DESIGN AND DEVELOPMENT OF A NAMED ENTITY BASED QUESTION ANSWERING SYSTEM FOR MALAYALAM LANGUAGE

Thesis submitted by

Ms. Bindu.M.S

in fulfilment of the requirements for

the award of the degree of

DOCTOR OF PHILOSOPHY
under the
Faculty of Technology

DEPT. OF COMPUTER SCIENCE

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

KOCHI 682022

2012

Certificate

This is to certify that the thesis entitled "DESIGN AND DEVELOPMENT OF A NAMED ENTITY BASED QUESTION ANSWERING SYSTEM FOR MALAYALAM LANGUAGE" submitted to Cochin University of Science and Technology, in partial fulfilment of the requirements for the award of the Degree of Doctor of Philosophy in Computer Science is a record of original research work done by Mrs. Bindu.M.S (REG NO:3705), during the period (2005-2012) of her study in the Department of Computer Science at Cochin University of Science and Technology, under my supervision and guidance and the work has not formed the basis for the award of any Degree or Diploma.

Signature of the Guide

Kochi- 22
Date:

Declaration

I, Bindu.M.S hereby declare that the thesis entitled "DESIGN AND DEVELOPMENT OF A NAMED ENTITY BASED QUESTION ANSWERING SYSTEM FOR MALAYALAM LANGUAGE" submitted to Cochin University of Science and Technology, in partial fulfilment of the requirements for the award of the Degree of Doctor of Philosophy in Computer Science is a record of original and independent research work done by me during the period 2005-2012 under the supervision of Dr. Sumam Mary Idicula, Professor, Department of Computer Science, Cochin University of Science and Technology, and it has not formed the basis for the award of any Degree or Diploma.

Signature of the Candidate

Kochi-22

Date:

Dedicated To

My Lord and Saviour

Jesus Christ

Acknowledgement

This thesis would not have been possible without the help and support of many people:

First of all I thank God for strengthening me physically and mentally without stumbling before the difficulties I faced during this PhD work.

I express my sincere gratitude and indebtedness to my guide ***Dr.Sumam Mary Idicula,*** *Professor, Department of Computer Science, Cochin University of Science and Technology for her valuable advice and support throughout.*

I would like to express my heartfelt gratitude to ***Dr. K. Paulose Jacob,*** *Head, Department of Computer Science, Cochin University of Science and Technology for providing me all help and facilities from the department.*

I want to thank all the staff members of Computer Science Department who have facilitated me to complete this work.

I am highly obliged to all my friends and all those who have prayed for me.

Finally I would like to thank my husband and children for bearing my difficulties, problems and burdens for the last six years.

Bindu.M.S

ABSTRACT

This is a Named Entity Based Question Answering System for Malayalam Language. Although a vast amount of information is available today in digital form, no effective information access mechanism exists to provide humans with convenient information access. Information Retrieval and Question Answering systems are the two mechanisms available now for information access. Information systems typically return a long list of documents in response to a user's query which are to be skimmed by the user to determine whether they contain an answer. But a Question Answering System allows the user to state his/her information need as a natural language question and receives most appropriate answer in a word or a sentence or a paragraph.

This system is based on Named Entity Tagging and Question Classification. Document tagging extracts useful information from the documents which will be used in finding the answer to the question. Question Classification extracts useful information from the question to determine the type of the question and the way in which the question is to be answered. Various Machine Learning methods are used to tag the documents. Rule-Based Approach is used for Question Classification.

Malayalam belongs to the Dravidian family of languages and is one of the four major languages of this family. It is one of the 22 Scheduled Languages of India with official language status in the state of Kerala. It is spoken by 40 million people. Malayalam is a morphologically rich agglutinative language and relatively of free word order. Also Malayalam has a productive morphology that allows the creation of complex words which are often highly ambiguous.

Document tagging tools such as Parts-of-Speech Tagger, Phrase Chunker, Named Entity Tagger, and Compound Word Splitter are developed as a part of this research work. No such tools were available for Malayalam language. Finite State Transducer, High Order Conditional Random Field, Artificial Immunity

System Principles, and Support Vector Machines are the techniques used for the design of these document preprocessing tools.

This research work describes how the Named Entity is used to represent the documents. Single sentence questions are used to test the system. Overall Precision and Recall obtained are 88.5% and 85.9% respectively. This work can be extended in several directions. The coverage of non-factoid questions can be increased and also it can be extended to include open domain applications. Reference Resolution and Word Sense Disambiguation techniques are suggested as the future enhancements.

Contents

LIST OF TABLES

Page No

LIST OF FIGURES

Page No

Chapter 1

INTRODUCTION

Contents

This introductory chapter provides essential background to the area of Question Answering. General Architecture of a Question Answering System is discussed. It also provides the motivation behind this research work and concludes the chapter with a description on organization of this thesis.

Communication with computers is a dream of mankind since the beginning of computer era. Since its inception in 1960 several key developments had happened. The notable developments are Natural Language Database Front Ends, Dialog systems, and Natural Language Understanding systems. To have accurate and effective communication, computer must understand Natural Language. Also it must produce responses in a natural way. Natural Language Understanding (NLU) is a branch of Artificial Intelligence (AI) that deals with the issues concerned with man-machine interface [1]. Information Retrieval (IR) and Question Answering Systems (QAS) are two examples of NLU systems.

Information Retrieval, an example of human computer interaction, is the art and science of retrieving from a collection of documents a subset that serves the user's purpose [2]. A system which has the capability to synthesize an answer to a query by drawing on bodies of information which reside in various parts of the knowledge base is called a Question Answering System [2]. IR systems do not have such deduction capability.

1.1 What is Question Answering?

Natural Language Processing (NLP) is a theoretically motivated range of computational techniques for analysing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human like language processing for a range of tasks or applications. It began as a branch of Artificial Intelligence. NLP gives machines the ability to read and understand the human language [1].

Information Retrieval is the area of study concerned with searching of documents for information within documents and information about documents (metadata). Information Extraction (IE) is a type of IR and its goal is to automatically extract structured information from unstructured or semi structured machine readable format. In most cases IE requires Natural Language Processing [2].

But a Question Answering System aims at automatically finding concise answers to arbitrary questions phrased in natural language. Compared to standard document retrieval systems which just return relevant documents to a query, a QAS has to respond with a specific answer to Natural Language (NL) query. Traditionally IR concentrates on finding whole documents while QAS tries to provide only one or a small set of specific answers to an input question [2].

The idea of using computers to search for relevant pieces of information was popularized through the article “As We May Think” [3] by Vennevar Bush in 1945. Early IR systems came into existence in 1950s and

1960s. By 1970, several different techniques were evolved. The QA System JASPER [4] was built for providing real time financial news to financial traders. By the beginning of 1987, IE was spurred by a series of Message Understanding Conferences (MUC). MUC [5] is a competition based conference that focuses on the domains such as Naval, Terrorism, Satellite Launch etc.

A question may be either a linguistic expression or a request made by such an expression. Questions are normally asked using interrogatives. Questions can be of different forms. This research work considers Factoid, List, Definition, and Descriptive questions [6]. Factoids are those for which the answer is a single fact. List questions are factoid questions that require more than one answer. Unlike definition questions descriptive questions require a more complex answer, usually constructed from multiple source documents.

Answers are given in response to questions or request for information. There are many ways of describing an answer. As per Text REtrieval Conference (TREC-8) definition, answer size is restricted to a string of up to 50 or 250 characters. This work also restricts the answer size according to TREC norms [6].

1.2 General Architecture of a Question Answering System

The major goal of a QA System is to provide an accurate answer to a user's question. General Architecture of a QA System is shown in Fig 1.1. The question analysis stage analyses the NL question and determines the type of expected answer type. Based on the question analysis, a retrieval query is formulated and posed to the retrieval stage. The retrieval component returns a set of ranked list of documents which are further analysed by the document analyzer based on the expected answer type. This component passes a list of candidate answers to the answer selection module. This final stage returns a single answer or a sorted list of answers to the user.

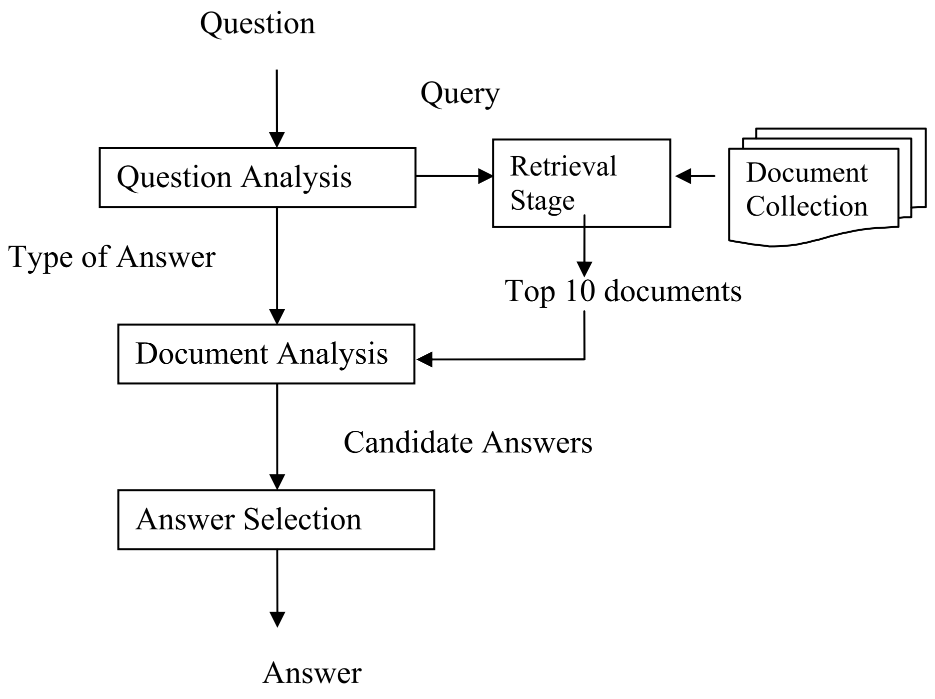


Fig 1.1 A Standard Question Answering System

1.3 Motivation and Scope

During Literature Survey it was noticed that no Question Answering System existed for Malayalam language. The QAS of other languages are not suitable for Malayalam due to the special complex features of this language. Most of the QAS available in other languages are only document retrieval systems whereas focus of this study is to develop an answer retrieval system for Malayalam. Many of the QAS existing in other languages employ keyword matching rather than Natural Language Understanding techniques.

Document preprocessing is an essential step in a QAS. But no tools were available for the preprocessing of Malayalam language. Hence preprocessing tools such as Part-of-Speech (POS) Tagger and Phrase Chunker are to be developed. Also these tools required word level analysis as most of the words in Malayalam language are compound words. Hence a Compound Word Splitter for Malayalam language was also essential without which evolution of above tools was impossible.

Named Entity (NE) is a meaning bearing word in a sentence. Meaning or semantics is an important point to be considered in a QAS. It is possible to include this fact in the development of NE-based index used for answer identification. But, to the best of our knowledge, a Malayalam NE Tagger was not available for this purpose.

This Question Answering System is designed to impart knowledge and insight in Malayalam to a naive user who knows only the regional language. Domain selected for this research work is the medical field dealing with the health issues, causes and remedies of lifestyle and infectious diseases.

The above ideas led to the formulation of certain important research questions like:

- How to develop a QAS for Malayalam language?
- How to store documents in a system?
- How to retrieve information?
- How to test and evaluate the system?

To solve the above given main research questions, following sub questions were also formulated

- What levels of NLP processing are required for this work such as morphological, syntactic, and semantic analysis?
- Which are the tools to be developed for the above NLP processing?
- What kind of document representation should be followed?
- How to analyse different types of Malayalam questions?
- What is the type of the answer to be returned, word, sentence or passage?
- What is the retrieval strategy to be followed?
- What are the metrics to be used for performance evaluation?

1.4 Objectives

As pointed out in the earlier section, this research work is an attempt to design and develop a closed domain, monolingual QAS which is capable of providing answers in a word/sentence for factoid and for a few non-factoid types of questions by the deep linguistic analysis of Malayalam documents in the corpus.

Therefore the main objectives of this research work are

- ❖ Conduct a detailed Literature Survey to understand the state of art in the field of QAS
- ❖ Development of language processing tools like
    - Compound Word Splitter
    - Parts-of-Speech Tagger
    - Phrase Chunker

  - Named Entity Tagger
- ❖ Creation of Malayalam Lexicon
- ❖ Collection and storage of Malayalam documents pertaining to lifestyle and infectious diseases
- ❖ Identification of pattern sets for question analysis
- ❖ Design of an answer retrieval scheme using double level index search.

1.5 Roadmap

This thesis is organized in ten chapters.

- **Chapter 1** provides the description of a standard Question Answering System.
- **Chapter 2** deals with the existing approaches to QAS. Also a few QAS available in other languages are discussed.
- An overview of Malayalam language is given in **chapter 3**.
- In **Chapter 4,** the architecture of Malayalam Question Answering System, MaQAS is described. MaQAS has three main modules, each one is explained with their design steps and working principles.
- A Compound Word Splitter using Finite State Transducer was developed and its details are given in **chapter 5**.
- **Chapter 6** describes Part-of-Speech Tagger, Tagset developed, methodology adopted and its performance evaluation.

- **Chapter 7** describes Phrase Chunker implementation. Artificial Immunity System Principles and its application for the development of Phrase Chunker are discussed in this chapter.
- Named Entity Tagger is described in **chapter 8**. Named Entity Tagset with 26 tags was identified for medical domain.
- Experimental environment and results are discussed in **Chapter 9.** Performance of MaQAS is mainly evaluated using the metrics precision and recall which is also explained in this chapter.
- **Chapter 10** concludes this work by summarizing the research achievements and suggesting directions for future research.

1.6 Chapter Summary

The background, motivation and objectives of the work are clearly mentioned in this chapter. General organization of a QAS is described. The chapter concludes with a layout of the thesis. Brief description of different approaches and types of QAS are discussed in chapter 2.

Chapter 2

LITERATURE SURVEY

Contents

Various approaches to Question Answering Systems are investigated. Both modern and earlier systems are discussed to have a clear distinction of their features, issues in their implementations, and the way in which they are handled.

Question Answering (QA), an important field of NLP, enables users to ask questions in natural language and get precise answers instead of long list of documents usually returned by search engines. When we trace back the history of computers we could come across the research stories and various developments that happened in various languages worldwide. QA Systems are also available in Indian languages such as Hindi [7], and Telugu [8]. However, no known work is available in Malayalam. Approaches to QA Systems and a few systems developed in various languages are described below.

2.1 Approaches to Question Answering Systems

QA System can be classified based on various factors such as domain of QA, language used for input query and the retrieved answer, types of question asked, kind of retrieved answers, levels of linguistics applied to the documents in the corpus, and the answer resources. Accordingly QA Systems fall into open domain or closed domain, monolingual or multilingual, factoid or non-factoid, document or passage or answer retrieval, deep or shallow, and database or Frequently Asked Questions (FAQ) or web QA Systems.

Open domain Question Answering System is an area of Natural Language Processing research, aimed at providing human users with a convenient and natural interface for accessing information. It deals with questions about nearly everything [9]. These systems usually have much more data available from which to extract the answer. ASKJEEVES [10] is the most well-known open domain QA System. To answer unrestricted questions, general ontologies and world knowledge would be useful. WordNet [11] and Cyc [12] are two popular general resources used in many systems. WordNet is a computational lexicon of English based on psycho-linguistic principles, created and maintained at Princeton University. It encodes concepts interms of sets of synonyms called synsets. Cyc is an AI project that attempt to assemble a comprehensive ontology, a knowledge base of everyday commonsense knowledge with the goal of enabling AI applications to perform human-like reasoning. Closed domain Question Answering Systems deal with questions under a specific domain (for example, medicine or automotive maintenance), and can be seen as an easier task because NLP systems can exploit domain-specific knowledge frequently formalized in ontologies [9]. Closed domain refers to a situation where only limited types of questions are accepted. In a closed domain QA, correct answers to a question may often be found in only very few documents since the system does not have large retrieval set. Green's BASEBALL [13] system is a restricted domain QA System that only answers questions about the US baseball league over a period of one year.

In a QA System, questions and answers are given in natural languages. Hence QA Systems can be characterized by the source (question) and the target (answer) languages. Based on these languages QA Systems are named as monolingual [14], multilingual or cross-lingual systems [15]. TREC QA [6] Track and New Trends in Computing and Informatics Research Question Answering Challenge (NTCIR QAC) [16] are all monolingual QA Systems which use the same source and target languages. Multilingual or cross lingual systems allow users to interact with machines in their own language thus providing easier and faster information access, but the documents in the corpus are in a different language. This idea emerged in the year 2000.

Question types can also be used to categorize QAS. Different question types may require different strategies to deal with them. There are three question types– Factoid, List and Description. Factoid QA is the simplest as the answers are named entities such as Location, Person, Organization etc. Some Factoid QA Systems return short passages as answers while others return exact answers. List QA is similar to Factoid QA except that a question may have more than one answer.

Description QA [17] is more complex because it needs answers that contain definitional information about the search term or describe some special events. Special summarization techniques are required to minimize the answer size.

Another classification is Shallow or Deep Systems, based on the level of processing applied to the questions and documents [18]. Some Shallow QAS use keyword based techniques to locate interesting passages and sentences from the retrieved documents based on the answer type. Ranking is then done based on syntactic features such as word order, location or similarity to query. But question reformulation is not sufficient for Deep QA; more sophisticated syntactic, semantic, and contextual processing must be performed to extract or construct the answer. These techniques include Named Entity Recognition and Classification (NERC).

The answer source is an important factor in designing a QA System. Databases are the most popular answer sources that store structured data. Structured Query Language (SQL) is used to retrieve data from databases. LUNAR [19] developed to answer NL questions about the geological analysis of rocks returned by the Apollo Moon Missions is an example of such a database system. The performance of this system was excellent in terms of accuracy achieved. FAQ represent another answer resource in various commercial/business customer service systems. FAQ systems only focus on processing input questions and matching them with FAQs. Like other systems they don't require question analysis and answer generation stages. For an input question, if an appropriate FAQ is found, then using lookup table method corresponding answer is retrieved [20]. Web QA uses search engines like Google, Yahoo, Alta-Vista etc. to retrieve web pages that contain answers to the questions. Some systems combine the web information with other answer resources to achieve better QA performance. The Web based QA Systems such as MULDER [21], NSIR [22], and ANSWERBUS [23] fall into the category of domain independent QA Systems while START [24] is referred as domain specific QA System. AQUAINT [25] corpus used in TREC QA Track consists of newswire text data drawn from three sources. These kinds of corpora are good sources for QA System research as the quality and quantity of the data are good. News Papers are good sources for open domain QA research as their contents are general and cover different domains.

Current QA Systems are either document/passage/sentence or answer retrieval systems. In these systems, operation starts when a user posts a question into the QA System. The QA System then analyses the question and finds one or more answer candidates from its input sources. Once the answer candidates are retrieved the QAS then evaluates the content of each one and scores them based on the quality of their content. For these QA Systems, output is a document or a passage or a sentence or the exact answer.

2.2 History of Question Answering Systems

Work on early QA Systems began in early 1960s. Two of the most famous early systems are SHRDLU [26] and ELIZA [27]. SHRDLU simulated the operation of a robot in a toy world (the "blocks world"), and it offered the possibility to ask the robot questions about the state of the world. Again, the strength of this system was the choice of a very specific domain and a very simple world with rules of physics that were easy to encode in a computer program. ELIZA, in contrast, simulated a conversation with a psychologist. ELIZA was able to converse on any topic by resorting to very simple rules that detected important words in the person's input. It had a very rudimentary way to answer questions, and on its own it led to the development of a series of chatter bots such as the ones that participate in the annual Loebner prize. These are examples of Dialog systems mostly influenced by Turing test suggested by Alan Turing [28].

Two of the most famous restricted domain QA Systems developed in the 1960s were BASEBALL and LUNAR. These systems were interfaced against databases. BASEBALL system developed by Green Chomsky and Laughery answered questions about the US baseball league over a period of one year. This was done by using shallow language parsing techniques. Another system similar to

BASEBALL was developed by Woods and was named LUNAR. Both QA Systems were very effective in their chosen domains. In fact, LUNAR was demonstrated at a lunar science convention in 1971 and it was able to answer 90% of the questions in its domain posed by people untrained on the system. Further restricted-domain QA Systems were developed in the following years. The common feature of all these systems is that they had a core database or knowledge system that was hand-written by experts of the chosen domain.

The 1970s and 1980s saw the development of comprehensive theories in computational linguistics, which led to the development of ambitious projects in text comprehension and question answering. One example of such a system was the Unix Consultant (UC), a system that answered questions pertaining to the UNIX operating system [29]. The system had a comprehensive hand-crafted knowledge base of its domain, and it aimed at phrasing the answer to accommodate various types of users. Another project was LILOG [30], a text-understanding system that operated on the domain of tourism information in a German city. The systems developed in the UC and LILOG projects never went past the stage of simple demonstrations, but they helped the development of theories on computational linguistics and reasoning [23].

Over a period of time many open domain QA Systems have been developed that allows questions on a multiple range of topics. Such systems included START, ANSERBUS, BrainBoost [31], EPHYRA [32] and Qualim [33]. START utilized a knowledge base to answer user's question. Knowledge base was first created automatically from unstructured Internet data. Then it was used to answer natural language questions.

With the increased popularity of QA Systems TREC started the QA track in 1999. In the late 1990s the annual Text Retrieval Conference included a question-answering track which has been running till today. Systems participating in this competition were expected to answer questions on any topic by searching a corpus of text that varied from year to year. This competition fostered research and development in open-domain text based question answering. The best system of the 2004 competition achieved 77% correct fact-based questions [34].

In 2007 the annual TREC included a blog data corpus for question answering. The blog data corpus contained both "clean" English as well as noisy text that include badly-formed English and spam. The introduction of noisy text moved the question answering to a more realistic setting. Real-life data is inherently noisy as people are less careful when writing in spontaneous media like blogs. In early years the TREC data corpus consisted of only newswire data that was very clean [35].

An increasing number of systems include the World Wide Web as one more corpus of text. Currently there is an increasing interest in the integration of question answering with web search. Ask.com is an early example of a system, which was followed in subsequent years by other natural language search engines. Google and Microsoft have also started to integrate question-answering facilities in their search engines. However, these tools mostly work by using shallow methods and return a list of documents.

2.3 Modern Question Answering Systems

Early systems mostly used keyword matching techniques while current systems are based on linguistic principles.

Information Retrieval community has investigated many different techniques to retrieve passages from large collections of documents for question answering. The work discussed in [36] quantitatively compares the impact of sliding windows and disjoint windows on the passage retrieval for question answering. For the TREC factoid QA task, retrieval of sliding windows outperforms retrieval of disjoint windows. For the task of retrieving answers to why-questions from wikipedia data, the best retrieval model is Term Frequency Inverse Document Frequency (TFIDF) and sliding windows give significantly better results than disjoint windows. Here experiment is conducted with three retrieval models, TFIDF, Okapi, and a language model based on the Killback-Leibler divergence [37].

In IR4QA system [38], first phase is query processing. This phase produces a set of keywords. They are passed to the retrieval model which outputs a list of relevant documents in order. The re-rank module adjusts the ranking of these relevant documents by considering various features such as frequency, position in paragraph, term's distribution etc. This system just provides satisfactory performance due to lack of query terms.

In the above systems, query processing units separate keywords and other unimportant words without considering any syntax or semantics of the question. First phase of the work described in [39] is natural language query processing which build the syntax representation of query, and transform it into a semantic representation using transformation rules. This phase is designed using fundamental ideas of W Cafe [40]. According to Chafe, syntax model is built on the concept of relationship between words such as object, subject, verb, and their Parts-of-Speech. The semantic model is built on Chafe's point of view about semantic structure; it is defined as the relationship between verb and its arguments. Syntax structure is transformed into semantic structures by the Semantic Deductive Generator component using predefined transformation rules. From the semantic representation model of query, database queries generator module will generate a set of database queries or SQL commands. These commands are executed to get results.

In [41] authors discuss a new model for question answering, which improved the two main modules of QA System, question processing and answer validation. This is an answer extraction model while previous systems were either document or passage retrieval systems. In this model, first of all questions are processed syntactically and semantically. This information is used to classify the question and to determine answer type. Then the query reformulation component converts the question into SQL statement. Then the search engine finds candidate answer documents and sends them to answer processing module to extract correct answer.

ASQA [42] is Question Answering System for complex questions. The question processing module of this system uses surface text patterns to retrieve a question's topic. Documents are indexed by character based Indexing scheme. This scheme is similar to the one used by the open source IR engine Lucene [43]. Lucene is a high performance, full featured text search engine library. Boolean search using AND, as well as OR, as keywords, are used to retrieve documents relevant to the query. After retrieving the documents they are split into several sentences. Sentence selection module uses co-occurrence based or entropy based methods to find relevant sentences. This system is a sentence retrieval system based on topics and uses Boolean strategy for retrieval where character index is adopted. Retrieval performance is only 65%. No syntax or semantic processing is used in this system.

A Question Answering System for Japanese is presented in [44]. First stage of this system is a passage selection module. Each passage is of size three consecutive sentences. After performing a preliminary analysis a passage selection algorithm is used for ranking all passages in each document and selects top N passages for further processing. Then each passage is scored using the count of query terms, their occurrence in the passages, and their inverse document frequency.

One of the important requirements for a QA System is to predict what type of answer the question requires: a person name, location or organization. In the above system they have defined 62 answer types and developed a method that classifies questions into the answer types using the LSP's (Lexico-Semantic Pattern). LSP is a pattern that is expressed by lexical entries, part-of-speech (POS), syntactic categories and semantic categories. Once answer type of a question is determined entities belonging to the answer type within the passages selected in the previous step are extracted. LSP grammar is constructed for this purpose. After extracting answer candidates, some of them are filtered out and the remaining answers are scored using a specific expression.

Work presented in [45] is a web based QA System which retrieves answers from web documents. The user's question is transformed into an IR query and delivered to the web search engines or ports. The retrieved documents are linguistically analysed and prepared a semantic representation. The semantic representations of questions and answers are compared to find answers.

DefArabicQA [46] is a definitional QA System for Arabic language. This system uses a pattern approach to identify exact and accurate definitions about organization using web resources. Question analysis module identifies expected answer type and topic using certain question patterns and interrogative pronoun of the question. The passage retrieval module collects the top n snippets retrieved by the web search engine. Then the definition extraction module extracts candidate definitions from these snippets based on the question topic. Definitions are identified with the help of lexical patterns. Definitions are ranked using statistical approach and top-5 definitions are presented to the user.

QArabPro [47] is a rule based QA System for Arabic. Question reformulation section process the input question and formulate the query. An IR system is used to search and retrieve relevant documents which are constructed using salton's statistical Vector Space Model (VSM). Then the rules for each WH questions are applied to the candidate document that contains the answer. Each rule awards a certain number of points to each sentence in the document. After applying the rules the sentence with the highest score is marked as the answer.

Samir Tartir et al. [48] presented a Hybrid Natural Language Question Answering System (NLQAS) on Scientific Linked Data sets as well as Scientific Literature in the form of publications. SemanticQA processes information need expressed in the form of NL query. Then it retrieves relevant answers from well established Linked Data Sets (LDS). If the answer is not found in LDS system it gathers all the relevant clues and conducts a semantic search on relevant publication. The answers are extracted from these documents and ranked using a novel measure the Semantic Answer score. This score returns the best answer from relevant documents.

A QA System for Portuguese language is described in [49]. Once the question is submitted, it is categorized according to question typology and through an internal query a set of potentially relevant documents is retrieved. Each document contains a list of sentences which were assigned the same category as the questions. Sentences are weighted according to their semantic relevance and similarity with the question. Next through specific answer patterns these sentences are again examined and the parts containing possible answers are extracted and weighted. Finally a single answer is chosen among all candidates.

MAYA [50] is a QA System for Korean language that uses a predictive answer indexer. Answer indexer extracts all answer candidates in a document in indexing time. Then it gives scores to the adjacent content words that are closely related with each answer candidate. Next it stores the weighted content words with each candidate into a database. During retrieval time MAYA just calculates the similarity between a user's query and the candidates. Therefore it minimizes the retrieval time and enhances the precision.

Log Answer [51] is QA System for German language. User enters a question into the interface and Log Answer presents a list of answers to the user. These are derived from an extensive knowledge base. This knowledge base is obtained by translating a snapshot of entire German Wikipedia into a semantic network representation in the Multi Net formalism. The question is analysed by linguistic methods and then translated into a Multi Net [52] and First Order Logic (FOL) representation. The Wikipedia contents are matched against the given query combining retrieval and shallow linguistic methods. They compute lists of features like the number of lexemes matching between passage and question or the occurrence of proper names in the passage. A Machine Learning based ranking technique uses these features to filter out the most promising text passages resulting in upto 200 text passages which might be relevant to the query. The FOL representation of these passages is individually tested by theorem prover E-KRHyper [53] each in conjunction with the background knowledge base and the logical query representation. The proofs are ranked by a classifier and the highest ranked proofs or candidates are translated back into NL answers that are displayed to the user.

RitsQA [54] is a system for non-factoid questions developed for Japanese language. Question analyzer analyses the question pattern and determines its type. IR module called Namazu is used to retrieve the top 100 documents. Also clue words are used to retrieve 10 snippets of Google search. Then similarities between these two results are measured to reorder the retrieved documents. Answer Extraction Module extracts paragraphs which include linguistic clues and some clue words of question sentence. Extracted paragraphs will be a target for answer strings.

Marsha QAS [55] is a Chinese Question Answering System. The query processing module recognizes known question types and formulates queries for the search engine. Most of these question types correspond to typical Named Entity classes used in IE systems.

2.4 QA Systems for Indian Languages

QA System described in [7] was developed for HINDI language. Here the question submitted by the user is analysed to identify its type. The question is then parsed to separate the important keywords by identifying the domain entities and filtering out stop words. The query formulation translates the question into a set of queries that is given to the retrieval engine. The engine returns top passages after weighting and ranking them. Finally answer selection is done by selected passage analysis. The retrieval process is carried out using a word level inverted index using all of the terms in the generated query. The selected documents are ranked by locality based similarity heuristic. The similarity between query and document is measured using the distance between the keywords.

Rami Reddy et al. [8] discusses a keyword based QA System for a huge domain (i.e. for Railways), which aims at replying user's questions in Telugu. Telugu is an important language in India belonging to the Dravidian family and spoken by the second large population in India. In this keyword based approach the input query statement is analysed by the query analyzer which uses domain ontology stored as knowledge base. The appropriate query frame is selected based on the keywords and the tokens in the query statement. Each query frame is associated with an SQL generation procedure which generates an SQL statement. Once the SQL statement is generated it is triggered on the database and the answer is retrieved. Each query frame has its corresponding answer generator. Template based answer generation method is used for answer generation. Each template consists of several slots. Those are filled by the retrieved answer and tokens generated from the query. The answer will be sent to the Dialogue Manager which will further send it to the user. This system showed 96.34% of precision and 88.66% of dialogue success rate.

2.5 Named Entity Based QA Systems

Some NE based QA Systems are described in this section.

The main objective of QA4MRE [56] is to develop a methodology for evaluating machine reading systems through question answering and reading comprehension tests. Machine reading task obtains an in-depth understanding of just one or a small number of texts. The task focuses on the reading of single documents and identification of the correct answer to a question from a set of possible answer options. The Conditional Random Field (CRF) -based Stanford

Named Entity Tagger 5 (NE Tagger) has been used to identify and mark the named entities in the documents and queries.

The system discussed in [57] is an Answer Retrieval system based on named entities. Here the NL question entered by the user is analysed and processed which determines the kind of answer expected. The first phase is a document retrieval phase that finds documents relevant to the question. Next is the sentence selection phase. From the relevant documents found by the first phase, all sentences are scored against the question. The sentences remaining after the sentence selection phase are then analysed for named entities. All named entities found in the sentences are considered to be possible answers to the user question. The best answer (i.e. with the highest score and matching the question type) is returned to the user. Instead of a list of relevant documents, this QA System tries to find an exact answer to the question

ArabiQA [58] is a named entity based QA System for Arabic language. Question analysis module of this system determines the type of the given question, question keywords and the named entities appearing in the question. Passage retrieval module retrieves passages which are estimated as relevant to contain the answer. It uses a Distance Density Model to compare the n-grams extracted from the question and the passage to determine the relevant passages. Java Information Retrieval System (JIRS) searches for relevant passages and assigns a weight to each one of them. The weight of a passage depends mainly on the relevant question terms appearing in the passage. Named Entity Recognition system tags all named entities within the relevant passage. Candidate answers are selected eliminating NE which do not correspond to the expected type of answer. Final list of candidate answers is decided by means of a set of patterns.

IRSAW [59] is a system that combines IR with a deep linguistic analysis of texts to obtain answers to NL questions in German language. The NL question is transformed into an IR query and Meta information such as question type and expected answer type is determined. Question and answer types are calculated using a Naïve Bayer's Classifier trained on features representing the first N words of the question [60]. Answer types are Locations (LOC), Persons (PER), Organizations (ORG) etc. Question types include yes-no questions, essay questions and questions starting with WH words. IR query is sent to external web sources which return result pages containing Uniform Resource Locators (URL). The web contents referred to by an URL are retrieved and converted into text. These texts are segmented into units and indexed and fed into the local database. Several methods are employed to pinpoint answers. In the InSicht subsystem a linguistic parser analyses the text segments and prepares semantic network representation. Then the representations of questions and texts are compared to find answers. The shallow technique for finding answers in IRSAW is based on pattern matching. Each word in the passage is assigned a set of tags including Part-of-Speech. These sequences of symbols are analysed using context window to locate certain patterns. The pattern matching returns an instantiation of the answer variables.

TextractQA [61] explains the role of IE in a QA application. There are two components in this system, the question processor and text processor. The question processing results are a list of keywords plus the information for asking point. The question processor scans the question to search for question words and maps them into corresponding NE types. On the text processing side the question is first sent to a search engine and obtains top 200 documents for further IE processing. This processing includes tokenization, POS Tagging, and NE Tagging. Then the text matcher attempts to match the question template with the processed documents for both the asking point and the keywords. There are three levels of ranking schemes. Primary ranking is a count of how many unique keywords are contained within a sentence. The secondary ranking is based on the order in which the keywords appear in the sentence compared to their order in the question. Tertiary ranking is based on whether there is an exact match or a variant match for the key verb.

2.6 Chapter Summary

QA Systems are categorized based on various input and output factors. The origin of QAS, and several systems developed later are described in detail. Certain QA Systems developed were mainly different in the methodology used and in the output. Even though many systems are available in English and European languages, no such systems are available in Malayalam. Since this research work is a NE based QA System, similar systems are also discussed in this chapter.

Chapter 3

OVERVIEW OF MALAYALAM LANGUAGE

Contents

Malayalam is the principal language of Kerala, the southern most state of India. The word Malayalam probably originated from the Malayalam/Tamil words "mala" meaning hill, and "elam" meaning region. The word "malaelam" (hill region) was used to refer to the land of Chera Kingdom. Kerala was a part of ancient Chera Kingdom and when Kerala became a separate entity "malaelam" became the name of its language ie "Malayalam". The name "Kerala" was derived from the word "Cheralam".

Dravidian Languages were first recognized as an independent family in 1816 by Francis W Ellis, a British Civil servant. The term Dravidian (adjective form of Dravida) was first employed by Robert A Caldwell. Dravidian Languages, a family of some 75 languages is spoken primarily in South Asia. These languages are divided into South, South-Central, Central and North groups; these groups are further organized into 24 subgroups. The four major literary languages – Telugu, Tamil, Malayalam, and Kannada – are recognized by the constitution of India [62].

Malayalam belongs to the Dravidian family of languages and is spoken by the people of Kerala. It is one of the 22 Scheduled Languages of India with official language status in the State of Kerala and Lakshadweep Islands. It is spoken by about 40 million people. In terms of the number of speakers Malayalam ranks eighth among the fifteen major languages of India [63].

Malayalam first appeared in writing in the *vazappalli* inscription which dates back about 830 AD. The ancient Malayalam script originated in 13th century from a script known as *vattezhuthu* (round writing) a descendant of the Brahmiscript. Now Malayalam character set consists of 73 basic letters [64] [65].

Malayalam is a morphologically rich agglutinative language and is relatively of free order. Also Malayalam has a productive morphology that allows the creation of complex words which are often highly ambiguous. Due to its complexity, development of an NLP system for Malayalam is a tedious and time consuming task. No tagged corpus or tag set is available for this language. NLP systems developed in other languages are not suitable for Malayalam language due to its differences in morphology, syntax, and lexical semantics.

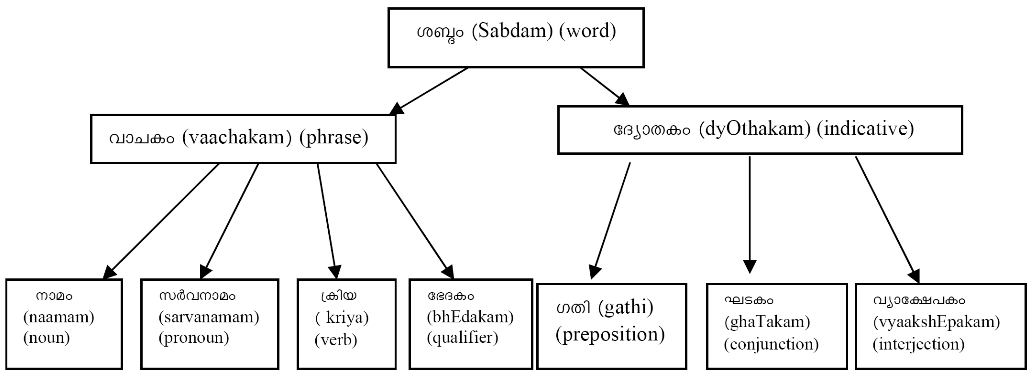

ശബ്ദം (Sabdam) (word)
വാചകം (vaachakam) (phrase)
ദ്യോതകം (dyOthakam) (indicative)
നാമം (naamam) (noun)
സർവനാമം (sarvanamam) (pronoun)
ക്രിയ (kriya) (verb)
ഭേദകം (bhEdakam) (qualifier)
ഗതി (gathi) (preposition)
ഘടകം (ghaTakam) (conjunction)
വ്യാക്ഷേപകം (vyaakshEpakam) (interjection)

Fig 3.1 Basic Parts-of-Speech in Malayalam Language

3.1 Basic Word Types

According to Keralapanineeyam written by Sri. A R Rajarajavarma [66], a Malayalam word may fall into one of the categories given in Fig 3.1.

In this thesis, words in Malayalam are represented in three forms– using Malayalam font, transliterated version, and in English where transliterated version is given in italics.

As given in Fig 3.1 Malayalam words (Sabdams) are classified as ‘vaachakam’ and ‘dyOthakam’. ‘vaachakam’ is further classified into namam (noun), sarvanamam (pronoun), kriya (verb), and bhEdakam (qualifier). ‘dyOthakam’ has three sub-categories namely gathi (preposition), ghaTakam (conjunction), and vyaakshEpakam (interjection). But most of the words in Malayalam are compound words and such a word may consist of an arbitrary number of prefixes, stems (nouns, verbs, pronouns etc.) and an arbitrary number of suffixes [67]. Unlike English, Malayalam does not contain spaces or other word boundaries between the constituents of the compound word.

Example of a compound word

പൊളിച്ചെഴുതണമെന്നാണ് (*poLicchezhuthaNamennaN~)* (It must be revised)

This word is a combination of five atoms as shown below.

പൊളി + എഴുത് +അണം +എന്ന് +ആണ് (verb +verb +suffix + dyOthakam +Aux-Verb)

Basic POS shown in Fig 3.1 are explained in the sub-sections below.

3.1.1 Nouns

Noun is classified into concrete noun and abstract noun. The subclasses of concrete nouns are proper nouns, common nouns, material nouns, and collective nouns. Abstract noun is further classified into quality nouns and verbal nouns.

Examples

1. Concrete Noun

കൊച്ചി	(*kochchi*)	(Cochin)	–	Proper Noun
പട്ടണം	(*paTTaNam*)	(city)	–	Common Noun
മണ്ണ്	(*maNN~*)	(sand)	–	Material Noun
കൂട്ടം	*(kooTTam)*	(group)	–	Collective Noun

2. Abstract Noun

ചിരി	(*chiri*)	(laugh)	–	Quality Noun
നടത്തം	(*naTaththam*)	(walk)	–	Verbal Noun

3.1.2 Pronouns

A pronoun is a word used instead of a Noun. This POS is mainly of three types; First Person, Second Person, and Third Person.

ഞാൻ	(*njaan)*	(I)	–	First Person
നിങ്ങൾ	*(ningngaL*)	(you)	–	Second Person
അവൻ	(*avan*)	(he)	–	Third Person

Third Person is again classified into ten different forms [66].

Definite Pronoun can be of four types.

ഏത്	*(Eth~)*	(which)	–	Interrogative Pronoun
അ	(*a*)	(that)	–	Demonstrative Pronoun
ഏ	(*ae*)	(who)	–	Relative Pronoun
തന്റെ	(*thante*)	(your)	–	Reflexive Pronoun

Indefinite Pronoun has six forms in Malayalam language.

ചില	*(chila)*	(some)	– നാനാ സർവനാമം (naanaa sarvanaamam)
ഇന്ന	(*inna)*	(what)	– നിർദ്ദിഷ്ടവാചി (nirddishTavaachi)
എല്ലാ	*(ellaa*)	(all)	– സർവ്വവാചി (sarvvavaachi)
മിക്ക	(*mikka)*	(most)	– അംശവാചി (amSavaachi)
മറ്റ്	(*mat~*)	(another)	– അന്യാർത്ഥകം (anyaarththhakam)
വല്ല	(*valla)*	(any)	– അനാസ്ഥവാചി (anaasthhavaachi)

3.1.3 Verbs

Verbs are divided into four categories based on their meaning, behaviour, feature, and importance. The first classification is transitive verbs and intransitive verbs. Another classification based on the behaviour is simple verbs and causatives. Third type of classification is strong and weak verbs. Last division is according to its importance and is named as finite and infinite verbs.

