JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-2, Feb 2019 E-ISSN: 2347-2693

Natural Language Interface for Querying Hardware and Software Configuration of a Local Area Network

Poornima G. Naik[1*], Santosh G. Patil[2], Girish R. Naik[3]

[1] Department of Computer Studies, Chh Shahu Institute of Business Education and Research, Kolhapur, India
[2]Chh Shahu Institute of Business Education and Research, Kolhapur, India
[3] Production Department, KIT's College of Engineering, Gokul Shirgaon, Kolhapur, India

**Corresponding Author: pgnaik@siberindia.edu.in, Tel.: +91-90499-61447*

DOI: https://doi.org/10.26438/ijcse/v7i2.949963 | Available online at: www.ijcseonline.org

Accepted: 24/Feb/2019, Published: 28/Feb/2019

***Abstract*—** In this digital era, it is virtually impossible to find even a single business premise that does not host a local area network in place. Knowledge of all softwares installed and all hardwares connected to LAN is extremely essential to a lab technician. Because on many occasions, any minor error in LAN requires the lab technician to check virtually all computers connected to LAN manually which is extremely time consuming and a lengthy process. In majority of cases, only few machines in the network contain softwares owning to the limited user licenses. Some machines contain softwares which are occasionally used. Locating such an information in a large local area network is an extremely time consuming task. The aim of current research is to design and develop an interface for LAN which accepts the queries pertaining to hardware and software installed in LAN in natural language (NL) which is parsed using NLP parser designed and developed by the authors for the domain under consideration for facilitating the human-machine interaction. For this a set of alphabets, a set of tokens, and a context free grammar is designed. The parts-of-speech tags are numerically encoded and a pattern is generated for a syntactically valid statements. A set of rules are generated by observing the statements which are syntactically valid. The research is carried out in two phases. In the first phase, the query entered by the user in a natural language is parsed using NLP parser. In the second phase of the research, the queries which are found to be syntactically correct in phase I are further evaluated by mapping them to the corresponding prolog queries which interfaces with the knowledge base for retrieving the desired set of information. Finally, the model is implemented in a hypothetical educational institute to evaluate the operational feasibility of the model.

***Keywords*—**Hardware Query Language, Knowledge base, Parse Tree, Parts-of-Speech Tokens, Prolog, Semantics.

I. INTRODUCTION

In many educational institutes' labs, LAN contains different softwares installed on different machines and also different hardwares such as printers, scanners etc. connected to specific machines, which poses a problem for locating machines where rarely used softwares are installed. The challenge continuously increases as new softwares are installed on different machines in a network and as the size of network increases beyond control. Further, technician is required to install newly acquired softwares on virtually every machine. Also during practical examination phase, the lab technician has to visit virtually each and every machine in the network to check whether the required softwares are installed on them or not. The time for the entire process is directly proportional to the size of the network. Lot of time can be saved if the whole process can be automated and centralized. Finding errors and searching any particular software or hardware in a large networked environment is extremely difficult task and also it requires sound knowledge of networking and technical queries to be fired against persisted data. In LAN it is extremely time consuming to find exact problem manually.

To address this issues the current research proposes a model for execution of human queries. This is achieved by storing the information pertaining to the machine, hardware and software information in a persistent Relational DataBase Management System (RDBMS) which is dynamic and is instantly updated as the new hardware is connected to LAN or a new software is installed. The end user instead of querying the database directly will use the natural language, termed as Hardware Query Language (HQL) and Software Query Language (SoQL) designed by the author, which is interfaced with RDBMS using prolog. To implement HQL and SoQL, a finite set of symbols, words and language rules are defined, which constitute HQL and SoQL grammar. The parse tree is constructed based on the grammar specified. The NLP query is parsed using NLP parser designed by the author and the queries which are successfully parsed are evaluated by mapping them to the corresponding prolog query using Java interface to Prolog (JPL). Prolog rules are stored in the corresponding prolog knowledge bases. NLP offers most flexible way to implement grammar which can be readily extended with least efforts and as such offers an efficient way of implementing rules in dynamically changing scenarios.

A. Introduction to Natural Language Processing and Prolog

Natural language processing (NLP) is a field that combines computer science technologies such as Artificial Intelligence, Statistical inference and Machine Learning with applied linguistics, concerned with the interactions between computers and natural human languages. Hence NLP is related to the area of human–computer interaction. It allows computer-supported understanding and processing of information expressed in human language for specific tasks, including machine translation, interactive dialog systems, opinion mining, etc. Many challenges in NLP involve natural language understanding by enabling computers to derive meaning from human or natural language input, and others involve natural language generation.

B. Introduction to Prolog

Prolog, Programming in Logic, is a special type of declarative type programming in which the various program elements and constructs are expressed in predicate logic. A program consists of mainly a number of declarations representing relevant facts and rules concerning the problem domain. The solution to be discovered is also expressed as a question to be answered or to be more precise the goal to be achieved based on the resolution method suggested by Robinson consisting of matching goals with facts and rules. A prolog program consists of a finite sequence of facts, rules and a query or goal statement. Prolog database or knowledge base consists of facts and rules. Prolog inferencing system mainly consists of three mechanisms viz.,

i. Backtracking
ii. Unification and
iii. Resolution.

Two interesting features of logic programming are non-determinism and backtracking. A non-deterministic program may find a number of solutions, rather than just one, to a given problem. Backtracking mechanism allows exploration of potentially alternative directions for solutions, when some direction currently being investigated, fails to find an appropriate solution.

The organization of the rest of the paper is as follows. Section II focuses on current state of research in the field of NLP. Section III presents a theoretical model for the conceptualization of NLP. Parse trees for querying hardware and software information of LAN are presented which is further employed in numerical encoding of parts-of-speech tags and pattern generation. Valid patterns are generated and rules are deduced for valid patterns. Section IV presents the implementation of the model presented in Section III and discusses few experimental results. Finally, Section V presents conclusions drawn from research and scope for future research in this area.

II. RELATED WORK

In recent years, automation plays a key role in virtually every field and owing to the dramatic decrease in hardware and software costs setting up LAN has become affordable even by the medium sized organization. Automation has a dual advantage, on one hand it helpful in reducing the time required towards LAN management and on the second hand it improves throughput by making accurate information available at fingertip.

The literature review focuses on application of NLP in different areas and automation procedures adopted by different researchers. Also, wireless network are targets to understand scope for future extension of research in this area. As a part of the current research the following papers were reviewed.

