters of our project consist of the following activities:

- Determining the parser model to be used using juypter notebook.
- Forecasting results about the grammar of marathi text.
- Performing Exploratory syntactic analysis on the data in order to obtain various results about the data.
- Determining parse tree.

CHAPTER VIII

8. References

[1] M. Rajani Shree, "Syntactic Parsing in Kannada Text/Natural Language Processing", Third International Conference on Intelligent computing information and control systems ICICCS, March 2022.

[2] Das BR, Singh D, Bhoi PC, Priyadarshini P "Parsing for Natural Language in Odia: a novel study", International Conference on Computer Science, Engineering and Applications (ICCSEA), March 2020.

[3] Teodora Dordevic and Suzana Stojkovic, "Different Approaches in Serbian Language Parsing Using Contextfree Grammars, Proceedings of the second international conference on data science, E-learning and ınformation systems, Sep2020.

[4] Denis Eka Cahyani, Langlang Gumilar, Ajie pages "Indonesian Parsing Using Probabilistic ContextFree Grammer and CYK", Third International Seminar on Research of Technology and Intelligent System, Dec 2020.

[5] Pandey V, Padmavati MV, Kumar R "Parsing Chhattisgarhi languages using CYK Algorithm and grammar rules", Int J Pure Appl Math ISSN 2018.

[6] Gajanan Rane, Nilesh Joshi, Geetanjali Rane, Hanumant Redkar, Malhar Kulkarni and Pushpak Bhattacharyya "Part-of-Speech Annotation Challenges in Marathi", Indian Institute of Technology Bombay, Mumbai, India.