കണ്ടു	(*kaNTu*)	(saw)	–	Transitive Verb
കുരക്കുന്നു	(*kurakkunnu*)	(barking)	–	Intransitive Verb
പാടുന്നു	(*paaTunnu*)	(singing)	–	Simple Verb
പാടിക്കുന്നു	(*paaTikkunnu*)	(make one sung)	–	Causatives
വായിക്കുന്നു	*(vaayikkunnu*)	(reading)	–	Strong Verb
തുന്നുന്നു	(*thunnunnu)*	(stitching)	–	Weak Verb
പറഞ്ഞു	(*paRanjnju*)	(told)	–	Finite Verb
ഓടുന്ന	(*OTunna)*	(running)	–	Infinite Verb

Infinite verb or participle is divided into പേരെച്ചം (*pErechcham*) (Adjectival Participle), and വിനയെച്ചം (*vinayechcham*) (Adverbial Participle).

3.1.4 Qualifiers

Three types of qualifiers are there in Malayalam– qualifiers of nouns (adjective), qualifiers of verbs (adverb), and qualifiers of qualifiers.

മിന്നുന്ന	(*minnunna*)	(glittering)	–	Adjective
ഉറക്കെ	(*uRakke)*	(loudly)	–	Adverb
വളരെ	(*vaLare*)	(too)	–	Qualifier of Qualifier

3.1.5 Dhyodhakam

'dyOthakam' is classified into prepositions, conjunctions and interjections [66]. In this work they are commonly referred as 'dhyodhakams'.

മുതൽ	(*muthal*)	(from)	– gathi (preposition)
ഉം	*(um)*	**(**and)	– ghaTakam (conjunction)
ആഹാ	*(aahaa*)	(sound showing wonder**)**	– vyaakshEpakam (interjection)

3.1.6 Affixes

Malayalam words are combinations of the above mentioned basic word types and affixes. Affixes are of three types- Prefix, Postfix, and Suffix [68]**.**

Prefixes are used to obtain a subdam (sound) from a root word with same or different meanings. Sometimes a new subdam might have an entirely different or opposite meaning. There are three types of prefixes.

First type	-	with opposite meaning.
Example	-	പ്രതിപക്ഷം *(prathipaksham*) (opposite party) where 'പ്രതി' (*prathi)* is the prefix.
Second type	-	same meaning but with emphasis
Example	-	സുസാധ്യം (*susaadhyam*) (that which is certainly possible) 'സു' (*su)* is the prefix
Third type	-	same meaning.
Example	-	പ്രഭാഷണം (*prabhaashaNam*) (speech) 'പ്ര' (*pra*) is the prefix

Postfix is mainly used for completing or changing the meaning of verbs. They are of four types. In the following examples underlined portion of the words are the postfixes.

1. കടന്നുകളഞ്ഞു (*kaTannukaLanju*) (escaped) –

   ഭേദകാനു പ്രയോഗം (*bhEdakaanuprayOgam*)

2. വരുന്നുണ്ട് (*varunnuNT~*) (coming) –

   കാലാനു പ്രയോഗം *(kaalaanuprayOgam*)

3. ഇല്ലായിരുന്നു *(illayirunnu*) (was not available) –

   പൂരണാനു പ്രയോഗം *(pooraNaanuprayOgam)*

4. വരരുത് *(vararuth)* (should not come) –

   നിഷേധാനുപ്രയോഗം (*nishEdhaanuprayOgam)*

Suffix-Words in Malayalam have a strong inflectional component. For verbs these inflections are based on tense, mood, aspect etc. For nouns and pronouns inflections distinguish the categories of gender, number, and case. These inflections called **Suffixes** are briefly described below**.**

A. NOUN- Suffixes

Table3.1 Gender Suffixes of Nouns and Pronouns

Gender / Word type	Masculine	Feminine	Neuter
Noun	അൻ (an)	ഇ (i)	അം (am)
Pronoun	അൻ (an)	അൾ (aL)	തു (thu)

1. **Nouns- Gender**

In Malayalam language the gender of nouns can be masculine, feminine, common, or neuter. Common gender suffixes are listed in Table 3.1.

അച്ഛൻ	*(achchhan)*	(father)	–	masculine
അമ്മ	(*amma*)	(mother)	–	feminine
വെള്ളം	(*veLLam*)	(water)	–	neuter
പക്ഷി	*(pakshi*)	(bird)	–	common

2. Nouns- Number

A noun can be either singular or plural. No number suffixes are required in singular form. അ (*a*), അർ *(aR*), മാർ (*maaR*), and കൾ (*kaL*) are the suffixes used to obtain plural forms of the nouns.

Table 3.2 Case-forms in Malayalam

Case	suffix	Example
നിർദ്ദേശിക (*nirddESika*) (Nominative)	No suffix	മരം (*maram*)
പ്രതിഗ്രാഹിക *(prathigraahika*) (Accusative)	എ	മരത്തെ *(maraththe)*
സംയോജിക (*samyOjika)* (Sociative)	ഓട്	മരത്തോട് *(maraththOT*)
ഉദ്ദേശിക(*uddESika*) (Dative)	ക്ക് ന്	മരത്തിന് (*maraththin~*)
പ്രയോജിക (*prayOjika*) (Instrumental)	ആ ൽ	മരത്താൽ (*maraththaal*)
സംബന്ധിക(*sambandhika)* (Possessive)	ഉടെ ന്റെ	മരത്തിന്റെ (*maraththinte)*
ആധാരിക *(aadhaarika*) (Locative)	ഇൽ കൽ	മരത്തിൽ *(maraththil*)

3. Nouns- Case

Suffixes used, to show the relationships of a noun to other words in the sentence are called case suffixes. There are seven case-forms possible for a noun. Table 3.2 shows the Case–forms and case suffixes available in Malayalam.

B. VERB- Suffixes

1. Verb- Tense

There are mainly three tenses- Present Tense, Past Tense, and Future Tense.

Table 3.3 gives a few examples of root verbs, their various tenses, and suffixes used.

Table 3.3 Verbs and Tenses

Root Verb	**ഭൂതം (*bhootham*) (Past)**	**വർത്തമാനം (*varththamaanam*) (Present)**	**ഭാവി (*bhaavi*) (Future)**
1) കൊട് (*koT*)	കൊടുത്തു *(koTuththu*)	കൊടുക്കുന്നു(*koTukkunnu)*	കൊടുക്കും (*koTukkum*)
(give)	(gave)	(gives)	(will give)
2) ഉറങ്ങ് (*uRangng*)	ഉറങ്ങി (*uRangngi)*	ഉറങ്ങുന്നു (*uRangngunnu*)	ഉറങ്ങും(*uRangngum*)
(sleep)	(slept)	(sleeps)	(will sleep)

Suffixes used in the first example are തു *(thu*), ഉന്നു (*unnu)*, and ഉം (*um*) for Past, Present, and Future tenses. In the second example suffixes are ഇ *(i*), ഉന്നു *(unnu*), and ഉം (*um)* respectively.

2. Verb- Mood

The different modes or manners in which a Verb may be used to express an action are called Moods. There are four Moods in Malayalam–Indicative, Imperative, Potential, and Permissive.

The Indicative Mood is used to make a statement of fact and to ask a question. No suffix is required for this purpose. Imperative Mood is used to express a command, order, advice, request or prayer. To make such a statement suffixes such as ആട്ടെ (*aaTTe*), ആലും (*aalum*), ഇൻ (*in)*, ഉക (*uka*) etc. are used. Potential Mood is to create a feeling of duty, fate or habbit by way of adding suffix `അണം' *(aNam*) to the root verb. Mood which shows permission is 'Permissive' and the suffix required is 'ആം' (*aam*).

എഴുതി	(*ezhuthi*)	(wrote)	–	Indicative
എഴുതട്ടെ	(*ezhuthaTTe)*	(shall I write)	–	Imperative
എഴുതണം	*(ezhuthaNam*)	(I have to write)	–	Potential
എഴുതാം	(*ezhuthaam*)	(I can write)	–	Permissive

3.Verb- Voice

Two voices are there for a verb – active and passive.

eg. രാമൻ രാവണനെ കൊന്നു (*raaman raavaNane konnu)*

(Rama killed Ravana) –Active Voice

eg. രാവണൻ രാമനാൽ കൊല്ലപ്പെട്ടു (*raavaNan raamanaal kollappeTTu*)

(Ravana was killed by Raman) –Passive Voice

A sentence in Active Voice is converted to Passive Voice by adding 'ആൽ' (*aal)* suffix to the Subject and 'പെട്ടു' (*ppettu)* suffix to the verb.

3.2 Phrase Types

Words are grouped together into phrases which are then joined to form sentences. A sentence consists of different types of phrases.

3.2.1 Noun Phrase

A noun phrase is a group of words that does the work of a noun. It contains information about the noun. Normally this noun is a participant of the action described by the verb. In the sentence ഇലയുടെ പച്ചനിറം ചെടിക്കു ഭക്ഷണംതയ്യാറാക്കുന്നു (*ilayuTe pachchaniRam cheTikku bhakshaNam thayyaARaakkunnu*)(Green color of the leaves prepare food for the plant), the underlined phrase is an example of noun phrase.

3.2.2 Verb Phrase

Verb phrase is a constituent of a sentence that contains at least one verb and its complements, objects or other modifiers that function syntactically as a verb. In the previous example ഭക്ഷണം തയ്യാറാക്കുന്നു *(bhakshaNam thayyaARaakkunnu*) (prepare food) is the verb phrase.

3.2.3 Adverbial Phrase

An adverbial phrase is a group of related words which plays the role of an adverb. In the example കുളത്തിലെ വെള്ളം വറ്റിയപ്പോൾ മൽസ്യങ്ങൾ കിടന്നു പിടച്ചു (*kuLaththile veLllLLam vatiyappOL malsyangaL kiTannu piTachchu)* (When the water in the pond dried up the fishes became restless), കുളത്തിലെ വെള്ളം വറ്റിയപ്പോൾ (*kuLaththile veLllLLam vatiyappOL*) is an example of adverbial clause of time.

3.2.4 Adjectival Phrase

An adjective phrase is a group of words in a sentence that can function in the same way as an adjective. Adjectives are used to modify nouns or pronouns.

They give additional details about the meaning of a noun. In the sentence below, underlined phrase is an example of an adjectival phrase.

വിവാഹം കഴിഞ്ഞ് നഗരത്തിലേക്ക് മടങ്ങിവന്ന രാമൻ അച്ഛന്റെ നിയോഗത്താൽ വനവാസത്തിനു പോയി.

(*vivaaham kazhinj nagaththilEkk maTangivanna raaman achchhante niyOgaththaal vanavaasaththinu pOyi*)

(Raman who returned to the city after marriage, went to the forest according to the decision of his father)

3.2.5 Postpositional Phrase

In Malayalam language, prepositions come after the dependant noun phrase and are called as postpositions. The postpositions indicate semantic relationship of the dependant noun phrase to the context.

In the sentence അച്ഛനും അമ്മയും കൂടി വന്നു (*achchhanum ammayum kooTi vannu*) (father and mother came together), ഉം (*um*) and കൂടി (*kooTi*) are the postpositions. അച്ഛനും അമ്മയും കൂടി (*achchhanum ammayum kooTi*) is the postpositional phrase.

3.3 Malayalam Sentences

A sentence is a group of words that makes complete sense. They can be classified according to their behaviour or construction.

3.3.1 Sentence Classification Based on Behaviour

Based on the behaviour, sentences can be classified into four types.

Assertive sentence: This is a sentence that makes a statement.

e.g. കളിസ്ഥലത്തിന് വേണ്ടത്ര നീളവും വീതിയും ഉണ്ട്

(kaListhhalaththinu~ vEnTathra nILavum vIthIyum uNt)

(Play ground has sufficient length and width.)

Interrogative sentence: This asks a question

e.g. ഹരി നിന്റെ അയൽക്കാരനണോ? *(Hari ninte ayal_kkaaranaNO)* (Is Hari your neighbor?)

Imperative sentence: This expresses a command, a request or a prayer.

e.g.ഞങ്ങൾക്കാവശ്യമുള്ള ആഹാരം ഞങ്ങൾക്കിന്ന് തരേണമേ

(njaNGaLkkaavaSyamuLLa Ahaaram njaNGaLKinn tharENamE)

(Give us this day our daily bread.)

Exclamatory sentence: They are used to express strong feeling, extreme happiness, sorrow, and wonder.

e.g. അല്ലാ! ആരാണ് ഈ വഴി വരുന്നത് *(allaa! aaraaN~ ee vazhi varunnath)*

(Oh! Who is coming this way?)

3.3.2 Sentence Classification Based on Construction

Sentences in Malayalam can be of three types based on the construction – simple, complex and compound.

Simple sentences: This type contains only one main clause**.**

e.g. അക്ബർ മഹാനായ ചക്രവർത്തി ആയിരുന്നു *(Akbar mahaanaaya chakravarththi aayirunnu)* (Akbar was a great emperor.)

Complex sentences: These sentences contain one principal clause and any number of subordinate clauses.

കുളത്തിലെ വെള്ളം വറ്റിയപ്പോൾ മൽസ്യങ്ങൾ കിടന്നു പിടച്ചു *(kuLaththile veLllLLam vatiyappOL malsyangaL kiTannu piTachchu)* (When the water in the pond dried up the fishes became restless)

Compound sentences: Compound sentences can have any number of principal clauses.

e.g. അയാൾ വീട്ടിലെത്തി ചായ തയ്യാറായിട്ടുണ്ടോ എന്ന് ഭാര്യയോട് ചോദിച്ചു

(ayaAL vITTileththi chaaya thayyaaRAyiTTuNTO enn bhaaryayOT chOdichchu)

(He reached home and asked his wife whether tea was ready)

3.4 Malayalam Sandhi

'Sandhi' means 'harmony'. When two words are united, certain changes occur to the last letter of the first word and the first letter of the second word. These changes are usually of four types.

1. ദ്വിത്വം *(dvithvam)* (Gemination*)*

ദ്വിത്വം (dvithvam) means two-fold; i.e. while joining two words, first letter of the second word doubles.

മരം *(maram)*+ കൊമ്പ് (*komp*) = മരക്കൊമ്പ് (*marakkomp*)

2. ലോപം *(lOpam)(*Elision)

If an elision of vowel or consonant occurs during the union of two words such a union is called 'lOpa sandhi'.

പോയി (*pOyi)* + ഇല്ല (*illa*) = പോയില്ല (*pOyilla)*

3. ആഗമം *(aagamam) (*Augmentation)

A new consonant is coming between the words and this consonant is either ‘യ’ *(ya*), ‘മ’ *(ma*), or ന (*na*).

അല (*ala*) + ആഴി (*aazhi*)	=	അലയാഴി (*alayaazhi)*
തിരു *(thiru*) + ആതിര (*aathira) =*		തിരുവാതിര (*thiruvaathira*)
കരി *(kari*)+ ചന്ത (*chantha*)	=	കരിഞ്ചന്ത (*karinchantha*)

4. ആദേശം *(aadESam) (*Substitution)

Substitution of a vowel or consonant occurs between the two words when they are combined and that is called ആദേശം (*aadESam).*

വിട് (*viT~*) + തു (*thu*)	=	വിട്ടു (*viTTu*)
കുളം *(kuLam*) + ഇൽ (*il*)	=	കുളത്തിൽ (*kuLaththil)*

3.5 Chapter Summary

Malayalam is a morphologically rich language. This chapter briefly describes the Parts-of-Speech, phrases and different kinds of sentences available in Malayalam language.

Chapter 4

MaQAS: A MALAYALAM QUESTION ANSWERING SYSTEM

Contents

Modern QA Systems are mostly document or passage retrieval systems. But users of QA Systems expect exact answers rather than long list of documents or snippets of documents. In this chapter, a QAS for Malayalam language which retrieves exact answers is described.

A monolingual QAS named MaQAS is developed and this work is capable of providing answers in a word or a sentence for factoid and a few non-factoid type questions by the deep linguistic analysis of Malayalam documents in the medical domain.

This work imparts knowledge to naive users regarding the causes, reasons, remedies, etc. of lifestyle and infectious diseases. MaQAS analyses the question given by the user and returns most relevant answer in a word or a phrase or a sentence. Since named entities help us to extract the essence of the text, MaQAS is developed as a Named Entity-based Question Answering System. The Architecture of MaQAS is explained in the following sections.

4.1 System Architecture

Overall Architecture of MaQAS is shown in Fig 4.1. It consists of three main modules– Indexing Module, Question Analysis Module, and Answer Extraction Module.

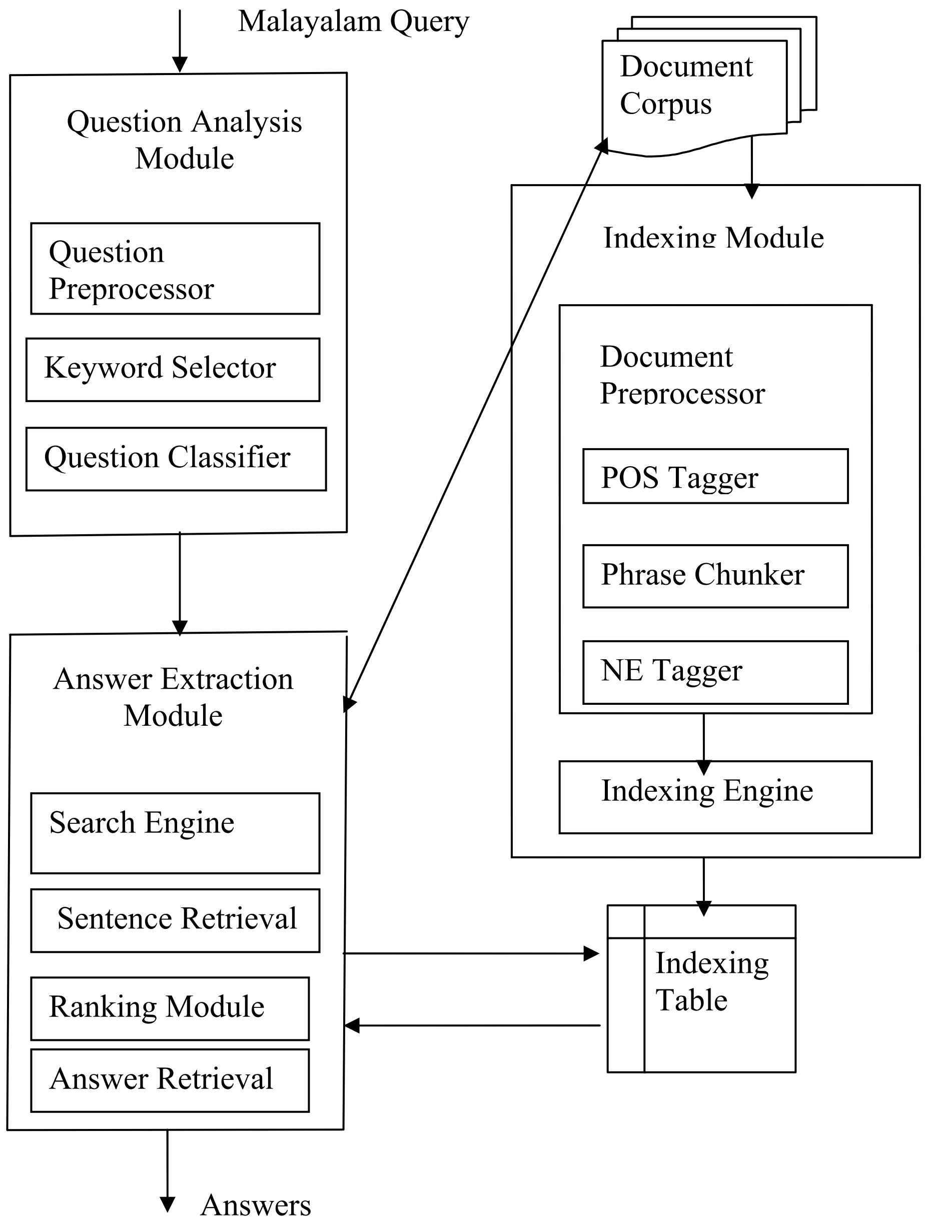


Fig 4.1 System Architecture of MaQAS

Indexing Module processes all the documents in the corpus and prepares a NE-based index. When a query is entered by the user, the Question Analysis Module analyses the query sentence and determines its type. Also it identifies keywords and other significant words. Keywords are useful in finding the entity type of the expected answer and significant words help in determining the final answer. These details are passed to the Answer Extraction Module. Answer Extraction Module uses this information to search through the index and extracts the most relevant answer from the corpus. Each module of MaQAS is described below.

4.1.1 Indexing Module

This module processes all the documents entered into the system and prepares an indexing scheme based on named entities.

There are two stages in this indexing module.

- Document Preprocessing
- Indexing Engine

Document Preprocessing

A general Question Answering System normally extracts keywords from the user query and retrieves documents in which the keywords are present. These retrieved documents are the candidate documents from which answer sentences are selected and returned to the user. But sometimes, keywords may not be directly present in the documents. At certain times, meanings of the keywords and that of the words present in the documents may not be the same. In such situations the results obtained are not accurate.

Keyword-matching technique is not used in MaQAS as Malayalam is an agglutinative language. Instead of using keyword matching technique, keywords meanings are used for answer retrieval. Such a system is implemented with the help of Named Entity principles. Named Entities are the important meaning bearing words in a sentence or a document. They aid in extracting word meanings in the user query as well as in the documents.

Example A രാമു പാമ്പിനെ വടികൊണ്ടു കൊന്നു

(raamu paampine vaTikoNTu konnu) (Ramu killed the snake with a stick)

The questions that can be asked, the expected answer type and respective answers with respect to Example A are listed in Table 4.1

Table 4.1 List of Possible Questions and Answers w.r.t Example A

Questions	Expected Answer Type	Expected Answer
1. ആരാണ് പാമ്പിനെ കൊന്നത് ? (aaraaN~ paampine konnath~?) (Who did kill the snake?)	AGENT	Ramu
2. രാമു പാമ്പിനെ എന്തുകൊണ്ടാണ് കൊന്നത് (raamu paampine enthukoNTaaN~ konnath~ (How did Ramu kill the snake?)	INSTRUMENT	Stick
3. രാമു എന്തിനെയാണ് കൊന്നത് (raamu enthineyaaN~ konnath~) (What did Ramu kill?)	OBJECT	Snake

For all the questions given in Table 4.1, answer words are nouns but their roles in the sentences are different. These words are Named Entities and their roles are AGENT, INSTRUMENT, and OBJECT respectively.

Mostly role-carrying agents are Nouns (Proper Noun, Common Noun, Pronoun), Adjectives and Adverbs. There are two stages in Named Entity-based systems – NE Recognition (NER) and NE Classification (NEC). NER stage identifies the Named Entity words and NEC assigns their roles. With respect to the example A, function of NER is to identify the Named Entities 'Ramu', 'Stick', and 'Snake'. NEC finds out their roles as 'AGENT', 'INSTRUMENT', and 'OBJECT'.

Example B

ആഹാരം കഴിച്ചശേഷം രാമു ഉറങ്ങി

(aahaaram kazhichchaSEsham raamu uRangngi) (Ramu slept after food)

Suppose the question asked with respect to **Example B** is 'എപ്പോഴാണ് രാമു ഉറങ്ങിയത്?' (eppOzhaaN~ raamu uRangngiyath~?) (When did Ramu sleep?), the expected answer is 'ആഹാരം കഴിച്ചശേഷം' (aahaaram kazhichchaSEsham) (After food) which is a phrase indicating 'TIME'. This example shows that Named Entity can also be a phrase while in **Example A** it was a word.

From the above examples it is clear that, NER require tools to identify nouns, adjectives, adverbs and phrases. Tools to identify these POS and phrase chunks are developed for Malayalam language. Word-level and context-level features are essential for POS Tagging and Phrase Chunking. Owing to the compounding nature, this feature extraction requires a Compound Word Splitter which is also developed. Implementation details of Compound Word Splitter are given in chapter 4. POS Tagger, Phrase Chunker, and NE Tagger are described in the chapters 5, 6, and 7 respectively.

Document preprocessing stage preprocesses the documents with the help of these tools and prepares NE tagged documents. Then the Indexing Engine scans these documents to prepare the NE-based index.

Indexing Engine

To facilitate fast and accurate Information Retrieval, document indexing techniques are used. Inverted Index is the most popular data structure used in document retrieval systems. Since MaQAS is a NE-based QAS, Named Entity-based indexing scheme is used.

Procedure for the preparation of NE-based index is given below. For every NE tagged document, named entity list (N), and sentence occurrence lists (O) are prepared and as shown in Fig 4.2.

Named Entity List (N1)	Occurrences (O1)
PERSON	1, 10, 15, 65......
ORGANISATION	14, 32...
DISEASE	0, 4, 17, 23...
VIRUS	11, 16...
DATE	6, 21...

Fig 4.2 Named Entities and their Occurrences in a Document D1

N1 and O1 are the lists corresponding to document D1. N1 shows the Named Entities that are identified in the document D1. Their point of occurrences (the sentences in which these NEs are present) in the document D1 are given in O1. For example, PERSON entity occurs in the sentences 1, 10, 15, 65 etc. DISEASE entity occurs in sentences 0, 4, 17, 23 etc. In the same way, NE lists and occurrence lists are prepared for the all 120 documents in the corpus. They are N1 to N120 and O1 to O120. By analysing the Named Entity lists N1 to N120 a common NE list E is prepared. For every entry in the list E, a list of documents in which that entity appears is maintained in the document list D. Fig 4.3 explains the preparation of NE-based index.

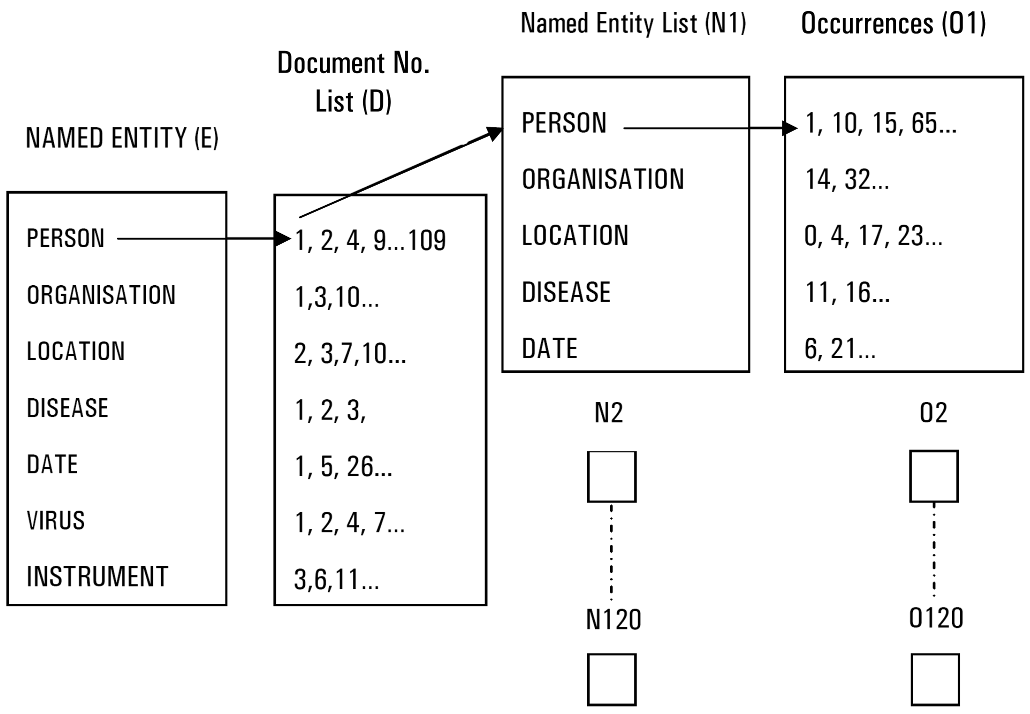


Fig 4.3 Index Preparation

List E in Fig 4.3 contains 6 NEs. For each NE in E, a document list is given in D. For example corresponding to PERSON entity in E, the list in D is 1, 2, 4, 9, and 109, i.e., PERSON entity exists in the documents D1, D2, D4, D9, and D109. Then corresponding to D1, N1 contains the PERSON entity and O1 contains the occurrence list. Thus the lists E and D form the first level index and the lists N and O form the second level index.

In short, document pre-processing stage, processes all the documents in the corpus and detects the named entities that occur in each one of them. Indexing stage comprehends in which documents and in which sentences these entities reside. Thus the document numbers and sentence numbers together create a double-level index.

4.1.2 Question Analysis Module

Question processing is one of the main tasks of Question Answering System and this is an initial step in the retrieval of relevant information. In order to answer a question correctly one needs to understand what type of information the question asks for. To identify the purpose of the question, it is analysed in a number of ways and all those features which are helpful to determine the expected answer type are extracted.

Main function of Question Analysis Module given in Fig 4.4 is to take the input set of questions and convert them into a form that can be processed by the Answer Extraction Module. This module extracts main keywords, expands keyword terms, determines question type, and builds the semantic context representation of the expected answer. This module consists of stages for Question Preprocessing, Keyword Selection, and Question Classification.

Question Preprocessor removes unimportant words, from the question that is to be submitted to the Keyword Selector stage. Function of Keyword Selector is to identify primary and secondary keywords. These keywords help the question classifier stage in determining the type of the question and the expected answer type. Each of these stages are described below.

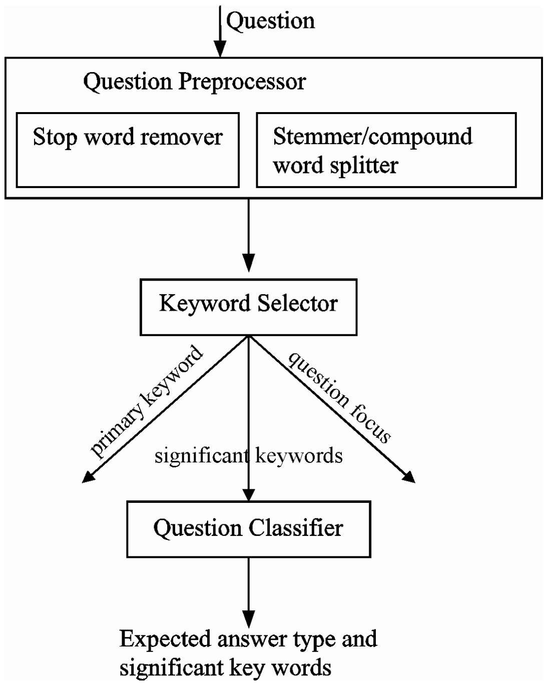


Fig 4.4 Question Analysis Module

Question Preprocessing

The Query/Question preprocessing includes stop word removal and stemming. Stop words [69] are words which are filtered out prior to or after processing of NL data or text. Any group of words can be chosen as the stop words. Normally less important or short function words are removed as stop words. In IR terms, common words which would appear to be of little value in helping select documents matching a user need are called stop words. Elimination of such words reduces the size of the index. Process of removing such words is called stop word removal. There are many such words in Malayalam language and a few are listed in Appendix A.

A stem is the portion of the word which is left after the removal of its affixes. Frequently the user specifies a word in a query but only a variant of this word is present in a relevant document. This reduces the chances of answer detection and retrieval. Stemming [69] is useful for improving retrieval performance because they reduce variants of the same root word to a common concept. Also compound words in the question are decomposed to obtain their root forms.

Keyword Selector

This module analyses each and every word of the question and identifies the keywords. Keywords are identified by the Pattern Matching technique. Table 4.2 shows various first level keywords and patterns used for their detection. These keywords are used by the Answer Extraction Module to find out the answers relevant to the question. Keywords are mainly used for detecting the question type.

Table 4.2 Patterns for Keyword Identification

Question Type	Pattern
ആര് (aar~) (who)	"\u201a\u00a5[\u00af \u00b8].*"
ഏത് (aeth~) (which)	"\u2021 [\u00b5]? \u02dc.*"
ആര്‍ക്ക് (aaRkk) (whom)	"\u201a\u00dc.*"
എത്ര (ethRa) (how much)	"\u2020\u00bd\u02dc.*"
എന്തുകൊണ്ട് (enthukonT~) (why)	"\u2020 [\u00b5]? \u00d0.*"
എപ്പോള്‍ (eppOL) (when)	"\u2020\u00b6\u00d4\u00af.*"
എങ്ങിനെ (eNGine)(how)	"\u2020\u00c4 [\u00b0]? \u00b5\u0153.*"
എവിടെ (eviTe) (where)	"\u2020\u00aa\u00b0\u00b5\u201c.*"
എന്ത് (enth~)(what)	"\u2020 [\u00b5]? \u00d0.*"
എന്ന് (enn~)(when)	"\u2020[\u00b5]?\u00d2\u00af.?[^\u00de].*"

Malayalam language keyboard and character encoding has been standardized as Indian Script Code for Information Interchange (ISCII) which is an 8-bit code. ISCII is defined in such a way that all Indian languages can use a single character encoding scheme. User interface of MaQAS is developed using Java that uses two byte Unicode characters. The Unicode standard is the universal character-encoding standard used for representation of text for computer processing. To render Malayalam text that uses MLB-TTIndulekha font developed by C-DAC to Java Graphical User Interface (GUI), a mapping from ISCII to Unicode is essential. This mapping table is given in Appendix B. Patterns for keyword identification are prepared by referring ISCII to Unicode mapping table.

Keyword Selector scans the input query and identifies primary keywords such as ആര് (aar~) (who), ഏത് (Eth~) (which), എപ്പോൾ (eppOL) (when), etc. These primary keywords give clear indication of expected answer type. But words like എത്ര (ethRa) (how much), ഏത് (aeth) (which), etc. are less clear about expected answer type. To solve such situation and to obtain most relevant answers for the question, additional keywords are required and they are called secondary or second level keywords. Second level keywords are also called Question Focus.

Question Focus [70] is a word or a phrase in the question that solves the question type ambiguity. This is a word close to the interrogative term that supplies additional information about the type of the expected answer. Detection of focus is important for extracting the answer from candidate paragraphs. In the absence of expected answer type, correct answer is retrieved using the noun phrase attached to the question phrase.

Working of Keyword Selector is explained with the following examples.

Example 1

ആരാണ് മലേറിയയുടെ വാക്സിൻ കണ്ടുപിടിച്ചത്?

(*aaraaN~ malERiyayuTe vaaksin kaNTupiTichchath~?)*

(Who invented the Vaccine for Malaria?)

Keyword Selector scans this question against the key patterns and identifies ആരാണ് (*aaraaN~*) **(who) as** the primary keyword. This primary keyword helps us to find out the question type as ‘who’ and expected answer type as ‘PERSON’.

Example 2

ഏത് രാജ്യക്കാരാണ് കോളറയുടെ പ്രതിവിധി കണ്ടുപിടിച്ചത്

(*Eth~ raajyakkaaraaN~ kOLaRayuTe prathividhi kaNTupiTichchath~)*
(Which country-men discovered the preventive for Cholera?)

Keyword selector identifies the primary keyword ഏത് (Eth~) (which). But this keyword does not give any clue regarding the question or answer type. Hence it is necessary to obtain secondary keywords. In the **Example 2** ‘രാജ്യക്കാരാണ്’ (*raajyakkaaraaN~)* (country-men) is the secondary keyword. Primary and secondary keyword together solves the problem of question type identification. Here, the words ഏത് രാജ്യക്കാരാണ് (*Eth~ raajyakkaaraaN~)* (which country-men) determines the question type as ‘which’ and question focus as ‘COUNTRY’.

Primary and secondary keywords determine the type of the question and the expected answer. But to obtain the most appropriate answer, another group of keywords are required. These key words are given more weightage than the normal keywords which are called significant keywords [71]. Normally words referring to the question object or noun phrases in the absence of question object are considered as significant. Words that refers to the object of the question is termed as Question object. In certain queries, Question focus and question objects are the same.

Example 3

ഏത് വൈറസാണ് ഇൻഫ്ളുവെൻസ ഉണ്ടാക്കുന്നത്?

(*Eth~ vaiRasaaN~ inphLuvensa uNTaakkuth~?)*

(Which virus causes Influenza?)

The keyword selector separates the primary keyword '**ഏത്**' (*Eth~)(*Which) and the secondary keyword '**വൈറസാണ്**' (*vaiRasaaN~*) (virus) from the question given in **Example 3**. For the same question, the word ഇൻഫ്ളുവെൻസ *(inphLuvensa)* (Influenza) is the question object. The keywords **ഏത് വൈറസാണ്** (*Eth~ vaiRasaaN~)* (which virus) gives a clear indication of both the question type and the expected answer type. But these keywords are not indicating anything about "which virus it is asking for". To answer such a question correctly, information regarding the question object is essential.

Example 4

ഏത് തരത്തിലുള്ള ഭക്ഷണമാണ് മഞ്ഞപ്പിത്ത രോഗി കഴിക്കേണ്ടത്

(*Eth~ tharaththiluLLa bhakshaNamaaN~ manjnjappiththa rOgi kazhikkENTath~)*

(What kind of food can a Jauntice patient eat?)

Primary keyword	– ഏത് (*Eth~*) (what)
Secondary keyword	– ഭക്ഷണമാണ് (*bhakshaNamaaN~*) (food)
Question object	– മഞ്ഞപ്പിത്ത രോഗി കഴിക്കേണ്ടത് (*manjnjappiththa rOgi kazhikkENTath~)* (Jauntice patient can take)

Again in Example 4, primary and secondary keywords determine the question and answer types but they do not give any indication of exact answer.

Question Classification

Keyword Selector obtains the primary, secondary, and the significant keywords from the user query and sends them to the 'Question Classification' stage.

Table 4.3 Classification of Question

Question Type	Question Focus	Category of Question
ആര് (aar~) (who)	**Not required**	PERSON
ഏത് (aeth~) (which)	രോഗം (rOgam)(disease)	DISEASE
	വൈറസ് (vaiRas)(virus)	VIRUS
	കാരണം (kaaraNam) (reason)	REASON
	ആഹാരം (aahaaram) (food)	FOOD
എത്ര (ethRa) (how much)	ദൂരം (duuram) (distance)	DISTANCE
	ദിവസം (divasam) (days)	DAYS
എന്തുകൊണ്ട് (enthukoNt~)(why)	Not required	REASON
എന്താണ് (enthaN~) (what)	Not required	DEFINITION/REASON
ആരാണ് (aaraaN~) (who is)	Not required	DEFINITION/PERSON
ആർക്ക് (aaRkk~) (whom)	Not required	PERSON
എപ്പോൾ (eppOL) (when)	Not required	TIME
എങ്ങിനെ (eNGine) (how)	Not required	DESCRIPTION
എവിടെ (eviTe) (where)	Not required	LOCATION
എന്ന് (enn~) (when)	Not required	DAY

Question Classification is the task that maps a question into one of the k predetermined classes. This stage is needed to understand the expected answer type before returning an answer. Assigning question classes can be accomplished in a variety of ways. In this work a rule-based classifier is used that classifies a question into fine grained categories and their corresponding coarse categories. Classification [72] is done for finding the type of questions and answers. The expected answer type is the semantic type of the entity expected as the answer to the question. The answer type is helpful for factual questions, but not for questions that require complex explanations.

Assigning a question type to the question is a crucial task as the entire answer extraction relies on finding the correct question type and hence the correct answer type. In some cases there are words that indicate the question type directly, i.e., "who", "where" etc. Some of these words can represent more than one type. In such situations, other words (the words that can indicate meaning) in the question need to be considered. Table 4.3 gives a list of question categories that is decided with the help of various keywords separated in the Keyword Selector stage.

Examples

A. 'എപ്പോൾ' (eppOL) (when) type Questions

These questions are of two types and always ask for a named entity DATE. There are few questions which require a day or days as the answer.

1. എന്നാണ് കുത്തിവെയ്പ് എടുക്കേണ്ടത്?

(*ennaaN~ kuththiveyp~ eTukkENTath~?)* (when to take the injection?)

2. എന്നൊക്കെയാണ് മരുന്ന് ലഭിക്കുന്നത്?

(*ennoakkeyaaN~ marunn~ labhikkuth~?*) (on what days medicine is available?)

3. എന്നു മുതൽ രോഗം പകരും?

(*ennu muthal rOgam pakarum?*) (from which day disease will spread?)

For these three questions given above, primary keywords are എന്നു (ennu) (when) and referring Table 4.3 the question category is determined as

'DAY'. In this type of questions secondary keywords are not required to find out the question and answer types.

B. 'ആര്'(aar~) (who) type Questions

There are three types of 'who' questions. Details are given below.

Who Definition

The question given below expects a 'DEFINITION' type answer.

4. ആരാണ് അശോകൻ? (*aaraaN~ aSOkan*) (who was Asoka?)

The primary keyword **'**ആര്' (aar~) (who) is used to identify the question type.

Who List

5. ആരൊക്കെയാണ് കോളറ പരീക്ഷണത്തിൽ പങ്കെടുത്തത്?

(*aarOkkeyaaN~ kOLaRa pareekshaNaththil pangkeTuththath~?)*

(Who all are participated in the cholera experiments?)

Answer type to 'question 5' is a 'PERSON' entity and it requires a list of persons. In this example the word 'ഒക്കെ' (okke) (all) is very important. The classifier uses the primary keyword **'**ആര്' (aar~) (who) and the secondary word **'**ഒക്കെ' (okke) (all) to identify the answer type. Then the significant keywords **കോളറ പരീക്ഷണത്തിൽ പങ്കെടുത്തത്** (kOLaRa pareekshaNaththil pangkeTuththath~) (participated in the cholera experiments) aids in answer extraction.