1. Design and Development of Network Monitoring and Controlling Tool for Domain Controller @Department of Computer Studies, CISBER using RMI Technology-A Case Study .

a) Dr.P.G.Naik b) Mr. M. B. Patil

In this research, the researchers provide information regarding how to manage and control the network by analyzing the collected information. Controlling is the task of taking specific actions against individual network and system components. Network monitoring is important to find network trends, network bottlenecks and locate network problems quickly.

2. Analysis of the Combination of Natural Language Processing and Search Engine Technology.

a) Wei Wang b) Huankai Shi

This research paper introduces current situation and process of natural language processing (NLP) and the effect of natural language processing in search engine. In terms of concepts and classifications, traditional search engine has many deficiencies which can not completely satisfy the requirements of users. The key technology of intelligent search engine and the development trend of the combination of natural language processing and search engine technology in the future are discussed.

3. Survivable LANs for distributed control systems.
a) JE Cooling

In this paper author discusses the need for, and methods of achieving, survivability in distributed control system networks. It is applicable to areas such as avionics, marine systems and industrial plants. Basic survival strategies are discussed in the context of specific network topologies, with emphasis on system design aspects. The strengths and weaknesses of the various approaches are discussed, together with the requirements and constraints of practical systems. Based on these, a general template for a survivable LAN is defined, accompanied by a set of recommendations for implementing specific survivability features.

4. A Subcategory-based Parser Directed to Generating Representations for Text Understanding
a) Yukoko Sasaki Alam

This paper describes a parser which is directed to generating representations for text understanding. For the purpose of reducing the proliferation of unwanted parse trees, and collecting information necessary for generating the semantic representations, the parser uses rules based on phrasal and lexical subcategories. These designs alleviate parsing problems such as PP attachment and coordination attachment, while capable of displaying the dependency of various types of phrases and clauses, thus facilitating the writing of grammar.

5. Annotating Topic Development in Information Seeking Queries.
a) Marta Andersson b) Adnan Ozturely c) Silvia Pareti

This paper contributes to the limited body of empirical research in the domain of discourse structure of information seeking queries. Researchers have described the development of an annotation schema for coding topic development in information seeking queries and the initial observations from a pilot sample of query sessions. The main idea that researchers explore, is the relationship between constant and variable discourse entities and their role in tracking changes in the topic progression. Researchers argue that the topicalized entities remain stable across development of the discourse and can be identified by a simple mechanism where anaphora resolution is a precursor. Researchers also claim that a corpus annotated in this framework can be used as training data for dialogue management and computational semantics systems. The main goal pursued there is to devise an annotation methodology that can capture the discourse structure in a set of successive queries where each information is seeking in structure. Researchers believe the methodology and that their research can serve well for preparing linguistic resources that can be used for training computational semantic applications, such as topic detection systems.

6. A Decomposable Attention Model for Natural Language Inference.
a) Ankur P. Parikh b)Dipanjan Das

In this paper, the researchers propose a simple neural architecture for natural language inference. The author's approach uses attention to decompose the problem into sub problems that can be solved separately, thus making it trivially parallelizable. On the Stanford Natural Language Inference (SNLI) dataset, the authors obtained state-of-the-art results with almost an order of magnitude fewer parameters than previous work and without relying on any word-order information. Adding intra-sentence attention that takes a minimum amount of order into account yields further improvements. Natural Language Inference (NLI) refers to the problem of determining entailment and contradiction relationships between a premise and a hypothesis. NLI is a central problem in language understanding and recently the large SNLI corpus of 570K sentence pairs was created for this task (Bowman et al., 2015). Researchers present a new model for NLI and leverage this corpus for comparison with prior work.

7. Multilingual Language Processing From Bytes.
a) Dan Gillick b) Cliff Brunk c) Oriol Vinyals d) Amarnag Subramanya

According to the researchers, the long-term trajectory of research in Natural Language Processing has seen the replacement of rules and specific linguistic knowledge with machine learned components. Perhaps the most standardized way that knowledge is still injected into largely statistical systems is through the processing pipeline: Some set of basic language-specific tokens are identified in a first step. Sequences of tokens are segmented into sentences in a second step. The resulting sentences are fed one at a time for syntactic analysis: Part-of-Speech (POS) tagging and parsing. Next, the predicted syntactic structure is typically used as features in semantic analysis, Named Entity Recognition (NER), Semantic Role Labeling etc. While each step of the pipeline now relies more on data and models than on hand-curate rules, the pipeline structure itself encodes one particular understanding of how meaning attaches to raw strings. One motivation for their work is to try removing this structural dependency rather than relying on the intermediate representations invented for specific subtasks.

8. Natural Language Processing (Almost) from Scratch.
a) Ronan Collobert b) Jason Weston c) Leon Bottou d) Michael Karlen

In their paper the researchers propose unified neural network architecture and learning algorithm that can be applied to various natural language processing tasks including part-of-speech tagging, chunking, named entity recognition, and semantic role labeling. This versatility is achieved by trying to avoid task-specific engineering and therefore disregarding a lot of prior knowledge. Instead of exploiting man-made input features carefully optimized for each task, their system learns internal representations on the basis of vast amounts of mostly unlabeled training data. The work is used as a basis for building a freely available tagging system with good performance and minimal computational requirements. A pertinent question that arises is a computer program is that “Will we ever be able to convert a piece of English text into a programmer friendly data structure that describes the meaning of the natural language text?” Unfortunately, no consensus has emerged about the form or the existence of such a data structure. Until such fundamental Artificial Intelligence problems are resolved, computer scientists must settle for the reduced objective of extracting simpler representations that describe limited aspects of the textual information. These simpler representations are often motivated by specific applications (for instance, bag of-words variants for information retrieval), or by the belief that they capture something more general about natural language. They can describe syntactic information (e.g., part-of-speech tagging, chunking, and parsing) or semantic information (e.g., word-sense disambiguation, semantic role labeling, named entity extraction, and anaphora resolution). Text corpora have been manually annotated with such data structures in order to compare the performance of various systems. The availability of standard benchmarks has stimulated research in Natural Language Processing (NLP) and effective systems have been designed for all these tasks. Such systems are often viewed as software components for constructing real-world NLP solutions.

9. Wireless LAN Security : IEEE 802.11g & VPN
a) Abu Taha Zamani b) Javed Ahmad

In this paper researcher begins by introducing the concept of WLAN(wirelessLAN). The introductory section gives brief information on the WLAN components and its architecture. In order to examine the WLAN security threats, researchers have looked at Denial of Service, Spoofing, and Eavesdropping. The researchers then focus on how Wired Equivalent Privacy (WEP) works, which is the IEEE 802.11b/WiFi standard encryption for wireless networking. The discussion of WEP continues by examining its weaknesses, which result in it being much less secured than what was originally intended. Their research also covers the new standards to improve the security of WLAN such as the IEEE 802.1x standard, which comprises of three separated sections: Point-to-Point Protocol (PPP), Extensible Authentication Protocol (EAP) and 802.1x itself. The 802.1x is actually included in 802.11i, a newly proposed standard for key distribution and encryption that play a big role in improving the overall security capabilities of current and future WLAN networks. The 802.11i standard provides two improved encryption algorithms to replace WEP, which are Temporal Key Integrity Protocol (TKIP) and CBC-MAC Protocol (CCMP). Finally the authors have listed down several products that assist users to protect their wireless networks from attacks, with concluding remarks on highlighted issues and solutions.

10. A Detailed Study on Wireless LAN Technologies.
a) Vijay Chandramouli

This research paper explains the evolution of network from early radio and telephone to current devices such as mobile Phones and laptops. Accessing the global network has become the most essential and indispensable as a part of our lifestyle. Wireless communication is an ever-developing field, and the future holds many possibilities in this area. One expectation for the future in this field is that, the devices can be developed to support communication with higher data rates and more security. Research in this area suggests that a dominant means of supporting such communication capabilities will be through the use of Wireless LANs. As the deployment of Wireless LAN increases well around the globe, it is increasingly important for us to understand different technologies and select the most appropriate one. This paper also provides a detailed study of the available wireless LAN technologies and the concerned issues. This is followed by a discussion on evaluating and suggesting a feasible standard for future.

From the review conducted it was concluded that there is lot of scope for research in application of natural language processing for accurate and fast retrieval of LAN information. The following gaps were identified.

1) There is less work on dynamic generation of LAN configuration information.

2) No model is generic and customizable which can address multiple organizations.

3) Review in this field dose not reports application of NLP for retrieving hardware and software related information through human-machine interaction.

The current research attempts to address these gaps.

III. METHODOLOGY

This section presents a theoretical model for the conceptualization of natural language parser for execution of human-like queries written in a natural language.

A. Theoretical Framework

Conceptual Model

A model is conceptualized for the generic environment in the following phases:

Phase 1 : Model for parser implementation and design and implementation of Hardware Query Language (HQL) and Software Query Language (SoQL) at high level.

Phase 2 : Design of grammar for HQL and SoQL.

Phase 3 : Design of an algorithm for NLP parser

Phase 4 : Design of an application framework and layered applications architecture.

B. Methodology For Model Development.

Phase 1 : Model for parser implementation and design and implementation of Hardware Query Language (HQL) and Software Query Language (SOQL) at high level.

For implementation of a generic framework, the application's dynamic data is separated out and stored in various XML configuration files. The reason for the selection of XML format is

- It is platform independent
- Enables storing document structure in Document Type Definition (DTD) and Schema
- Large numbers of XML parsers are available for parsing data stored in XML file.

The structure of various configuration files storing information pertaining to the instantiate activities selected by an end user is shown below along with their DTD.

The structure of the XML file for storing institute-specific is as shown below:

```
<institute>
<name>CSIBER, Kolhapur</name>
  <lab>
    <name>MCA Computer Lab</name>
  </lab>
</institute>
```

The corresponding Document Type Definition (DTD) is as shown below:

```
<?XML version="1.0" ?>
<!ELEMENT institute(name,lab+)>
<!ELEMENT name(#PCDATA)>
<!ELEMENT lab(name)>
<!ELEMENT name(#PCDATA)>
```

C. Model For Hardware Query Language.

To assist human-machine interaction, a query language is designed for enabling quick location of desired hardware and software in a local area network without involving into technicalities of information retrieval. For achieving this a set of alphabets, a set of tokens, and a context free grammar is designed which is parsed using Natural Language Parser (NLP). The query language so developed is referred to as "Hardware Query Language" (HQL) and Software Query Language (SoQL). HQL/SoQL is designed for organizations which renders the end user free from the intricacies involved in SQL syntax for filtering, joins etc., employing the following different approaches.

- Developing a parse tree and using NLP parser for parsing the query.
- Developing a grammar for human interface to database which involves execution of human-like queries for information retrieval from database.
- Defining grammar for human-like queries using Natural Language Processing (NLP).

D. Design of NLP Parser

Syntax and Semantics of Natural Language

Languages are defined by their legal sentences. Sentences are sequences of symbols. The legal sentences are specified by a grammar.

Our first approximation of natural language is a context-free grammar. A context-free grammar is a set of rewrite rules, with non-terminal symbols transforming into a sequence of terminal and non-terminal symbols. A sentence of the language is a sequence of terminal symbols generated by such rewriting rules. For example, the grammar rule

sentence→noun_phrase, verb_phrase

means that a non-terminal symbol sentence can be a noun_phrase followed by a verb_phrase. The symbol "→" means "can be rewritten as." The complete set of grammar for HQL is depicted below.

```
sentence --> noun_phrase, verb_phrase, terminator.

noun_phrase --> proper_noun, adjective.

noun_phrase --> determiner, noun.

verb_phrase --> intransitive_verb.
verb_phrase --> intransitive_verb, preposition, determiner.

verb_phrase --> transitive_verb, helping_verb, noun.

verb_phrase --> transitive_verb, helping_verb, noun, conjunction, noun.

verb_phrase --> transitive_verb, helping_verb, noun, conjunction, noun, conjunction, noun.

verb_phrase --> transitive_verb, helping_verb, noun, conjunction, noun, conjunction, noun, conjunction, noun.

verb_phrase --> transitive_verb, helping_verb, noun, conjunction, noun, conjunction, noun, conjunction, noun, conjunction, noun.

conjunction --> [and].

conjunction --> [or].

preposition --> [in].

terminator --> ['.'].

terminator --> [].
```

Parse Tree for Querying the Hardware and Software Information of a Local Area Network.

Consider the following query in natural language:

Which machine has maximum { ram | harddisk | processor}

{ ram? ' ' : harddisk? Capacity : Speed}

In the above syntax, the sixth token depends on fifth token, if fifth token is "ram", then the sixth token is null. If the fifth token is "harddisk" then the sixth token is "capacity", otherwise the sixth token is "speed" . The conditional expression is utilized in the sixth token.