Who factoid

6. ആരാണ് കോളറ കണ്ടുപിടിച്ചത്?

(*aaraaN~ kOLaRa kaNTupiTichchath~?*)(Who discovered Cholera?)

Classification module assigns PERSON category to ‘question 6’. Even though the primary keyword is the same as in ‘question 4’, answer type identified is different since there is a significant keyword **കണ്ടുപിടിച്ചത്** (*kaNTupiTichchath~*) (discovered) after the secondary keyword **കോളറ** *(kOLaRa*) (Cholera).

C. ‘ആരെ/ ആരുടെ’ (aare /aaruTe) (Whom/Whose) type Questions

Answer type for these questions are either OBJ or PERSON**.**

7. ആർക്കാണ് കോളറ ആദ്യമായി വന്നത്

(*aarkkaaN~ kOLaRa aadyamaayi vannath~)* (Who was affected by the cholera the first time?)

8. ആരുടെ മേൽനോട്ടത്തിലാണ് പരീക്ഷണം തുടങ്ങിയത്

(aaruTe mElnO'ththilaaN~ pareekshaNam thuTangngiyath~)

(Under whose supervision was experiments started?)

Classifier assigns ‘OBJ’ category to question 7 and ‘PERSON’ category to question 8.

D. ‘എവിടെ’ (eviTe) (Where) type Questions

Answer type to this question is ‘LOCATION’.

9. എവിടെയാണ് കരൾ സ്ഥിതിചെയ്യുന്നത്

(eviTeyaaN~ karaL sthhithicheyyuth~) (Where in the human body liver lies?)

The primary keyword ‘എവിടെ’ (*eviTe*) (Where) is sufficient to identify the answer type. Then the remaining words in the question (significant words) help in answer extraction.

E. 'ഏത്' (aeth) (Which) type Questions

Answer type is decided by the question focus.

10. ഏത് വൈറസാണ് ഫ്ളു ഉണ്ടാക്കുന്നത്?

(Eth~ vaiRasaaN~ phLu uNTaakkuth~?) (Which virus causes flu?)

11. ഏത് സ്ഥലത്താണ് ചിക്കൻ ഗുനിയ ആദ്യമായി കണ്ടത്?

(*Eth~ sthhalaththaaN~ chikkan guniya aadyamaayi kaNTath~)*

(At which place Chikunguniya was first noticed?)

12. ഏത് സമയത്താണ് മഞ്ഞപ്പിത്തം പകരുന്നത്?

(*Eth~ samayaththaaN~ manjnjappiththam pakaruth~?)*

(In which season Jauntice spreads?)

In each of these interrogatives 10, 11 and 12, underlined words determines the expected answer type. They are VIRUS, LOCATION and TIME respectively.

F. 'എന്ത്' (enth~) (What) type Questions

13. എന്താണ് മഞ്ഞപ്പിത്തം? (*enthaaN~ manjnjappiththam*?) (what is Jauntice?)

14. എന്താണ് കോളറയുടെ കാരണം? (*enthaaN~ kOLaRayuTe kaaraNam*?)

(what is the cause of cholera?)

In the questions 13 and 14, answer types are decided by the key words and the question focus. 'Question 13' expects 'DEFINITION' type answer while 14 expects 'REASON' type.

G. 'എത്ര' (eethRa) (How Many) type Questions

15. എത്ര ദിവസം രോഗം നിലനില്ക്കും?

(*ethra divasam rOgam nilanilkkum*?) (How many days the disease last?)

16. എത്ര ദൂരം പോകണം?

(*ethra dooram pOkaNam*?) (How much distance to cover?)

17. എത്ര സമയം വിശ്രമിക്കണം?

(ethra samayam viSramikkaNam?) (How much time to take rest?)

In questions 15, 16, and 17 the underlined words is the question focus but, that itself is not sufficient to finalize the answer. Significant keywords in the question are essential to determine the exact result.

H. 'എന്തുകൊണ്ട്' (*enthukoNT~*) (Why) questions/ 'എങ്ങിനെ'(*engngine*) (How) questions

These are the easiest question words to process and answer type to these questions are REASON/DESCRIPTION.

18. എന്തുകൊണ്ട് എയ്ഡ്സ് ബാധിക്കുന്നു?

(*enthukoNT~ eyDs~ baadhikkunnu*)(Why AIDS affects?)

19. എങ്ങിനെയാണ് കോളറ പകരുന്നത് ?

(*engngineyaaN~ kOLaRa pakaruth~*) (How Cholera spreads?)

Primary keywords in the questions 18 and 19 are used to determine the question type and classifier assigns their answer types. This information along with the significant keywords is sent to the answer extractor.

As explained above, Question Analysis Module scrutinizes the input question and extracts all the information that can aid the answer retrieval process. This module also finds out the question and answer types. Knowledge that collected and derived from the input is then transferred to the Answer Extraction Module which is described in the next section.

4.1.3 Answer Extraction Module

In many QA Systems answer extraction identifies the documents or paragraphs in the document set that are likely to contain an answer [73]. Also in the absence of indexing scheme the search engine scans every document in the corpus which would take a considerable amount of time and computing power. Answer Extraction Module in Fig 4.1 is redrawn and shown below.

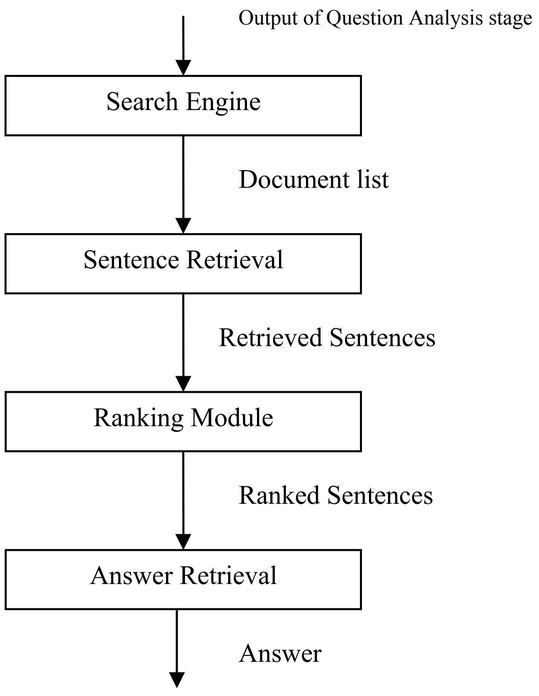


Fig 4.5 Answer Extraction Module

In this work, function of Answer Extraction Module is to produce exact responses rather than paragraphs. This aim is facilitated by the four stages of this module namely Search Engine, Sentence Retrieval, Ranking Module, and Answer Retrieval.

Search Engine

A search engine's job is to retrieve and present links to information as relevant to search query as possible.

Suppose the input query is ആരാണ് മഞ്ഞപ്പിത്തം കണ്ടുപിടിച്ചത്

(aaraaN~ manjnjappiththam kaNTupiTichchath~)(Who discovered Jaundice?)

Question Analysis stage analyses the above question and identifies the primary keyword ആരാണ് (*aaraaN~)* (who). This information is further used, to determine the question type ആര് (*aar~)* (who) and expected answer type 'PERSON'.

The search engine refers the index table given in Fig 4.3 and finds the named entity 'PERSON' in the entity list E. For every entry in the entity list E there is a corresponding entry in the D list. The D list entry shows that the 'PERSON' entity is present in the documents D1, D2, D4, D9 and D109. This list of document numbers is fed to the Sentence Retrieval stage.

Search Engine stage uses first level index. First level index contains Named Entities and pointer to documents where that entity exist.

Sentence Retrieval

The sentence retrieval stage retrieves the sentences using the second level index. Second level index contains Named Entity and pointer to sentences. The entries in the document list D are used for identifying entity lists and respective sentence lists. This sentence list is used for retrieving candidate sentences from the documents. For the example mentioned above, the document numbers 1, 2, 4, 9 and 109 are used for identifying entity lists N1, N2, N4, N9, and N109 and sentence lists O1, O2, O4, O9, and O109. By referring these lists all the sentences are retrieved in which the entity 'PERSON' is present. From these five documents, 16 sentences are retrieved which are ranked by the ranking module.

Ranking Module

Significant keywords are used for ranking the retrieved sentences. The sentences retrieved are weighted according to the presence of these significant words or terms. For Example, sentence in which 6 significant terms are present is assigned a weight 6; a sentence with 3 terms is assigned a weight 3, and so on. Then these sentences are assigned a rank based on the weight. The top ranked 5 sentences are selected and used for the final answer retrieval.

Answer Retrieval

This final component of Answer Extraction Module selects the answer from the top ranked sentence. The word or phrase who's named entity matches with the expected answer type is selected as the answer.

In short, Answer Retrieval Module identifies candidate sentences by searching through the prior prepared double level index using the facts obtained from the Question Analysis stage. These sentences are ranked based on various factors and top ranked sentence is selected as the final candidate. With the help of expected answer type answer is derived from the candidate sentence.

Consider the question

കരൾ സ്ഥിതിചെയ്യുന്നത് എവിടെ? (*karaL sthhithicheyyunnath~ eviTe*)

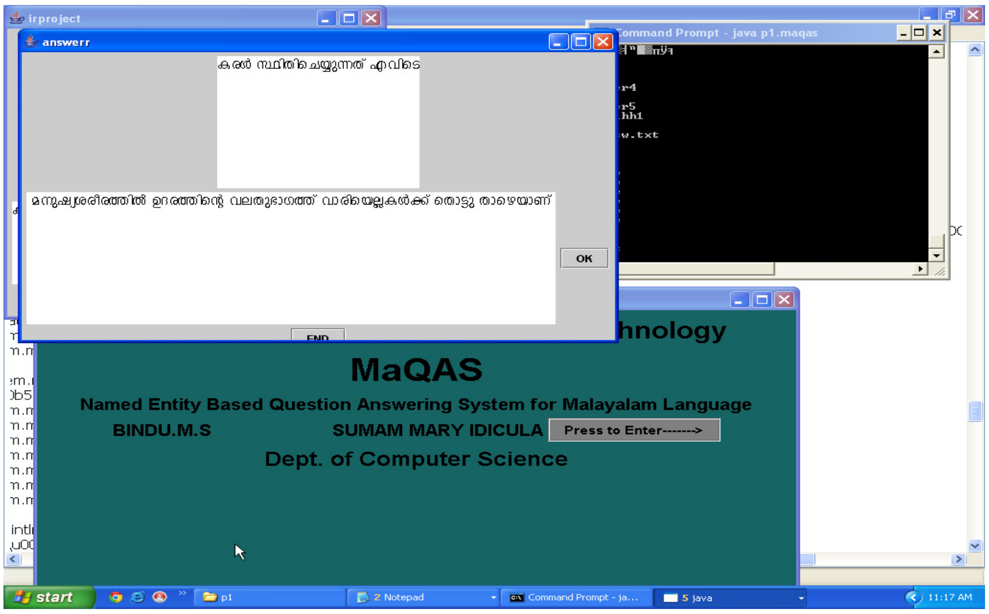
(where does Liver exist?).

For the above question, sentence retrieval stage retrieves the following five sentences.

Table 4.4 Outputs of Various Stages of Answer Extraction Module

Sentences Retrieved	Entity Identified	Weight	Rank
1. മനുഷ്യശരീരത്തിൽ ഉദരത്തിന്റെ വലതുഭാഗത്ത് വാരിയെല്ലകൾക്ക് തൊട്ടു താഴെയാണ് കരൾ സ്ഥിതിചെയ്യുന്നത്.	മനുഷ്യശരീരത്തിൽ ഉദരത്തിന്റെ വലതുഭാഗത്ത് വാരിയെല്ലകൾക്ക് തൊട്ടു താഴെയാണ്	2	1
2. മഞ്ഞപിത്തരോഗിയുടെ തൊലിയും കൃഷ്ണമണിയുടെ വെളുത്തമണ്ഡലവും മഞ്ഞനിറമാകുന്നു.		0	100
3. പ്രധാനമായും മലിനജലത്തിലൂടെ പകരുന്ന ഹെപ്പറ്റൈറ്റിസ്എ അവികസിത രാജ്യങ്ങളിലാണ് കൂടുതലായി കാണപ്പെടുന്നത്.		0	100
4. ഏതു പ്രായത്തിലുള്ളവരെയും ഹെപ്പറ്റൈറ്റിസ്എ മഞ്ഞപിത്തം ബാധിക്കാമെങ്കിലും കുട്ടികളിലാണ് കൂടുതൽ രോഗസാദ്ധ്യത		0	100
5. മനുഷ്യശരീരത്തിൽ ഉദരത്തിന്റെ വലതുഭാഗത്ത് പ്രാചീരത്തിനു തൊട്ടു താഴെയാണ് ശരീരത്തിലെ രാസപ്രക്രിയകളിൽ നല്ലൊരു ഭാഗം നടക്കുന്ന കരൾ സ്ഥിതി ചെയ്യുന്നത്.		2	2

Sentences retrieved, their weights and ranks assigned are given Table 4.4. Sentences with zero rank (no significant words present) are assigned a bigger number as Rank. In this case it is 100. From the top ranked sentence (rank=1), the phrase corresponding to the expected named entity 'LOCATION' is separated and it is displayed as the answer. Screen shot for the above example is given in Fig 4.6. A few screenshots are given in Appendix I.



കരൾ സ്ഥിതിചെയ്യുന്നത് എവിടെ

മനുഷ്യശരീരത്തിൽ ഉദരത്തിന്റെ വലതുഭാഗത്ത് വാരിയെല്ലുകൾക്ക് തൊട്ടു താഴെയാണ്

MaQAS

Named Entity Based Question Answering System for Malayalam Language

BINDU.M.S SUMAM MARY IDICULA

Dept. of Computer Science

Fig 4.6 Output of MaQAS

4.2 Chapter Summary

Architecture of MaQAS is depicted in this chapter. Indexing Module preprocesses all Malayalam documents and prepares a double level index. Question Analysis Module analyses the user query and understands the type of the expected answer. Then Answer Extraction Module searches through the index and determines the location of candidate sentences. These sentences are retrieved from the NE tagged documents and rank them. The Named Entity which is expected as the result is extracted from the top ranked sentence and returned to the user.

Chapter 5

COMPOUND WORD SPLITTER

Contents

Understanding each and every word or sentence of the documents in the corpus is essential to find an exact answer. Malayalam being an agglutinative language most of the words in a Malayalam document are compound words. The meanings of compound words are not usually available in a dictionary or in a catalog. This fact shows how important a compound word splitter is!

Compound word is a word composed of two or more words either in the closed form, hyphenated form or in an open form [74]. Compounding is a common phenomenon in Malayalam, Bengali, Greek etc. Owing to this compounding or agglutinative nature 80-85% words in Malayalam documents are compound words. Hence a compound word splitter is essential to split these words into their components to deduce the word meaning.

IR and QA, two major tasks of NLU, require extensive knowledge of the compound words to have effective communication between computers and human. Compound word splitter is also an important preprocessing tool in NLP applications.

5.1 Malayalam Compound Word

In Malayalam language, a compound word might contain any number of prefixes, postfixes, suffixes, verbs, nouns, dhyodhakam, pronouns, adjectives, adverbs and qualifiers. Unlike English, Malayalam compound words exist in the closed form which does not contain spaces or other word boundaries.

Examples

ആസൂത്രണവൈഭവത്തോടുകൂടിയ

(*aasUthraNavaibhavaTHOTukUTiya*) (with planning capability)

This Malayalam word is a combination of 5 atoms of different categories.

ആസൂത്രണം + വൈഭവം + ഓടു് + കൂടി + അ

(noun + noun + suffix + verb + dhyodhakam)

പൊളിച്‌ചെഴുതണമെന്നാണ് (*poLicchezhuthaNamennaN~*)

(The write-up is to be renewed)

പൊളി + എഴുത് + അണം + എന്ന് + ആണ്

(verb + verb + suffix + dhyodhakam + Aux)

Malayalam has a large number of compound words. All these are obtained by suitably combining the 10 atom types mentioned above [66]. A few examples are given below.

1) Verb + Verb = Verb

കേട്ടു (heard) + പഠിക്കണം(study) =കേട്ടുപഠിക്കണം (study by hearing)

(*kETTu + paTiKaNam = kETTu paTiKaNam*)

2) Adverb + Verb = Verb

പതുക്കെ(slowly) + ഓടി (ran) = പതുക്കെയോടി(ran slowly)

(*pathuKe + OTi= pathuKeyOTi*)

3) Noun + Noun = Noun

പന (palm tree) + കുരു (nut) = പനങ്കുരു(Seed of palm tree)

(*Pana + kuru = panankuru*)

4) Adjective + Noun +Verb + postfix = Verb

കറുത്ത + ആന + ഓടി + പോയി = കറുത്തയാനയോടിപോയി*(*black elephant ran away)

(*KaRuTHa + aana+ OTi+ pOyi = kaRuTHayaanayOTipoyi*)

5.2 Methods for Compound Word Splitting

To split the compound word, different methods are available.

5.2.1 Most Probable Word Technique

Find all possible ways of breaking the compound word (C) into pieces and choose the one with highest probability [75].

C = അപകടത്തിലാക്കുമെന്നും (*ApakaTatthilaaKumennum)*

(Will also be put into trouble)

Some possible ways of breaking this word include

1) അപ കടം ഇൽ ആക്ക ഉം എന്ന ഉം

(apa kaTam il aakka um enn um)

2) അപകടം ഇൽ ആക്ക ഉം എന്ന ഉം

(*apakaTam il aakka um enn um)*

3) അപകടം ഇ ലാക്ക് ഉം എന്ന ഉം

(*apakaTam E laaK um enn um)*

If the components are taken as S_1 to S_N and the corresponding probability as $Pr(S_1)$ to $Pr(S_N)$ then the overall probability of the string is $Pr(S_1 \ldots S_N) = Pr(S_1)*Pr(S_2)*\ldots\ldots\ldots\ldots*Pr(S_N)$. Here the probabilities are estimated by counting the occurrences in a selected corpus. The second case has the highest probability as probability of അപകടം (*apakaTam*) is greater compared to the probability of അപ (*apa*). Also the probability of ആക്ക (*aaKa*) is greater than that of ലാക്ക് (*laaK*) in the selected corpus. This approach produces many more entries, unnecessarily increasing the search time.

5.2.2 N- Gram Technique

The "N-gram" technique [76] is one approach to languages such as Japanese, Chinese and Korean which use no spaces between words. It requires less programming work on the back end and requires no dictionary or linguistic rules. N-gram segmentation works by chopping text into segments n-characters long, usually two to three characters. When "n" is two, a sliding window isolates strings of two-characters long from the input text. Using n-gram technique the above given word can be segmented into bigrams as follows: അപ പക കട ടത്ത ത്തി ില ലാ ാക്ക ക്കു ുമെ മെ മന്ന ന്നു ും (*apa paka kaTa Taththa ththi ila laa aakka kku ume me manna nnu um)*. The number of "false words" created by this segmentation process is relatively high and also in this example number of genuine words is nil. This method gives many outputs but poor result.

5.2.3 Longest Match Technique

The basic idea is to search for the longest parts of the word (from left to right). On a given string, a longest string-matching is first performed from left hand side. Let the given compound word be S. Then find the longest string X in the dictionary that is a prefix of S. Break this string off the beginning. Repeat this process until there is no such string in which case the remainder of the string is the last subcomponent [74].

e.g. കാരണങ്ങളുണ്ടാവാമെങ്കിലും (*kaaraNangaLunTaavaamenkilum)*

(though there may be reasons)

This word is split into sub-components as given below.

കാരണം   കൾ ഉണ്ട് ആവ ആം എങ്കിൽ ഉം

(kaaraNam kaL uNT~ aava aam engkil um) where ആവ (*aava*) is not the correct sub-component.

5.2.4 Baseline Technique

A simple approach is to choose the splits with as few components as possible or selecting the split with the longest first component in case of a tie [77]. This approach works quite well giving an accuracy of 85% on the ambiguous compounds in our corpus.

e.g. പെരുമാറിയിട്ടുള്ളയാളാണ് (*perumaaRiyiTTuLLayaaLaaNa~*)

(is a person who behaved)

Two possible splits of above word is given below

1. പെരുമ ആറി ഇട്ട് ഉള്ള ആൾ ആണ് (6 words)

   (*peruma aaRi itt uLLa aaL aaN~)*

2. പെരുമാറ് ഇ ഇട്ട് ഉള്ള ആൾ ആണ് (6 words)

   (perumaaR~ i itt uLLa aaL aaN~)

There are many possible combinations for this compound word. If we consider the above two combinations with 6 components each, baseline method selects the second suggestion since it has the longest first component.

5.2.5 Finite State Transducer (FST)

FST uses morphological analysis, requires a dictionary and the intelligence to recognize features of the language: punctuation, actual words, word forms and affixes. For the non-weighted FST, a simple morphological parsing method is used with the grammar rules where atoms are being classified by their functionality.

The decomposition algorithm and simple FST methods try to find substrings of the given words. The process looks for all valid decompositions of the given word.

eg.നേടിയെടുക്കുകയുമാവണം *(nETiyeTukkukayumaavaNam)* (it should be achieved)

The decompositions obtained are

1. നേടി എടു് ക്ക് ഉക ഉം ആ അണം

   (nETi eTu~ kk uka um aa aNam)

2. നേട് ഇയ എടു് കുക ഉം ആവണം

   (nET iya eTu~ kuka um aavaNam)

3. നേട് ഇയ എടു് കുക ഉം ആവ അണം

(*nET iya eTu~ kuka um aava aNam*)

To choose the correct interpretation some other mechanism is to be used.

5.2.6 Ad Hoc Rules

During the formation of compound words sometimes agamasandhi (augmentation) [66] must be used and there will be addition of consonants 'യ' (ya), 'മ' (ma), or 'വ' (va). But this will occur with certain word classes only. Same way if lopasandhi (elision) [66] is used certain consonants or vowels will be removed. Hence by checking the presence or absence of these letters, correct interpretation of the suggestion can be made.

e.g. എന്നിവിടങ്ങളിലേക്ക് (*enniviTaNGaLilEK)* (to the places mentioned)

1. എന്ന ഇവിടം കൾ ഇല ഏ ക്ക്

(*enna iviTam kaL ila ae kk)*

2. എന്ന ഇവിടം കൾ ഇൽ ഏ ക്ക്

(*enna iviTam kaL il ae kk)*

Here the ambiguity is resolved by checking the presence of 'യ' (ya) after 'ല' (la). For the first combination, the compound word would be

എന്ന + ഇവിടം + കൾ + ഇല + ഏ + ക്ക് = എന്നിവിടങ്ങളിലയേക്ക്

(enna + iviTam + kaL + ila + ae + kk = enniviTaNGaLilayEyK)

The sub-steps of forming the compound word are given below

a) എന്ന + ഇവിടം = എന്നിവിടം (*enna + iviTam = enniviTam*)

b) എന്നിവിടം + കൾ = എന്നിവിടങ്ങൾ (*enniviTam + kaL = enniviTangngaL*)

c) എന്നിവിടങ്ങൾ + ഇല = എന്നിവിടങ്ങളില

(enniviTangngaL + ila = enniviTangngaLila)

d) എന്നിവിടങ്ങളില + ഏ = എന്നിവിടങ്ങളിലയേ

(*enniviTangngaLila + ae = enniviTangngaLilayE*)

e) എന്നിവിടങ്ങളിലയേ + ക്ക് = എന്നിവിടങ്ങളിലയേക്ക്

(enniviTangngaLilayE + kk = enniviTaNGaLilayEyK)

This compound word obtained is different from the original compound word since (ഇല + ഏ = ഇലയേ) *(ila + ae = ilayE)*

The second group of elements gives

എന്ന + ഇവിടം + കൾ + ഇൽ + ഏ + ക്ക് = എന്നിവിടങ്ങളിലേക്ക്

(*enna + iviTam+ kaL+ il + ae + kk = enniviTaNGaLilEyK)*

since (ഇൽ + ഏ= ഇലേ) *(il + ae = ilE)*

5.2.7 Hybrid Method

Since the above methods make errors on the interpretation of different compound words, various methods are combined to obtain higher accuracy than individual methods. The hybrid system developed combines the baseline technique with POS and Ad Hoc rules. This method has an accuracy of 92% on the ambiguous compounds.

തിരിച്ചുകൊണ്ടുവരാനുമുള്ള (*thiricchukonTuvaraanumuLLa)*

(to bring back also)

1. തിര ഇ ച്ച് കൊണ്ടു് വര് ആൻ ഉം ഉള്ള (8)

   (thira i chch konTu~ var~ aan um uLLa)

2. തിരി ച്ച് കൊണ്ടു് വര ആൻ ഉം ഉള്ള (7)

   (thiri chch konTu~ vara aan um uLLa)

3. തിരി ച്ച് കൊണ്ട ് വര് ആൻ ഉം ഉള്ള (8)

   (thiri chch konTa u~ var~ aan um uLLa)

4. തിരി ച്ച് കൊണ്ടു് വര് ആൻ ഉം ഉള്ള (7)

   (thiri chch konTu~ var~ aan um uLLa)

In the previous example, 4 suggestions are given. (2) and (4) have less number of components. Hybrid method takes combinations (2) and (4) and checks the tagging of each component. 'തിരി'(*thiri*) can be a noun (candle) or a verb (turn) and the suffix 'ച്ച് '(*chch)* is possible only with a verb. Same way there is ambiguity between വര (*vara)* and വര് (*var~*). വര (*vara*) means line. വര് means come. But suffix 'ആൻ' *(aan)* can be used only with a verb. Hence the ambiguity is solved and (4) is selected as the correct interpretation.

5.2.8 Weighted FST

Weighted FSTs (WFST) are like ordinary FSTs with input and output symbols on arcs but they contain in addition a weight on every arc and every final state. These weights are combined during traversal of the automation to compute a weight for each path. To calculate weights various possibilities are tried.

- Number of segments
- Assigning probability to each grammar rule
- Component frequency

1) Compound segment with equal weight

In this method decomposition with less number of segments is selected as the correct morphological analysis [78]. To implement this technique equal weights are assigned to each segment.

നേടിയെടുക്കുകയുമാവണം (*nETiyeTukkukayumaavaNam*) (it should be achieved also)

With a segment weight 0.25 example given above gives overall weight as follows.

നേടി എടു് ക്ക് ഉക ഉം ആ അണം

(nETi eTu~ kk uka um aa aNam)

<0.25*0.25*0.25*0.25*0.25*0.25*0.25>=<0.000061025> (1)

നേടി എടു് ക്ക് ഉക ഉം ആവണം

(nETi eTu~ kk uka um aavaNam)

<0.25*0.25*0.25*0.25*0.25*0.25>=<0.0002441> (2)

നേട് ഇയ എടു കുക ഉം ആവ അണം

(nETi iya eTu~ kuka um aava aNam)

<0.25*0.25*0.25*0.25*0.25*0.25*0.25>=<0.000061025> (3)

2) Assigning probability to each grammar rule

FST uses grammar rules or morphological information to split the compound words. When many decompositions are possible, probability of grammar rule is considered to resolve the ambiguity [79].

3) Component Frequency

Given the count of words in a corpus the splits with highest geometric mean frequency of its parts Pi is chosen. The more frequent a word occurs in a training corpus the bigger the statistical basis to estimate translation probabilities [80].

കാരണങ്ങളുണ്ടാവാമെങ്കിലും (though there may be reasons)

(kaaraNangaLunTaavamenkilum)

കാര ണം കൾ ഉണ്ട് ആവ ആം എങ്കിൽ ഉം (4)

(kaara Nam kaL uNT~ aava aam engkil um)

കാരണം കൾ ഉണ്ട് ആവ ആം എങ്കിൽ ഉം (5)

(kaaraNam kaL uNT~ aava aam engkil um)

കാരണം കൾ ഉണ്ട് ആ ആം എങ്കിൽ ഉം (6)

(kaaraNam kaL uNT~ aa aam engkil um)

Component frequency is used for the calculation of probability of occurrence of each word. During the analysis, (6) is selected as the correct interpretation as കാരണം (*kaaraNam*) has a higher frequency than കാര *(kara*) in (4), also ആ (*aa*) in (6) has been referred to more times than ആവ (*aava*) in (4) and (5). Accuracy comparison of above mentioned methods are given in table 5.1.

Table 5.1 Accuracy Comparison of Compound Word Splitting Methods

Method	Accuracy
Most probable	70%
N-gram	25%
Longest Match	80%
Finite State Transducer	90%
Hybrid Method	92%
Weighted FST	98%

5.3 Compound Word Analyzer for Indian Languages

T. N. Vikram and Shalini R [81] developed a prototype of morphological analyzer for Kannada language based on Finite State Machine. This is just a prototype based on Finite State Machines and can simultaneously serve as a stemmer, Part-of-Speech tagger and spell checker. This analyzer has the capability to handle around 7000 distinct words from 500 distinct noun and verb stems. But it is far from a being full- fledged morph analyzer as pronoun and adjective morphology have not been included in this work. Also it does not handle compound formation.

Shambhavi B. R and Dr. Ramakanth Kumar (2011) [82] developed a paradigm based morphological generator and analyzer for Kannada language using a trie based data structure. The disadvantage of trie is that it consumes more memory as each node can have at most ‘y’ children, where y is the alphabet count of the language. As a result it can handle up to maximum 3700 root words and around 88K inflected words.

MORPH- A network and process model for Kannada morphological analysis/ generation was developed by K. Narayana Murthy (2001) [83] and the performance of the system is 60 to 70% on general texts.

Ramasamy Veerappan et.al [84] explains the working of MAG- a system for morphology analyzer and generator. Morphological analyzer was developed using FST. This project was developed as a part of MT system from English to Kannada. This model includes a lexicon, morphotactic, and orthographic rules. It handles only simple morphology.

Uma Maheshwar Rao G. and Parameshwari K. of CALTS, University of Hyderabad (2010) [85] attempted to develop a morphological analyzer and generators for South Dravidian languages.

Kiranmai.G et.al [86] presented a morphological analyzer for the classical Dravidian language Telugu using machine learning approach. The developed morphological analyzer is based on sequence labeling and training by kernel methods, it captures the non-linear relationships and various morphological features of Telugu language in a better and simpler way. This approach is more efficient than other morphological analyzers which were based on rules. In rule based approach every rule is depends on the previous rule. So if one rule fails, it will affect the entire rule that follows. Regarding the accuracy this system significantly achieves a very competitive accuracy of 94% and 97% in case of Telugu Verbs and nouns.

[87] describes the development of an open-source morphological analyzer for Bengali language using finite state technology. The morphological analyzer/generator was developed as a part of a new language pair for Apertium; hence the analyzer data conforms to Apertium's data format. The data for the analyzer/generator is contained in the Bengali monolingual dictionary. This morphological analyzer could also be used as a stemmer for any search engine for Bengali language. It could also double as a spell checker.

For Bengali, unsupervised methodology is used in developing a morphological analyzer system [88] and two-level morphology approach was used to handle Bengali compound words. This paper introduces a simple, yet highly effective algorithm for unsupervised morphological learning for Bengali, an Indo–Aryan language that is highly inflectional in nature. When evaluated on a set of 4,110 human-segmented Bengali words, our algorithm achieves an F-score of 83%, substantially outperforming Linguistica, one of the most widely-used unsupervised morphological parsers, by about 23%.

Rule based Morphological Analyzer were developed for Oriya and Sanskrit languages. [89] deals with the analysis and design of Oriya Morphological Analyzer (OMA). The major contents on which this OMA has been built up with i) Pronoun Morphology, ii) Inflectional Morphology and iii) Derivational Morphology. The OMA system is designed according to the object oriented approach to increase its reusability, robustness and extensibility. It also provides sufficient interface for applications involved in Oriya Machine Translation (OMT), Word-Net for Oriya (OriNet), Oriya Spell Checker and Oriya Grammar Checker.

[90] describes a Sanskrit morphological analyzer that identifies and analyzes inflected noun-forms and verb-forms in any given sandhi-free text. This system checks and labels each word as three basic POS categories - subanta, tiṅanta, and avyaya. Thereafter, each subanta is sent for subanta processing based on an example database and a rule database. The verbs are examined based on a database of verb roots and forms as well by reverse morphology based on Paninian techniques.

Authors of [91] describe a morphological analyzer for Marathi language that is a paradigm based inflectional system combined with FSMs for modeling the morphotactics. This morphological analyzer used a lexicon and inflection rules for all paradigms. The accuracy obtained was as high as 97.18%. This analyzer could not handle Derivational Morphology and compound words.

In Tamil language, the first step towards the preparation of morphological analyzer for Tamil was initiated by Anusaraka group. Ganesan (2007) [92] developed morphological analyzer for Tamil to analyze CIIL corpus. Phonological and morphophonemic rules are taken into for building morphological analyzer for Tamil. Resource Centre for Indian Language Technological Solutions (RCILTS) -Tamil has prepared a morphological analyzer (***Atcharam***) for Tamil. Finite automata state-table has been adopted for developing this Tamil morphological analyzer [93].

Parameswari.K (2010) [94] developed a Tamil morphological analyzer and generator using APERTIUM tool kit. This attempt involves a practical adoption of *lttoolbox* for the modern standard written Tamil in order to develop an improvised open source morphological analyzer and generator. The tool uses the computational algorithm Finite State Transducers (FST) for one-pass analysis and generation, and the database is developed in the morphological model called word and paradigm.

Vijay Sundar Ram R et.al (2010) [95] was designed Tamil Morphological Analyzer using paradigm based approach and Finite State Automata, which works efficiently in recursive tasks and considers only the current state for having a transition. In this approach complex affixations are easily handled by FSA and in the FSA, the required orthographic changes are handled in every state. In this approach, they built a FSA using all possible suffixes, categorize the root word lexicon based on paradigm approach to optimize the number of orthographic rules and use morpho-syntax rules to get the correct analysis for the given word.

Akshar bharati et.al (2001) [96] developed an algorithm for unsupervised learning of morphological analysis and generation of inflectionally rich languages. This algorithm uses the frequency of occurrences of word forms in a raw corpus. They introduce the concept of "observable paradigm "by forming equivalence classes of feature-structures which are not obvious. Frequency of word forms for each equivalence class is collected from such data for known paradigms. This method only depends on the frequencies of the word forms in raw corpora and does not require any linguistic rules or tagger. The performance of this system is dependant on the size of the corpora.

M.Ganesan (2007, 2009) [92] explained about a rule-based system for the analysis and generation of Tamil corpora. Successive split method is used for splitting the Tamil words. Initially all the words are treated as stems. In the first pass these stems are split into new stems and suffixes based on the similarity of the characters. They are split at the position where the two words differ. The right substring is stored in a suffix list and the left sub string is kept as a stem. In the second pass, the same procedure is followed, and the suffixes are stored in a separate suffix list.

Uma Maheswar Rao (2004) [85] proposed a modular model that is based on a hybrid approach which combines the two basic primary concepts of analyzing word forms and Paradigm model. This architecture involves the identification of different layers among the affixes, which enter into concatenation to generate word forms.

Menon et.al (2009) [97] developed a Finite State Transducer (FST) based morphological analyzer and generator. They have used AT &T Finite State Machine to build this tool. The system is based on lexicon and orthographic rules from a two level morphological system.

Deepa. S.R et.al [98] presented an algorithm for Hindi compound word splitting. This algorithm has shown a correct split rate of about 85%. A trie like structure is used to store and efficiently match the words. Also few modifications are suggested to handle compound words.

The work presented in [99] is a morphological analyzer for Hindi language. This uses the SFST (Stuttgart Finite State Transducer) tool for generating the FST. A lexicon of root words is created. Rules are then added for generating inflectional and derivational words from these root words. The morph analyzer developed was used in a Part Of Speech Tagger based on Stanford POS Tagger. It was tested with about 4000 inflectional, derivational and compound words and gives approximately 97% correct results.

5.4 Description of Malayalam Compound Word Splitter

A compound word M is decomposed into its constituents M_1 to M_i.To find each constituent, the longest match method is adopted. When one component, M_i, is separated, the remaining portion is sent to the Addition/

Deletion algorithm. The component M_i is first searched in the lexicon and if it is not found, Transformation algorithm is applied to obtain various forms of M_i and again searching is carried out. Even after transformation, if not found in the lexicon, process is repeated with next smaller substring of M. Working of Transformation and Addition Deletion algorithms are explained in Table 5.2.

Methodology: Finite State Transducer (FST)

Formally, a Finite State Transducer T is a 6-tuple (Q, Σ, Γ, I, F, δ) such that:

- Q is a finite set, the set of states;
- Σ is a finite set, called the input alphabet;
- Γ is a finite set, called the output alphabet;
- I is a subset of Q, the set of initial states;
- F is a subset of Q, the set of final states; and δ is the transition relation.

FST is a machine which accepts a string and translates it into another string. FST can also be used for generating and checking sequences [100].

A compound word is a sequence of Malayalam characters. To split this sequence into substrings FST can be used.

Compound word splitter uses an FST with the following definition.

In the 6-tuple, set of states Q	=	{A, B, C, D, E, F, G, H}
Initial state I	=	{A}
Final states F	=	{C, D, E, F, G, H}
Input alphabet Σ	=	{compound words}
Output alphabet Γ	=	{valid simple Malayalam words}
Transition function δ	=	{NOUN, VERB, ADJECTIVE,…, SUFFIX}

Table 5.2 Examples of Transformation and Addition/Deletion Algorithms

Compound Word	Component to be transformed	Transformation	Output of Transformation Algorithm	Component to be modified	Addition/ Deletion	O/P of Add/Del Algorithm
1. വൈഭവത്തോട്=വൈഭവ+ ത്തോട്	വൈഭവ	അം	വൈഭവം	ത്തോട്	Delete ത്തോ Add ഓ	ഓട്
2. വന്നിരുന്നു = വന്ന + ിരുന്നു	വന്ന	ഉ്	വന്ന്	ിരുന്നു	Delete ി Add ഇ	ഇരുന്നു
3. ചെയ്യാതായി = ചെയ്യാത + ായി	ചെയ്യാത	എ	ചെയ്യാതെ	ായി	Delete ാ Add ആ	ആയി
4. പാക്കഞ്ഞി = പാ + ക്കഞ്ഞി	പാ	ൽ	പാൽ	ക്കഞ്ഞി	Delete ക്ക Add ക	കഞ്ഞി
5. അവിടം = അ + വിടം	അ	No change	അ	വിടം	Delete വി Add ഇ	ഇടം
6. വെള്ളപ്പളുങ്ക് = വെള്ള + പ്പളുങ്ക	വെള്ള	No change	വെള്ള	പ്പളുങ്ക്	Delete പ്പ Add പ	പളുങ്ക്
7. തേങ്കനി = തേ + ങ്കനി	തേ	ൻ	തേൻ	ങ്കനി	Delete ങ്ക Add ക	കനി

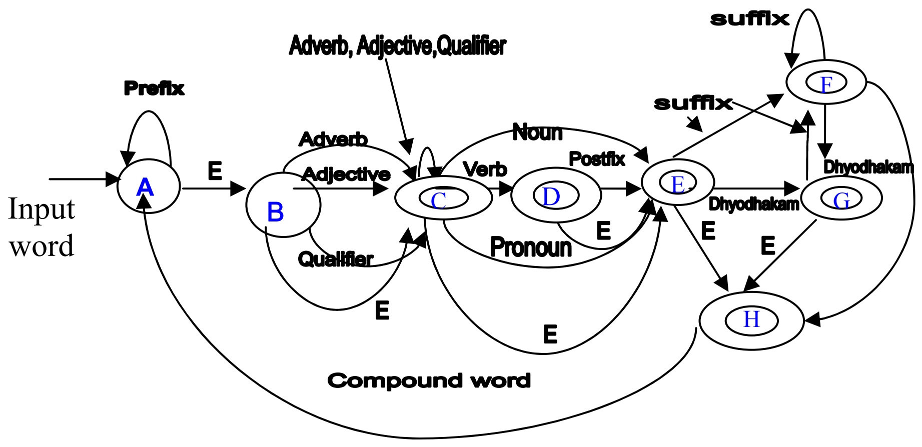


Fig 5.1 FST for Compound Word Splitter

The machine starts at state A and repeats the following process till the word size becomes zero. At the current state, word type of the input word is matched against the labels on the outgoing arcs and if a match is found, cross that arc and move to the next state. The word whose match is found is separated from the compound word and the remaining portion is taken as the input word to the next state. If no match is found, longest match algorithm is used to obtain the next longest word and continue the same process. Likewise, all possible combinations are tried and still if no match is found go to the next state without making any changes to the input word.

Referring to Fig 5.1, at state A all prefixes present consecutively are separated from the incoming word W, and the remaining portion of W or new W is sent to state B. At this state, consecutive adverbs or adjectives or qualifiers are removed. Then automation enters into state C. If a verb or a noun or a pronoun is present in the word W, it is separated and either enters into state D (if the match is a verb) or E. At state D, search for postfix is carried out and at E comparison is made for suffix/dhyodhakam. At state H if word size is zero (x=0) process stops or else goes to the initial state.

Table 5.3 State Table for the FST in Fig 5.1

Present State	Next state for X=1										Next state for X=0
	Prefix	Adj	Adv	Quafr	Noun	Verb	Postfix	P.Noun	Suffix	Dhyo.	Any Symbol
A	A	B	B	B	B	B	B	B	B	B	Error
B	C	C	C	C	C	C	C	C	C	C	Error
C	C	C	C	C	E	D	E	E	E	E	C
D	E	E	E	E	E	E	E	E	E	E	D
E	H	H	H	H	H	H	H	H	F	G	E
F	H	H	H	H	H	H	H	H	F	G	F
G	H	H	H	H	H	H	H	H	F	F	G
H	A	A	A	A	A	A	A	A	A	A	H

e.g. ആസൂത്രണവൈഭവത്തോടുകൂടിയ (*aasUthraNavaibhavaTHOTukUTiya)*
(with planning capacity)

Working of compound word splitter for this input word is explained below.