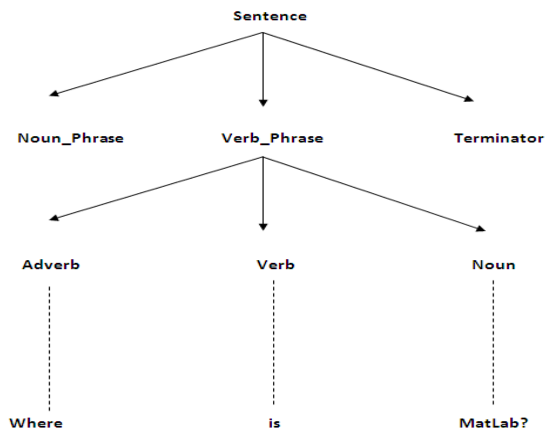
Further, in the above syntax, the following notations are used.

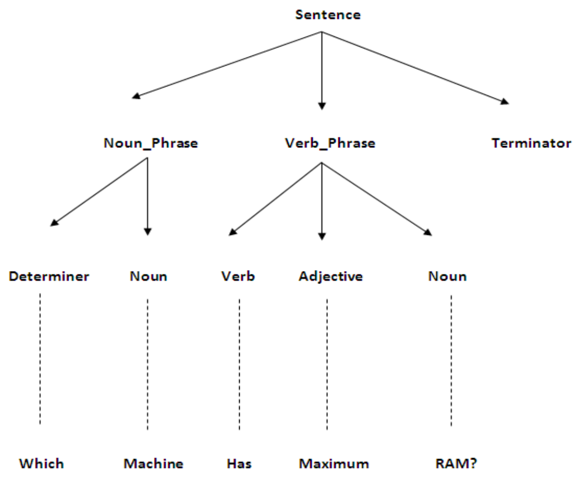
{a|b|…} → One clause from the group of clauses separated by | must be selected.

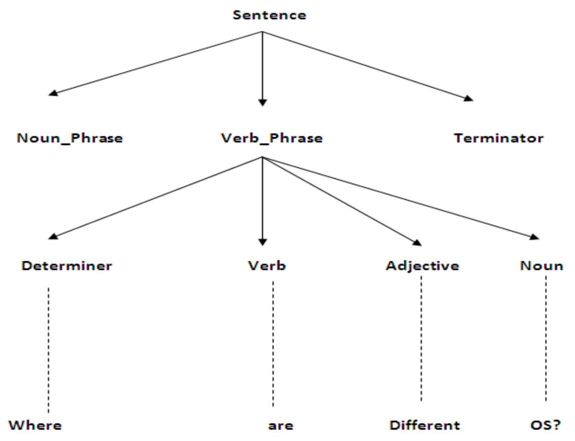
[..] → The clause specified is optional

The above semantics generates the following queries.

The corresponding NLP parse trees are depicted in Figures 1(a)-1(e)







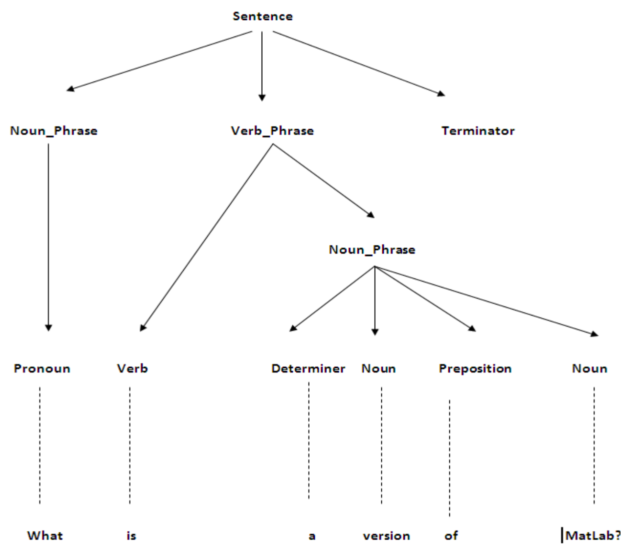


Figure 1(a) - 1(e) NLP Parse Trees

The parts-of-speech tags required for parsing are identified and are numerically encoded as shown below:

E. Numerical Encoding of Pos Tags

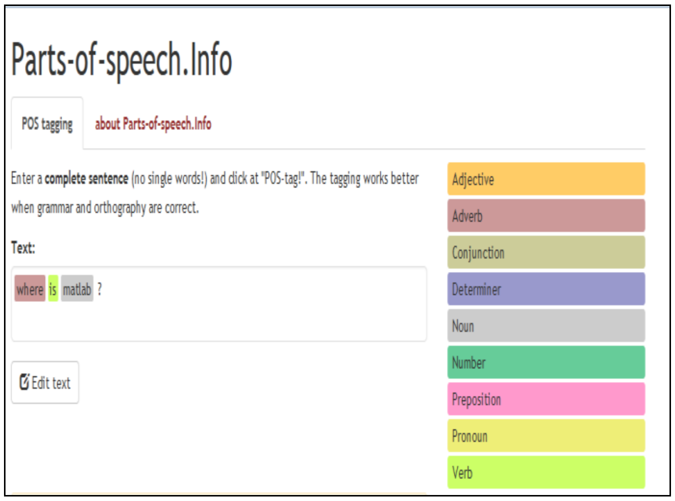
The numerical encoding of POS tags is depicted in Table 1.

Table 1. Numeric Encoding of POS Tags

POS Tag	Numeric Encoding
Verb	1
Adverb	2
Noun	3
Pronoun	4
Adjective	5
Determiner	6
Preposition	7

For identifying POS tags in a given sentence the www.parts-of-speech.info site is employed which uses different color code schemes for identifying POS tags in a given sentence as depicted in Figure 2.

Nine sample sentences are depicted in Figure - which an analyzed for identifying POS tags employed.


Parts-of-speech.Info

POS tagging about Parts-of-speech.Info

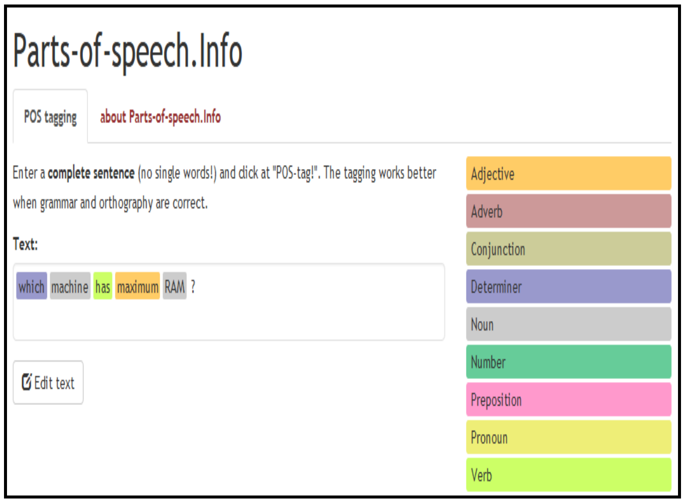
Enter a **complete sentence** (no single words!) and click at "POS-tag!". The tagging works better when grammar and orthography are correct.

Text:

where is matlab ?

Edit text

Adjective
Adverb
Conjunction
Determiner
Noun
Number
Preposition
Pronoun
Verb


Parts-of-speech.Info

POS tagging about Parts-of-speech.Info

Enter a **complete sentence** (no single words!) and click at "POS-tag!". The tagging works better when grammar and orthography are correct.

Text:

which machine has maximum RAM ?

Edit text

Adjective
Adverb
Conjunction
Determiner
Noun
Number
Preposition
Pronoun
Verb

Figure 2. Breaking Natural Language Sentence into Parts-of-speech Tokens

F. Pattern Generation

For generating valid patterns, each sentence is broken into POS tags and employing the numeric encoding depicted in Table 2, the corresponding pattern is generated.

Table depicts the POS tag for each token along with its numeric encoding:

Table 2. Pattern Generation using Numeric Encoding

Sentence 1		
Where	Adverb	2
Is	Verb	1
*	Noun	3
Sentence 2		

Which	Determiner	6
Machine	Noun	3
Has	Verb	1
RAM	Noun	3
Sentence 3		
Which	Determiner	6
Machine	Noun	3
Has	Verb	1
Maximum	Adjective	5
Harddisk	noun	3
Capacity	noun	3
Sentence 4		
Which	Determiner	6
Machine	Noun	3
Has	Verb	1
Maximum	Adjective	5
Processor	Noun	3
Speed	Noun	3
Sentence 5		
What	Determiner	6
Are	verb	1
Different	Adjective	5
OS	noun	3
Sentence 6		
What	Pronoun	4
Is	Verb	1
A	Determiner	6
Version	Noun	3
Of	Preposition	7
*	noun	3
Sentence 7		
What	Determiner	6
Are	Verb	1
Different	Adjective	5
Machine	Noun	3
Brands	Noun	3
Sentence 8		
What	Determiner	6
Are	Verb	1
Different	Adjective	5
Processor	Noun	3
Types	Noun	3
Sentence 9		
Which	Determiner	6
Softwares	Noun	3
Are	Verb	1
Installed	Verb	1
On	Preposition	7
*	Noun	3

Assigning POS tag to each token and numerically encoding each, yields the following patterns.

Sentence No.	Pattern
1	213
2	63153
3	631533
4	631533
5	6153
6	416373
7	61533
8	61533
9	631173

Few observations deduced from the patterns are listed below:

- Minimum length of the pattern is 3
- Maximum length of the pattern is 6.
- First character of a pattern is one of 2, 4 or 6.
- Last character of a pattern is 3.
- Second character of a pattern is one of 1 or 3.
- Third character of a pattern is one of 1, 3, 5 or 6.
- Fourth character of a pattern is one of 1, 3 or 5.
- Fifth character of a pattern is one of 3 or 7.
- Sixth character of a pattern is 3.

Rules deduced by observing the patterns are listed below:

Rule 1 :If the length of a sentence is not in the range 3-6, the sentence is invalid.

Rule 2: If the numeric encoding of a first word is not one of 2, 4 or 6, the sentence is invalid.

Rule 3: If the numeric encoding of a last word is not3, the sentence is invalid.

Rule 4: If the numeric encoding of a second word is not one of 1 or 6, the sentence is invalid.

Rule 5: If the numeric encoding of a third word is not one of 1, 3, 5 or 6, the sentence is invalid.

Rule 6: If the numeric encoding of a fourth word is not one of 1, 3 or 5, the sentence is invalid.

Rule 7: If the numeric encoding of a fifth word is not one of 3 or 7, the sentence is invalid.
Rule 8: If the numeric encoding of a sixth word is not3, the sentence is invalid.

G. Token Categorization

Further the tokens are categorized into different POS tags as depicted in Table 3.

Table 3. Token Categorization in POS Tags

Parts of Speech	Tokens
Verb	is, has, are, installed
Adverb	Where
Noun	*, machine, RAM, harddisk, capacity, processor, speed, OS, version, brands, types, softwares
Pronoun	What
Adjective	maximum, different
Determiner	which, what, a
Preposition	of, on