Step1

Input word at state A is ആസൂത്രണവൈഭവത്തോടുകൂടിയ (*aasUthraNavaibhavaTHOTukUTiya*) and no match is found at this state against the list of PREFIX. Hence automation goes to state B.

Step 2

Same word is processed at state B. Lists of ADJECTIVES, ADVERBS and QUALIFIERS are compared with the compound words of longest to smallest size. Again no match is found hence goes to state C.

Step 3

Above word is checked against the lists of NOUN, VERB and PRONOUN. As mentioned before, various possibilities are compared and the word ആസൂത്രണം (*aasUthraNam*) is separated from this compound word ആസൂത്രണവൈഭവത്തോടുകൂടിയ *(aasUthraNavaibhavaTHOTukUTiya*). When the longest word ആസൂത്രണ *(aasUthraNa*) is taken for comparison, different combinations are tried and an identical entry is located only for the word ആസൂത്രണം (*aasUthraNam*). Various alterations of words are obtained by the transformation algorithm whose working is explained in table 5.2. Automation continues at state E with the remaining word വൈഭവത്തോടുകൂടിയ (*vaibhavaTHOTukUTiya*).

Step 4

From state E process goes to state H. As the word size is not 0 at the present state next state is A as given in table 5.3.

Above process is repeated 3 times detaching ആസൂത്രണം (*aasUthraNam*) in the first cycle വൈഭവം *(vaibhavam)* and ഓട് (*Oat)* in the second cycle കൂടി (*kUTi*) and അ(*a*) in the third cycle.

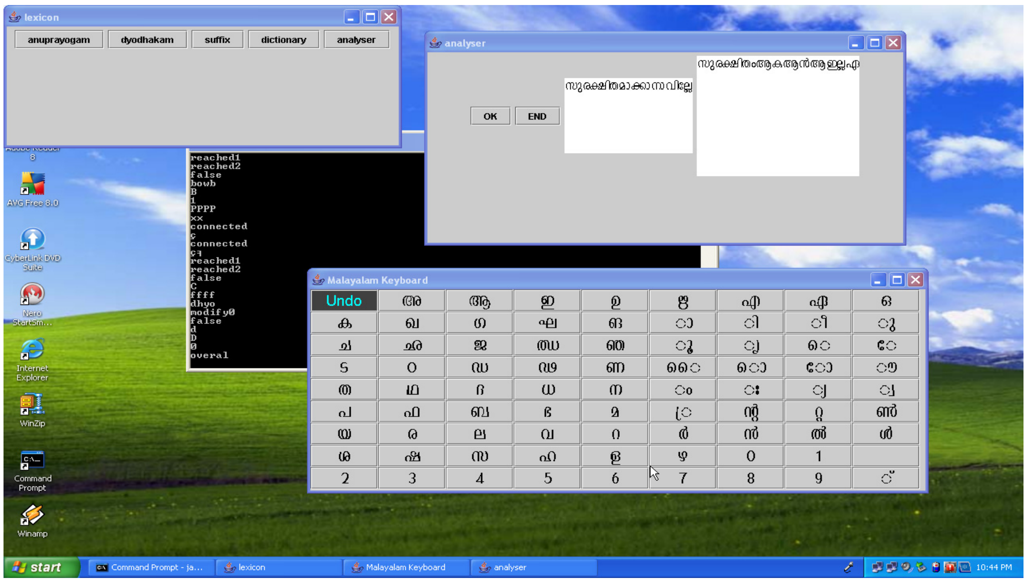
The compound word splitter splits the word ആസൂത്രണവൈഭവത്തോടുകൂടിയ (*aasUthraNavaibhavaTHOTukUTiya*) into five components as follows.

ആസൂത്രണം+ വൈഭവം+ ഓട് + കൂടി + അ

(*aasUthraNam + vaibhavam + Oat + kUTi+ a)*

(noun +noun + suffix+ verb+ dhyodhakam)

The traversal path of the machine during the decomposition of the above compound word is A-B-C-D-E-H-A-B-C-E-F-H-A-B-C-D-E-G-H.



സുരക്ഷിതമാക്കാനാവില്ലേ

സുരക്ഷിതംആകആൻആഇല്ലഏ

```
reached1
reached2
false
bowb
B
1
PPPP
xx
connected
ç
connected
ç¬
reached1
reached2
false
C
ffff
dhyo
modify0
false
d
D
0
overal
```

Fig 5.2 Output of Compound Word Splitter

Fig 5.2 shows the result of compound word splitter for the compound word സുരക്ഷിതമാക്കാനാവില്ലേ (*surakshithamaakkaanaavillE*). Output of splitter for this word is displayed as

സുരക്ഷിതം ആക്ക ആൻ ആ ഇല്ല ഏ (*surakshitham aakka aan aa illa E*)

Table 5.4 Examples of Compound Words and Components

Malayalam	Sub components								
Compound Words	Verb	Noun	Pronoun	Adj	Adv	Qualifier	Suffix	Prefix	Postfix	Dhyodhakam
പട്ടികയിലുൾപ്പെടുന്നതിനാലാണ് *(pattikayiluLppeTunnathinaalaaN~)* *(pattika+il+uL+peT+ Unna+ath+in+aal+aaN~)*	പെട് *(peT)* ആണ് *(aaN~)*	പട്ടിക *(pattika)* ഉൾ *(uL)*	അത് *(ath)*				ഇൽ *(il)* ഉന്ന *(inna)* ഇൻ *(in)* ആൽ *(aal)*			
ഇനങ്ങളിലൊക്കെപ്പെടുത്തി *(inangaLiloKeppeTuththi)* *(inam+ kaL+ il+ okke +peT+ th+ i)*	പെട് *(peT)*	ഇനം *(inam)*	ഒക്കെ *(okke)*				കൾ *(kaL)* ഇൽ *(il)* ഉ *(u)* ത് *(th)* ഇ *(i*			
തടസ്സപെട്ടിരിക്കുകയാണ് *(thaTassapeTTirikkukayaaN~)* *(thaTassam+ peTT+ iri + kk~+uka+ aaN~)*	പെട് *(peT* ഇരി *(iri)* ആണ് *(aaN~)*	തടസ്സം *(thaTassam)*					ക്ക് *(kk)* ഉക *(uka)*			
കളഞ്ഞുപോയിയെന്നുപറഞ്ഞു *(kaLanjnjupOyiyennupaRanjnju)* *(kaLa+njnj~+ pOyi+ enn+ paRa+njnju)*	കള *(kaLa* പറ *(paRa)*				എന്ന് *(enn)*		ഞ്ഞ് *(njnj)* ഉ *(u)*		പോയി *(pOyi)*	
വലിയകറുത്തവാഹനങ്ങൾക്കി ടയിലൂടെയോടിയെത്തി *(valiyakaRuththavaahanangaLkkiTayil uTeyOTiyeththi)(valiya +kaRuththa+ Vaahanam + kaL + kk + iTa + il + uuTe + oaTi + eth th + i)*	ഓടി *(oaTi)* എത്ത് *(eththi~)*	വാഹനം *(vaahanam)* ഇട*(iTa)*		വലിയ *(valiya)*		കറുത്ത *(kaRuththa)*	കൾ *(kaL)* ക്ക് *(kk)* ഇൽ *(il)* ഊടെ *(uuTe)*			ഇ *(i)*

A few examples of, compound words and their constituents obtained by the compound word splitter are listed in Table 5.4.

5.5 Performance Evaluation

The FST for compound word splitter was tested for about 4000 words and 96% of the splits gave correct result. For testing the system, words are selected from different corpuses and are analysed. Then a list of compound words is prepared. The corpuses used are

- Malayala Manorama: compounds related to current affairs (Editorial page) (year 2008).
- Mathrubhumi: compounds from the weekly (year 2008).
- Vanitha fortnightly.

Even though the selected lists are independent, 90% of compound words are common in all the three lists.

5.6 Chapter Summary

Importance of compound word splitting in the field of QA and various methods of the same are comprehended and learned. FST method is comparatively superior to all other methods which gave an accuracy of 96%. Complexity of Malayalam compound words is well studied and a FST based compound word splitter is developed.

Chapter 6

PART-OF-SPEECH TAGGER

Contents

Part-of-Speech Tagging is an essential and important pre-processing step in many NLP systems. Tagged corpora play a significant role in Machine Translation, Information Retrieval, Data Mining and QA Systems. In this QA System, POS Tagging is needed for document pre-processing and index preparation. Each document in the corpus is tokenized and tagged with an appropriate POS tag. POS is one of the important features used in the NE Recognition and Classification.

POS Tagging, also called grammatical tagging, is a principal issue in Natural Language Processing. The purpose of this task is to assign part-of-speech or other lexical class markers to each and every word in a document. In English there are eight parts-of-speech such as noun, verb etc. But POS tags vary according to the language and application [100].

Since 100 B.C humans are aware that language consists of several distinct parts called POS. Those POS play a crucial role in many fields of linguistics. POS is based on both its definition and its context or relationship with adjacent and related words in a phrase, sentence or paragraph. Part-of-speech Tagging is harder because some words can represent more than one POS at different times [101]. The significance of part-of-speech for language processing is the large amount of information they give about a word and its neighbour.

6.1 Related Work

The first major corpus of English for computer analysis was Brown corpus [100] which was tagged with POS markers over many years. In the mid 1980s researchers in Europe began to use Hidden Markov Model (HMM) [102] to disambiguate POS. In 1987 Steven J. DeRose [103] and Kenneth Ward Church [104] independently developed algorithms to solve this problem.

A lot of work has been done in Parts-of-Speech Tagging of western languages. These taggers mostly are implemented using stochastic or rule-based methods. In [105] author projects the use of Transformation-Based Learning (TBL) method to bootstrap the POS annotation results of English POS Tagger by exploiting the POS information of the corresponding Vietnamese words. The basic tagging step is achieved through the available POS Tagger (Brown) and the correction step is achieved through the Transformation-Based Learning method in which the information on the corresponding Vietnamese is used through available word alignment in the English Vietnamese parallel Corpus (EVC). A genetic algorithm for POS implementation is presented in [106] which correctly tag 93.8% of sentences using bigrams and word POS frequencies. This system is able to integrate statistical and rule-based approaches into one system. Eric Bril described a rule-based POS Tagger which automatically acquires rules and tags [101].

Rule-based techniques, and finite state machine morphological dictionary are some of the techniques used in ANUBHARATI [107], ANUVADHINI Machine Translation (MT) system for Bengali [108], and

Tamil spell checker [109]developed at Anna University, Chennai. In Tamil spell checker, morphological dictionary is internally represented by a set of FSTs that are automatically generated from a more general dictionary containing morphological syntactic information. A Punjabi spell checker [110] has been developed using Rule cum Dictionary-based method. It is a dictionary with search algorithms, which search string matching, fuzzy search, suffix stripping etc. Morphological stripping methods and paradigm-based FSTs are some other techniques used in Text Analyzers. In Indian languages Natural Language Processing tools are less as compared to English and other European languages. In Hindi a rule-based POS Tagger developed by IIT, Bombay, has been used in stemmer and morphological analyzer for Word Net project [111]. In Tamil, Anna university- K B Chandrasekhar (AU-KBC) research centre developed a morph analyzer and a POS Tagger [112]. But it is a rule-based system and its accuracy is less, and also showed word sense disambiguation problem for machine translation system [113].

Automatic Part-of-Speech Tagging is an area of Natural Language Processing where statistical techniques have been more successful than rule-based methods. Rule-based Taggers have many advantages over these taggers including vast reduction in stored information, the perspicuity of small set of meaningful rules, ease of finding and implementing improvements to the tagger and better probability from one tagset corpus genre or language to another. Stochastic approaches have often been preferred to rule-based approaches because of their robustness and their automatic training capabilities.

Smriti Singh et.al [114] have proposed a tagger for Hindi, that uses the affix information stored in a word and assigns a POS tag using no contextual information. By considering the previous and the next word in the verb group, it correctly identifies the main verb and the auxiliaries. Lexicon lookup was used for identifying the other POS categories.

Hidden Markov Model based tagger for Hindi was proposed by Manish Shrivastava and Pushpak Bhattacharyya [115]. The authors attempted to utilize the morphological richness of the languages without resorting to complex and expensive analysis. The core idea of their approach was to explode the input in order to increase the length of the input and to reduce the number of unique types encountered during learning. This in turn increases the probability score of the correct choice while simultaneously decreasing the ambiguity of the choices at each stage.

Nidhi Mishra and Amit Mishra [116] proposed a Part-of-Speech Tagger for Hindi Corpus in 2011. In the proposed method, the system scans the Hindi corpus and then extracts the sentences and words from the given corpus. Also the system search the tag pattern from database and display the tag of each Hindi word like noun tag, adjective tag, number tag, verb tag etc.

Based on lexical sequence constraints, a POS tagger algorithm for Hindi was proposed by Pradipta Ranjan Ray (2003) [117]. The proposed algorithm acts as the first level of POS Tagger, using constraint propagation, based on ontological information, morphological analysis information and lexical rules. Even though the performance of the POS tagger has not been statistically tested due to lack of lexical resources, it covers a wide range of language phenomenon and accurately captures the four major local dependencies in Hindi.

Sivaji Bandyopadhyay et.al (2006) [118] came up with a rule-based chunker for Bengali which gave an accuracy of 81.64 %. The chunker has been developed using rule-based approach since adequate training data was not available. The list of suffixes has been prepared for handling unknown words.

They used 435 suffixes; many of them usually appear at the end of verb, noun and adjective words.

For Bengali, Sandipan et al., (2007) [119] have developed a corpus based semisupervised learning algorithm for POS tagging based on HMMs. Their system uses a small tagged corpus (500 sentences) and a large unannotated corpus along with a Bengali morphological analyzer. When tested on a corpus of 100 sentences (1003 words), their system obtained an accuracy of 95%.

Hammad Ali [120] developed a Baum-Welch trained HMM tagger for the Bangla language. Training is performed in order to learn the underlying HMM parameters. This HMM could then be used to perform tagging on a corpus and tested to obtain the accuracy figures. The complete work was divided into the following phases: 1) collect corpus and tagset, 2) search for implementation of Baum-Welch algorithm, 3) perform training and 4) test against gold standard for accuracy.

Debasri Chakrabarti [121] proposed a rule based Parts-of-Speech tagger for Bangla with layered tagging. There are four levels of Tagging which also handles the tagging of multi verb expressions. In the first level ambiguous basic category of a word is assigned. Disambiguation rules are applied in the second level with more detail morphological information. At the third level multi word verbs are tagged and the fourth or the final level is the level of local word grouping or chunking.

A rule based part-of-speech tagging approach was used for Punjabi [122]. This tagger uses handwritten linguistic rules to disambiguate the part-of-speech information, which is possible for a given word, based on the context information. A tagset for use in this part-of-speech tagger has also been devised to incorporate all the grammatical properties that will be helpful in the later stages of grammar checking based on these tags.

Sreeganesh implemented a rule-based tagger for Telugu [123]. In the initial stage Telugu morphological analyzer analyses the input text. Then tagset is added. Around 524 morpho-syntactic rules do the disambiguation.

Avinesh et.al [124] proposed a Telugu tagger with a performance of 77.37%. In [125] three taggers are developed with accuracies 98.0%, 92.1%, and 87.8% respectively.

A SVM based POS tagger is explained in [126]. Multiclass SVM is used for classification. The tagset contained only 8 tags and overall accuracy obtained was 86.25% for passages which contains unknown words.

Approach presented in the paper [127] is a machine learning model. It uses supervised as well as unsupervised techniques. It uses a CRF to statistically tag the test corpus. The CRF is trained using features over a tagged and untagged data. Initially a rule based tagging code is run on the test data. This code used both machine learning and rule based features for tagging. It gave an accuracy of 86.43%. Then a CRF tool is used to test the data. It gave an accuracy of 89.90%.

A morphology driven Manipuri tagger is explained in [128]. This tagger was tested with 3784 sentences containing 10917 unique words. This tagger showed accuracy of 69%. Further this tagger is improved using CRF and SVM [129] and obtained accuracies of 72.04% and 74.38% respectively.

Antony P J and Soman KP [130] of Amrita University, Coimbatore proposed statistical approach to build a POS tagger for Kannada language using SVM. They have proposed a tagset consisting of 30 tags. The proposed POS tagger for Kannada language is based on supervised machine learning approach. The Part-of-Speech tagger for Kannada language was modeled using SVM kernel.

A rule-based POS tagger for Tamil was developed and tested by Dr.Arulmozhi P et.al [131]. This system gives only the major tags and the sub tags are overlooked during evaluation. A hybrid POS tagger for Tamil using HMM technique and a rule based system was also developed [132]. Parts of speech tagging scheme, tags a word in a sentence with its parts of speech. It is done in three stages: pre-editing, automatic tag assignment, and manual post-editing. In pre-editing, the corpus is converted to a suitable format to assign a POS tag to each word or word combination. Because of orthographic similarity one word may have several possible POS tags. After the initial assignment of possible POS, words are manually corrected to disambiguate words in texts.

Kathambam attaches parts of speech tags to the words of a given Tamil document. It uses heuristic rules based on Tamil linguistics for tagging and does not use either the dictionary or the morphological analyzer. It gives 80% efficiency for large documents, uses 12 heuristic rules and identifies the tags based on gender, tense and case markers. Standalone words are checked with the lists stored in the tagger. It uses 'Fill in rule' to tag 'unknown words. It also uses bigram for identifying the unknown word using the previous word category.

Lakshmana Pandian S and Geetha T V (2009) [133] developed CRF Models for Tamil Part of Speech Tagging and Chunking. This method avoids a fundamental limitation of Maximum Entropy Markov models (MEMMs) and other discriminative Markov models. The Language models are developed using CRF and designed based on morphological information of Tamil.

Selvam and Natarajan (2009) [134] have developed a rule based morphological analyzer and POS Tagger for Tamil. They improved the above systems using Projection and Induction techniques. Rule based morphological analyzer and POS tagger can be built from well defined morphological rules of Tamil. They applied alignment and projection techniques for projecting POS tags, and alignment, lemmatization and morphological induction techniques for inducing root words from English to Tamil. Categorical information and root words are obtained from POS projection and morphological induction respectively from English via alignment across sentence aligned corpora. They generated more than 600 POS tags for rule based morphological analysis and POS tagging.

6.2 Malayalam POS Tagging

POS Tagging is the process of identifying and labelling each word in a sentence with corresponding POS. Fixed word order languages like English have specific structure for a sentence. Therefore POS assignment in these languages can be done using the grammar rules. Malayalam is a free word order language; hence words can appear in any order in a sentence. But within a phrase, words are in a specific related order.

Since most of the words in a Malayalam document are compound words decomposition of these words into their constituents is extremely necessary for finalizing their POS tag. Sometimes more than one morphological analysis and hence more than one POS may occur for a single word. A correct resolution of this kind of ambiguity for each occurrence of the word is crucial in QA Systems. A large percentage of words also show ambiguity regarding lexical category. Hence to determine the POS of a word in Malayalam language, grammar rules themselves are not sufficient but both word and contextual levels of information are necessary.

Currently available tagsets for other languages are only giving importance to the morphological and syntactical properties of the language while the tagset developed considers the semantic features of the language. In many other languages, they have used tagset derived from Penn Tree Bank [100] or Brown corpus for POS Tagging. But these tagsets are not sufficient for POS Tagging in Malayalam. MaQAS needed a POS tagset which gives importance to semantics in its development.

6.2.1 POS Tagset for Malayalam

A tagset with 52 tags is developed by manually tagging different documents from various newspapers and various fields. This tagset contains POS tags necessary for Information Retrieval and Question Answering tasks in Malayalam language. POS tagset developed for this work is given in Table 6.3. The method used to find the POS tags and a few tagging examples are given below.

Noun

In this system nouns without suffixes and nouns with gender or plural suffixes are considered as ‘NOUN’ whereas the Penn tagset, a standard tagset used for English, makes distinction between noun singular, noun plural, common nouns, proper nouns etc. Nouns which are acting as agents are given the tag, NOUN which has three subcategories proper noun, pronoun and noun.

Example 1: കുട്ടി പാൽ കുടിച്ചു (*kuTTi paal kuTichchu*) (child drank milk)

In this example, the words, both child and milk, are nouns without suffixes. But the noun, child, is the agent of the action and milk is the object of the action; therefore 'NOUN' tag is assigned to child and 'OBJECT' (OBJ) tag is assigned to milk.

Example 2. കുട്ടികൾ പാൽ കുടിച്ചു (*kuTTikaL paal kuTichchu*)(children drank milk)

In example 2 the word 'children' (noun with plural suffix) is also assigned the POS tag 'NOUN'.

Noun with Case Suffix

When a noun occurs in a sentence with case suffix, POS tag of that noun is determined by the associated case. Here the case suffix changes the role of the word in the sentence. Case indicates a relation between noun and a verb [67]. Relation indicated by each case suffix is different. Hence separate POS tags are assigned to each unique case which are shown in Table 6.1.

Examples

രാജുവിന്റെ അമ്മ പോയി (*raajuvinte amma pOyi*) (Raju's mother has gone)

രാജു അമ്മയെ അടിച്ചു (*raaju ammaye aTichchu*) (Raju slapped his mother)

രാജു അമ്മയോട് സംസാരിച്ചു (*raaju ammayOT samsaarichchu)* (Raju talked to his mother)

രാജു അമ്മക്ക് സാരി വാങ്ങി (*raaju ammaykk saari vaangi*) (Raju bought a saree for his mother)

രാജു വടിയാൽ മകനെ അടിച്ചു *(raaju vaTiyaal makane aTichchu*) (Raju beat his son with a stick)

കമല അമ്മയുടെ സാരി വാങ്ങി (*kamala ammayuTe saari vaangi*) (Kamala got her mother's saree)

കമല ദൈവത്തിൽ ആശ്രയിച്ചു (*kamala daivaththil aaSrayichchu)* (Kamala depended on God)

Table 6.1 Example of Case Relations

Case Relation	suffix	Example	POS Tag
Nominative	No suffix	അമ്മ (*amma)*	NOUN
Accusative	എ	അമ്മയെ(*ammaye)*	ACC
Sociative	ഓട്	അമ്മയോട്(*ammayOT)*	SOC
Dative	ക്ക് ന്	അമ്മയ്ക്ക്(*ammaykk)*	DAT
Instrumental	ആ ൽ	അമ്മയാ ൽ(*ammaayal)*	INST
Possessive	ഉടെ ന്റെ	അമ്മയുടെ(*ammayuTe)*	GEN
Locative	ഇൽ കൽ	അമ്മയി ൽ(*ammavil)*	

All the underlined words in the above examples except the first are nouns but with case suffix. They are not agents of action but their case endings show the relationship of theirs to the agent and the main verbs.

Table 6.2 Examples of VERB Tags

Example	Role of underlined word	Tag
രാജുവിന്റെഅമ്മ പോയി	Verb	VERB
വാങ്ങിയ പേന	Adjectival Participle	AdjP
വരാൻ പറഞ്ഞു	Adverbial Participle	AdvP

Verbs

Finite verbs are tagged with VERB tag. Infinite verbs are divided into adjectival participle and adverbial participle. Adjectival participle relies on nouns and adverbial participle on verbs. All the three forms function distinctly [67]. Table 6.2 gives examples of VERB tags.

Examples

1. രാജുവിന്റെ അമ്മ പോയി (*raajuvinte amma pOyi*) (Raju's mother has gone)

Here the word പോയി is marked as finite verb and the corresponding POS tag is 'VERB'.

2. വാങ്ങിയ പേന (*vaangiya pEna*) (pen which was bought)

Adjectival participle വാങ്ങിയ is given the tag ‘AdjP’.

3. <u>വരാൻ</u> പറഞ്ഞു (*varaan paRanju*) (told to come)

The tag ‘AdvP’ is assigned to the Adverbial participle വരാൻ.

Table 6.3 POS Tags

TAG	DESCRIPTION	TAG	DESCRIPTION
NOUN	Noun	AdvP	Adverbial Participle
PN	Proper Noun	AdvT	Adverbial clause of Time
RN	Pronoun	AdvPl	Adverbial clause of Place
ACC	Accusative	AdvPr	Adverbial clause of Purpose
DAT	Dative	AdvR	Adverbial clause of Result
GEN	Genitive or Possessive	AdvSe	Adverbial clause of Reason
LOC	Locative	AdvCo	Adverbial clause of Comparison
SOC	Sociative	Adv-S	Adverbial clause of Supposition
INST	Instrumental	AdvC	Adverbial clause of Condition
OBJ	Object	AdjQl	Adj clause of Quality
RES	Reason	AdjQn	Adj clause of Quantity
PSP1	Postposition Type1	AdjD	Adj clause of Description
PSP2	Postposition Type2	AdjN	Adj clause of Number
PSP3	Postposition Type3	AdjI	Adj clause of Interrogation
PSP4	Postposition Type4	AdjE	Adj clause of Exclamation
PSP5	Postposition Type5	SourceP	Source- Place(Noun+PSP2)
PSP6	Postposition Type6	DestP	Destination-Place (Noun+PSP3)
PSP7	Postposition Type7	LikeP	Word Showing Similarity (Noun+ PSP 9)
PSP8	Postposition Type8	ListP	Word indicating a list(Noun+PSP11)
PSP9	Postposition Type9	TimeP	Word indicating Time(Noun +PSP 8)
PSP10	Postposition Type10	ThruP	Word indicating a path or friendship (Noun+PSP5)
PSP11	Postposition Type11	SYM	Symbol
VERB	Verb	CN	Cardinal Number
AuxV	Auxiliary Verb	ON	Ordinal Number
AdjP	Adjectival Participle	INT1	Wh Words
Adv	Adverb	INT2	Yes/No Words

Postposition

This is a word used along with a noun or a pronoun to show how it is related to something else. Each postposition is assigned a different POS as they serve different roles. ‘PSP1’ to ‘PSP11’ are the tags assigned to different postpositions which are listed in table 6.4.

Table 6.4 Examples of Postpositions

Word	Tag	Word	Tag
കൊണ്ട്	PSP1	കാരണം	PSP7
മുതൽ	PSP2	തുടങ്ങി	PSP8
വരെ	PSP3	പോലെ	PSP9
നിന്ന് വച്ച് മേൽ	PSP4	നേരേ	PSP10
കൂടി	PSP5	തൊട്ട്	PSP11
പറ്റി വേണ്ടി മറിച്ച്	PSP6		

Adjectives

An adjective is a word that adds to the meaning of the noun. It is working as a qualifier of the noun. There are different kinds of adjectives. Adjective indicates quality, number, quantity etc. The POS tag varies according to the type of the adjective. Examples are AdjQl, AdjN, AdjQn.

Adverbs

An adverb adds something to the meaning of a verb, adjective or another adverb. Adverb indicates time, condition, reason etc. POS tags corresponding to these are AdvT, AdvC, and AdvR.

Examples of Adjectives and Adverbs

Adjectives

- Adjective of Quality - AdjQ നല്ല

❖ Adjective of Quantity	-	AdjQn നിറയെ
❖ Adjective of Number	-	AdjN ഓരോന്ന്
❖ Demonstrative Adjective	-	AdjD ഇത്

Adverbs

❖ Adverbial clause of Time	-	AdvT മുതൽ
❖ Adverbial clause of Place	-	AdvPl അവിടെ
❖ Adverbial clause of Condition	-	AdvC ആൽ
❖ Adverbial clause of Purpose	-	AdvPr ആയി

Examples of other POS

❖ SourceP	- കൊച്ചിമുതൽ
❖ DestP	- കൊല്ലംവരെ
❖ LikeP	- അശോകനെപോലെ
❖ ListP	- ഉപ്പുതൊട്ട്
❖ TimeP	- അന്നതുടങ്ങി
❖ ThruP	- ആകാശത്തിലൂടെ

6.2.2 Part-of-Speech Tagger for Malayalam

Fig.6.1 shows the block diagram of the POS Tagger which was developed. Working of this tagger is as follows. The input document is divided into tokens. Then each token is sent to the Word Analyzer for the detailed analysis. Word Checker checks the word and determines whether it is a compound word or root word. If it is a root word, the Word Type Identifier assigns suitable tags using the information from the lexicon. A view of lexicon, prepared for this work is given in Appendix C.

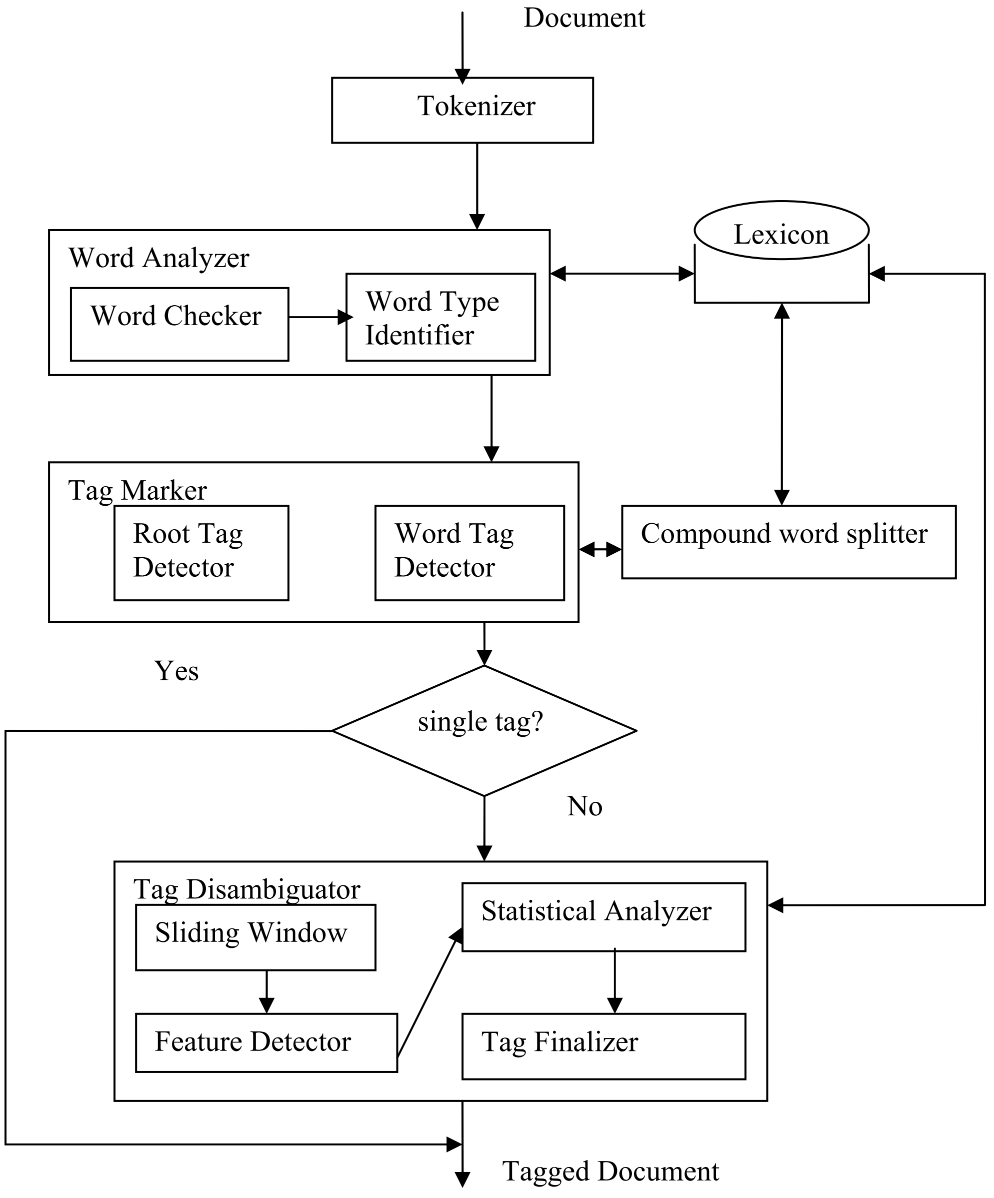


Fig 6.1 Block Diagram of POS Tagger

If the token is a compound word, the morphological details and the constituents of it are determined by the Compound word splitter. The Tag Marker assigns to each token, all possible POS tags with the help of information provided by the Word Analyzer and the Compound word splitter. Root Tag Detector assigns POS tags of the root words and the Word Tag Detector assigns POS tags of the compound words. If the tag is not unique or ambiguous, ambiguity is resolved by the Tag Disambiguator. Extended Conditional Random Field (ECRF) is the methodology used for the Tag Disambiguator. It uses contextual information to solve the ambiguity and eliminates all but one tag.

Tokenizer

Complete document can be assumed as a string of characters. Tokenization is the process of splitting a string of characters into lexical elements such as words and punctuations. This task is done by locating word boundaries.

Input to the Tokenizer block in Fig 6.1 is a document in Malayalam. During the tokenization process each sentence of the document is taken and splits into words or co-occurrence patterns. Multi part words are one of the main issues arise while tagging. These words are considered as single words to keep the semantic information intact for the purpose of efficient information extraction.

Word Analyzer

Word Analyzer checks each token to see whether it is present in the lexicon or not. Lexicon has all the root words along with its POS information. If it is present in the lexicon then it is a simple word and the word is labelled with POS details retrieved from the lexicon. Else the token is a compound word and it is labelled with <CW> tag.

Table 6.5 Output of Word Analyzer for Example 3

Token	Tag
കമല (*kamala)*	< Noun>
അമ്മയുടെ (*ammayuTe)*	<CW>
സാരി(*saari)*	< Noun>
വാങ്ങി (*vaangi)*	<CW>

Example 3 കമല <u>അമ്മയുടെ</u> സാരി വാങ്ങി (*kamala ammayuTe saari vaangi)* (Kamala got her mother's sari)

For this sentence, Word Analyzer produces the output shown in Table 6.5.

Tag Marker

Tag Marker uses the information provided by the Word Analyzer and marks the token with the most appropriate POS tag. It is designed with a Finite State Transducer as shown in Fig 6.2. Tokens marked with <CW> tags are sent to the Compound word splitter one by one and then receive the constituents of the words along with their POS information. Next the Tag Marker assigns suitable tags to each compound word based on the constituents. Sometimes there will be different valid decompositions possible for a compound word and in such cases the word will be marked with multiple tags.

For the FST in Fig 6.2 following elements form the tuple.

Initial state	–	A
Final states	-	{B, C,..Z}
Input Alphabet	-	{NOUN, VERB…, DHYODHAKAM}
Output alphabet	-	{Any POS from the tagset}
Transition function	-	{any valid POS}

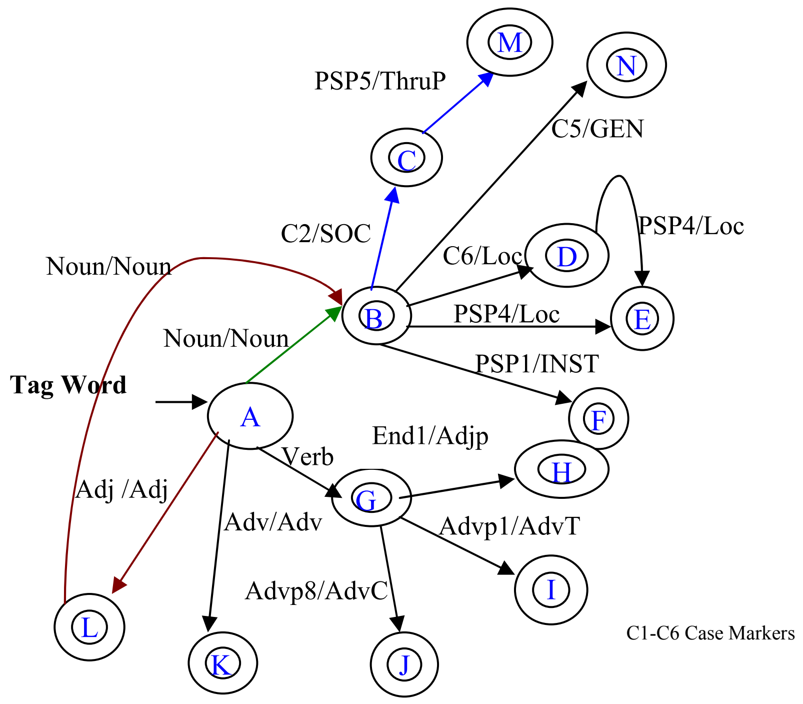


Fig 6.2 FST for Tag Marker

Example

For the word നഗരത്തിലേക്ക് (*nagaraththilEkk)* (towards the city), Compound word splitter produces the following output.

നഗരം<NOUN> ഇൽ<C6> ഏക്ക്<PSP4>

(*nagaram*<NOUN> *il* <C6> *aekk*<PSP4>)

Then the FST for the Tag Marker takes <NOUN> <C6> <PSP4> as the input string and by traversing the path A-B-D-E produces the output 'LOC'. i.e., the POS for the word നഗരത്തിലേക്ക് (*nagaraththilEkk*) is 'LOC'. Output of Tag Marker for a few examples are given in table 6.6.

Table 6.6 Output of Tag Marker

Word	POS string	Traversal Path	POS tag
കുട്ടി (*kutti)* (child)	Noun	A-B	NOUN
ചെറിയകുട്ടി(*cheRiyakutti)*(small child)	Adj + Noun	A-L-B	NOUN
ചെറിയകുട്ടിയുടെ(*cheRiyakuttiyuTe)* (of the small child)	Adj + Noun+Suffix-C5	A-L-B-N	GEN
ചെറിയകുട്ടിയോടുകൂടെ (*cheRiyakuttiyOTukUTe)* (with the small child)	Adj +Noun+Suffix-C2+PSP5	A-L-B-C-M	ThruP

Tag Disambiguator

Tag Marker assigns each input token a POS tag or multiple tags. Tokens with multiple tags are sent to the Disambiguator to solve the tag ambiguity which removes all tags except one. Output of Tag Disambiguator is a string of all tokens along with their POS tags.

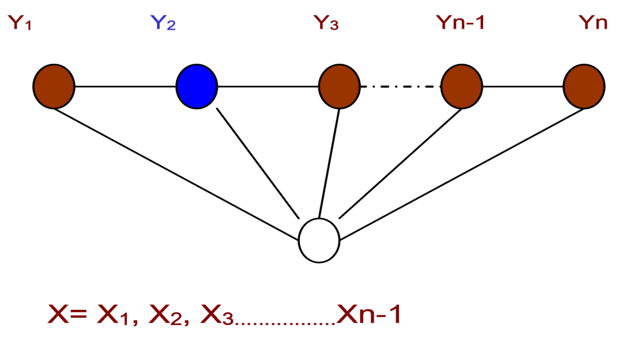


Fig 6.3 Graphical structure of chain-structured CRFs

Methodology

Tag Disambiguator is implemented using High Order Conditional Random Field or Extended CRF. It is an undirected graphical model in which each vertex represents a random variable whose probability distribution is to be inferred and each edge represents a dependency between two random variables [135][136].

Considering Fig 6.3, X={X_1 …X_N} and Y= {Y_1…Y_N} are two sets of random fields. For the given input sequence X, Y represents a hidden state variable and CRF's define conditional probability distributions P (Y|X) over the input sequence. Sometimes the conditional dependency of each Y_i on X will be defined through a fixed set of feature functions (potential functions) of the form f (i, Y_{i-1}, Y_i, X). The model assigns each feature a numerical weight and combines them to determine the probability of a certain value for Y_i. CRF's can contain any number of feature functions and the feature function can inspect the entire input sequence X at any point during inference. CRF's are extended into high order models by making each Yi dependent on a fixed number of previous variables Y_{i-o} ... Y_{i-1}.

POS Tagging can be modelled as a sequence labelling task where X=$X_1X_2X_3$...Xn represents an input sequence of words and Y= $Y_1Y_2Y_3$...Yn represents corresponding POS label sequence. The general label sequence Y has the highest probability of occurrence for the word sequence X among all possible label sequences, that is, Y = argmax {Pr (Y|X)}. Referring Fig 6.3, a word X can be assigned any one of the labels Y_1 to Yn and the relation between X and these labels are determined by the various feature functions. The relation which computes to the maximum probability is the best choice and hence corresponding label or tag is selected.

Main features for POS Tagging have been identified based on the word combination and word context. Following are the features used for POS Tagging in Malayalam.

- Constituents of current word: These determine the POS tag of the word as noun, verb etc.

- Context word features: Preceding (pw) and following words (nw) of the current word. pw1, pw2, pw3, nw1, nw2, nw3 are the features selected for this work.
- POS information: POS of previous words and successive words
- Digits or symbols present: If the word contains digits they are marked with 'NUMBER' POS (cardinal number (CN) or ordinal number (ON)).
- Lexicon feature: It contains Malayalam root words and their basic POS information such as noun, verb, adjective, adverb etc.
- Inflection lists: After analysing various classes of words inflection lists of nouns, verbs, and participles are prepared to improve the performance of the POS Tagger.

Example: അവൻ വെള്ളമുണ്ട് ഉടുത്തു. (*avan veLLamunT uTuththu)*

(He wore white Dhoti.)

Word Analyzer checks each word of this sentence and produces the following output.

അവൻ< NOUN> വെള്ളമുണ്ട്<CW> ഉടുത്തു<CW>

(*avan* <NOUN> *veLLamunT* <CW> *uTuththu*<CW>)

Then the Tag Marker sends the last two words with the <CW> tag to the Compound word splitter which decomposes these words into its constituents as shown below.