H. Design of NLP Parser For Parsing HQL/SoQL

The model for the NLP parser and executor is implemented by employing the algorithm given below:

```
/* A function for transforming the sentence entered by the user in Natural Language into a Prolog Query and executing the query. The query checks the grammar of the sentence by retrieving the information stored in Prolog knowledgebase mqlgrammar.pl
Input    – HQL/SOQL query entered by the user in Natural Language
Output - 1 indicates that the query is successfully executed
         0 indicates that the query contains syntactical errors
 */
int function parseQuery(String sentence)
{
  tokens = sentence.split(" ");
   query = "[";
   for (i=0;i<tokens.length;i++)
   {
     query=query + tokens[i];
   }
  q1 = new Query("consult('hqlgrammar.pl')");
  q2 = new Query(query);
  if (q2.query()== 1)
     return 1;
else
     return 0;
}
/*
A function for evaluating the query using Natural Language Interface to Database. MQL query is converted into the corresponding Prolog query and the query on successful execution retrieves the contents of Prolog knowledgebase rules.pl and methodrules.pl
Input   – Successfully parsed MQL query
Output - Results of the query stored in a string array.
*/
char[][] function evaluateQuery(char[] query)
{
   tokens = query.split(" ");
   l = tokens.length();
  if (i == 3)
  {
          q3 = new Query("Software_on_machine(X,tokens[2]).");
          result = q3.allSolutions();
  }
  else if (i == 5 && tokens[2].equals("OS") )
  {
          q3 = new Query("OS_on_machine(X,tokens[4]).");
          result = q3.allSolutions();
  }

  else if (i == 5 && tokens[4].equals("brands"))
  {
          q3 = new Query("machine_by_vendor(X,Y).");
          result = q3.allSolutions();
  }
  else if (i == 5 && tokens[3].equals("processor"))
  {
          q3 = new Query("processor_of_machine(X,Y).");
          result = q3.allSolutions();
  }
else if (i == 5 && tokens[3].equals("windows"))
  {
        q3 = new Query("os_on_machine(X,windows).");
        result = q3.allSolutions();
  }
else if (i == 5 && tokens[3].equals("linux"))
  {
        q3 = new Query("os_on_machine(X,linux).");
        result = q3.allSolutions();
  }
else if (i == 5 && tokens[1].equals("IP"))
  {
        q3 = new Query("ip_of_machine(X,Y).");
        result = q3.allSolutions();
  }
else if (i == 5 && tokens[2].equals("monitorsize"))
  {
        q3                    =                    new
Query("machine_attribute(tokens[4],monitorsize,X).");
        result = q3.allSolutions();
  }
else if (i == 5 && tokens[2].equals("resolution"))
  {
        q3                    =                    new
Query("machine_attribute(tokens[4],resolution,X).");
        result = q3.allSolutions();
  }
  else if (i == 6 && tokens[2].equals("version"))
  {
        q3                    =                    new
Query("version_of_software(X,tokens[5]).");
        result = q3.allSolutions();
  }
  else if (i == 6 && tokens[3].equals("installed"))
  {
        q3   =   new   Query("software   _on_machine
(tokens[5],X).");
        result = q3.allSolutions();
  }
  else if (i == 6 && tokens[2].equals("IP"))
  {
        q3 = new Query("ip_of_machine(tokens[5],Y).");
        result = q3.allSolutions();
  }

     return result;
}
```

I. *Prolog Rules*

General format of the prolog rule is

predicate(phrase1, phrase2, ..., phraseN)

where, phrase1, phrase2, ..., phraseN can be constants or variables (facts or rules) .

J. *Natural Language Queries*

The set of natural language queries are depicted below:

1. Where is *? (* → Software Name)
2. Which machine has maximum RAM?
3. Which machine has maximum harddisk capacity?
4. Which machine has maximum processor speed?
5. What are different OS?
6. What is the version of *? (* → Software Name)
7. Which machine has JDK version 1.6?
8. What are different machine brands?
9. What is brand of *? (* → Machine Name)
10. What are different processor types?
11. What is processor type of *? (* → Software Name)
12. List all softwares installed on *? (* → Machine Name)
13. Which machine has Windows OS?
14. Which machine has Linux OS?
15. List IP address of machines
16. What is IP address of *? (* → Machine Name)
17. What is monitorsize of *? (* → Machine Name)
18. What is resolution of *? (* → Machine Name)

K. *Prolog Queries*

The equivalent Prolog queries are:

1. version_of_software (<softwarename>,<versionno>,<machinename>)
2. machine_by_vendor(<machinename>,<vendorname>)
3. software _on_machine (<machinename>,<softwarename>)
4. processor_of_machine(<machinename>,<processortype>)
5. os_on_machine(<machinename>,<osname>)
6. ip_of_machine(<machinename>,<ipaddress>)

7. machine_attribute(<machinename>,<attributename>,<attributevalue>)

Table 4 furnishes mapping of Natural Language query to the corresponding Prolog query where N/A represents the situation where the direct maping is not appliable.

Table 4 Mapping Table for Natural Language Query to Prolog Query

Sr.No.	Natural Language Query	Equivalent Prolog Query
1	Where is *? (* → Software Name) Example : Where is Python?	Software_on_machine(X,python)
2	Which machine has maximum RAM?	N/A
3	Which machine has maximum harddiskcapacity?	N/A
4	Which machine has maximum processor speed?	N/A
5	What are different OS?	OS_on_machine(X,Y).
	What is OS on *? (* → Machine Name) Example: What is OS on Dell4?	N/A
6	What is the version of *? (* → Software Name) Example: What is version of JDK?	version_of_software (jdk,X,Y)
7	Which machine has JDK version 1.6?	version_of_software (jdk,1.6,Y)
8	What are different machine brands?	machine_by_vendor(X,Y)
9	What is brand of *? (* → Machine Name) Example: What is brand of DELL5?	machine_by_vendor(dell5,Y)
10	What are different processor types?	processor_of_machine(X,Y)
11	List all softwares installed on *? (* → Machine Name) Example: List all softwares installed on DELL5	software _on_machine (dell5,X)
12	Which machine has Windows OS?	os_on_machine(X,windows)
13	Which machine has Linux OS?	os_on_machine(X,linux)
14	List IP address of machines	ip_of_machine(X,Y)
15	What is IP address of *? (* → Machine Name) Example: What is IP address of DELL5?	ip_of_machine(dell5,Y)
16	What is monitorsize of *? (* → Machine Name) Example: What is monitorsize of DELL5?	machine_attribute(dell5,monitorsize,X)
17	What is resolution of *? (* → Machine Name) Example: What is resolution of DELL5?	machine_attribute(dell5,resolution,X)

For example the Prolog query corresponding to the Natural Language Query,
What is version of jdk?
is shown below:

Software_version (software, version, machine name)
Machine_vender (machine name, vender name)
Machine software (machine name, software name)

IV. Results and Discussion

The model proposed above is implemented using Java and Prolog. The structure of the database for storing hardware and software configuration information in MS-Access database is depicted in Figure 3.