വെള്ളമുണ്ട്← വെള്ളം+ ഉണ്ട് (noun + verb)

വെള്ള+മുണ്ട് (adjective +noun)

veLLamunT ← *veLLam+unT* (noun+verb)

veLLa +munT (adjective +noun)

ഉടുത്തു ← ഉടു + ഉ (verb +suffix)

uTuththu ←*uTu +u* (verb+suffix)

The Tag Marker finalizes the POS tag of the word ഉടുത്തു *(uTuththu*) as 'VERB'. The word വെള്ളമുണ്ട് (*veLLamunT)* can be a verb or a noun as the POS of the compound word is based on the POS of the tail component of the compound word. This situation is solved by the Disambiguator. For the Disambiguator, input sentence represents X and corresponding POS represents Y. According to the principles of CRF, each POS Yi, is dependant on the corresponding word of the sentence X. But in Malayalam language, Yi is not only based on Xi but also on certain other features mentioned above.

Probability Calculation

In the sentence അവൻ വെള്ളമുണ്ട് ഉടുത്തു (*avan veLLamunT uTuththu*) the word വെള്ളമുണ്ട് *(veLLamunT*) has two POS tags VERB and NOUN. For this sentence Tag Marker produces the following output.

അവൻ < NOUN>   വെള്ളമുണ്ട് < VERB/ NOUN>ഉടുത്തു <VERB>

(avan   <NOUN> *veLLamunT* <VERB/NOUN> *uTuththu* <VERB>)

Table 6.7 gives possible POS tags of successive and preceding words of VERB and that of a NOUN in question.

Table 6.7 Tagging Features

VERB	NOUN
POS $_{i-1}$ = adv	POS $_{i-1}$ = adj
POS $_{i-1}$ = noun	POS $_{i-1}$ =adjP
POS $_{i-1}$ = advP	POS $_{i-1}$ = noun
POS $_{i-1}$ = verb	POS $_{i+1}$ = noun
POS $_{i-1}$ =PSP	POS $_{i+1}$ =verb
POS $_{i+1}$ = AuxV	POS $_{i+1}$ = adv
POS $_{i+1}$ = verb	POS $_{i+1}$ = advp
	POS $_{i+1}$ = adj
	POS $_{i+1}$ = PSP
Probability of POS$_{i-1}$ = 1/5	Probability of POS$_{i-1}$ = 1/3
Probability of POS $_{i+1}$= ½	Probability of POS $_{i+1}$= 1/6

Considering the above example

P (VERB| വെള്ളമുണ്ട്(*veLLamunT*)) = sum of the probabilities of the features

= probability of the preceding word + probability of the following word

= 1/5+0=1/5=0.2 ---- (1)

Similarly P (NOUN | വെള്ളമുണ്ട് (*veLLamunT*)) = 1/3+1/6=1/2=0.5----------(2)

P (Y|X) = maximum of (1) and (2) and hence POS label NOUN is assigned to the word വെള്ളമുണ്ട് (*veLLamunT*).

When this same word occur in a sentence അവിടെ വെള്ളമുണ്ട് (*aviTe veLLamunT)* Disambiguator calculates the conditional probability P (verb| വെള്ളമുണ്ട് (*veLLamunT)*) =1/5 and P (noun| വെള്ളമുണ്ട് (*veLLamunT)*) = 0 and then finalizes the POS label as VERB. Above two examples of Tag Disambiguation are summarized in Table 6.8.

Table 6.8 Example of POS Tag Disambiguation

Word	Sentence	Probability Calculation	Max Value	POS Tag
വെള്ളമുണ്ട് (*veLLamunT*)	1) അവൻ വെള്ളമുണ്ട് ഉടുത്തു. (*avan veLLamunT uTuththu*)	P (VERB| *veLLamunT)* =1/5+ 0 = 1/5 =0.2 P (NOUN | *veLLamunT*) = 1/3+1/6=1/2=0.5	0.5	NOUN
	2) അവിടെ വെള്ളമുണ്ട് (*aviTe veLLamunT*)	P (verb| *veLLamunT)* =1/5 =0.2 P (noun| *veLLamunT*) = 0	0.2	VERB

E.g. വിവാഹം കഴിഞ്ഞ് നഗരത്തിലേക്ക് മടങ്ങിവന്ന രാമൻ അച്ഛന്റെ നിയോഗത്താൽ വനവാസത്തിനു പോയി

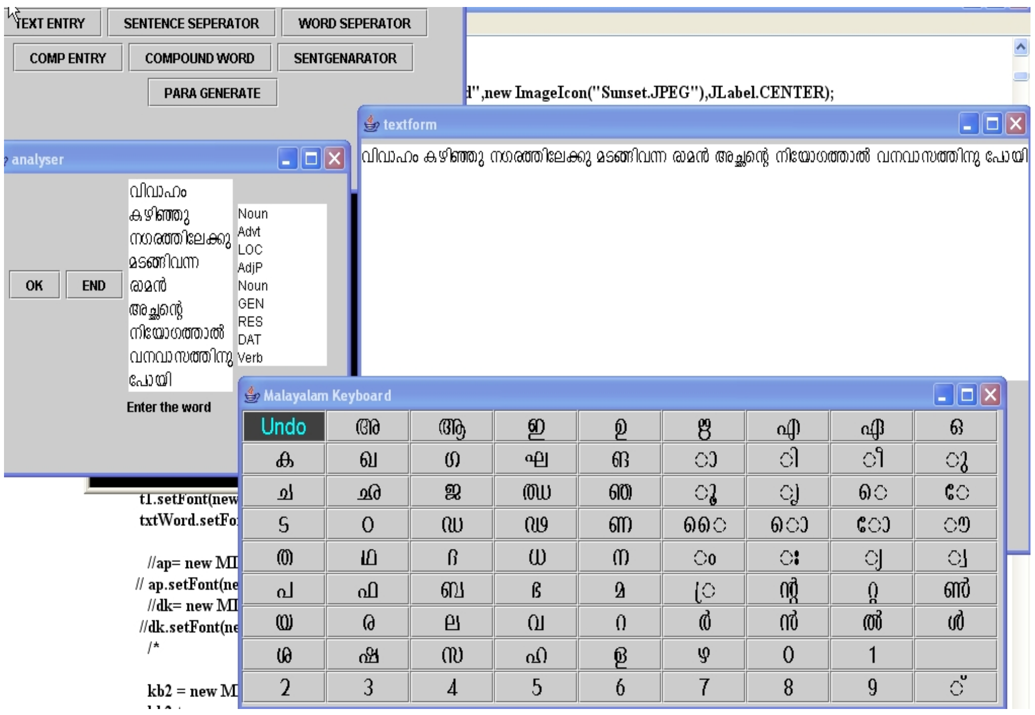
(vivaaham kazhinj nagaththilEkk maTangivanna raaman achchhante niyOgaththaal vanavaasaththinu pOyi)

(Raman who returned to city after marriage, went to lead forest-life by the instruction of his father)

Table 6.9 lists the outputs of Word Analyzer, Compound word splitter and Tag Marker for sentence given above. Fig 6.4 shows the output of POS Tagger for the same sentence.

Table 6.9 Ouput of POS Tagger

Stage 1 Output of Word Analyzer	Stage 2 Output of Compound Word Splitter	Stage 3 Output of Tag Marker
വിവാഹം <NCW Noun> (vivaaham<NCW Noun>)		വിവാഹം<NOUN> (vivaaham< NOUN>)
കഴിഞ്ഞ് <CW> (kazhinj <CW>)	കഴി <verb> ഞ്ഞ്<suffix> (kazhi <verb> nnj <suffix>)j	കഴിഞ്ഞ് <AdvT> (kazhinj <AdvT>)
നഗരത്തിലേക്ക് <CW> (nagaththilEkk <CW>)	നഗരം<Noun> ഇൽ <suffix> ഏക്ക്<suffix> (nagaram<Noun>il <suffix> aek <suffix>)	നഗരത്തിലേക്ക്<LOC> (nagaththilEkk <LOC>)
മടങ്ങിവന്ന <CW> (maTangivanna <CW>)	മടങ്ങി<verb> വന്ന<AdjP> (maTangi <verb>vanna<<AdjP>)	മടങ്ങിവന്ന<AdjP> (maTangivanna < AdjP>)
രാമൻ <NCW Proper noun> (raaman <NCW Proper noun>)		രാമൻ <Noun> (raaman <Noun>)
അച്ഛന്റെ <CW> (aachchhante<CW>)	അച്ഛൻ<Noun> ന്റെ<suffix> (achchhn<Noun>nte <suffix>)	അച്ഛന്റെ<GEN> (achchhnte<GEN>)
നിയോഗത്താൽ <CW> (niyOgaththaal <CW>)	നിയോഗം <Noun>ആൽ<suffix> (niyOgam <Noun> aal<suffix>)	നിയോഗത്താൽ <RES> (niyOgaththaal <RES>)
വനവാസത്തിനു <CW> (vanavaasaththinu<CW>)	വനവാസം<Noun> ന്<suffix> (vanavaasam<Noun> ni~<suffix>)	വനവാസത്തിനു<DAT> (vanavaasaththinu<DAT>)
പോയി <CW> (pOyi<CW>)	പോ<verb> ഇ<suffix> (pO<verb>I <suffix>)	പോയി <VERB> (pOyi<VERB>)



```
d",new ImageIcon("Sunset.JPEG"),JLabel.CENTER);
```

വിവാഹം കഴിഞ്ഞു നഗരത്തിലേക്കു മടങ്ങിവന്ന രാമൻ അച്ഛന്റെ നിയോഗത്താൽ വനവാസത്തിനു പോയി

വിവാഹം
കഴിഞ്ഞു
നഗരത്തിലേക്കു
മടങ്ങിവന്ന
രാമൻ
അച്ഛന്റെ
നിയോഗത്താൽ
വനവാസത്തിനു
പോയി

Noun
Advt
LOC
AdjP
Noun
GEN
RES
DAT
Verb

```
t1.setFont(new
txtWord.setFo

  //ap= new MI
// ap.setFont(ne
  //dk= new MI
 //dk.setFont(ne
  /*

  kb2 = new M
```

Fig 6.4 Screen shot of POS Tagger Output

6.3 Results and Discussion

The performance evaluation of the POS Tagger is carried out using the standardized evaluation techniques like precision and recall, where precision is defined as the ratio of correct number of token tag pair sequence in the output to total number of token tag pair that appears in the output and recall is the ratio of correct number of token tag pair sequence to the number of correct token tag pair that is possible [137].

POS Tagger results were analysed using a confusion matrix or contingency table. A contingency table is a simple performance analysis tool typically used in classification tasks and IR scenarios.

Table 6.10 A Typical Contingency Table

Contingency Table		Actual Class	
		Positive	Negative
Predicted Class	Positive	True Positive (Correct Result)	False Positive (Unexpected Result)
	Negative	False Negative (Missing Result)	True Negative (Correct absence of Result)

A 2*2 contingency table is shown in Table 6.10. The entries in the table have the following meaning in the context of POS labelling.

True Positive	– number of correct predictions that an instance is +ve
False Positive	– number of incorrect predictions that an instance is +ve
False Negative	– number of incorrect predictions that an instance is -ve
True Negative	– number of correct predictions that an instance is –ve

The **recall** or **true positive rate** is defined as the ratio of the number of true positives to the elements actually belonging to the positive class.

Recall = True Positive / (True Positive + False Negative)

The **precision** or **positive predictive value** is defined as the ratio of the number of true positives to the elements labeled as belonging to the positive class.

Precision = True Positive / (True Positive + False Positive)

Documents from Malayalam dailies Malayala Manorama, Mathrubhumi, Karshaka Sree, and few text books pertaining to five different fields were selected as the test corpus. These documents were texts with simple, compound and complex sentences and 95% of these words were compound words. Totally 2352 sentences were tested and obtained an average precision of about 92% and recall of 94%. Some words were not correctly tagged since all features were not included in the system. Performance evaluation details are listed in Appendix D.

Table 6.11 Overall Performance of the POS Tagger

Predicted ↓	Actual		
	Pos	Neg	Total
Pos	10947	951	11898
Neg	698	53	751
Total	11645	1004	12649

Average precision and recall can be obtained from the contingency table shown in Table 6.11.

TP = 10947 FP = 951 FN = 698 TN = 53

Average Precision = 10947 / (10947+951) = 92.0 %

Average Recall = 10947 / (10947+698) = 94.0 %

- This approach is accurate and efficient for a language like Malayalam since it considers language level and word level features.
- Malayalam language has no specific structure for a sentence. Hence it is very difficult to assign POS tag for a word; many times a word will have multiple tags. This problem is solved in this system by taking into account the contextual information.
- A POS tagset is developed which is unique in its nature as it reflects morphological, syntactical and semantic features of the language.

6.4 Chapter Summary

POS Tagging is the first and one of the important stages of document preprocessing. POS tags necessary for Malayalam document tagging and their method of development is described in this chapter. A study of, POS Taggers available in other languages are done and is also given in this chapter. A detailed discussion of Malayalam POS Tagger design, development and implementation is conducted.

Chapter 7

PHRASE CHUNKER

Contents

In this work, phrase chunking is the second phase of document preprocessing. Chunker identifies and divides sentences into syntactically correlated word groups. An Artificial Immunity System (AIS)-based phrase chunker is discussed which identifies and labels each phrase chunk with suitable phrase tag. Phrase tag is one of the important language dependant feature used for Named Entity Recognition and Classification.

QA is the task of automatically answering a question given in natural language. In order to find the answer, the question is analysed, and the type of expected answers is determined. Then the answer retrieval module retrieves either a document, passage, or a phrase as the answer to the query. Questions such as ‘when’, ‘where’, ‘what’, ‘why’ etc. mostly return an answer that contains a noun, a noun phrase or a prepositional phrase. Hence identification of phrases is an essential preprocessing step in QA Systems.

Phrase chunks can be identified and separated either by chunking or by way of full parsing. Abney [138] has proposed text chunking as a useful step for full parsing since it lays foundation for further levels of analysis. Abney describes chunking as a natural phenomenon in the following words.

“(when I read) (a sentence) (I read it) (a chunk) (at a time)”

Full parsing [139] is the syntactic analysis of a text, made up of tokens to determine the grammatical structure with respect to a given formal grammar. By this analysis it will be clear how words are combined to get phrases and phrases into sentences. Full parsing is an extremely difficult process for morphologically rich language like Malayalam. Also it is expensive and is not very robust. But chunking is more robust, efficient, faster, and sufficient for many applications.

Chunking is a shallow parsing technique also called light parsing for extracting non-overlapping segments from a stream of data [140]. A typical chunk consists of a single context word surrounded by a constellation of function words and each chunk contains a head word. Some words in a sentence may not be grouped into a chunk. A phrase chunker identifies phrases like verb phrase, noun phrase etc. in a sentence but does not either specify their internal structure or their role in the main sentence.

7.1 Related Work

The Artificial Immunity System Principle is explored in intelligent robotics system [141]. Authors of [141] have implemented a behaviour arbitration mechanism for the robots to choose the best option when an abnormal situation arises. Nasser Omer Sahel Ba-Karait et al. [142] explains the use of negative selection algorithm for the classification of Electroencephalography (EEG) signals and proved that this method reveals very promising performance. The general purpose algorithm based on the clonal selection and affinity maturation process in an adaptive immunity system is presented in [143] which is capable of solving complex engineering tasks like multimodal and combinatorial optimization. Clonal selection algorithms are applied for edge detection problems in pattern recognition and computer vision [144]. The principle of hyper mutation and receptor editing are exploited for the above purpose. Akshat Kumar and Shivashankar B Nair described the working of English Grammar checking system using adaptive immunity system [145].

In Hindi [146], Tamil [147], Telugu [148], Bengali [149] languages several works are reported but they are mainly using statistical approaches like Conditional Random Field (CRF), Hidden Markov Model (HMM) and Maximum Entropy Markov Models (MEMM). In Tamil a few taggers are developed employing Support Vector Machines (SVM) [150] and they obtained a tagging accuracy of 95.82%.

Many researchers have used various machine learning methods and their combinations for chunking. Noun phrase chunking is an important and useful task in many natural language applications and it is studied well in languages such as English [151] and French [140]. But for a less privileged language like Malayalam, Corpus-based NLP tasks are at deadlock due to the unavailability of lexicons and taggers.

[152] describe experiences in building an HMM based Part-Of-Speech (POS) tagger and statistical chunker for 3 Indian languages-Bengali, Hindi and Telugu. For chunking, the training data is used to extract chunk pattern templates defined as a sequence of POS tags. These templates, in conjunction with the POS tag of the word following the chunk, are used to decide chunk boundaries in the unannotated text. A dynamic programming algorithm is used to find the best possible chunk sequence. The chunk accuracies obtained are 67.52 for Bengali, 69.98 for Hindi and 68.32 for Telugu.

Pattabhi R.K Rao et.al [153] presents a Transformational-Based Learning (TBL) approach for text chunking. In this technique of chunking, a single base rule (or a few base rules) is provided to the system, and the other rules are learned by system itself during the training phase for reorganization of the chunks. This system is designed to work with three of the Indian languages namely Hindi, Bengali and Telugu.

In [154] authors describe a rule-based chunker that has been developed and tested on the Bengali development test set and demonstrated 85.88% accuracy. This chunker has then been tested on the Hindi and Telegu development sets and demonstrated 73.88% and 55.96% accuracies. The chunking system has demonstrated 80.63%, 71.65% and 53.15% accuracies with the unannotated Bengali, Hindi and Telegu test sets.

Paper [155] presents the building of POS Tagger and Chunk Tagger using Decision Forests and also focuses on the investigation towards exploring different methods for Parts-Of-Speech Tagging of Indian languages using sub-words as units. For chunking this system used 2-tag scheme. Features used for chunking are 2-level context of POS Tags namely present, previous, previous-previous, next and next-next word POS Tags. The two models POS Tagger and Chunk Tagger were tested with 3 different Indian languages Hindi, Bengali, Telugu and achieved the accuracies as 69.92%, 70.99%, 74.74% and 69.35%, 60.08%, 77.20% respectively.

A computational framework for chunking based on *valency theory* and *feature structures* has been described in [156]. This paper also draws an analogy of chunk formation in free order languages with the bonding of atoms, radicals or molecules to form complex chemical structures. A chunker has been implemented for Bengali using this approach with considerably good accuracy.

A rule based model [157] is developed using 21 linguistic rules for automatic VP chunking. A 100,000 word Urdu corpus is manually tagged with VP chunk tags. The corpus is then used to develop a hybrid approach using HMM based statistical chunking and correction rules. The technique is enhanced by changing chunking direction and merging chunk and POS tags. The automatically chunked data is compared with manually tagged held-out data to identify and analyze the errors. Based on the analysis, correction rules are extracted to address the errors. By applying these rules after statistical tagging, further improvement is achieved in chunking accuracy. The results of all experiments are reported with maximum overall accuracy of 98.44% achieved using hybrid approach with extended tagset.

In this work phrase chunker is implemented using a novel approach based on Artificial Immunity System. AIS is a new branch of AI used for intelligent problem solving technique in optimization and scheduling using the theory of human immunity system. AIS algorithms are more efficient than the classical heuristic scheduling algorithms. Also they are more successful than Genetic algorithms [158]. There are three algorithms widely applied: clonal selection algorithms [143], immune network algorithms [159] and negative selection algorithms [142]. AIS theory is not yet used for the development of phrase chunker. It is easy and possible to make any modifications to an AIS-based system as this system works on the principle of generation and deletion of self/non-self patterns.

7.2 Malayalam Phrase Chunker

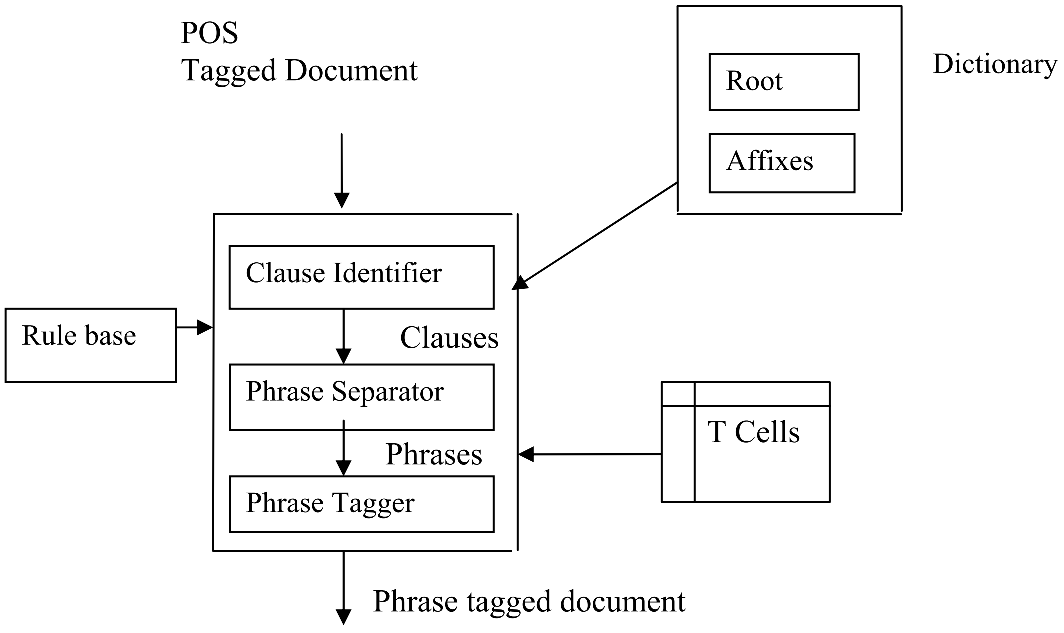


Fig 7.1 Detailed Architecture of Phrase Chunker

Phrase Chunker which is the second stage of document preprocessing module takes the output of POS Tagger and labels it with phrase tags. Clause identifier in Fig 7.1 identifies and separates the clauses from the sentences using a rule base. Then the phrase separator separates the phrases from each of these clauses and phrase tagger attaches appropriate phrase tags to them.

7.2.1 Clause Identifier

Malayalam sentences are mostly complex or compound sentences containing multiple clauses. Hence clause identification is an essential step of phrase chunking. Clause identifier identifies and separates clauses from the sentences using a rule base and gives the output to the phrase separator.

To obtain the clauses it is necessary to find out where a clause begins and where it ends. The POS tags assigned to each and every token in the sentence is used to determine these boundaries. Normally main clause ends with a verb and a subordinate clause ends with an infinite verb. This module identifies clauses with the help of handcrafted linguistic rules. A clause may end either with a verb, Auxiliary verb, Adjectival participle or an Adverbial participle.

In the rules given below, N, N+1, N+2, N+3 are the positions of the current word and the three words following the current word. Examples of the rules are:

Rule 1: IF N+1 adjp   N+2 Noun then separate clause

Rule 2: IF N+1 adjp N+2 adjp N+3 Noun then separate clause

Rule 3: IF N+1 advp then separate clause

7.2.2 Phrase Separator

Each clause obtained in the previous step is taken one by one and sent to the phrase separator. All the phrases corresponding to each clause is identified and separated. Then they are labelled with phrase tags.

Methodology

Phrase Separator (PS) is implemented using the principles of human immune system. There are two levels of defence mechanism in human body. First level of defence called innate immune system is available since birth and is regulated by white blood cells. Innate immune system defends the external organisms by making barriers such as mucus, low PH, saliva etc. Second level of defence is called adaptive immune system. This defence is pathogen specific and is controlled by Lymphocytes consisting of T-cells and B-cells. When a specific pathogen enters the human body the T-cells with the specific receptors recognizes and tries to destroy them. In case, the number of cells to attack the invading pathogen is insufficient then they are multiplied by clonal selection and attack the pathogen [143].

In a phrase separator a POS tagged clause is the input. Using the phrase grammar the POS tag string can be analysed to identify the phrase chunks. In an AIS-based phrase chunker the POS tag string can be assumed as a pathogen. A pathogen can be identified by the corresponding T-cells. Since the number of valid phrase chunks identified for Malayalam language is thirty two, there are 32 different T-cells to identify the phrase chunks. T-cells which recognize the valid phrase chunks are called self cells and others are called non self cells [160]. POS patterns for phrase identification are given in Appendix E. Once T-cells are created, the system is ready to work like an Artificial Immune System. i.e. the T-cells are mature enough to detect the presence of incoming phrase chunks (pathogens). If the chunks are detected, they are separated (thrown out) from the incoming clause.

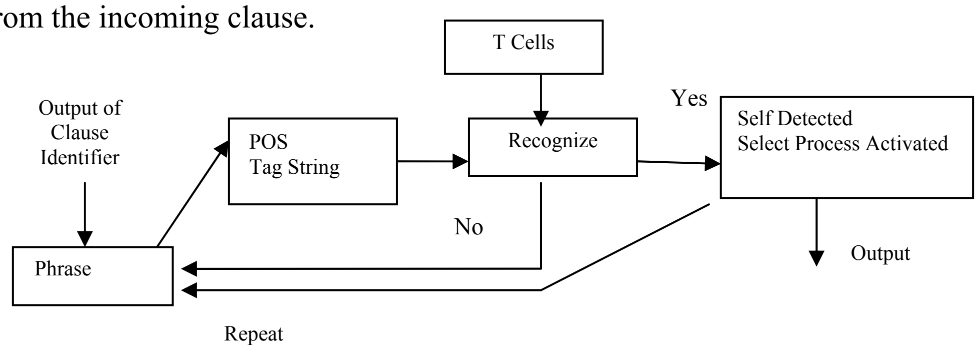

T Cells
Output of Clause Identifier
POS Tag String
Recognize
Yes
Self Detected Select Process Activated
No
Phrase
Output
Repeat

Fig 7.2 Working of Phrase Separator

Working of Phrase Separator is described in Fig 7.2. The PS module accepts a stream of words with the corresponding POS tags as input and group sub-sequences of words and tags that most likely form a phrase. Here, the phrase identifier groups every sequence and POS tag string separator separates the POS tag string from the word-tag sequence. Every subgroup of tags is compared to every pattern in the self set. If there is no match found then the tested pattern is of non self and a process is activated to reject the incoming phrase. If the tested pattern matches any self pattern then “accept” process is activated. This action is continued until all the clauses are split into phrases.

Table 7.1 Phrase Tags

Tag	Descriptions		Tag	Description
NP	Noun phrase		AdvpC	Adverbial clause of Condition
VP	Verb Phase		AdvpRe	Adverbial clause of Reason
NP-Acc	Noun phrase-Accusative		AdvpS	Adverbial clause of Supposition
NP-Dat	Noun phrase-Dative		AdvpCo	Adverbial clause of Comparison
NP-Gen	Noun phrase-Genitive		AdjpQl	Adjectival phrase of Quality
NP-Loc	Noun phrase-Locative		AdjpQn	Adjectival phrase of Quantity
NP-Soc	Noun phrase-Sociative		AdjpN	Adjectival phrase of Number
NP-Obj	Noun phrase-Object		AdjpE	Adjectival phrase of Emphasis
NP-Inst	Noun phrase-Instrumental		AdjpI	Adjectival phrase of Imperative
NP-Res	Noun phrase-Reason		AdjpD	Adjectival phrase of Description
AdjNP	Adjectival Participle of phrase		ThruPP	Noun Phrase- Path
AdVP	Adverbial Participle of phrase		ListPP	Noun Phrase -Group
AdvpT	Adverbial clause of Time		LikePP	Noun Phrase -Similarity
AdvpPr	Adverbial clause of Purpose		SourcePP	Noun Phrase -Source
AdvpR	Adverbial clause of Result		DestPP	Noun Phrase -Destination
AdvpP	Adverbial clause of Place		TimePP	Noun Phrase -Time

7.2.3 Phrase Tagger

This module assigns suitable phrase tags to the output of Phrase Separator. Phrase tags are listed in Table 7.1.

Examples

1. നീ നന്നായി പഠിച്ചാൽ നിനക്ക് ഫസ്റ്റ്ക്ലാസ് ലഭിക്കും.

(*nI nannaayi paThichchaal ninakk fast klaas labhikkum*

(If you study well you will get first class.)

POS Tagger produces the following output for the above sentence.

നീ <RN>നന്നായി <Adv> പഠിച്ചാൽ <AdvC> നിനക്ക് <Dat> ഫസ്റ്റ്ക്ലാസ് <OBJ> ലഭിക്കും<VERB>

nI <RN> *nannaayi* <Adv> *paThichchaal*<AdvC> *ninakk*<Dat> *fast klaas* <OBJ> *labhikkum*<VERB>

This POS tagged sentence is divided into two clauses as follows.

Clause 1: നീ നന്നായി പഠിച്ചാൽ (*nI nannaayi paThichchaal*) (subordinate clause)

Clause 2: നിനക്ക് **ഫസ്റ്റ്ക്ലാസ്** ലഭിക്കും **(***ninakk fast klaas labhikkum*) (main clause)

Corresponding to Example 1 two clauses are identified and they are given as input to phrase seperator. From each of those clauses, phrases are separated and labelled with phrase tags.

Input to phrase separator

Clause 1: നീ <RN>നന്നായി <Adv> പഠിച്ചാൽ <AdvC>

nI <RN> ***nannaayi*** <Adv> ***paThichchaal*** <AdvC>

Clause 2: **നിനക്ക്<DAT>** ഫസ്റ്റ്ക്ലാസ്**<OBJ> ലഭിക്കും <VERB>**

ninakk **<DAT>*****fast klaas*** <OBJ>***labhikkum***<VERB>

Steps of processing clause 1

Step1: form the POS tag string **<RN><Adv><AdvC>** from clause 1

Step 2: this POS pattern is checked with all entries in the self set. Since this POS tag string forms an invalid pattern no match is found and hence this phrase is rejected.

Step 3: Next the pattern <**RN><Adv>** is checked which is also rejected.

Step 4: Pattern <**RN** > is sent to the self set, match is found. As <RN> forms a valid noun phrase it is recognized by the self set and "accept" process is activated, i.e. **നീ** **(*nI*)** **<RN>** is a noun phrase and this word is identified as **<NP>.**

Step 5: **<Adv><AdvC>** pattern is then sent to the test, since it is a valid phrase, chunker marks **നന്നായി പഠിച്ചാൽ** (*nannaayi paThichchaal*) with phrase tag **<AdvpC>.**

Similarly, corresponding to clause 2, three phrases are obtained which are shown below.

നിനക്ക്<NP-Dat> ഫസ്റ്റ്ക്ലാസ്< NP-Obj> ലഭിക്കും<VP>

ninakk **<NP-Dat>** *fast klaas* **<NP-Obj>** *labhikkum* <VP>

Phrase chunker output for the above example is listed in table 7.2.

Table 7.2 Phrase Chunker output for Example 1

Clauses	POS Tag String	Detected String	Phrase Tag
1) നീ <RN> **നന്നായി** <Adv> **പഠിച്ചാൽ** (<AdvC>	RN Adv AdvC	RN	NP
(nI <RN> ***nannaayi*** <Adv> ***paThichchaal*** <AdvC>		Adv AdvC	AdvpC
		DAT	NP-Dat
2)**നിനക്ക്**<DAT> **ഫസ്റ്റ്ക്ലാസ്**<OBJ> **ലഭിക്കും** <VERB>	DAT OBJ VERB	OBJ	NP-Obj
(*ninakk* <DAT> *fast klaas* <OBJ> *labhikkum* <VP>		VERB	VP

2. പിറ്റേന്ന് എല്ലാവരും വെളുപ്പിനെഴുന്നേൽക്കണമെന്ന് അപ്പൂപ്പൻ ആജ്ഞാപിച്ചു

(*pitEnn ellaavarum veLuppinezhunnElkkaNamenn appUppan aajnjaapichchu*)

(Grand father instructed all of them to get up next day early in the morning)

POS Tagger produces following output for the above sentence.

പിറ്റേന്ന്<Adv>എല്ലാവരും<RN> വെളുപ്പിനെഴുന്നേൽക്കണമെന്ന്<VERB>

അപ്പൂപ്പൻ <NOUN>ആജ്ഞാപിച്ചു <VERB>

PitEnn<Adv> *ellaavarum*<RN> *veLuppinezhunnElkkaNamen~*<VERB> *appUppan* <NOUN> *aajnjaapichch u*<VERB>

Then the clause identifier separates the clauses as shown below.

Clause1:പിറ്റേന്ന് എല്ലാവരും വെളുപ്പിനെഴുന്നേൽക്കണമെന്ന്

(*PitEnn ellaavarum veLuppinezhunnElkkaNamenn*)

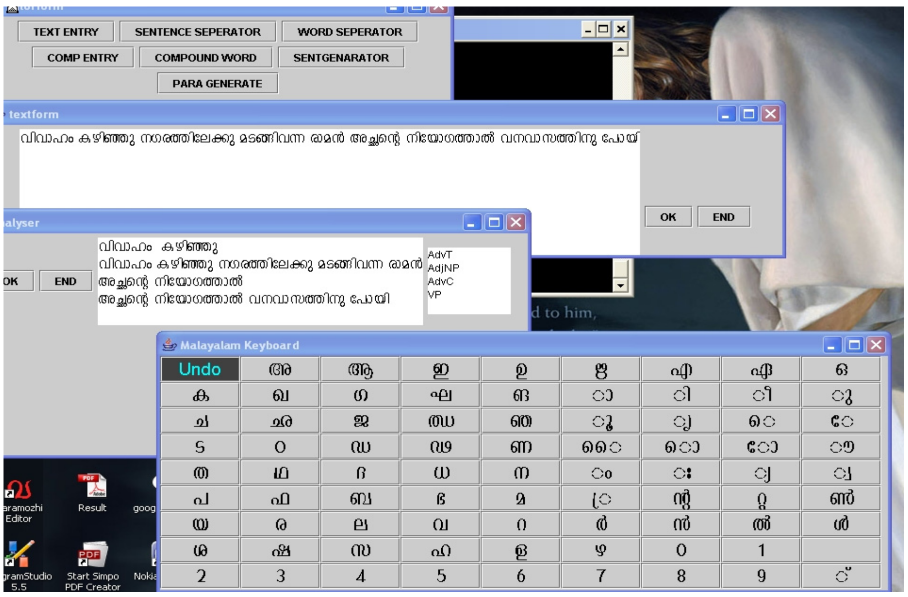
Clause 2: അപ്പൂപ്പൻ ആജ്ഞാപിച്ചു.

(*appUppan aajnjaapichchu)*

For the above two clauses phrase chunker produces the output as shown in table 7.3

Table 7.3 Phrase Chunker output for Example 2

Clauses	POS string	Detected String	Phrase Tag
1) പിറ്റേന്ന് <Adv>എല്ലാവരും<RN> വെളുപ്പിനെഴുന്നേൽക്കണമെന്ന്<VERB> (*PitEnn*<Adv> *ellaavarum*<noun-RN> *veLuppinezhunnElkkaNamenn*~<VERB>)	Adv RN VERB	Adv VERB RN	VP NP
2) അപ്പൂപ്പൻ<NOUN>ആജ്ഞാപിച്ചു <VERB> *appUppan* <NOUN> *aajnjaapichchu*<VERB>	NOUN VERB	NOUN VERB	NP VP



വിവാഹം കഴിഞ്ഞു നഗരത്തിലേക്കു മടങ്ങിവന്ന രാമൻ അച്ഛന്റെ നിയോഗത്താൽ വനവാസത്തിനു പോയി

വിവാഹം കഴിഞ്ഞു
വിവാഹം കഴിഞ്ഞു നഗരത്തിലേക്കു മടങ്ങിവന്ന രാമൻ
അച്ഛന്റെ നിയോഗത്താൽ
അച്ഛന്റെ നിയോഗത്താൽ വനവാസത്തിനു പോയി

AdvT
AdjNP
AdvC
VP

Fig 7.3 Screen shot of Phrase Chunker

Fig 7.3 describes the output of Phrase Chunker for the sentence

3. വിവാഹം കഴിഞ്ഞ് നഗരത്തിലേക്ക് മടങ്ങിവന്ന രാമൻ അച്ഛന്റെ നിയോഗത്താൽ വനവാസത്തിനു പോയി

(vivaaham kazhinj nagaththilEkk maTangivanna raaman achchhante niyOgaththaal vanavaasaththinu pOyi)

(Raman who returned to city after marriage, went to forest according to the decision of his father)

The phrases identified are listed below in table 7.4

Table 7.4 Final Output of Phrase Chunker

Phrases	Phrase Tag
1)വിവാഹം കഴിഞ്ഞ് *(vivaaham kazhinj)*	AdvT
2)വിവാഹം കഴിഞ്ഞ് നഗരത്തിലേക്ക് മടങ്ങിവന്ന രാമൻ *(vivaaham kazhinj nagaththilEkk maTangivanna raaman)*	AdjNP
3)അച്ഛന്റെ നിയോഗത്താൽ (*achchhante niyOgaththaal*)	AdvC
4)അച്ഛന്റെ നിയോഗത്താൽ വനവാസത്തിനു പോയി (*achchhante niyOgaththaal vanavaasaththinu pOyi)*	VP

7.3 Performance Evaluation

The performance of the phrase chunker was evaluated using the standardized techniques like precision, recall, and F-score where precision is defined as a ratio of number of correct chunks to the number of chunks in the output and recall is the ratio of number of correct chunks to the number of chunks in the test data.

Table 7.5 Performance Evaluation of the AIS-based Phrase Chunker

Chunk	Precision	Recall	F-Score
NP	93.5%	92.6%	93.0%
VP	96.7%	97.1%	96.9%
NP-Obj	92.0%	92.2%	92.1%
NP-Dat	96.4%	97.0%	96.7%
NP-Acc	94.3%	95.2%	94.7%
NP-Gen	91.0%	90.5%	90.7%
NP-Loc	87.9%	88.3%	88.1%
NP-Soc	95.0%	95.0%	95.0%
NP-Inst	92.8%	91.4%	92.1%

F-score = 2*recall*precision/ recall+ precision [161]. Precision, recall and F-score obtained for certain types of chunks are shown in table 7.5. Complete list is given in appendix F.

Documents related to five different fields were selected as the test corpus. All types of sentences were tested. Average precision is 91.3% and recall 90.6%. Out of 5647 phrase chunks tested, 4671 chunks gave the results correctly. 447 failure cases are identified where phrases were not in proper order and also overlapping phrases were present. To separate the phrases correctly, the word order within the phrase has to be corrected.

Table 7.6 Overall Performance of the Phrase Chunker

Predicted ↓	Actual		
	Pos	Neg	Total
Pos	4671	447	5118
Neg	482	47	529
Total	5153	494	5647

Average precision and recall can be obtained from the contingency table shown in table 7.6.

TP = 4671 FP = 447 FN = 482 TN = 47

Average Precision = 4671 / (4671+447) = 91.3 %

Average Recall = 4671 / (4671+482) = 90.6 %

This phrase chunker is less complex and has high speed due to the reduced number of computations. Without much difficulty, system can be updated just by updating the self/non self cell patterns.

7.4 Chapter Summary

Phrase chunking and its importance in a QA System are discussed. Various approaches to chunking, and methods adopted in different languages are described. A thorough investigation of Artificial Immune Systems is done and its application for phrase chunking is explained. A phrase tagset with 32 tags is also developed.

-

Chapter 8

NAMED ENTITY TAGGER

Contents

Named Entity Recognition and Classification is an important preprocessing step in all NLU applications. Named Entities are words or word sequences which usually cannot be found in common dictionaries and yet encapsulate important information that can be useful for the semantic interpretation of texts. Hence they are helpful in understanding natural language documents. Support Vector Machine-based NE Tagger for Malayalam language is discussed in this chapter.

A QA System becomes most effective when it understands the question given by the user and returns the appropriate answer from the collection of documents in the corpus. Answer selection is another important aspect that requires understanding of NL documents. Both of these issues demand the identification of semantic information which can be decided by the Named Entities present in the question or texts in the corpus.

Named Entities are words or word sequences which usually cannot be found in common dictionaries and yet encapsulate important information that can be useful for the semantic interpretation of texts. The term “named entity” now widely used in natural language processing was first introduced in 1995 by the sixth Message Understanding Conference (MUC-6) [162]. At that time MUC was focusing on the information retrieval tasks, where structured information of company and defence related activities were extracted from unstructured text such as newspaper articles. Then people noticed that references to the entities such as names including person names, location names, organization names and numerical expressions including date, time, amount etc. were essential to recognize information units for accurate information retrieval. Identifying these entities is one of the important tasks of IE and was called Named Entity Recognition and Classification. The NE task that was first introduced as a part of the MUC 6 (MUC 1995) evaluation exercise was continued in MUC 7(MUC 1998) [163]. This formulation of NE task defined 7 types of NE: PERSON, ORGANISATION, LOCATION, DATE, TIME, MONEY and PERCENTAGE.

NEs are theoretically identified and classified using phonological, morphological, semantic, and syntactic properties of linguistic forms that act as the targets of linguistic rules and operations. Two kinds of features are commonly used – internal and external. Internal features are provided from within the sequence of words that constitute the entity and external features are those that can be obtained by the context in which entities appear [164].

8.1 Related Work

Various techniques available for solving NER problems are statistical machine learning techniques, rule-based systems and hybrid approaches. Machine learning methods are using either supervised learning or unsupervised learning techniques. Statistical methods require large amount of manually annotated training data. A few commonly used statistical methods are Hidden

Markov Model, Maximum Entropy Markov Model, and Conditional Random Field. Sequence labelling problem can be solved very efficiently with the help of HMM. The conditional probabilistic characteristics of CRF and MEMM are very useful for the development of Named Entity Recognition (NER) systems. MEMM is having a label biasing problem. But all the machine learning techniques require large relevant corpuses which is unavailable in Malayalam. Machine learning methods are cost effective and do not need much of language expertise. In [165] authors describe an NER system using CRF. This system uses different contextual information of the words along with both language independent and language dependant features. GuoDong Zhou and Jian Su [166] described a HMM based on the mutual information independence assumption where they claimed that their system reached 'near human' performance. NER system based on MEMM is presented in [167].