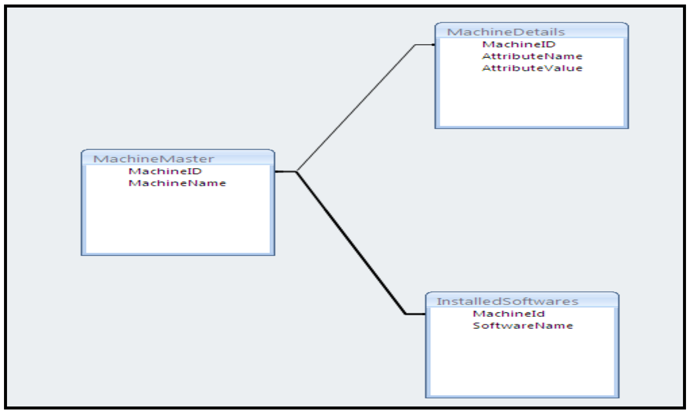


Figure 3. Structure of MS-Access Database for Storing Hardware/Software Configuration Information

A. Structure and Relationship Between Tables

The structure of different tables in the database along with the relationship between them is depicted below:

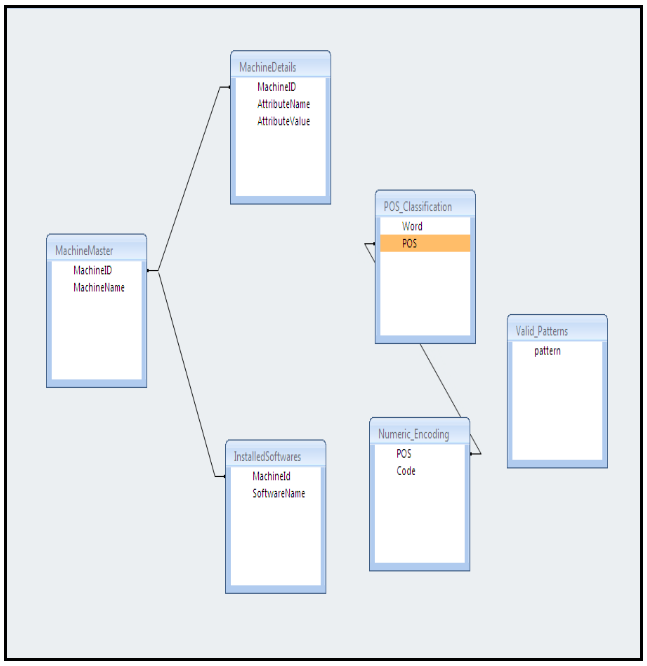
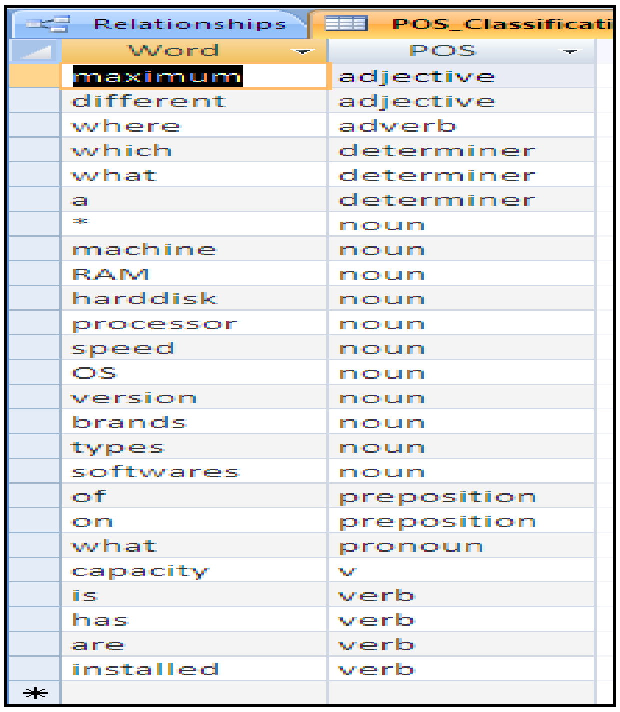


Table Name : POS_Classification

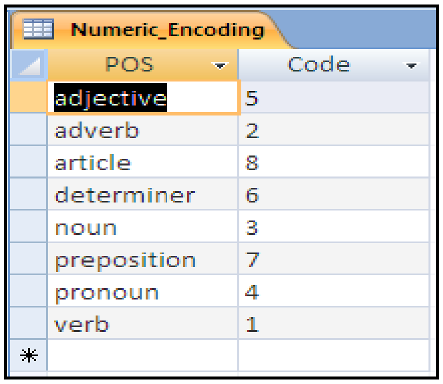
Table Structure:


Relationships
POS_Classificati

Word	POS
maximum	adjective
different	adjective
where	adverb
which	determiner
what	determiner
a	determiner
*	noun
machine	noun
RAM	noun
harddisk	noun
processor	noun
speed	noun
OS	noun
version	noun
brands	noun
types	noun
softwares	noun
of	preposition
on	preposition
what	pronoun
capacity	v
is	verb
has	verb
are	verb
installed	verb

Table Name : Numeric_Encoding

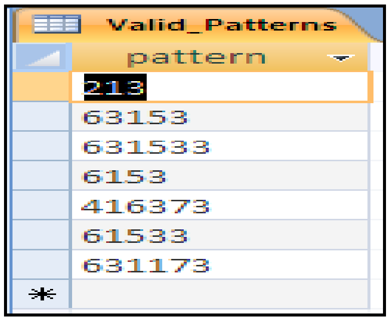
Table Structure


Numeric_Encoding

POS	Code
adjective	5
adverb	2
article	8
determiner	6
noun	3
preposition	7
pronoun	4
verb	1

Table Name : Valid_Patterns

Table Structure


Valid_Patterns

pattern
213
63153
631533
6153
416373
61533
631173

B. Implementation of Crisp Expert System for the Selection of Manufacturing Method Based on Single Objective.

An expert system is implemented in VB for querying hardware and software information in a LAN. Figure 4 shows the architecture of Expert System designed for the purpose and Figure 5 depicts the multi-tier architecture employed by the expert system.

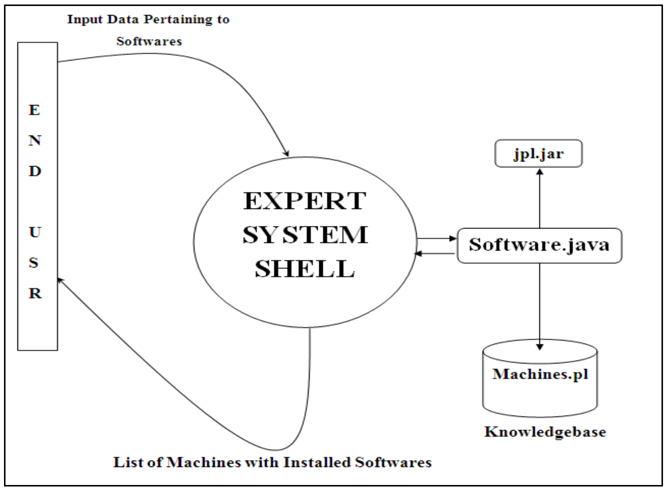

Input Data Pertaining to Softwares
E N D U S R
EXPERT SYSTEM SHELL
jpl.jar
Software.java
Machines.pl
Knowledgebase
List of Machines with Installed Softwares