Grammar-based techniques are used for creating NER systems that obtain better precision but at the cost of lower recall and months of work by experienced computational linguistics [168]. Rule-based approaches lack the ability of coping with the problems of robustness and portability. Each new source of text requires significant tweaking of rules to maintain optional performance and the maintenance cost is quite high. Rule-based systems perform the best especially for specialized applications. [169] introduces a rule-based system that used handcrafted rules and this approach gave them better performance than the CRF method.

Hybrid methods either use combinations of different machine learning methods or combinations of rule-based and machine learning methods. In [170] authors present an NE recognition tool for Portuguese. It has two components– rule-based components for recognition of number expressions and hybrid component for names. Lot of work has been reported in the field of NER for English and European languages.

Research indicates that NER systems developed for one domain do not typically perform well on another domain.

Asif Ekbal et.al [171] reports the development of a Named Entity Recognition (NER) system for Bengali using the statistical Conditional Random Fields (CRFs). The system makes use of the different contextual information of the words along with the variety of features that are helpful in predicting the various named entity classes. A portion of the partially NE tagged Bengali news corpus, developed from the archive of a leading Bengali newspaper available in the web, and has been used to develop the system. The training set consists of 150K words and has been manually annotated with a NE tagset of seventeen tags. Experimental results of average Recall, Precision and F-Score values of 93.8%, 87.8% and 90.7%, respectively.

Bengali NER system in [172] used SVM approach for its development. Training set contained 130,000 words with 16 NE tags using BIE model for PERSON, LOC, and ORG entities. This model includes gazetteers for names of persons, locations, organization, and miscellaneous. The evaluation reported a goo F-Measure of 91.8%.

Hasan et.al (2009) [173] presented a learning-based Named Entity Recognizer for Bengali that donot rely on manually constructed gazetteers in which they developed two architectures for the NER system. The corpus consisting of 77942 words is tagged with one of 26 tags in the tagset defined by IIT Hyderabad where they used CRF++ to train the POS tagging model. Evaluation results shows that the recognizer achieved an improvement of 7.5% in F-measure over a baseline recognizer.

B. B. Chaudhuri and S. Bhattacharya [174] have proposed a three-stage approach of named- entity detection. The stages are based on the use of Named-Entity (NE) dictionary, rules for named-entity and left-right co- occurrence statistics. Corpus of Anandabazar Patrika has been used from the year 2001-2004. Experimental results have shown the average recall, precision and f-measure to be 85.50%, 94.24% and 89.51%.

Vijayakrishna and Sobha L [175] developed a domain focused Tamil NER for tourism using CRF. The tag set contained106 tags. Morphological analysis, POS tagging, NP Chunking and NE annotation are done manually on the corpus. This corpus is divided into training data (80%) and test data (20%). CRF model is then trained with the training data. A total of 4059 NEs are tested and obtained an overall F-Measure of 80.44%.

S Lakshmana Pandian [176] describes a hybrid 3-stage NER system for Tamil. There are 3 phases namely shallow parsing, shallow semantic parsing, and statistical processing. Statistical processing uses E-M algorithm which takes inputs from both shallow parsing and semantic parsing stages. Average F-Measure values obtained is 72.72%. This system uses both linguistic and statistical methods.

[177] describes NER system for Hindi using CRF approach. The training set has been manually annotated with a NE tag set of 12 tags. The performance of the system has shown improvements by using the Part-of-Speech information of the current and surrounding words, name list, location name list, organization list, person prefix gazetteers list etc. It has been observed that using prefix and suffix feature helped a lot in improving the results. This system has achieved Precision, Recall and F-score of 72.78%, 65.82% and 70.45% respectively. CRF++ toolkit was used for training and testing data.

Saha et.al [178] describes the development of Hindi NER using MEMM approach. The training data consisted about 234K words collected from the newspaper “Dainik Jagaran”. It was evaluated against a blind test corpus of 25K words having four classes –PERSON, ORGANISATION, LOCATION, and DATE. This system is able to achieve an f-value of 81.52%, using a hybrid set of features including traditional NER features augmented with gazetteer lists and extracted context patterns.

Amit Goyal describes [179] a NER system for Hindi language using CRF approach which also identifies nested entities. This method was evaluated on test set 1 and test set 2 and attains a maximal F1 measure around 49.2 and nested F1 measure around 50.1 for test-set 1; maximal F1 measure around 44.97 and nested F1 measure 43.70 around for test-set 2 and F-measure of 58.85% on development set.

Authors of [180] describes the application of Conditional Random Fields (CRFs) with feature induction to a Hindi named entity recognition task. They discover relevant features by providing a large array of lexical test and using feature induction to construct the features that increases the conditional likelihood. Combination of Gaussian prior and early-stopping based on the results of 10-fold cross validation is used to reduce over fitting.

Gupta and Arora (2009) [181] describes the observation made from the experiment conducted on CRF model for developing Hindi NER. It shows some features which makes the development of NER system complex. It also describes the different approaches for NER. The data used for the training of the model was taken from Tourism domain and it is manually tagged in IOB format.

Biswas et.al [182] presented a hybrid system for Oriya NER that applies both ME and HMM and some handcrafted rules to recognize NEs. Firstly the ME model is used to identify the named entities from the corpus and then this tagged corpus is regarded as training data for HMM which is used for final tagging. The annotated data used in the system is in IOB format. The F-Measure obtained was between 75% and 90%.

[183] explains a Telugu NER system by using MEMM approach. The corpus was collected from various newspapers. The system makes use of the different contextual information of the words and gazetteer list was also prepared manually or semi-automatically from the corpus and came out with an F-Measure of 72.07% for PERSON, 60.76%, 68.40%, and 45.28% for ORG, LOC, and others respectively.

A CRF-based NER system is explained in [184]. This describes the working of a noun tagger. This system was trained with a manually tagged data of 13425 words and tested on a data set of 6223 words. F-Measure obtained in various experiments is between 80% and 97%. This work is only for single word NEs. No POS Tagger or Chunker information is used for this work.

In [185] authors present the experiments conducted as a part of NER for South and South East Asian Languages (NERSSEAL) Competition with various feature combinations for Telugu NER. It is also observed that the prefix and suffix information helps a lot in finding the class of the token. The best performing model gave an $F\beta=1$ measure of 44.91.

Authors of [186] describe a handcrafted rule-based NER for Assamese. A corpus of about 50000 words of Assamese online Pratidin article was first manually tagged. This tagged corpus is then analyzed to form the rules.

8.2 Difficulties in Finding Named Entities in Malayalam Language

NER in Malayalam language is a difficult task since it is not case sensitive as English and other European languages. Owing to its agglutinative nature most of the words are compound words and hence NE of these words cannot be determined directly. The world of NE is "open" since new Person, Location, and Organization names are coming up regularly. It is impossible to add all these NEs to a dictionary and it is a time consuming task. Another problem is that the same word can be grouped into several NE types.

8.3 Methodology- Support Vector Machines

Support Vector Machine proposed by Vapnik is a set of machine learning algorithms based on statistical methods. It is known as one of the best supervised learning algorithms and has been successfully applied to natural language tasks such as text categorization [187], phrase chunking [188], POS Tagging [189], data classification [190] etc. This algorithm analyses data and recognizes patterns used for statistical classification and regression analysis [191].

Assigning a Named Entity label to a word in the sentence can be taken as a classification problem. SVM is normally used for solving binary classification problems but can be extended to solve multiclass problems such as Named Entity Classification (NEC). Linguistic features are mapped to SVM feature vectors and setting these features to 1 if it exists or else 0 otherwise. These feature vectors are used to decide whether a word is an NE or not, and the specific class of NE to which the word belongs.

Multiclass SVM

Multiclass SVMs are usually implemented by combining several two-class SVMs. Popular methods for doing this are: one-versus-all method using Winner-Takes-All (WTA) strategy, one-versus-one method implemented by Max-Wins Voting (MWV), Directed Acyclic Graph-SVM (DAG-SVM), and error-correcting codes [192].

For a given multiclass problem, M will denote the number of classes and ω_i, $i = 1 \ldots M$ will denote the M classes. For binary classification the two classes are referred as *positive* and *negative.* A binary classifier will be assumed to produce an output function that gives relatively large values for examples from the positive class and relatively small values for examples belonging to the negative class.

WTA SVM

WTA SVM constructs M binary classifiers. The i^{th} classifier output function ρ_i is trained, taking the examples from ω_i as positive and the examples from all other classes as negative. For a new example **x**, WTA SVM strategy assigns it to the class with the largest value of ρ_i.

MWV SVM

This method constructs one binary classifier for every pair of distinct classes and so, all together $M(M-1)/2$ binary classifiers are constructed. The binary classifier C_{ij} is trained in taking the examples from ω_i as positive and the examples from ω_j as negative. For a new example **x**, if classifier C_{ij} says **x** is in class ω_i, then the vote for class ω_i is added by one. Otherwise, the vote for class ω_j is increased by one. After each of the $M(M-1)/2$ binary classifiers makes its vote, MWV strategy assigns **x** to the class with the largest number of votes.

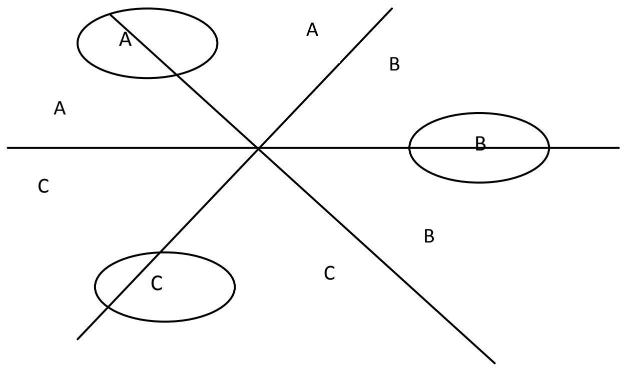


Fig 8.1 Diagram of Pairwise SVM Decision boundaries on a basic Problem

For example, in Fig 8.1 there are three classes A, B, and C. Using MWV strategy, 3*(3-1)/2= 3 binary classifiers are created namely $C_{AB,}$ $C_{BC,}$ and $C_{CA.}$ C_{AB} classifies word in the test data as of class A or class B by increasing the count corresponding to A or B. C_{BC} and C_{CA} repeats the same process and increases the count corresponding A or B or C. Repeating this strategy, each word in the test data is assigned a class label either as A or B or C.

8.4 Malayalam NE Tagger

This tagger takes a document marked with POS and phrase tags, and produces an NE tagged document. Block diagram of Malayalam NE Tagger is given in Fig 8.2. There are mainly four modules, NE marker, NE identifier, NE classifier, and NE Disambiguator. NE Marker determines whether a token is a Named Entity or not. NE identifier categorizes an entity as sole-entity, constituent-entity, or dependant-entity. If a particular token is an entity, then it is classified as one of the 26 predefined categories. Sometimes it might not be possible to tag a token with a single NE tag, and then NE Disambiguator solves the issue using the Gazetteer lists and assigns the token an NE tag.

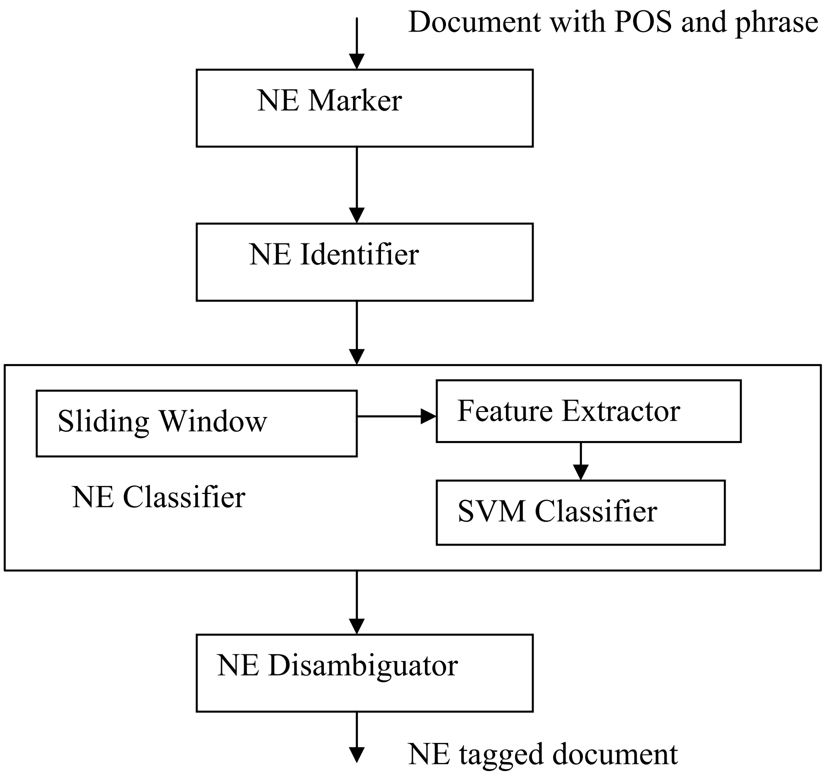


Fig 8.2 Block Diagram of NE Tagger

8.4.1 NE Marker

The function of this module is to classify the incoming tokens into an NE or NOT-AN-NE (NAN). For this purpose, it checks the POS tag of each token in the input document. Words which have POS other than ‘VERB’, ‘AuxV’, and postpositions are decided as named entities and label them with ‘NE’ tag while all other tokens are assigned a ‘NAN’ tag.

8.4.2 NE Identifier

Once an ‘NE’ tag is assigned to a token, next step is to determine its type whether it is sole entity, constituent entity, or dependant entity. A word which can act as an entity irrespective of its surroundings is called a sole entity. A word which cannot stand as a sole NE can be combined with other words to form an NE. Such words are constituent entities. Owing to the presence of certain surrounding words, a word can become an entity which is called dependant entity.

Table 8.1 Named Entity Tagset

NE Tagset	
DISEASE	MEDIUM
MEDICINE	FUNCTION
TREATMENT	REST
ORGANISATION	DESCRIPTION
PERSON	SYMPTOM
LOCATION	TIME
VIRUS	DATE
PRECAUTION	COLOR
FOOD	TEST
DEFINITION	OBJECT
REASON	JOURNAL
ELEMENT	COUNTRY
AGE	AGENT

8.4.3 NE Classifier

The NE Classifier determines the NE classes of all entity tokens in the document. An NE tagset is designed for this work and it contains 26 NE tags which are listed in Table 8.1.

To classify an entity into one of these 26 tags, SVM is used. During the training phase, features are extracted from the training data and classifiers are designed for each named entity. For classification, features are extracted from the current document and are encoded as vectors. If the entity is a sole entity internal or external features are not required for classification since the entity itself determines its NE. In the case of a constituent entity, internal features (internal to the word) are used for the classification while a dependant entity uses external features or surrounding features. The features used for NER and classification are given below.

For a constituent entity internal features are extracted to form a feature vector. Dependant entity uses a sliding window and feature extractor to obtain the external features. Once the feature vector of the input token is ready, WTA strategy of Multiclass SVM can be used to determine the NE label.

Features used for Named Entity Identification and Classification

In this system clues present as inner word (internal feature) and context word (external features) are used for NE identification and classification. Some of these features are Language Independent Features while others are Language Dependant Features [193].

Language Independent Feature

Word Prefix/Suffix

Prefix/Suffix information of a word and its surrounding words are useful for highly inflected language like Malayalam. Word suffix information is useful to identify the named entities. Variable length suffixes can be matched against the list of suffixes for different classes of NEs. A list of linguistic suffixes (verbs, adjectives, adverbs, nouns) is prepared which helps to recognize 'Not a Named Entity' cases. Certain suffixes are helpful in detecting Named Entities such as PERSON, LOCATION etc.

Digit Information

This gives word level orthographic information. If a word contains digits or special symbols, corresponding feature is set to 1; otherwise 0. Three features are considered depending upon the presence of digits and/or the number of tokens, combination of digits and symbols. These features are helpful in recognizing DATE expressions, TIME expressions, AGE expressions etc.

Length of the Word

If the length of the word is greater than two, the feature 'Length' is set to 1; otherwise it is set to 0. Named Entities, since they are nouns/noun phrases or open class entities they are rarely shorter words.

Word position

There are two features corresponding to word position. The feature 'First Word' is set to 1 if the current word is the first word of the sentence; or else set to 0 and the feature 'Last Word 'is set to 1 if the given word is the last word of the sentence.

Frequent Word List

According to Luhn, words with very high frequency and very low frequency are not sense carrying agents, they are either rare words or closed class words.

Surrounding Words

Previous and subsequent words of a particular word are used as features. This feature is multivalued. Different window sizes were used for different domains in different experiments.

NE Information

The NE tags of the previous words are used as a dynamic feature.

Language Dependant Feature

Parts of Speech Information

POS of the current word and the surrounding words are important to recognize NEs. POS also indicates whether the word is a 'stand alone word' or part of a phrase.

Phrase Chunk Information

Certain NEs are noun phrases which appear either directly or as a part of other phrases such as postpositional phrases.

8.4.4 NE Disambiguator

Gazetteer lists are created for the names of diseases, persons and locations. In case NE Classifier cannot decide the final NE tag of the token, decision is made by this module.

8.5 Performance Evaluation

Its performance is evaluated using standardized techniques precision, recall and F-score where precision is defined as a ratio of number of correct NE tags in the output to the number of NE tags in the output and recall is the ratio of number of correct NE tags in the output to the number of NE tags in the test

data. Fscore = 2*recall*precision/ (recall+ precision) [194]. Documents in the corpus were selected from the medical domain. Then 4000 sentences were randomly selected for training and 700 sentences as test set. Precision, recall, and F-score obtained for certain NE tags are shown in Table 8.2. Highest recall is 96.21% for the entity LOCATION and highest precision for the entity JOURNAL. Highest F-Score is obtained for the named entity JOURNAL.

Table 8.2 NER Performance by Named Entity Type

Named Entity	Recall	Precision	F-Measure
PERSON	95.32	94.28	94.78
ORGANISATION	89.25	88.46	88.85
LOCATION	96.21	95.30	95.75
DISEASE	88.41	89.53	88.97
SYMPTOM	86.38	84.14	85.25
REASON	82.18	83.60	82.88
AGE	87.64	89.43	88.52
TIME	90.40	92.52	91.44
DATE	92.67	91.45	92.05
DESCRIPTION	91.20	90.16	90.68
VIRUS	95.47	94.50	94.98
JOURNAL	96.00	97.23	96.61
MEDICINE	92.87	94.75	93.80
MEDIUM	74.49	76.28	75.37
PRECAUTION	78.34	75.65	76.97
TREATMENT	88.00	86.50	87.24
FOOD	87.04	92.56	89.72
DEFINITION	87.00	91.03	88.97
ELEMENT	88.24	89.21	88.77
FUNCTION	84.52	85.31	84.91
REST	87.64	91.45	89.50
COLOR	79.28	82.71	80.96
TEST	81.65	82.27	81.46
OBJECT	78.43	81.10	79.74
COUNTRY	79.36	85.42	82.28
AGENT	80.14	81.16	80.65
Average	86.90	87.90	87.70

Table 8.3 Overall Performance of the Named Entity Tagger

Predicted ↓	Actual		
	Pos	Neg	Total
Pos	743	102	845
Neg	113	53	166
Total	856	155	1011

The contingency table given above shows the overall performance of Named Entity Tagger.

TP = 743 FN= 113 FP = 102 TN= 166

Average Precision = 743 / (743+102) = 87.9 %

Average Recall = 743 / (743+113) = 86.9 %

Table 8.4 gives a few examples of Named Entities identified and corresponding NE tags assigned. Underlined word/phrase/sentence is the Named Entity identified.

8.6 Chapter Summary

NER plays an important role in the design of an accurate answer retrieval and efficient MT system. The objective of NER is to categorize all NE words in a document into predefined classes like PERSON, LOCATION, DISEASE etc.

Twenty six Named Entity tags were identified for a closed domain QA System in the medical domain. An SVM based NER system is designed and its details are discussed in this chapter. Results show that Multiclass SVM is a promising method for Malayalam NE classification and recognition. Approach presented here requires linguistic preprocessing of the document text such as morphological analysis, POS Tagging and phrase chunking.

Table 8.4 NE Tag Examples

NE TAG	Examples and the Named Entity
TREATMENT	പൂർണ്ണവിശ്രമവും ഉപ്പും കൊഴുപ്പും കുറഞ്ഞ ഗ്ലൂക്കോസ് കൂടുതലുള്ള ആഹാരവും ആണ് ഈ രോഗം കഠിനമല്ലാത്തവർക്കുള്ള മുഖ്യചികിൽസാ നിർദ്ദേശങ്ങൾ *(pURNNa viSramavum uppum kozhuppum kuRanja glukkOs kUTuthaluLLa aahaaravum aaN~ ii rOgam kaThinamallaaththavaRakkuaLLar mukhyachikilsaa nirddESangaL)*
SYMPTOM	ചർദ്ദി ഓക്കാനം വയറെരിച്ചിൽ എന്നീ രോഗലക്ഷണങ്ങൾ ഉണ്ടാകാം (*chhaRddi oaakkaanam vayaRerichchil ennI rOgalakSHaNangaL unTaakaam)*
MEDICINE	ഇന്റർഫെറോൺ എന്ന മരുന്ന് *(intaRpheRON enna marunn*)
REST	മഞ്ഞപിത്തം തീർച്ചപ്പെടുത്തിയാൽ മൂന്നു നാല് ആഴ്ച വിശ്രമമെടുക്കണം. (*manjappiththam thIRchchappeTuththiyaal mUnn naal~ aazhcha viSRamameTukkaNam)*
AGE	ഏതു പ്രായത്തിലുള്ളവരെയും ഹെപ്പറ്റൈറ്റിസ്എ മഞ്ഞപിത്തം ബാധിക്കാമെങ്കിലും കുട്ടികളിലാണ് കൂടുതൽ രോഗസാദ്ധ്യത. (*aethu praayaththiluLLavareyum heppataitis A manjappiththam baadhikkaamenkilum kuTTikaLilaaN~ kUTuthal rOgasaaddhyatha)*
ELEMENT	രക്തത്തിൽ ബെൽറൂബിൻ അതികരിക്കുന്നതു മൂലമാണ് ശരീരത്തിന് മഞ്ഞനിറം ഉണ്ടാകുന്നത *(rakthaththil belRUbin athikarikkunnathu mUlamaaN~ SarIraththin~ manjaniRam unTaakunnath~)*
DEFN	മഞ്ഞപിത്തം കരളിനെ ബാധിക്കുന്ന രോഗമാണ് *(manjappiththam karaLine baadhikkunna rOgamaAN~)*
TEST	എച്ച്ബിഎസ്എടി (*echch~ bi es~ e Ti)*

Chapter 9

PERFORMANCE EVALUATION

Contents

In this chapter various performance evaluation techniques that can be used for QA Systems are discussed. The experimental details and various methods of MaQAS evaluation are also described.

In Information Retrieval systems, user query request is inherently vague and the retrieved documents are not exact answers and hence they have to be ranked according to their relevance to the query. Such relevance ranking introduces a component which plays central role in IR that determines how precise the answer set is. This type of evaluation is referred to as retrieval performance evaluation [195]. Such an evaluation is based on a test reference collection and on an evaluation measure. The test reference collection consists of a collection of documents, a set of sample information requests, and a set of relevant documents for each sample information request. Given a retrieval strategy S, the evaluation measure quantifies the similarity between the set of documents retrieved by S and the set of relevant documents.

When considering the performance evaluation of IR systems, we should first consider the retrieval task that is to be evaluated. For instance, the retrieval task could simply consist of a query processed in batch mode or of a whole interactive session. Batch and interactive query tasks are quite distinct processes and thus their evaluations are also distinct. Besides the nature of the query request one has to consider the environment where the evaluation will take place and the type of interface used. Evaluation of experiment performed in a laboratory might be quite distinct from evaluation of experiments carried out in real life situation.

9.1 General Methods of Evaluation

From previous chapters it is evident that the QA Systems are not just made up of a single component but a series of components working together to achieve the final goal. Though the final goal of a QA System is to obtain a correct answer to the question asked, each of the individual components within the system has their own goals which eventually lead to the final goal. Thus the performances of the individual components are likely to influence the entire QA task.

Black-box Evaluation Approach –Here the performance of the system is considered as whole without caring much about the performance of individual components [196]. Thus the final answer from the system is compared with the question asked to evaluate the QA System. A correct answer is what is expected from a QA System but there could be more than one correct answer for a given question. Thus the QA System needs to find out all possible final answers.

Glass-box Evaluation Approach – Here each of the individual components of QA Systems are evaluated with appropriate methods particular to that component [196]. The goal is to have optimal performances for each individual component which will ultimately lead to a better overall performance of the QA System.

Interactive Setting - Evaluation Scenario (setting) determines the appropriate evaluation metrics. An example task for interactive setting is finding out the answers for a questionnaire. The performance of the system is measured in terms of the average number of questions attempted, the average number of questions answered correctly, and the time taken to fill out the questionnaire.

Non-Interactive Setting - In a batch (or non-interactive) setting, the ability of the system is measured by finding the relevant documents corresponding to a query and ranking them accordingly [197].

9.2 Evaluation Metrics

To accurately measure the performance of a QA System we need matrices that can provide good indication on how the system would perform in real world scenario. In QA it is important to retrieve documents that contain the answer or part of the answer that will satisfy the user question. There are a number of evaluation measures that can be used to compare the performance of the various retrieval techniques. Each measure highlights a different aspect and use of several measures to describe the performance of a system is more revealing than using a single measure. On the other hand, when comparing systems, it is often more convenient to use a single measure and the choice depends on the purpose of the retrieval system and the context in which it is used.

Precision and Recall

These are the most commonly used indicators to measure IR retrieval quality. Precision of the system is the fraction of retrieved material that is actually relevant. Recall is defined as the fraction of relevant material actually retrieved in answer to a search request [198]. Precision can be seen as a measure of exactness or quality whereas recall is a measure of completeness or quantity. High recall means that a system returned most of the relevant results. High precision means that the system returned more relevant results than irrelevant. Precision and recall are depicted in fig 9.1.

Consider a sample information request I. Given that $|R_r|$ be the number of relevant documents retrieved, $|R_j|$ be the number of irrelevant documents retrieved and $|N_r|$ be the number of relevant documents not retrieved, then the precision and recall are given by the following equations.

Precision= $|R_r|/ (|R_r|+|R_j|) = A \cap B / B$

Recall= $|R_r|/ (|R_r|+|N_r|) = A \cap B / A$

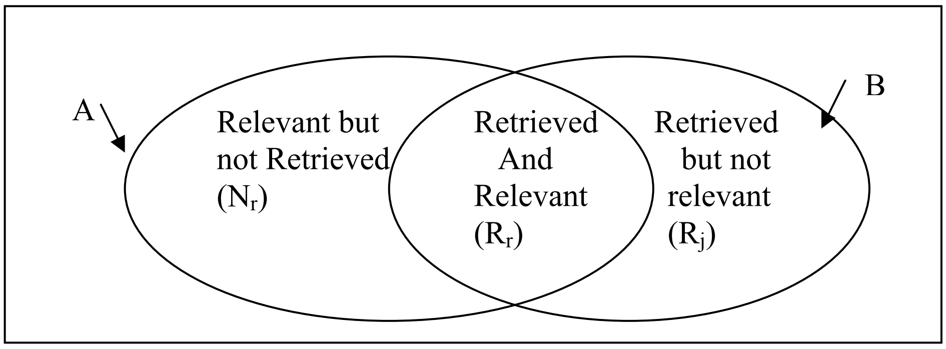

A
B
Relevant but not Retrieved (N_r)
Retrieved And Relevant (R_r)
Retrieved but not relevant (R_j)

Fig 9.1 Set Diagram Showing Elements of Precision and Recall

Precision and recall can also be defined by a contingency table. This table is normally used in classification tasks. A typical contingency table used in IR systems is shown below.

Table 9.1 Contingency Table

Total Documents(D)	Relevant (A)	Non-Relevant (A′)
Retrieved (Predicted +ve) (B)	TP = $A \cap B$ (true +ves)	FP= $A' \cap B$ (false +ves)
Not Retrieved (Predicted -ve) (B′)	FN= $A \cap B'$ (false -ves)	TN= $A' \cap B'$ (true –ves)

TP	=	Number of true positives
FP	=	Number of false positives
TN	=	Number of true negatives
FN	=	Number of false negatives

True positives are documents deemed relevant by both the human expert and the information retrieval system. False positives are returned by the IR system, but were reckoned irrelevant to the query by the human expert. False negatives are documents relevant to the query which are not found by the system. True negatives are not returned by the system and are considered irrelevant by the human expert.

Accuracy	= (TP+TN)/ (TP+FP+TN+FN)
Precision	= TP / (TP+FP)
Recall	= TP / (TP+FN)

F-Measure

F-Measure or F-Score can be defined as harmonic mean of precision and recall. It is a measure of system's accuracy. It considers both the precision P and recall R to compute the score.

F-Measure F = 2* P*R / (P + R)

F-Measure reaches its best value at 1 and worst score at 0. It is 0 when no relevant documents have been retrieved and is 1 when all ranked documents are relevant. The harmonic mean F assumes a high value only when both recall and precision are high [69].

9.3 MaQAS –Implementation

MaQAS accepts questions in simple sentences, analyses them, and returns answers in a single word, phrase, or sentence. The platform used for the implementation of MaQAS is given below.

Platform Used

An Intel Core2 Duo CPU 2.00 GHz with 2GB RAM and 150GB Hard disk was used. Operating System was Windows XP.

Database: My SQL Version 4.1.14

Language: J2SDK 1.4.2

Testing Environment

The corpus was prepared by collecting documents from medical books, journals, and health magazines like "Arogyam" (health). Test questions were gathered from various users around us. This system was tested with hundreds of different types of questions. A sample document and a set of sample questions are given below.

Sample Document

മനുഷ്യശരീരത്തിൽ ഉദരത്തിന്റെ വലതുഭാഗത്ത് വാരിയെല്ലുകൾക്ക് തൊട്ടു താഴെയാണ് കരൾ സ്ഥിതിചെയ്യുന്നത്. പ്രധാനമായും ഹെപ്പറ്റൈറ്റിസ് എ വൈറസുകളാണ് പടർന്നുപിടിക്കുന്ന മഞ്ഞപിത്തത്തിനു കാരണം. പ്രധാനമായും മലിനജലത്തിലൂടെ പകരുന്ന ഹെപ്പറ്റൈറ്റിസ് എ അവികസിത രാജ്യങ്ങളിലാണ് കൂടുതലായി കാണപ്പെടുന്നത്. ഏതു പ്രായത്തിലുള്ളവരെയും ഹെപ്പറ്റൈറ്റിസ് എ മഞ്ഞപ്പിത്തം ബാധിക്കാമെങ്കിലും കുട്ടികളിലാണ് കൂടുതൽ രോഗസാദ്ധ്യത. പകർച്ചവ്യാധിയായ മഞ്ഞപിത്തം വർഷത്തിലെല്ലാക്കാലവും പടർ ന്നുപിടിക്കാമെങ്കിലും മഴക്കാലത്താണു രോഗം കൂടുതൽ വ്യാപകമാകുന്നത്. മലിനജലം ഉപയോഗിച്ച് കഴുകിയ പഴവർഗ്ഗങ്ങൾ, പച്ചക്കറികൾ ഇവ ഉപയോഗിക്കുന്നതിലൂടെയും രോഗം പകരാം.

Transliterated version of above document is given below.

manushyaSrareeraththil udaraththinte valathubhaagathth~ vaariyellukaLkk~ thottu thaazheyaaN~ karaL sthhithicheyyuth~. pradhaanamaayum heppataitis~ e vaiRasukaLaaN~ paTarnnupiTikkunna manjnjapiththaththinu kaaraNam. pradhaanamaayum malinajalaththilooTe pakarunna heppataitis~ e avikasitha raajyangngaLilaaN~ kooTuthalaayi kaaNappeTunnath~ Ethu praayaththiluLLavareyum heppataitis~ e manjnjappiththam baadhikkaamengkilum kuTTikaLilaaN~ kooTuthal rOgasaaddhyatha pakarchchavyaadhiyaaya manjnjapiththam varshaththilellaakkaalavum paTar nnupiTikkaamengkilum mazhakkaalaththaaNu rOgam kooTuthal vyaapakamaakunnath~. malinajalam upayOgichch~ kazhukiya pazhavarggangngaL, pachchakkaRikaL iva upayOgikkunnathilooTeyum rOgam pakaraam.

Sample Questions

a. എന്താണ് മഞ്ഞപ്പിത്തം? (*enthaaN~ manjnjappiththam*?) (what is Jauntice?)

b. മഞ്ഞപ്പിത്തം എന്നാൽ എന്താണ്? (*manjnjappiththam ennaal enthaaN~?)* (what is meant by Jauntice?)

c. എങ്ങനെ ചികിത്സിക്കണം? (*engngane chikithsikkaNam*?) (how to treat?)

d. എന്താണ് മരുന്ന്? (*enthaaN~ marunn~*?) (what is the medicine?)

e. എന്തു ചികിത്സയാണ് മഞ്ഞപ്പിത്തത്തിനുള്ളത്? (*enthu chikithsayaaN~ manjnjappiththaththinuLLath~ ?)*(what is the treatment for Jauntice?)

An example of a document available in the corpus and types of questions used for testing the system are listed in Appendix G and H respectively.

MaQAS performance was evaluated employing various measures and its details are vividly portrayed below.

9.4 Performance Evaluation of MaQAS

Evaluation criteria do a major job in the completeness of performance analysis. The question set was first divided into ten different categories based on the question type and then into 26 categories based on the expected NE type. These questions were then fed into the QAS one by one and the retrieved answers were analysed. MaQAS was evaluated using the measures precision, recall, and F-Measure. Table 9.2 shows the test results of MaQAS for a particular run. It gives the following information.

Total number of questions asked	= 200
Number of answers present in the corpus	= 171 (147+24)
Number of questions correctly answered	= 147
Number of questions wrongly answered	= 19
Number of answers present in the corpus but not retrieved	= 24
Answers which were not relevant	= 10

Table 9.2 Contingency Table Showing MaQAS output

Number of Answers	Relevant	Not-relevant
Retrieved	147	19
Not retrieved	24	10

True Positives (TP)	= 147
False Positives (FP)	= 19
True Negatives (TN)	= 10
False Negatives (FN)	= 24

Precision (P) = TP / (TP+FP) =147/ (147+19) = 88.5%

Recall (R) = TP / (TP+FN) =147/ (147+24) = 85.9%

F-Measure = 2*P*R/ (P+R) = 2*88.5*85.9/ (88.5+85.9) = 87.2%

9.5 Analysis and Discussion of Results

Performance of MaQAS was evaluated by measuring its ability to retrieve all and only relevant information. MaQAS performance is strongly dependant on NE Tagging and correct processing of the queries. The system achieved an overall precision of 88.5% and 85.9% of recall. There is a tradeoff between precision (P) and recall(R). Higher the value of P lower will be the value of recall and vice versa. The F-Measure is the harmonic mean of P and R.

Referring to Table 9.2 it is clear that for 200 questions asked, 166 answers were retrieved out of which 147 answers were relevant to the query and 19 were non-relevant. Even though 24 other relevant answers existed in the corpus they were not extracted because some of the question patterns were not recognized properly or semantics was not sufficient enough to identify the required NE. In this work, retrieval scheme is purely based on named entities. Hence all the sentences with the candidate entities are retrieved. But all these entities might not be the answers to the question.

Table 9.3 Performance of MaQAS

Questions	Relevant Answer	Retrieved Answers	Correct Output	Recall	Precision	F-Measure
200	173	166	147	85.9	88.5	87.2

The overall performance of MaQAS is given in Table 9.3. Precision of 88.5% shows that the answers retrieved are correct answers and only very few non-relevant answers are retrieved. Percentage of recall is less than precision and is 85.9%. This can certainly be improved by providing a WordNet facility. While raising the questions no restriction is kept to avoid any bias that may affect the system performance. Question patterns were prepared just considering the question keyword. No considerations were given to other words which are present in the question as different users might use different combinations of words or phrases. It is not possible to consider all word combinations that can occur in a query and also is not an efficient practice. If a WordNet is present, it can be used during the identification of secondary keywords of question analysis phase which will certainly increase the recall value.

Table 9.4 describes the performance of MaQAS based on question type. Factoid and certain non-factoid questions were only considered in this work. Yes/No questions are not considered in the design of MaQAS and hence still it remains as a research topic. Poor performances of other question types are mainly due to the disability of correct identifications of NE from the question analysis phase. Handling of 'HOW' and 'WHY' type questions are the most difficult because they mostly require answers spreading over more than one sentence or paragraph. These questions sometimes require deep semantic processing of the sentences and identification of more keywords to detect the presence of explanations, intentions, justifications etc.

Table 9.4 Performance According to Question Type

Question type	No.of questions	Retrieved Answers	Correct Answers	Wrong Answers	Relevant Not-Retrieved	Precision	Recall	F-Measure
Who	5	4	3	1	0	75	100	85.7
Where	11	6	5	1	2	83.3	71.4	76.9
What	48	44	42	2	2	95.5	95.5	95.5
Why	6	4	3	1	2	75.0	60.0	66.7
Which	53	49	46	3	4	93.9	92.0	92.0
When	14	12	10	2	2	83.3	83.3	83.3
How	26	23	18	5	5	78.3	78.3	78.3
How	9	4	4	0	1	100	80.0	88.9
much	18	15	13	2	4	86.7	76.5	81.3
Others	10	5	3	2	2	60.0	60.0	60.0
Yes/No								
Total	200	166	147	19	24			

Performance based on expected answer type is listed in Table 9.5. No answer is retrieved for 'ORGANISATION' type questions. This is because no organization NE is present in the index. Though this entity is not present in the documents presently tagged, it might be required in future. Named Entity 'ORGANISATION' is considered in the tag list since it is noticed during the question analysis phase that many user questions had this. At the same time 100% accuracy is obtained for 'REST' type questions. The failures in NE Tagging, question analysis phase and unavailability of appropriate answers in the corpus are the reasons behind the reduced performance rate in other question types.

Table 9.5 Performance According to Answer Type

Answer Type	No. of questions	Retrieved Answers	Correct Answer	Wrong Answer	Relevant Not-Retrieved
DISEASE	6	4	4	1	0
MEDICINE	8	6	6	0	1
TREATMENT	12	10	9	1	1
ORGANISATION	4	0	0	0	0
PERSON	6	5	4	1	0
LOCATION	11	9	7	2	2
VIRUS	7	6	5	1	1
PRECAUTION	14	12	11	1	2
FOOD	11	11	10	1	1
DEFINITION	7	6	6	0	0
REASON	13	11	10	1	4
ELEMENT	4	4	4	0	0
AGE	4	3	3	0	0
MEDIUM	10	9	7	2	1
FUNCTION	5	4	4	0	0
REST	8	8	8	0	0
DESCRIPTION	6	6	4	2	2
SYMPTOM	6	4	4	0	1
TIME	10	9	8	1	1
COLOR	6	5	5	0	0
TEST	8	6	6	0	0
OBJECT	2	2	2	0	0
JOURNAL	4	4	3	1	1
COUNTRY	4	3	2	1	2
AGENT	9	8	8	0	1
DATE	6	5	4	1	1

Table 9.6 Performance of Different Runs

Run ID	Total Questions	Retrieved Answers	Correct Answers	Wrong Answers	Relevant	Precision	Recall	Accuracy
Run 1	112	98	90	8	108	91.8	83.3	80.3
Run 2	97	92	82	10	95	89.1	86.3	84.5
Run 3	63	57	45	12	60	78.9	75.0	71.4
Run 4	150	141	121	20	148	85.1	81.7	80.7
Run 5	170	122	104	18	134	85.2	77.6	61.2
Run 6	45	39	38	1	40	97.4	95.0	84.4
Run 7	200	166	147	19	171	88.5	85.9	78.5

Questions were collected from seven users of different walks of life and the respective results are shown in Table 9.6.

9.6 Chapter Summary

Performance evaluation of MaQAS is discussed in this chapter. General measures used to evaluate the performance of QA Systems are described. Then the analysis and discussion of experiments conducted to test the system MaQAS is presented. The experimental results show that the Named Entity Based approach is a successful method for QAS. In addition to keyword matching, use of the deep analysis of the text helps to improve the performance of the system. Double level index not only reduced the run time overhead but also increased the speed of answer retrieval.

Chapter 10

CONCLUSION AND FUTURE WORK

Contents

Main contributions of this research work are highlighted in this chapter. Some open issues in the design of Malayalam QAS are discussed and the ways in which they could be addressed in future work are proposed.

This research work is about the development of a QAS for Malayalam language. MaQAS extracts answers suitable to the type of question using Named Entity techniques. Question Answering Systems available in other languages are mostly document or passage retrieval systems or sometimes extract answers just by keyword or pattern matching techniques. In this system, question is analysed and type of the expected answer is identified. Then double level indexing scheme is used to extract possible candidate sentences from the preprocessed documents. These sentences are ranked and top scored sentence is used as answer candidate from which the Named Entity that matches the question type is separated.

In this approach, various Natural Language Processing tools are used for the analysis of both the question and the documents. The most important part of this system is the document preprocessor which performs morphological analysis, syntactic analysis and semantic analysis. These stages tag each word of the document with POS, phrase, and NE tags. Next important stage is a question classifier which uses pattern matching and rules that have been formulated by observing a bank of sample questions. This system classifies the input queries by the question type and the type of answer. From the question, main keywords and secondary keywords or phrases are extracted to retrieve the answer from the preprocessed documents.

This system performs deep linguistic processing of the documents and prepares NE based index. During runtime, no overhead is present for document processing. MaQAS when tested with Factoid, Definition, List, and Descriptive questions gave a recall of 85.9% and precision of 88.5%.

10.1. Contributions

A major contribution of this work is the design and development of an innovative QA System for Malayalam language which focus on efficiency and robustness. The most relevant contributions of MaQAS are highlighted below.

- ➢ Question Answering Systems require understanding of both questions and documents. This needs morphological, syntactical and semantic analysis of natural language sentences. Since no NLP tools were available for Malayalam language, tools like Compound Word Splitter, POS Tagger, Phrase Chunker and Named Entity Tagger were developed.
- ➢ Special POS and Phrase Tagsets are designed.
- ➢ Most of the QA Systems perform document retrieval and sequential search of the query over the entire document collection. This system adopted a double level indexing scheme which facilitates the direct retrieval of candidate sentences.

- Traditional systems consider only proper nouns as NEs while MaQAS define every role carrying agent as a NE.
- No need to restructure the question entered by the user while majority of the systems translate the input questions to a Boolean query or a SQL statement.

10.2. Future Work

QAS developed in this work is restricted to handle questions from the domain of lifestyle and infectious diseases. This can be extended to any area (open domain QA System) making it beneficial to the society at large.

This research work can be extended to include Anaphora Resolution techniques since it presents a big challenge to QAS. Anaphora is a natural language expression used, to refer an entity that has been previously introduced in the document or in the question. The pronouns and definite noun phrases found during the pattern matching process of question analysis phase have to be linked to the correct antecedent in order to identify the exact question target. This will be helpful in extracting complete and meaningful answers. Identification of the “references” is essential to understand and process the documents in the corpus. Accuracy of NE identification and index preparation is solely dependant on proper understanding of the documents. Hence Reference Resolution has an important role in the index preparation stage.

Another direction in which this work can be extended is to include the facility of Word Sense Disambiguation (WSD). Word and their meanings provide appropriate bits necessary to construct the NE-based document representation and for key word extraction. A word can have multiple senses and hence, it is necessary to discover which sense of the word is used in the documents and in the query. Various relations exist between words and their senses. Some of them are Homonymy, Polysemy, Synonymy, and Hyponymy. Homonymy and Polysemy issues reduce the precision of a QAS by leading a system to return answers irrelevant to the user's information need. Synonymy and Hyponymy reduce the recall by causing the retrieval system without identifying the relevant answers. WordNet and WSD techniques can solve such word sense issues.

REFERENCES

[1] James Allen, "Natural Language Understanding", The Benjamin Cummings Publishing Company, 2nd Edition,1995

[2] Dan Moldovan, Mihai Surdeanu, " On the Role of Information Retrieval and Information Extraction in Question Answering Systems", LNAI 2700 pp. 29-147, 2003, © Springer-Verlag Berlin Heidelberg 2003

[3] Vennevar Bush, " As We May Think", The Atlantic, July 1945

[4] Peggy M Anderson, Philip J Hayes,Alison K Huettner, Linda M Schmandt, Irene B Nirenburg,Steven P Weinstein, " Automatic Extraction of Facts from Press Releases to Generate News Stories", 3rd Conference on Applied NLP, 1992, pp. 170-177, © ACL

[5] Beth M Sundheim, Nancy M Chinchor, " Survey of Message Understanding Conferences" , Proceedings of the Workshop on Human Language Technology, 1993, pp. 56-60

[6] E.M. Voorhees, D. Harman, "Overview of eighth Text REtrieval Conference", TREC-8, National Institute of Standards and Technology, November, 1999.

[7] Praveen Kumar, Shrikant Kashyap, Ankush Mittal.Sumit Gupta, "A Query Answering system for E- Learning Hindi Documents", South Asian Language Review, Vol XIII Nos 1&2, January- June 2003.

[8] Rami Reddy, Nandi Reddy, Sivaji Bandyopadhyay, " Dialogue Based Question Answering System in Telugu", EACL 2006 Workshop on Multilingual Question Answering - MLQA06 pp 53-60

[9] Tomek Strzalkowski, Sanda M. Harabagiu, “Advances in Open Domain Question Answering”, Text, Speech and Language Technology Series, SpringerLink: Bucher, Vol. 32, Springer 2006.

[10] “Ask Jeeves”, http://www.ask.com

[11] George A Miller, “WordNet: A Lexical Database for English”, Communication of ACM, Vol. 38 (11), pp 39-41, Nov 1995

[12] Douglas. B. Lenal, “Cyc - : A Large Scale Investment in Knowledge Infrastructure”, Communication of ACM, Vol. 38 (11), pp 33-38, Nov 1995

[13] Green W, Chomsky C, and Laugherty K., “BASEBALL: An automatic question answerer”, Proceedings of the Western Joint Computer Conference, p.p. 219-224 San Francisco, CA, USA, 1961

[14] Edward Whittaker, J Hamonic, D Yang, Tor Klingberg, Sadaoki Furui, “Monolingual Web-based Factoid Question Answering in Chinese Swedish English and Japanese”, Workshop on Multilingual QA MLQA06, pp 45-52, 2006

[15] Nikesh P L, Sumam Mary Idicula, David Peter S, “English-Malayalam Cross-Lingual Information Retrieval – an Experience”, IEEE International Conference on Electro/Information Technology, Iowa University, Ames, IA, USA, May18-20,2008

[16] Noriko Kando, “Overview of 7th NTCIR Workshop”, Proceedings of NTCIR-2 Workshop, Tokyo, Japan, 2001

[17] Abolfazl Keighobadi Lamjiri, Julien Dubac, Leila Kosseim, Sabine Bergler, “Indexing Low Frequency Information for Question

Answering", Conference RIAO2007, Pittsburg P A, USA, May 30 - June 1,2007

[18] Chirstan Grant, Clint P.George, Joir-dan Gumbs,Joseph N.Wilson, Peter J.Dobbins, "Morpheus : A Deep Web Question Answering System", iiWAS2010,8-10 Nov,2010, Paris, France, © ACM

[19] W.A. Woods, "Semantics and Quantification in Natural Language Question Answering", Advances in Computers,Vol.17,1978,Academic Press

[20] Robin D. Burke, Kristian J. Hammond, Vladimir Kulyukin, Steven L. Lytinen, Noriko Tomuro, Scott Schoenberg , **"**Question Answering from Frequently Asked Question Files", AI Magazine, Vol.18, No.2,1997, AAAI

[21] Codi Kwok, Oren Etzioni, Daniel S.Weld, "Scaling Question Answering to the Web", ACM Transactions on Information Systems, Vol.19, No.3, July 2001 pp. 242-262

[22] Dragomir Radev, Weiguo Fan, Hong Qi, Harris Wu, Amardeep Grewal, "Probabilistic Question Answering on the Web", WWW2002,May 7-11,2002, Honolulu, Hawaii,USA, ACM

[23] Zhiping Zheng, "AnswerBus Question Answering System", Proceedings of Second International Conference on Human Language Technology HLT '02, pp 399-404, 2002

[24] Boris Katz, Gary Borchardt, Sue Felshin, "Natural Language Annotations for Question Answering", AQUAINT Phase II , AAAI, 2006

[25] Pawel Kowalezyk, Ingrid Zukerman, Michael Niemann, " Analysing the Effects of Query Class on Document Retrieval Performance", AI- 2004, LNAI 3339, pp 550-561, 2004

[26] Terry Winograd, "Understanding Natural Language", University Press.

[27] Waizenbaum, Joseph, " Eliza-A Computer Program for the study of Natural Language Communication between Man and Machine", Communications of the Association for Computing Machinery , 9 1996,pp. 36-45

[28] RamiReddy NandiReddy, Sivaji Bandyopadhyay, " Dialogue based Question Answering System in Telugu" , EACL 2006 Workshop on Multilingual Question Answering -MLQA06

[29] David N.Chin, "Knowledge Structures in UC, the UNIX Consultant", National Science Foundation

[30] Otthein Herzog, Claus- Rainer Rollinger, "Text Understanding in LILOG, Integrating Computational Linguistics and AI", LNCS 546 Springer 1991.

[31] BrainBoost, Question Answering Systems: website: http://www.answer.com

[32] N. Schlaefer, P. Gieselmann, and G. Sautter. "The Ephyra QA system at TREC 2006", Proceedings of the Fifteenth Text REtrieval Conference, 2006.

[33] Wendy G. Lehnert, "The Process of Question Answering - A Computer Simulation of Cognition", American Journal of Computational Linguistics, Volume 6, Number 3-4, July - December 1980

[34] E.M. Voorhees, Buckland, Proceedings of 13th Text Retrieval Conference, TREC 2004

[35] E.M. Voorhees, "Overview of TREC 2007", Proceedings of 16th Text Retrieval Conference, TREC 2007

[36] Mahboob Alam Khalid,Susan Verberne, " Passage Retrieval for Question Answering using Sliding Windows",Coling 2008, Proceedings of the 2nd

workshop on Information Retrieval for Question Answering (IR4QA) pp (26-33) Manchaster,UK,August 2008.

[37] Graham Bennett, Falk Scholar, Alexandra Uitdenbogerd , “A comparative study of Probabilistic and Language Models for Information Retrieval”,19th Australian Database Conference (ADC 2008),Wollongong, Australia, January 2008 Kill

[38] I-chien Liu, Lun-Wei Ku, Kuang-Lua Chen and Hsin-Idsi Chen. “NTUBROWS System for NTCIR-7 IR for Question Answering”, Proceedings of NTCIR-7 Workshop meeting, December 16- 19, 2008 Tokyo, Japan.

[39] Nguyen Tuan Dang, Do Thi Thanh Tuyen, “Natural Language Question Answering model Applied to Document Retrieval System”. World Academy of Sciences, Engineering and Technology 51, 2009

[40] W.cafe, “Linguistics and Human Knowledge”, Georgetown University Monograph Series on Language and Linguistics 24, pp 57-69

[41] Mohammed Reza Kangavari, Samira Ghandchi, Manak Golpour, “ Information Retrieval: Improving Question Answering systems by Query Reformulation andAnswer Validation”, World Academy of sciences, Engg.and Technology 48,2008

[42] Yi-Hsun Lee, Cheng Wei Lee,Cheng-Lung Sung,Mon-Tin Tzou, “Complex Question Answering System with ASQA at NTCIR7 ACLIA”, Proceedings of NTCIR workshop meeting, Dec 16-19,2008,Tokyo,Japan

[43] Christian Middleton, “Open Source Search Engines- Modern Information Retrieval”, Addison Wesley, pp 1-25,2010

[44] Seungwoo Lee, Gary Geunbae Lee, “SiteQ/J:A Question Answering System for Japanese”, 3rd NTCIR Workshop Sep-Oct 2002

[45] Johannes Leveling, “Role of IR in the Question Answering System IRSAW”, Information Retrieval workshop, 9-11 Oct 2006,Hildeshiem

[46] Omar Trigui, Lamia Hadrich, Belguith Paolo Rosso, “DefArabicQA:Arabic Definition Question Answering System”, TREC-2009.

[47] Mohammed Akour, Sameer Abufardeh, Kenneth Magel, Qasem Al-Radaideh, “QArabPro:A Rule Bsed Question Answering System”,American Journal of Applied Sciences 8(6), pp 652-661,2011

[48] Samir Tartir, I.Budak Arpinar,Mustafa Nural, “Question Answering in Linked Data for scientific Exploration”, Web Science Conference, 2010, April 26-27, 2010, USA

[49] Carlos Amaral, Helena Figueira, Andre Martins Afonso Mendes, Pedro Mendes, Claudia Pinto, “Priberam’s Question Answering System for Portuguese”,1999

[50] Harksoo Kim, Kyungsun Kim, Gary Geunbae Lee,Jungyun Seo, “ MAYA: A Fast Question Answering System Based on a Predictive Answer Indexer”, in proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL'01), Workshop on Open-Domain Question Answering, 2001

[51] Tiansi Dong, Ulrich Furbach, “ A Natural Language Question Answering System as a Participant in Human Q&A Portals”, Proceedings of 22nd International Joint Conference on Artificial Intelligence,pp 2430-2435

[52] Hermann Helbig, "Knowledge Representation and the Semantics of Natural Language", Springer, 2006.

[53] P. Baumgartner, U. Furbach, M. Gross-Hardt, T. Kleemann and C. Wernhard, "KRHyper Inside - Model Based Deduction in Applications", Proc. CADE-19 Workshop on Novel Applications of Deduction Systems, 2003.

[54] Junichi Fukumoto, " Question Answering System for Non-factoid Type Questions and Automatic Evaluation Based on BE Method", Proceedings of NTCIR-6 Workshop Meeting, May 15-18, 2007, Tokyo, Japan

[55] Xiaoyan Li and W.Bruce Soft, "Evaluating Question Answering Techniques in Chineese", HLT '01 Proceedings of the 1st International Conference on Human Language Technology Research pp 1-6, ACL,USA 2001

[56] Partha Pakray, Pinaki Bhasker, Somanath Banerjee, Bidhan Chandrapal , Sivaji Bandyopadhayay, Alexander Gelbukh, "A Hybrid Question Answering System Based on Information Retrieval and Answer Validation", QA4MRE@CLEF 2011, 19-22 September 2011

[57] Diego Molla, Menno Van Zannen, "Answer Finder", CLEF 2007 Workshop, Budapet, Hungary,2007

[58] Yassine Benajiba, Paolo Rosso, Abdelouahid Lyhyaoui, "Implementation of ArabiQA Question Answering System's Components" , Proceedings of International Colloquium on Arabic Language Processing 2007.

[59] Johannes Leveling, "Intelligent Information Retrieval on the Basis of a Semantically Annotated Web", CNI Spring 2007 Task Force Meeting, April 16-17, 2007, Phoenix, AZ

[60] Sa-Jeong Ko, Jung Hyun Lee, “Feature Selection Using Association Word Mining for Classification”, Proceedings of the 12th International Conference on Database and Expert System Applications pp 211-220

[61] Rohini Srihari,Wei Li, “A Question Answering System Supported by Information Extraction”, Proceedings of 6th Conference on Applied Natural Language Processing, pp- 166-172, USA, 2000

[62] Bhadriraju Krishnamurthi, “The Dravidian Languages”, Cambridge University Press, 2003

[63] http://www.kerala/language.htm

[64] http://www.karmakeralam.com/literature/language-literature-of-kerala.html

[65] http://www.keralacafe.com/kerala-language/index.htm

[66] A .R .Rajarajavarma, “ Keralapanineeyam”, National Book Stall, Kottayam, 2000.

[67] Dr.C.K.Chandrasekharan Nair, “Adisthana Vyakaranam”, Kerala Bhasha Institute, Thiruvananthapuram, 1997

[68] Prof. K.S.Narayana Pillai(1995), “Adhunika Malayala Vyakaranam”, Kerala Bhasha Institute, Thiruvananthapuram

[69] Ricardo A Baeza-Yates, Berthier Riberio Neto, “ Modern Information Retrieval”, Addison-Wesley Longman Publishing Co. Inc, Boston,USA,1999.

[70] Marcin Skowron,Kenji Araki , “Effectiveness of Combined Features for Machine Learning Based Question Classification”, Journal of Natural Language Processing 12(6), pp 63-83, 2005

[71] Vicado Jose Luis , Ferranadez Antonio, "A Semantic Approach to Question Answering Systems", TREC -9, 2000

[72] Xin Li, Dan Roth, "Learning Question Classifiers", Proceedings of the 19th International Conference on Computational Linguistics (COLING), Taipei,Taiwan,2002 pp- 556-562

[73] Alessander Moschitti, Sanda Harabagiu, "A Novel Approach to Focus Identification in Question Answering Systems", Workshop on Pragmatics of Question Answering", 2004

[74] Ralf D Brown, "Corpus Driven Splitting of Compound Words", In Proceedings of Ninth international Conference on Theoretical and Methodological Issues in Machine Translation (TMI-2002), Keihanna, Japan, March 13-17

[75] Philippi Koehn, Kevin Knight, "Emprical Methods for compound Word Splitting", Proceedings of 11th Conference of European Chapter of Association of Computational Linguistics, pp. 187-193, 2003

[76] Robert Hecht, Jürgen Riedler, Gerhard Backfried, "Fitting German into N-Gram Language Models", LNAI 2448, pp- 341-346,2002,Springer-Verlag

[77] Kann, V., R. Domeij, J. Hollman, M. Tillenius, " Implementation Aspects and Applications of a Spelling Correction Algorithm", NADA report TRITA-NA-9813, 1998.

[78] Anne schiller, "German compound word analysis with wfsc", Proceedings of the Fifth International Workshop of Finite State Methods in Natural Language Processing,Helsinki,2005

[79] Ulrike Rackow, Ido Dagan, Ulrike Schwall, "Automatic Translation of Noun Compounds", Proceedings of COLING, Aug 23-28, 1992 , pp. 1249-1253

[80] L. Karttunen, " Applications of FST in Natural Language Processing", Proceedings of CIAA-2000, LNCS, Springer Verlag

[81] T.N. Vikram & Shalini R, (2007), "Development of Prototype Morphological Analyzer for the South Indian Language of Kannada", Lecture Notes in Computer Science: Proceedings of the 10th international conference on Asian digital libraries: looking back 10 years and forging new frontiers. Vol.4822/2007, 109-116.

[82] Shambhavi. B.R, Dr. Ramakanth Kumar P, Srividya K, Jyothi B J , Spoorti Kundargi, Varsha shastri, "Kannada Morphological Analyser and Generator using Trie", International Journal of Computer Science and 297 Network Security (IJCSNS) Vol 11 No. 1, Jan 2011, pp 112-116

[83] K. Narayana Murthy, "Issues in the Design of a Spell Checker for Morphologically Rich Languages", 3rd International Conference on South Asian Languages, ICOSAL-3, 4th to 6th January 2001, University of Hyderabad

[84] Ramasamy Veerappan, Antony P.J, S.Saravanan, and Dr.Soman.K.P, "A Rule-based Kannada Morphological Analyser and Generator using FST", IJCA vol 27(10) August 2011.

[85] Uma Maheswara Rao G, Parameshwari K: CALTS, University of Hyderabad, "On the description of morphological data for morphological analyzers and generators: A case of Telugu, Tamil and Kannada", 2010.

[86] Kiranmai.G, Mallika K, Anandkumar N, Dhanalakshmi V, and Soman K.P, "Morphological Analyser for Telugu Using Support Vector Machines", ICT 2010, pp.430-433, Springer-Verlag.

[87] Abu Zaher, Md.Faridee, Francis M Tayers, "Development of a Morphological Analyser for Bengali", Proceedings of First International Workshop on Free/Open Source Rule-based Machine Translation", pp. 43-50, November 2009.

[88] Sajib Dasgupta and Vincent Ng., ".Unsupervised Morphological Parsing of Bengali". In the journal of Language Resources and Evaluation, 2007. 40:3-4, pp 311-330

[89] Mohanty, S., Santi. P.K., Adhikary, K.P.D. 2004., "Analysis and Design of Oriya Morphological Analyser: Some Tests with OriNet", In Proceeding of symposium on Indian Morphology, phonology and Language Engineering, IIT Kharagpur

[90] Girish Nath Jha., Muktanand Agarwal., Subash., Sudhir K Mishra.,DiwakarMishra., Manji Bhadra Surjit K Singh, "Inflectional Morphology Analyzer for Sanskrit", Sanskrit CL, pp. 219-238, 2009, Springer-Verlag

[91] Mugdha Bapat , Harshada Gune, and Pushpak Bhattacharyya, "A Paradigm-Based Finite State Morphological Analyzer for Marathi", Proceedings of the First Workshop on South and Southeast Asian Natural Language Processing, pp. 26-34, 23rd International Conference on CL (COLING), August 2010.

[92] Ganesan M (2007), "Morph and POS Tagger for Tamil" (Software), Annamalai University, Annamalai Nagar.

[93] Anandan P, Ranjani Parthasarathy and Geetha T.V (2002), "Morphological Analyzer for Tamil", ICON 2002, RCILTS-Tamil, Anna University, India.

[94] Parameshwari K, "An Implementation of APERTIUM Morphological Analyzer and Generator for Tamil", Language in India

www.languageinindia.c o m. 11:5 M ay 2011, Special Volume: Problems of Parsing in Indian Languages, 2011

[95] Vijay Sundar Ram R, Menaka S and Sobha Lalitha Devi (2010), “Tamil Morphological Analyser”, In Mona Parakh (ed.) Morphological Analyser For Indian Languages, CIIL, Mysore, pp. 1 -18.

[96] Akshar Bharat, Rajeev Sangal, S. M. Bendre, Pavan Kumar and Aishwarya, “Unsupervised improvement of morphological analyzer for inflectionally rich languages,” Proceedings of the NLPRS, pp. 685-692, 2001.

[97] Menon A. G.; Saravanan S; Loganathan R; Soman K. P. (2009): Amrita Morph Analyzer and Generator for Tamil: A Rule-Based Approach, TIC, Cologne, Germany, pp. 239-243.

[98] Deepa. S.R, Kalika Bali, A.G. Ramakrishnan, and Partha Pratim Talukdar, “Automatic Generation of Compound Word Lexicon for Hindi Speech Synthesis”, Language Resources and Evaluation Conference (LREC), 2004.

[99] Deepak Kumar, Manjeet Singh, and Seema Shukla, “FST Based Morphological Analyzer for Hindi Language”, First International Conference on Emerging Trends in Engineering and Technology, IEEE Computer Society-2008.

[100] Daniel Jurafsky and James H Martin, “Speech and Language Processing”,AI Pearson Education Series in AI.,First Indian Print 2002.

[101] Eric Brill, “A Simple Rule Based POS Taggar”, Proceedings of the third Conference on Applied Computational Linguistics, Torento, Italy, 1992.

[102] J Kupiec, “Robust Part-of-Speech Tagging using a Hidden Markov Model”, Computer Speech and Language, vol 6, pp 225-242, 1992

[103] Steven J. DeRose, “Grammatical Category Disambiguation by Statistical Optimization”, Computational Linguistics Vol 14(1), pp 31-39

[104] Kenneth Ward Church, “A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text”, Proceedings of the Second Coference on Applied Natural Language Processing ANLC’ 88,1988, PP 136-143

[105] Dinh Dien, Hong Kien, “Part of Speech Taggar for English Vietnamese Bilingual Corpus”,HLT-NAACL 2003 Workshop Building and Using Parallel Texts Data Driven Machine Translation and Beyond pp88-95,Edmonton,May-June 2003.

[106] K.T Lua, “POS Tagging of Chinese Sentences using Genetic Algorithms”, Conference on Chinese Computing 1996 4-7 June National University of Singapore pp 45-49.

[107] R. M. K Sinha, “An Engineering Perspective of Machine Translation, AnglaBharti-II and AnuBharti-II Architectures”,Proceedings of International Symposium on MT, NLP and Translation Support Systems”, Tata McGraw Hill, NewDelhi, pp 134-138,2004

[108] Kommaluri Vijayanad, Ramalingam Subramanian, “Anuvadini:An Automatic example-based Machine Translation System for Bengali into Assamese and Oriya”, Proceedings of First National Symposium on Modelling and Shallow Parsing of Indian Languages, Bombay,India, 2006

[109] Vijay Sunder Ram R, ChandraMouli N, Bhuvaneswari P, Ananda Priya J, B Kumara Shanmugam, “Hybrid Approach for Developing a Tamil Spell Checker”, Proceedings of International Conference on Natural Language Processing, pp 111-114, 2005

[110] Gurpeet Singh Lehal, “Design and Implementing of Punjabi Spell Checker”,International Journal of Systemics, Cyberrnetics and Informatics, pp 70-75, 2007

[111] Manish Shrivastava and Pushpak Bhattacharyya, “Hindi POS Tagger Using Naive Stemming: Harnessing Morphological Information Without Extensive Linguistic Knowledge”, International Conference on NLP (ICON08), Pune, India, December, 2008

[112] Dhanalakshmi V, Anand Kumar , Shiva Prasad G, Soman K.P , Rajendran S, “Tamil POS using Linear Programming”, International Journal of Recent Trends in Engineering vol 1 no.2 May 2007.

[113] Chirag Patel and Karthik Gali, “POS tagging for Gujarathi using CRF”,Proceedings of the IJCNLP-08 Workshop on NLP for lLess Privileged Languages pp 117-122.

[114] Smriti Singh, Kuhoo Gupta, Manish Shrivastava and Pushpak Bhattacharyya (2006), “Morphological richness offsets resource demand – experiences in constructing a pos tagger for Hindi”, Proceedings of the COLING/ACL 2006,Sydney, Australia Main Conference Poster Sessions, pp. 779–786.

[115] Manish Shrivastava and Pushpak Bhattacharyya, “Hindi POS Tagger Using Naive Stemming: Harnessing Morphological Information Without Extensive Linguistic Knowledge”, International Conference on NLP (ICON08), Pune, India, December, 2008.

[116] Nidhi Mishra Amit Mishra (2011), “Part of Speech Tagging for Hindi Corpus”, International Conference on Communication Systems and Network Technologies.

[117] Pradipta Ranjan Ray, Harish V., Sudeshna Sarkar and Anupam Basu, (2003) "Part of Speech Tagging and Local Word Grouping Techniques for Natural Language Parsing in Hindi" , Indian Institute of Technology, Kharagpur,INDIA

[118] Sivaji Bandyopadhyay, Asif Ekbal and Debasish Halder (2006), "HMM based POS Tagger and Rule-based Chunker for Bengali", Proceedings of NLPAI Machine Learning Workshop on Part Of Speech and Chunking for Indian Languages.

[119] Sandipan Dandapat (2007), "Part Of Speech Tagging and Chunking with Maximum Entropy Model", Proceedings of IJCAI Workshop on Shallow Parsing for South Asian Languages.

[120] Hammad Ali (2010), "An Unsupervised Parts-of-Speech Tagger for the Bangla language", Department of Computer Science, University of British Columbia. 2010.

[121] Debasri Chakrabarti (2011), "Layered Parts of Speech Tagging for Bangla", Language in India www.languageinindia.com, May 2011, Special Volume:Problems of Parsing in Indian Languages.

[122] Dinesh Kumar and Gurpreet Singh Josan,(2010), "Part of Speech Taggers for Morphologically Rich Indian Languages: A Survey", International Journal of Computer Applications (0975 – 8887) Volume6– No.5, September, 2010

[123] T. Sreeganesh, "Telugu POS Tagging in WSD", Language of India, vol 6(8) August 2006.

[124] Avinesh PVS and Karthik Gali, "POS Tagging and Chunking Using CRF and TBL", Proceedings of the IJCAI and the Workshop on SPSAL, pp. 21-24.

[125] Rama Sree, R.J Kusuma Kumari P, “Combining POS Taggers for Improved Accuracy to Create Telugu Annotated Text for IR”, Tirupati.

[126] G. Sindiya Binulal, P. Anand Goud, K.P Soman, “A SVM Based Approach to Telugu POS Tagging Using SVM Tool”, International Journal of Recent Trends in Engg.”, Vol 1(2), May 2009

[127] Chirag Patel and Karthik Gali, “POS Tagging for Gujarati Using CRF”, Proceedings of the IJCNLP-08 workshop on NLP for Less Privileged Languages”, Hyderabad, pp. 117-122.

[128] Thodam Doren Singh and Sivaji Bandyopadhayay, “Morphology Driven Manipuri POS tagger”, Proceedings of IJCNLP-08 Workshop on NLP, pp. 91-98

[129] Thodam Doren Singh and Sivaji Bandyopadhayay, “Manipuri POS Tagging Using CRF and SVM: A Language Independent Approach”, Proceedings of ICON-2008: Sixth International Conference on NLP.

[130] Antony P.J and K.P. Soman. 2010, “ Kernel Based Part of Speech Tagger for Kannada”, In Machine Learning and Cybernetics (ICMLC), 2010 International Conference on, volume 4, pages 2139 –2144, July.

[131] Arulmozhi P, Sobha L, Kumara Shanmugam. B (2004), “Parts of Speech Tagger for Tamil”, Proceedings of the Symposium on Indian Morphology, Phonology & Language Engineering, Indian Institute of Technology, Kharagpur.

[132] Arulmozhi P and Sobha L (2006), “A Hybrid POS Tagger for a Relatively Free Word Order Language”, Proceedings of MSPIL-2006, Indian Institute of Technology, Bombay.

[133] Lakshmana Pandian S and Geetha T V (2009), “CRF Models for Tamil Part of Speech Tagging and Chunking “, Proceedings of the 22nd ICCPOL

[134] M. Selvam, A.M. Natarajan (2009), “Improvement of Rule Based Morphological Analysis and POS Tagging in Tamil Language via Projection and Induction Techniques”, International Journal of Computers, Issue 4, Vol 3, 2009.

[135] Isabelle Tellier , Iris Eshkol Samer Taalab and Jean –Philippe Prost, “POS tagging for Oral text with CRF and Category Decomposition”, 11th International Conference on Intelligent Text Processing and Computational Linguistics Romania-2010.

[136] Hanna.M.Wallach, “Conditional Random Fields”,University of Pennsylvania CIS Technical Report MS-CIS-04-21.

[137] Ghassan Kannan, Riyad-al-Shalabi and Majdi Sawalha, “Improving Arabic Information Retrieval System Using Part of Speech Tagging”,Information Technology Journal 4(1) 32-37,2005

[138] Steven P. Abney, “ Parsing by chunks”, Kluwer Academic Publishers (1991), pp. 257–278.

[139] Yoshimasa Tsurnoka , Junichi Tsujii, Sophia Ananiadou, “Fast ful Parsing by Linear Chain Conditional Random Field(2005)”,Proceedings of 12th Conference of the European Chapter of the ACL, pp 790-798.

[140] Ramashaw, M. P. Marcus(1995), “ Text chunking using transformation-based learning”, Proceedings of the Third ACL Workshop on Very Large Corpora (1995).

[141] Chingtham Tejbanta Singh , Shivashankar.B Nair(2005), “An Artificial Immune system for a Multi agent Robotics System”,World Academy of sciences Engineering and Technology 2005.

[142] Nasser Omer Sahel Ba-Karait, Siti Mariyam Shamsuddin, Rubita Sudirman (2009), "Swarm Negative Selection Algorithm for Electroencephalogram Signals Classification",Journal of Computer Science 5(12) pp 998-1005,2009.

[143] Leandro N De Castro, Fernando J Van Zuben(2002), "Learning and Optimization using the Clonal Selection Principle", IEEE Transactions on Evolutionary Computation Special Issue on Artificial Immune System vol 6 No.3 pp 239-251,2002.

[144] Fangjia Li,Shangee Gao,Wee Wang and Zheng Tag(2007), "An Adaptive clonal Selection selection Algorithm for Edge Linking Problem", IJCSNS International Journal of Computer Science and Network Security vol 9 , No.7 July 2007.

[145] Akshat Kumar, Shivashankar B Nair(2007), "Artificial Immune System Based Approach for English Grammar Checking", LNCS pp348-387.

[146] Akshay Singh, Sushma Bendre, Rajeev Sangal, "HMM Based Chunker for Hindi", Proceedings of IJCNLP-05: The Second International Joint Conference on Natural Language Processing, October, 2005

[147] Sobha. L, Vijay Sunder Ram R, "Noun Phrase Chunking in Tamil", Proceedings of the First National Symposium on Modelling and Shallow Parsing of Indian Languages MSPIL-06,April 2006

[148] Avinesh . PVS, Karthik G, "POS Tagging and Chunking using CRF and Transformatiom Based Learning", Proceedings of Workshop on Shallow Parsing for South Asian Languages IJCAI, 2007

[149] Sandipan Dandapat, "POS Tagging and Chunking With Maximum Entropy Model", Proceedings of Workshop on Shallow Parsing for South Asian Languages IJCAI, 2007

[150] Danalakshmi. V, PadmavathY. P, Anand K.M, Soman K.P, Rajendran S, “Chunker for Tamil”, International Conference on Advances in Recent Technologies in Communication and Computing, 2009

[151] Ruy L Milidiu, Cicero Nogueira dos Santos, Julio C Duarte, “Phrase Chunking using Entropy Guided Trandformation Learning”, Proceedings of ACL-08, Human Language Technology, pp 647-655, June 2008

[152] G.M. Ravi Sastry , Sourish Chaudhuri and P. Nagender Reddy, “An HMM based Part-Of-Speech tagger and statistical chunker for 3 Indian languages”, SPSAL 2007

[153] Pattabhi R K Rao T, Vijay Sundar Ram R, Vijayakrishna R and Sobha L (2007), “A Text Chunker and Hybrid POS Tagger for Indian Languages”, AU-KBC Research Centre, MIT Campus, Anna University, Chromepet, Chennai, 2007.

[154] Asif Ekbal, Samiran Mandal and Sivaji Bandyopadhyay (2007), “POS Tagging Using HMM and Rule-based Chunking”, Workshop on shallow parsing in South Asian languages.

[155] Sathish Chandra Pammi and Kishore Prahallad (2007), “POS Tagging and Chunking using Decision Forests”, Workshop on shallow parsing in South Asian languages, 2007. shiva.iiit.ac.in/SPSAL2007/proceedings.php.

[156] Dipanjan Das, Monojit Choudhary, Sudeshna Sarkar, and Anupam Basu, “An Affinity Based Greedy Approach Towards Chunking for Indian Languages”, Proceedings of International Conference on Natural Language Processing, ICON 2005.

[157] Wajid Ali, Sarmad Hussaion, “A Hybrid Approach to Urdu verb Phase Chunking”, Proceedings of the 8th Workshop on Asian Language Resources, pages 136–142,Beijing, China, 21-22 August 2010.

[158] Dong Hwa Kim, Kye Young Lee(2002), “Neural Networks Control by Immune Network Algorithm Based Auto Weight Function Tuning”, Proceedings of IJCNN, Vol.2, pp. 1469-1474.

[159] Leandro N De Castro, Jon Timmis, “An Immune Network for Multimodal Function Optimization”, Proceedings of IEEE Congress on Evolutionary Computation CEC ’02, pp 699-674, 2002

[160] Stephani Forest, Alan S Perelson, Lawrence Allen,Rajesh Cherukuri(1994), “Self-Nonself Discrimination in a computer”, Proceedings of 1994 IEEE Symposium on Research in Security and Privacy.

[161] Simon M Garrett (2005), “How do we evaluate Artificial Immune System”, Evolutionary Computation 13(2):145-178.

[162] Ali Elsebai, Farid Meziane , Fatina Zihra Belkredim, “A Rule Based Persons Names Arabic Extraction System”, Communications of IBIMA, vol.11, 2009

[163] Mark Stevenson, Robert Gaizauskas, “Improving Named Entity Recognition Using Annotated Corpora”, Proceedings of the LREC Workshop –Information Extraction meets Corpus Linguistics,2000, Athens, Greece

[164] Asif Ekbal, Rejwanul Haque, Amitava Das, Venkateswarlu Poka, Sivaji Bandyopadhyay, “Language Independent Named Entity Recognition in Indian Languages”, Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian Languages, pages 33–40,Hyderabad, India, January 2008.

[165] Burr Settles, “ Biomedical Named Entity Recognition Using Conditional Random Fields and Rich Feature Sets”, Proceedings of the COLING 2004 International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA). Geneva, Switzerland. 2004.

[166] GuoDong Zhou, Jian Su, “ Named Entity Recognition using an HMM-based Chunk Tagger”, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 473-480

[167] Mohammad Hasanuzzaman, Asif Ekbal, Sivaji Bandyopadhyay, “ Maximum Entropy Approach for Named Entity Recognition in Bengali and Hindi”, International Journal of Recent Trends in Engineering, Vol. 1,No.1, May 2009

[168] Hayssam Traboulsi, “Arabic Named Entity Recognition: A Local Grammar Based Approach”, Proceedings of the International Multiconference on Computer Science and Information Technology, pp 139-143, October 2009

[169] Kashif Riaz , “Rule-based Named Entity Recognition in Urdu”, Proceedings of the 2010 Named Entities Workshop, ACL 2010, pp 126–135, Uppsala, Sweden, 16 July 2010.

[170] E. Ferreira, J. Balsa, A. Branco, “Combining Rule-based and Statistical methods for Named Entity Recognition in Portuguese”, V Workshop em Tecnologia da Informa¸c˜ao e da Linguagem Humana, pages 1615–1624, 2007.

[171] Asif. Ekbal, R. Haque, and S. Bandyopadhyay, “Named Entity Recognition in Bengali: A Conditional Random Field,” in *Proceedings of ICON*, India, pp. 123–128.

[172] Asif. Ekbal and S. Bandyopadhyay, “Bengali Named Entity Recognition using Support Vector Machine,” in *Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian languages*, Hyderabad, India, January 2008, pp. 51–58.

[173] K.S. Hasan, M. U.R Rahman, and V. Ng, “Learning -Based Named Entity Recognition for Morphologically-Rich Resource-Scare Languages”, In Proceedings of the 12th Conference of the European Chapter of the ACL, Athens, Greece, 2009, pp. 354–362

[174] B. B. Chaudhuri and S. Bhattacharya, “An Experiment on Automatic Detection of Named Entities in Bangla,” in *Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian languages*, Hyderabad, India, January 2008, pp. 75–82.

[175] Vijayakrishna. R and Sobha. L, “Domain focused Named Entity Recognizer for Tamil using Conditional Random Fields,” in *Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian languages*, Hyderabad, India, 2008, pp. 59–66.

[176] S. Lakshamana Pandian, Krishna. Aravind. Pavithra, and T.V. Geetha, “Hybrid Three-stage Named Entity Recognizer for Tamil,” *INFOS2008*, March 2008.

[177] Rajesh Sharma, Vishal Goyal, “NER System for Hindi Using CRF Approach”, ICISIL, 2011, PP. 31-35, Springer-Verlag.

[178] S. K. Saha, S. Sarkar, and P. Mitra, “A Hybrid Feature Set based Maximum Entropy Hindi Named Entity Recognition,” in *Proceedings of the 3rd International Joint Conference on NLP*, Hyderabad, India, January 2008, pp. 343–349.

[179] Amit Goyal, “Named Entity Recognition for South Asian Languages,” in *Proceedings of the IJCNLP-08 Workshop on NER for South and South-East Asian Languages*, Hyderabad, India, Jan 2008, pp. 89–96.

[180] W. Li and A. McCallum, “Rapid Development of Hindi Named Entity Recognition using Conditional Random Fields and Feature Induction (Short Paper),” *ACM Transactions on Computational Logic*, pp. 290–294, Sept 2003.

[181] P. K. Gupta and S. Arora, “An Approach for Named Entity Recognition System for Hindi: An Experimental Study,” in *Proceedings of ASCNT-2009*, CDAC, Noida, India, pp. 103–108.

[182] S.Biswas, S.P.Mohanty, S.Acharya, and S.Mohanty, “A Hybrid Oriya Named Entity Recognition system,” in *Proceedings of the CoNLL*, Edmonton, Canada, 2003.

[183] G. Raju, B.Srinivasu, D. S. V. Raju, and K. Kumar, “Named Entity Recognition for Telegu using Maximum Entropy Model,” *Journal of Theoretical and Applied Information Technology*, vol. 3, pp. 125–130, 2010.

[184] P.Srikanth and K. N. Murthy, “Named Entity Recognition for Telegu,” ,in *Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian languages*, Hyderabad, India, Jan 2008, pp. 41–50.

[185] P. M. Shishtla, K. Gali, P. Pingali, and V. Varma, “Experiments in Telegu NER: A Conditional Random Field Approach,” in *Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian languages*, Hyderabad, India, January 2008, pp. 105–110.

[186] Padmaja Sharma, Utpal Sharma, and Jugal Kalita, “The First Towards Assamese Named Entity Recognition”, Brisbane Convention Centre, Australia, September 2010

[187] Thorsten Joachims, “Text Categorization with Support Vector Machines: Learning with Many Relevant Features”, European Conference on Machine Learning”, 1998

[188] Taku Kudoh, Yuji Matsumoto, “Chunking with Support Vector Machines”, Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies NAACL ’01, pp 1-8, 2001

[189] Tetsuji Nakagawa, Taku Kudoh, Yuji Matsumoto, “Unknown Word Guessing and POS Tagging with Support Vector Machines”, Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium, 2001

[190] Durgesh K.Srivastava, Lekha Bhambhu, “Data Classification using Support Vector Machines”, Journal of Theoretical and Applied Information Technology, pp 1-7, vol 12(1), 2009

[191] Hyeran Byun, Seong-Whan Lee, “Applications of Support Vector Machines for Pattern Recognition: A Survey”, LNCS 2388, pp 213-236, 2002

[192] Gjorgji Madzarov, Dejan Gjorgjevikj and Ivan Chorbev, “ A Multi-class SVM Classifier Utilizing Binary Decision Tree”, Informatica 33 (2009) 233-241

[193] Asif Ekbal and Sivaji Bandyopadhyay, “Named Entity Recognition Using Appropriate Unlabeled Data, Post-processing and Voting”,Informatica 34 (2010) 55–76

[194] Sujan Kumar Saha , Shashi Narayan, Sudeshna Sarkar, Pabitra Mitra, “ A composite kernel for named entity recognition”, Pattern Recognition Letters (2010),Elsevier

[195] Christopher D Manning, Prabhakar Raghavan, Hinrich Schutze, "Introduction to Information Retrieval", Cambridge University Press, 2008.

[196] Sarra El Ayari, Brigitte Grau, "A Framework of Evaluation for Question Answering Systems", ECIR 2009, LNCS 5478, pp. 744-748, 2009, © Springer-Verlag Berlin Heidelberg 2009.

[197] S.Quarteroni, S.Manandher, "Designing an Interactive Open Domain Question Answering System", Natural Language Engineering 1(1): 1-23, 2008, Cambridge University Press.

[198] Ghassan Kannan, Awni Hammouri, Riyad Al-shalabi, Majdi Swaha, "A New Question Answering System for the Arabic Language", American Journal of Applied Sciences 6(4), pp. 797-805,2009

PUBLICATIONS

A) Publications in International Refereed Journals

1. Bindu.M.S, Sumam Mary Idicula, "Compound Word Generation and Analysis in Malayalam for the Purpose of Information Retrieval", International Journal of Computer Science and Information Technology **(IJCSIT**), Vol.2, No.2, Dec. 2009.

2. Bindu.M.S, Sumam Mary Idicula, "Named Entity Recognizer Employing Multiclass Support Vector Machines for the Development of Question Answering Systems", International Journal of Computer Applications **(IJCA),** Vol.25, No.10, July 2011.

3. Bindu.M.S, Sumam Mary Idicula, "High Order Conditional Random Field Based Part of Speech Tagger for Malayalam -a Highly Agglutinative Language", International Journal of Advanced Research in Computer Science (**IJARCS**), Vol.2, No.5, Sep. 2011.

4. Bindu.M.S, Sumam Mary Idicula, "A Hybrid Model for Phrase Chunking Employing Artificial Immunity System and Rule Based Methods", International Journal of Artificial Intelligence and Applications (**IJAIA**), Vol 2, No.4 October 2011.

5. Bindu.M.S, Sumam Mary Idicula, "Named Entity Identifier for Malayalam Using Linguistic Principles Employing Statistical Methods", International Journal of Computer Science Issues (**IJCSI**), vol.8, No.4 July 2011.

B) Papers in International Conferences

1. Bindu.M.S, Sumam Mary Idicula, "Analysis of Malayalam Compound Words and Implementation of a Compound Word Splitter Tool Using Finite State Models", International Conference on Modeling and Simulation (MS 09) India 1-3 Dec 2009.Organised by CET, Trivandrum and AMSE, France.
2. Bindu.M.S, Sumam Mary Idicula, "Experimental Analysis of N-gram and Vector Space Language Models and Development of a Hybrid Model for Malayalam Document Representation", International Conference on Modeling and Simulation (MS '09) India 1-3 Dec 2009.
3. Bindu.M.S, Sumam Mary Idicula, "Malayalam Text Classification Using an Enhanced N-gram Based Vector Space Model for Health Applications", The third International Conference on Semantic E-Business and Enterprise Computing,15-17 September, 2010. Organized by AJCE, Kanjirappally and Kingston University, UK

C) Papers in National Conferences

1. Bindu.M.S, Sumam Mary Idicula, "Compound Word Generation and Analysis in Malayalam for the Purpose of Information Retrieval", A National Conference on Computer Science and Information Technology", 25-26 November 2009. Organized by University of Calicut and NIMIT (**Best Paper Award**)

APPENDIX A
STOP WORD LIST

സാധാരണ ഗതിയിൽ (saadhaaraNa gathiyil)
വരെ (vare)
ഈ (ee)
ഇളം (iLam)
ചെറിയ (cheRiya)
അന്ന് (ann~)
അതിനിടെ (athiniTe)
തന്നെ (thane)
പെട്ടെന്ന് (peTTenn~)
ഇത്തരം (iththaram)
അവിടെ (aviTe)
അതുപോലെ (athupOle)
ഇതിന്റെ (ithinte)
എത്രയോ (ethrayO)
ധാരാളം (dhaaraaLam)
ഇതോടെ (ithOTe)
അതിനെകുറിച്ച് (athinekkuRichch~)
താമസിയാതെ (thaamasiyaathe)
അല്പമായ (alpamaaya)
അങ്ങേയറ്റം (angngEyatam)
ഒരു (oru)
ഇനി (ini)
പല (pala)
ക്രമേണ (kramENa)
പ്രസതുത (prasathutha)
തുടർന്ന് (thuTarnn~)
എന്നാൽ (ennaal)
ഏതാനും (Ethaanum)
ഉപയോഗിച്ചു (upayOgichchu)
പൂർണ്ണമായി (poorNNamaayi)
കഴിവതും (kazhivathum)
തൽഫലമായി (thalphalamaayi)
പരിപൂർണ്ണമായോ paripoorNNamaayO

പിന്നീട് (pinneeT~)
എന്നിവ (enniva)
തീരെ (theere
കൂട്ടമായി (kooTTamaayi)
മുതലുള്ള (muthaluLLa)
മറ്റ് (mat~)
ഇതിനിടെ (ithiniTe)
ചിലപ്പോൾ (chilappOL)
വളരെ (vaLare)
ഇവിടെ (iviTe)
ഒട്ടേറെ (oTTERe)
അതിന്റെ (athinte)
ഉണ്ടാകുന്ന (uNTaakunn)
എന്നിവയിലെ (ennivayile)
പ്രധാന(pradhaana)
മുൻപേതന്നെ (munpEthanne)
പരമാവധി (paramaavadhi)
സഹായിക്കും (sahaayikkum)
ആവശ്യം (aavaSyam)
പരിധിവരെ (paridhivare)
ഇതല്ലാതെ (ithallaathe)
മാത്രം (maathram)
അതേ(athE)
കൂടുതലായി (kooTuthalaayi)
ആയ (aaya)
വളരെ (vaLare)
ഒരിക്കലും(orikkalum)
നിമിഷങ്ങൾക്ക് (nimishangngaLkk~)
പലപ്പോഴും (palappOzhum)
വേണ്ടവിധം(vEnTavidham)
അഥവാ(athhavaa)
പരിധിവരെ (paridhivare)
ഭാഗികമായോ(bhaagikamaayO)

പരസ്പരം (parasparam)
നിശ്ശേഷം (niSSEsham)
മൊത്തമായോ(moththamaayO)
വ്യക്തമായി (vyakthamaayi)
ശ്രദ്ധകൊടുത്ത് (SraddhakoTuthth~)
ഘടനാപരമായി (ghaTanaaparamaayi)
പലതും (palathum)
തമ്മിൽ (thammil)
ഗുരുതരമായ (gurutharamaaya)
വേഗത്തിൽ (vEgaththil)

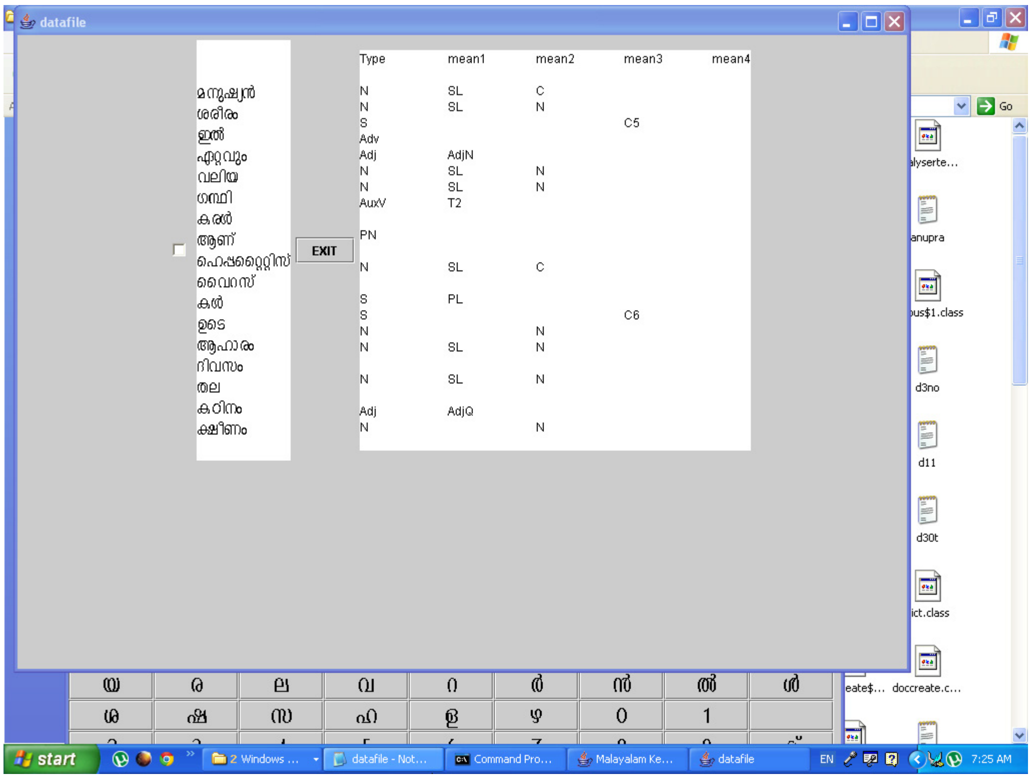
APPENDIX B

Malayalam ISCII-Unicode Mapping Table

-127 അ 0081	-126 ആ 201A	-125 ഇ 0192	-124 ഉ 201E	-123 ഋ 2026	-122 എ 2020	-121 ഏ 2021	-120 ഒ 02C6	-119 ക 2030	-118 ഖ 0160
-117 ഗ 2039	-116 ഘ 0152	-115 ങ 008D	-113 ഛ 008F	-112 ജ 0090	-111 ഝ 2018	-110 ഞ 2019	-109 ട 201C	-108 ഠ 201D	-107 ഡ 2022
-106 ഢ 2013	-105 ണ 2014	-104 ത 02DC	-103 ഥ 2122	-102 ദ 0161	-101 ധ 203A	-100 ന 0153	-99 പ 009D	-96 0 00A0	-95 ബ 00A1
-94 ഭ 00A2	-93 മ 00A3	-92 യ 00A4	-91 ര 00A5	-90 റ 00A6	-89 ല 00A7	-88 ള 00A8	-87 ഴ 00A9	-86 വ 00AA	-85 ശ 00AB
-84 ഷ 00AC	-83 സ 00AD	-82 ഹ 00AE	-81 . ാ 00AF	-80 ി 00B0	-79 ീ 00B1	-78 ു 00B2	-77 ൂ 00B3	-76 ൃ 00B4	-75 െ 00B5
-74 േ 00B6	-73 ൗ 00B7	-72 ് 00B8	-71 ം 00B9	-70 ഃ 00BA	-69 ്യ 00BB	-68 ്വ 00BC	-67 ്ര 00BD	-66 ക്ക 00BE	-65 ക്ല 00BF
-64 ക്ഷ 00C0	-63 ഗ്ഗ 00C1	-62 ഗ്ല 00C2	-61 ങ്ക 00C3	-60 ങ്ങ 00C4	-59 ച്ച 00C5	-58 ഞ്ച 00C6	-57 ഞ്ഞ 00C7	-56 ട്ട 00C8	-55 ൺ 00C9
-54 ണ്ട 00CA	-53 ണ്ണ 00CB	-52 ത്ത 00CC	-51 ദ്ദ 00CD	-50 ദ്ധ 00CE	-49 ൻ 00CF	-48 ന്ത 00D0	-47 ന്ദ 00D1	-46 ന്ന 00D2	-45 ന്മ 00D3
-44 പ്പ 00D4	-43 പ്ല 00D5	-42 ബ്ബ 00D6	-41 ബ്ല 00D7	-40 മ്പ 00D8	-39 മ്മ 00D9	-38 മ്ല 00DA	-37 യ്യ 00DB	-36 ർ 00DC	-35 റ്റ 00DD
-34 ൽ 00DE	-33 ല്ല 00DF	-32 ൾ 00E0	-31 ള്ള 00E1	-30 വ്വ 00E2	-29 ശ്ല 00E3	-28 ശ്ശ 00E4	-27 സ്ല 00E5	-26 സ്സ 00E6	-25 ഹ്ല 00E7
-24 സ്റ്റ 00E8	-23 ഡ്ഡ 00E9	-22 ക്ട 00EA	-21 ബ്ധ 00EB	-20 ബ്ദ 00EC	-19 ച്ഛ 00ED	-18 ജ്ജ 00EE	-17 ണ്മ 00EF	-16 സ്ഥ 00F0	-15 ന്ഥ 00F1
-14 ജ്ഞ 00F2	-13 ത്ഭ 00F3	-12 ധ്മ 00F4	-11 ശ്ച 00F5	-10 ണ്ഡ 00F6	-9 ത്മ 00F7	-8 ക്ത 00F8	-7 ഗ്ന 00F9	-6 ന്റ 00FA	-5 ഷ്ട 00FB
-4 ച 00FC	-3 ഫ 00FD								ISCII CHAR Unicode

APPENDIX C

A View of Lexicon used in MaQAS



മനുഷ്യൻ
ശരീരം
ഇൽ
ഏറ്റവും
വലിയ
ഗ്രന്ഥി
കരൾ
ആണ്
ഹെപ്പറ്റൈറ്റിസ്
വൈറസ്
കൾ
ഉടെ
ആഹാരം
ദിവസം
തല
കഠിനം
ക്ഷീണം

Type	mean1	mean2	mean3	mean4
N	SL	C		
N	SL	N		
S			C5	
Adv				
Adj	AdjN			
N	SL	N		
N	SL	N		
AuxV	T2			
PN				
N	SL	C		
S	PL			
S			C6	
N		N		
N	SL	N		
N	SL	N		
Adj	AdjQ			
N		N		

Type field	-	Noun/verb/Adj/Adv/PSP/Suffix
Mean 1	-	SL/PL/T1/T2/T3
Mean 2	-	M/F/N/C
Mean 3	-	C1-C6 / R1-R2
Mean 4	-	0 or 1(Human or non-human)
SL	-	Singular
PL	-	Plural
T1	-	Past tense
T2	-	Present Tense
T3	-	Future Tense
M	-	Masculine Gender
F	-	Feminine Gender
C	-	Common Gender
N	-	Neuter Gender
C1-C6	-	Case suffixes
R1-R2	-	Active/Passive voice

APPENDIX D

Performance of POS Tagger

POS Tag	Precision (%)	Recall (%)	F-Score (%)
NOUN	87.3	94.6	90.8
PN	98.0	97.1	97.6
RN	90.2	95.2	92.6
ACC	95.6	97.0	96.3
DAT	92.3	96.2	94.2
GEN	91.0	95.5	93.2
LOC	90.5	94.3	92.4
SOC	89.1	95.0	92.0
INST	95.5	91.4	93.4
OBJ	93.3	89.7	91.5
RES	90.6	88.5	89.5
PSP1	94.0	95.8	94.9
PSP2	95.6	93.2	94.4
PSP3	94.1	94.0	94.1
PSP4	95.3	93.1	94.2
PSP5	96.4	95.4	95.9
PSP6	92.3	97.9	95.0
PSP7	94.5	92.1	93.3
PSP8	96.6	93.6	95.1
PSP9	95.2	92.3	93.7
PSP10	94.4	95.2	94.8
PSP11	94.0	91.9	92.9
VERB	86.5	94.2	90.2
AdjP	84.9	91.4	88.0
Adv	83.7	89.6	86.6
AdvP	92.3	96.7	94.5
AuxV	89.2	94.3	91.7
AdvT	89.7	95.3	92.4
AdvPl	91.4	97.3	94.3
AdvPr	94.1	92.9	93.5
AdvR	90.5	95.5	92.9
AdvSe	91.9	93.0	92.5
AdvCo	92.4	95.1	93.7
Adv-S	93.2	96.2	94.7

AdvC	90.3	95.7	92.9
AdjQl	94.6	93.6	94.1
AdjQn	90.5	92.2	91.3
AdjD	89.6	90.7	90.2
Adjn	90.3	96.0	93.1
AdjI	91.5	92.0	91.8
AdjE	88.4	90.4	89.4
SourceP	95.6	93.4	94.5
DestP	92.5	96.7	94.6
LikeP	87.4	93.0	90.1
ListP	88.7	91.5	90.1
TimeP	90.5	93.0	91.7
ThruP	93.1	92.5	92.8
Sym	96.6	94.7	95.6
CN	94.3	96.2	95.2
ON	87.4	93.6	90.4
INT1	96.0	97.4	96.7
INT2	93.4	94.3	93.8
Average Value	92.0	94.0	93.0

APPENDIX E

Patterns for Phrase Identification

Sl. No	Phrase	POS Patterns
1	NP	NOUN/ PN/ RN
2	VP	VERB/AuxV
3	NP-Acc	ACC
4	NP-Dat	DAT
5	NP-Gen	GEN
6	NP-Loc	LOC/NOUN PSP4/NOUN PSP2 NOUN PSP3/ SourceP NOUN PSP3/SourceP DestP
7	NP-Soc	SOC
8	NP-Obj	OBJ/ACC PSP6
9	NP-Inst	INST/ GEN INST/NOUN PSP1/PN PSP1/RN PSP1
10	NP-Res	RES/NOUN PSP7/PN PSP7/RN PSP7
11	AdjNP	AdjP NOUN/AdjP PN/AdjP RN
12	AdVP	AdvP / Adv AdvP
13	AdvpT	AdvT / Adv AdvT
14	AdvpPr	AdvPr / Adv AdvPr
15	AdvpR	AdvR / Adv AdvR
16	AdvpP	AdvPl / Adv AdvPl
17	AdvpC	AdvC / Adv AdvC
18	AdvpRe	AdvRe / Adv AdvRe
19	AdvpS	AdvS / Adv AdvS
20	AdvpCo	AdvCo / Adv AdvCo
21	AdjpQl	AdjQl / Adj AdvQl
22	AdjpQn	AdjQn / Adj AdjQn
23	AdjpN	AdjN / Adj AdjN
24	AdjpE	AdjE / Adj AdjE
25	Adjpl	Adjl / Adj Adjl
26	AdjpD	AdjD / Adj AdjD
26	SourcePP	NOUN PSP2/PN PSP 2/RN PSP 2/SourceP
27	ListPP	NOUN PSP11 NOUN PSP3/NOUN PSP2 DestP
29	LikePP	ACC PSP 9
30	ThruPP	LOC PSP5
31	DestPP	NOUN PSP3/PN PSP3/GEN PSP10/DestP
32	TimePP	RN PSP8/AdjN SourceP PSP8

APPENDIX F

Performance Evaluation of the AIS-based Phrase chunker

Chunk	Precision (%)	Recall (%)	F-Score (%)
NP	93.5	92.6	93.0
VP	96.7	97.1	96.9
NP-Obj	92.0	92.2	92.1
NP-Dat	96.4	97.0	96.7
NP-Acc	94.3	95.2	94.7
NP-Gen	91.0	90.5	90.7
NP-Loc	87.9	88.3	88.1
NP-Soc	95.0	95.0	95.0
NP-Inst	92.8	91.4	92.1
NP-Res	90.7	89.7	90.2
AdjNP	89.7	88.5	89.1
AdvP	93.5	91.8	92.6
AdvpPr	86.3	85.2	85.7
AdvpR	93.4	92.0	92.7
AdvpP	94.6	93.1	93.8
AdvpC	88.3	85.4	86.8
AdvpS	88.0	87.9	88.0
AdvpT	87.2	86.1	86.6
AdvpRe	92.8	93.6	93.2
AdvpCo	91.3	92.3	91.7
AdjpQ	91.4	89.5	90.4
AdjpQn	89.2	88.9	89.0
AdjpN	95.1	94.2	94.6
AdjpE	92.3	91.4	91.8
Adjpl	87.9	86.6	87.2
AdjpD	87.4	86.0	86.7
ListPP	88.6	87.3	87.9
ThruPP	85.5	86.7	86.0
LikePP	87.8	88.1	87.9
TimePP	94.1	94.7	94.4
SourcePP	92.7	91.1	91.9
DestPP	93.2	89.8	91.5
Average Value	91.3	90.6	90.9

APPENDIX G

A Sample Malayalam Document

കോളറ വിബ്രിയോ എന്നറിയപ്പെടുന്ന അണുക്കളാണ് കോളറയുണ്ടാക്കുന്നത്. ചെറുകുടലിനെയാണ് ഈ അണൂക്കൾ ബാധിക്കുന്നത്.ഈ അണുക്കൾ അതിശക്തമായ വിഷമയം ഉണ്ടാക്കുന്നു. ചെറുകുടലിലെ കോശങ്ങളെ ആക്രമിക്കുന്നതിന്റെ ഫലമായി ശരീരത്തിനാവശ്യമായ സോഡിയത്തിന്റെ ആഗിരണം തടസ്സപ്പെടുകയും ജലവും ക്ലോറൈഡ് അംശവും നഷ്ടപ്പെടുകയും ചെയ്യുന്നു.ഈ രോഗത്തിന്റെ അണുക്കൾ ശരീരത്തിലെത്തി മണിക്കുറുകൾക്കുള്ളിൽ രോഗം പ്രത്യക്ഷപ്പെടുന്നു.

രോഗിയുടെ വിസർജ്യവസ്തുക്കളിൽനിന്നും പ്രാണികൾ വഴി ഭക്ഷണത്തിലും വെള്ളത്തിലുമെത്തുന്ന അണുക്കളാണ് രോഗമുണ്ടാക്കുന്നത്.മലീമസമായ ചുറ്റുപാടുകളിൽ തിങ്ങിപ്പാർ ക്കുന്ന കുടുംബങ്ങളിലെ കുട്ടികളിലും മുതിർന്നവരിലും ഈ അസുഖം വളരെ പെട്ടെന്ന് പടർന്നുപിടിക്കാറുണ്ട്.

വളരെ പെട്ടെന്നു ണ്ടാകുന്ന വയറിളക്കവും ഛർദ്ദിയുമാണ് ഇതിന്റെ ലക്ഷണങ്ങൾ. വയറുവേദന, മൂത്രതടസ്സം, മസിലുകളിലെ കൊളുത്തിപ്പിടിക്കൽ എന്നിവയുമുണ്ടാകും. തുടക്കത്തിൽ മലത്തിന് മഞ്ഞനിറം കാണുന്നു. ഛർദ്ദിയുടെ കൂടെത്തന്നെ വയറിളക്കവും കാണുന്നു. പീന്നിട് മലവും ഛർദ്ദിയും അരിക്കാടിപോലെയായിത്തീരുന്നു.

ഛർദ്ദിയുടെ കൂടെ ബൈൽ കാണുകയില്ല. ഛർദ്ദിച്ചും അതിസരിച്ചും ക്രമേണ ശരീരത്തിലെ ജലാംശം നഷ്ടപ്പെടുന്നതു കൊണ്ട് രോഗി അപകടനിലയിൽ എത്താറുണ്ട്. തണുത്ത് വരണ്ട തൊലി, കുഴിഞ്ഞു താണ കണ്ണുകൾ, ഉണങ്ങിവരണ്ട നാക്ക്, ചുണ്ടുകൾ, പനി എന്നിവ അപകടകരമായ ലക്ഷണങ്ങളാണ്. രക്തമർദ്ദം കുറഞ്ഞ് പൾസ് കിട്ടാതിരിക്കുക, കിതപ്പ് എന്നീ ലക്ഷണങ്ങളും അപകടകരങ്ങളാണ്.

കോളറയെതുടർന്ന് കിഡ്നിയുടെ പ്രവർത്തനം നിലച്ച് യുറീമിയ, രക്തചംക്രമണം നിലക്കുക, ഗ്രാൻ ഗ്രീൻ, നിമോണിയ വയർ, ഗാൾബ്ലാഡർ, കണ്ണുകൾ, പരോട്ടിഡ് ഗ്രന്ഥി എന്നിവയിലെ അണുബാധ, ശക്തമായ പനി എന്നിവ ഉണ്ടാകാം.

ഒ ആർ എസ് ലായനിയും കരിക്കിൻ വെള്ളം, പഞ്ചസാരയും ഉപ്പും ചേർത്ത വെള്ളം എന്നിവയെല്ലാം ഉപയോഗിക്കണം. ഖരരൂപത്തിലുള്ള ഭക്ഷണം ഒഴിവാക്കി ലായനികൾതന്നെ കഴിക്കുന്നതാണ് ആദ്യദിവസങ്ങളിൽ ഉചിതം. മലീമസമായ ചുറ്റുപാടുകളിൽനിന്ന് രോഗിയെ മാറ്റിപ്പാർപ്പിക്കണം. പൂർണവിശ്രമം ആവശ്യമാണ്.

Transliterated document

kOLaRa vibriyO ennaRiyappeTunna aNukkaLaaN~ kOLaRayuNTaakkunnath~. cheRukuTalineyaaN~ ee aNookkaL baadhikkunnath~.ee aNukkaL athiSakthamaaya vishamayam uNTaakkunnu.cheRukuTalile kOSangngaLe aakramikkunnathinte phalamaayi SareerathThinaavaSyamaaya sODiyathThinte aagiraNam thaTassappeTukayum jalavum kLORaiD~ am_Savum nashTappeTukayum cheyyunnu.ee rOgathThinte aNukkaL SareerathThilethThi maNikkuRukaL kkuLLil rOgam prathyakshappeTunnu.

rOgiyuTe visar_jyavasthukkaLil_ninnu praaNikaL vazhi bhakshaNathThilum veLLathThilumethThunna aNukkaLaaN~ rOgamuNTaakkunnath~.maleemasamaaya chutupaaTukaLil thingngippaar kkunna kuTum_bangngaLile kuTTikaLilum muthir_nnavarilum ee asukham vaLare peTTenn~ paTar nnu piTikkaaRuNT~.

vaLare peTTennuNTaakunna vayaRiLakkavum chhar_ddiyumaaN~ ithinte lakshaNangngaL. vayaRuvEdana, moothrathaTassam, masilukaLile koLuthThippiTikkal ennivayumuNTaakum.thuTakkathThil malathThin~ manjnjaniRam kaaNunnu. chhar_ddiyuTe kooTethThanne vayaRiLakkavum kaaNunnu. peenniT~ malavum chhar_ddiyum arikkaaTipOleyaayithTheerunnu.

chhar_ddiyuTe kooTe bail kaaNukayilla. chhar_ddichchum athisarichchum kramENa SareerathThile jalaam_Sam nashTappeTunnathu koNT~ rOgi apakaTanilayil ethThaaRuNT~. thaNuthTh~ varaNTa tholi, kuzhinjnju thaaNa kaNNukaL, uNangngivaraNTa naakk~, chuNTukaL, pani enniva apakaTakaramaaya lakshaNangngaLaaN~. rakthamar_ddam kuRanjnj~ paLs~ kiTTaathirikkuka, kithapp~ ennee lakshaNangngaLum apakaTakarangngaLaaN~.

kOLaRayethuTarnn~ kiDniyuTe pravar_thThanam nilachch~ yuReemiya, rakthacham_kramaNam nilaykkuka, graan green, nyumONiya vayar, gaaL_bLaaDar, kaNNukaL, parOTTiD~ granthhi ennivayile aNubaadha, Sakthamaaya pani enniva uNTaakaam.

o aar es~ laayaniyum karikkin veLLam, panjchasaarayum uppum chEr_thTha veLLam ennivayellaam upayOgikkaNam. khararoopathThiluLLa bhakshaNam ozhivaakki laayanikaL_thanne kazhikkunnathaaN~ aadyadivasangngaLil uchitham. maleemasamaaya chutupaaTukaLil_ninnu rOgiye maatippaar_ppikkaNam. poor_NaviSramam aavaSyamaaN~

APPENDIX H
LIST OF SAMPLE QUESTIONS

മഞ്ഞപ്പിത്തം എങ്ങനെയാണ് ഉണ്ടാകുന്നത്?

(manjnjappiththam engnganeyaaN~ uNTaakuth~ ?)

എപ്പോഴാണ് മഞ്ഞപ്പിത്തം പകരുന്നത് ?

(eppOzhaaN~ manjnjappiththam pakaruth~ ?)

എന്താണ് മഞ്ഞപ്പിത്തത്തിന്റെ കാരണങ്ങൾ ?

(enthaaN~ manjnjappiththaththinte kaaraNangngaL ?)

എതു മരുന്നാണ് മഞ്ഞപ്പിത്തത്തിന് ഉപയോഗിക്കുന്നത് ?

(ethu marunnaaN~ manjnjappiththaththin~ upayOgikkuth~ ?)

എന്തൊക്കെയാണ് പ്രതിവിധികൾ ?

(enthokkeyaaN~ prathividhikaL ?)

ചികിത്സരീതികൾ ഏതൊക്കെ ?

(chikithsareethikaL Ethokke ?)

എതു പ്രായക്കാരെയാണ് മഞ്ഞപ്പിത്തം ബാധിക്കുന്നത് ?

(ethu praayakkaareyaaN~ manjnjappiththam baadhikkuth~ ?)

ഏതു വൈറസാണ് മഞ്ഞപ്പിത്തം ഉണ്ടാക്കുന്നത് ?

(Ethu vaiRasaaN~ manjnjappiththam uNTaakkuth~ ?)

ഏത്ര ദിവസം കൊണ്ട് മഞ്ഞപ്പിത്തം ഭേദമാകും ?

(Ethra divasam koNT~ manjnjappiththam bhEdamaakum ?)

എങ്ങിനെയാണ് ഈ രോഗം പകരുന്നത് ?

(engngineyaaN~ ee rOgam pakaruth~ ?)

ഏതു ടെസ്റ്റാണ് മഞ്ഞപ്പിത്തം തിരിച്ചറിയാൻ ഉപയോഗിക്കുന്നത് ?

(Ethu TestaaN~ manjnjappiththam thirichchaRiyaan upayOgikkuth~ ?)

എന്തഹാരം കഴിക്കാം ?(enthahaaram kazhikkaam ?)

എന്തു മുൻകരുതലാണ് വേണ്ടത് ?

enthu mungkaruthalaaN~ vENTath~ ?

എന്താണ് ലക്ഷണങ്ങൾ ?

enthaaN~ lakshaNangngaL ?

ഏതു സമയത്താണ് അസുഖം പകരുന്നത് ?

Ethu samayaththaaN~ asukham pakaruth~ ?

മഞ്ഞപ്പിത്തം എന്നാൽ എന്താണ് ?

manjnjappiththam ennaal enthaaN~ ?

ആരാണ് മഞ്ഞപ്പിത്തത്തിന്റെ മരുന്ന് കണ്ടുപ്പിടിച്ചത് ?

aaraaN~ manjnjappiththaththinte marunn~ kaNTuppiTichchath~ ?

പ്രതിരോധകുത്തിവെയ്പുണ്ടോ ?

prathirOdhakkuththiveypuNTO ?

എങ്ങിനെയാണ് ഈ രോഗം പകരുന്നത് ?

engngineyaaN~ ee rOgam pakaruth~ ?

എതു അവയവത്തെയാണ് ഈ രോഗം ബാധിക്കുന്നത് ?

ethu avayavaththeyaaN~ ee rOgam baadhikkuth~ ?

എതു കമ്പനിയാണ് ഇതിന് മരുന്ന് ഉണ്ടാക്കുന്നത് ?

ethu kampaniyaaN~ ithin~ marunn~ uNTaakkuth~ ?

എതു ജീവിയാണ് ഈ രോഗം പകർത്തുന്നത് ?

ethu jeeviyaaN~ ee rOgam pakar_ththuth~ ?

ആരാണ് മഞ്ഞപ്പിത്തം കണ്ടുപിടിച്ചത് ?

aaraaN~ manjnjappiththam kaNTupiTichchath~ ?

എന്താണ് മഞ്ഞനിറത്തിനു കാരണം ?

enthaaN~ manjnjaniRaththinu kaaraNam ?

മഞ്ഞപ്പിത്തത്തിന്റെ അനന്തഫലങ്ങൾ ഏവ ?

manjnjappiththaththinte ananthaphalangngaL Eva ?

എങ്ങിനെയാണ് മഞ്ഞപ്പിത്തം തിരിച്ചറിയുക ?

engngineyaaN~ manjnjappiththam thirichchaRiyuka ?

ഏതു രാജ്യക്കാരാണ് ഈ രോഗം കണ്ടുപിടിച്ചത് ?

Ethu raajyakkaaraaN~ ee rOgam kaNTupiTichchath~ ?

എപ്പോഴാണ് കുത്തിവെയ്പ് എടുക്കേണ്ടത് ?

eppOzhaaN~ kuththiveyp~ eTukkENTath~ ?

കരളിന്റെ ഉപയോഗം എന്താണ് ?

karaLinte upayOgam enthaaN~ ?

എവിടെയാണ് കരൾ സ്ഥിതിചെയ്യുന്നത് ?

eviTeyaaN~ karaL sthhithicheyyuth~ ?

ഏതു വഴിയാണ് ഇതു പകരുന്നത് ?

Ethu vazhiyaaN~ ithu pakaruth~ ?

ആഹാരത്തിൽ എന്തൊക്കെ ശ്രദ്ധിക്കണം?

aahaaraththil enthokke SraddhikkaNam?

ഏതു കമ്പനിയാണ് മഞ്ഞപ്പിത്തത്തിന്റെ മരുന്ന് ഉണ്ടാക്കുന്നത്

Ethu kampaniyaaN~ manjnjappiththaththinte marunn~ uNTaakkuth~

മഞ്ഞപ്പിത്തം തിരിച്ചറിയാനുള്ള ടെസ്റ്റ് ഏതാണ്

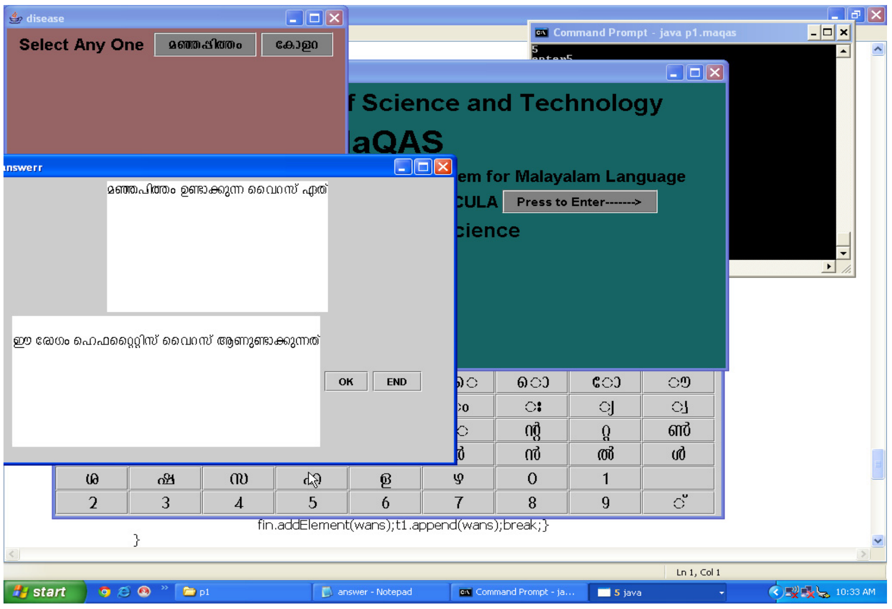
manjnjappiththam thirichchaRiyaanuLLa Test~ EthaaN~

ഏതു സമയത്താണ് മഞ്ഞപ്പിത്തം പകരുന്നത്

Ethu samayaththaaN~ manjnjappiththam pakaruth~

Appendix I

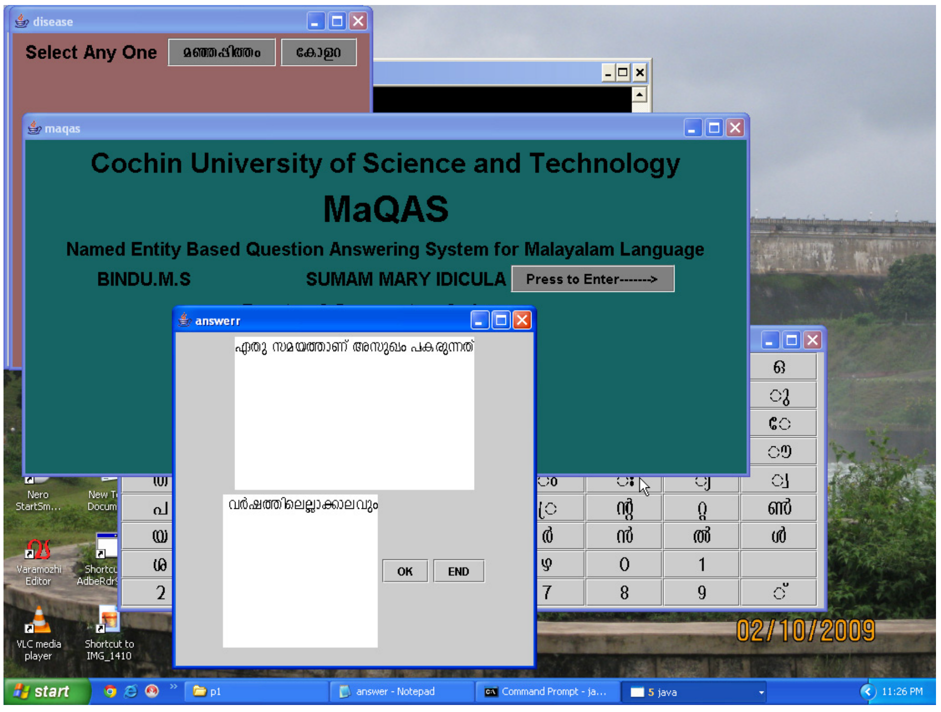
Screen shots showing Output of MaQAS



മഞ്ഞപിത്തം ഉണ്ടാക്കുന്ന വൈറസ് ഏത്

ഈ രോഗം ഹെഫറ്റൈറ്റിസ് വൈറസ് ആണുണ്ടാക്കുന്നത്

```
fin.addElement(wans);t1.append(wans);break;}
}
```



Cochin University of Science and Technology

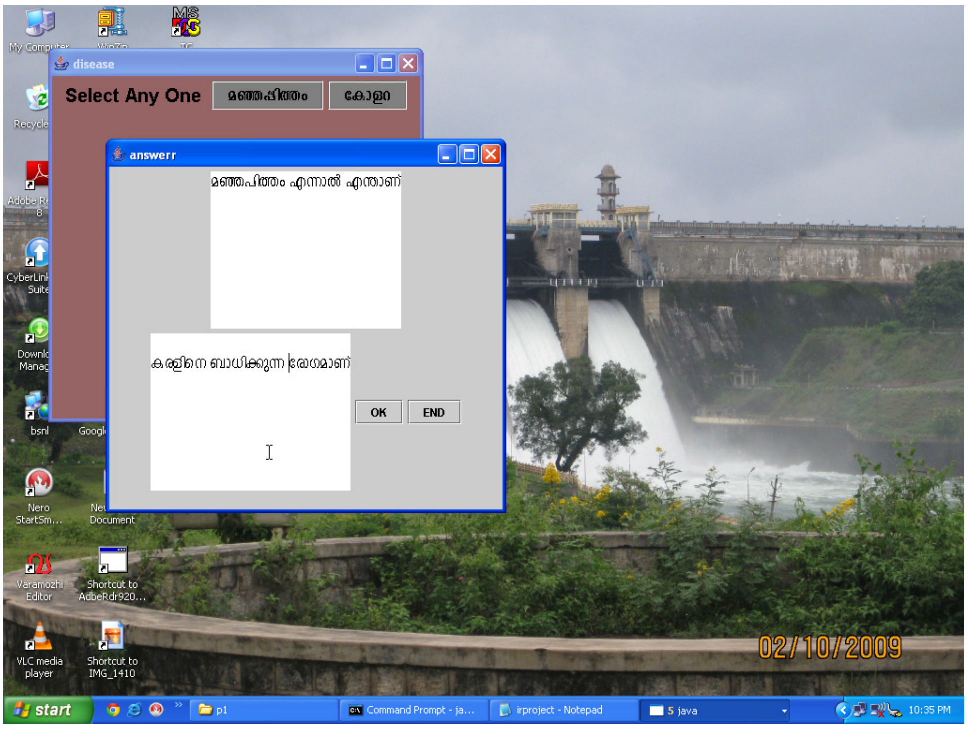
MaQAS

Named Entity Based Question Answering System for Malayalam Language

BINDU.M.S SUMAM MARY IDICULA Press to Enter------->

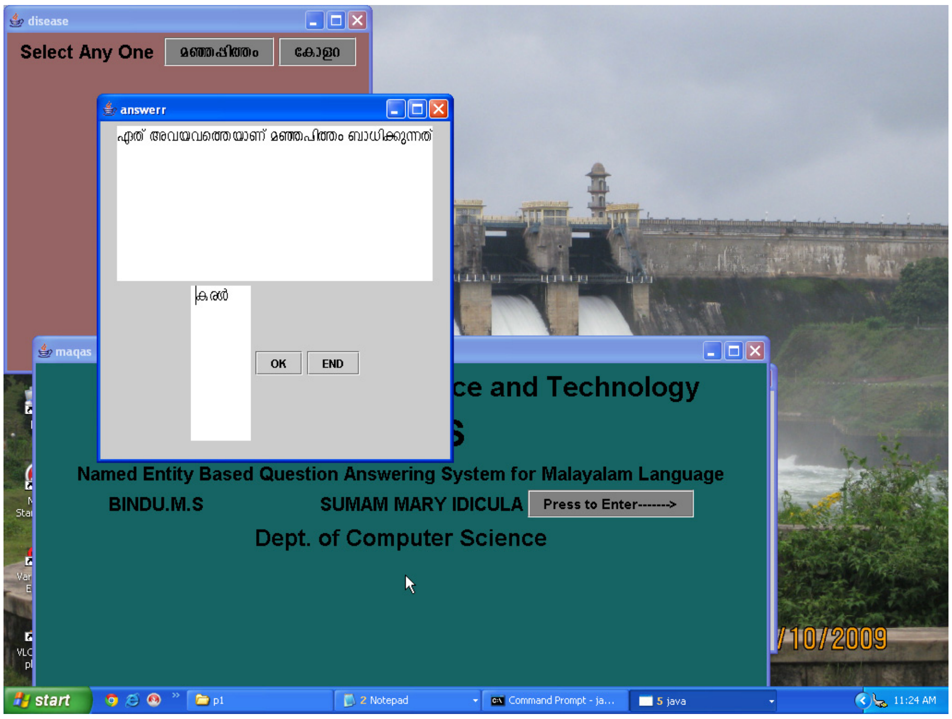
ഏതു സമയത്താണ് അസുഖം പകരുന്നത്

വർഷത്തിലെല്ലാക്കാലവും



മഞ്ഞപിത്തം എന്നാൽ എന്താണ്

കരളിനെ ബാധിക്കുന്ന രോഗമാണ്



എത് അവയവത്തെയാണ് മഞ്ഞപിത്തം ബാധിക്കുന്നത്

കരൾ

ce and Technology

Named Entity Based Question Answering System for Malayalam Language

BINDU.M.S SUMAM MARY IDICULA

Dept. of Computer Science