Figure 4 Multi-tier Architecture of Expert System

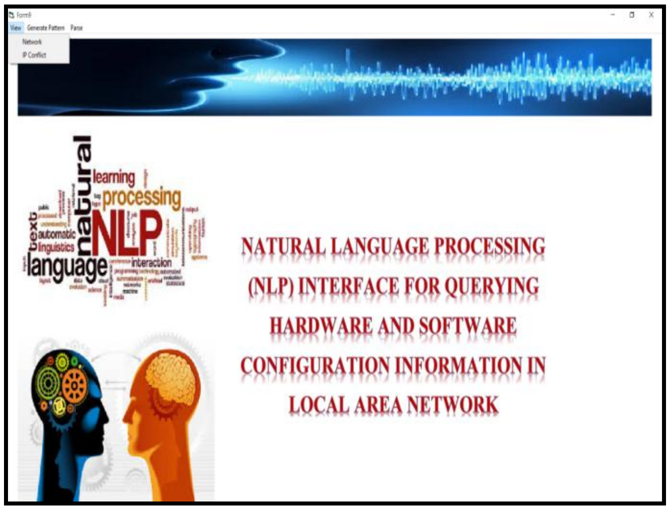
User Interface in Visual Basic
Java application interfacing with Prolog using JPL
Prolog
Prolog Database or Knowledgebase

Figure 5. Layered Architecture of Expert System

An expert system shell with Graphical User Interface (GUI) is presented to the end user. Figure 6. shows the startup screen for NLP interface for querying hardware and software configuration information in a Local Area Network(LAN). Figure 7(a)-(b). depicts the corresponding menu structure. The start-up script retrieves the information about various machines connected to LAN and the data pulled out is persistently stored in a database management system for querying the data in future. Figure 8 depicts the LAN configuration of department of Computer Studies of CSIBER. The snapshot is taken at a particular point of time and only the machines which are online at that point of time are listed by the startup script

Form9
View Generate Pattern Parse
natural learning processing NLP text automatic linguistics language interaction
NATURAL LANGUAGE PROCESSING
(NLP) INTERFACE FOR QUERYING
HARDWARE AND SOFTWARE
CONFIGURATION INFORMATION IN
LOCAL AREA NETWORK

Figure 6. Startup Screen for NLP Interface


Form9
View Generate Pattern Parse
Network
IP Conflict
NATURAL LANGUAGE PROCESSING
(NLP) INTERFACE FOR QUERYING
HARDWARE AND SOFTWARE
CONFIGURATION INFORMATION IN
LOCAL AREA NETWORK

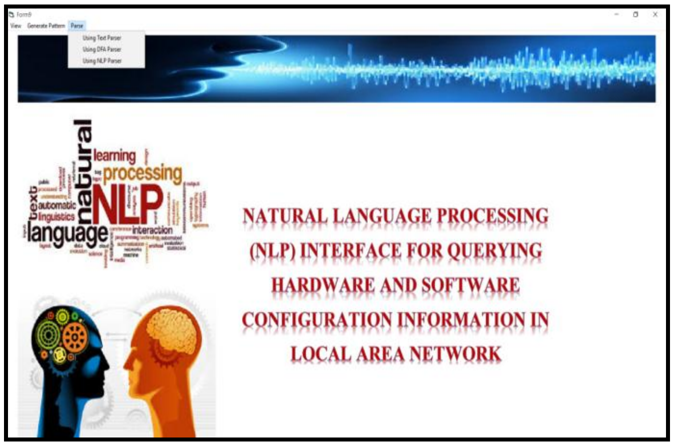

Form9
View Generate Pattern Parse
Using Text Parser
Using DFA Parser
Using NLP Parser
NATURAL LANGUAGE PROCESSING
(NLP) INTERFACE FOR QUERYING
HARDWARE AND SOFTWARE
CONFIGURATION INFORMATION IN
LOCAL AREA NETWORK

Figure 7(a)-7(b). Interface Menu Designs

The structure of the Local Area Network with the corresponding organization of the machines in LAN is depicted in Figure 8 .

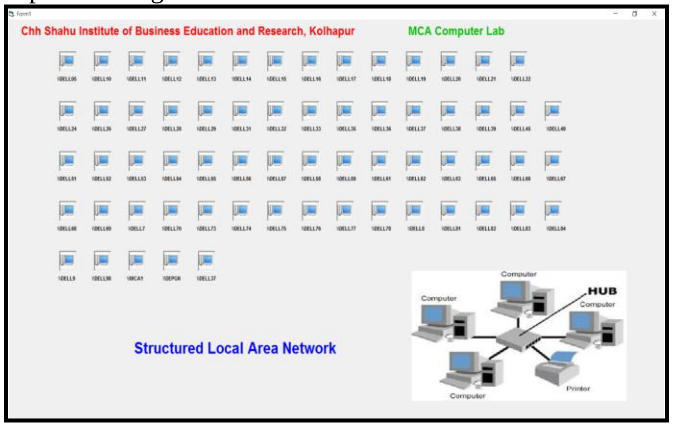
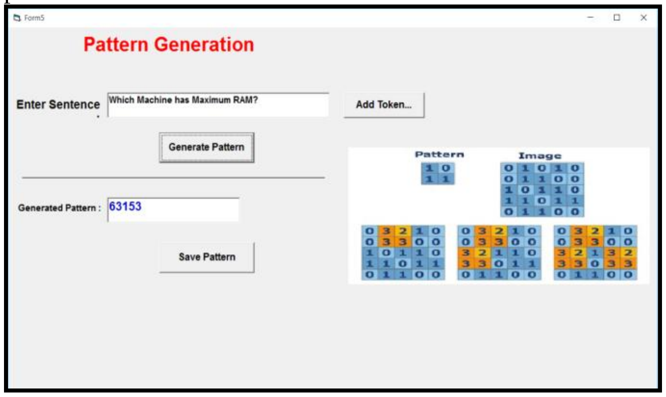

Chh Shahu Institute of Business Education and Research, Kolhapur
MCA Computer Lab
Structured Local Area Network
Computer
HUB
Printer

Figure 8 Structure of the Local Area Network Generated by the Tool.

The model designed and presented above is applied to the pattern generation which is shown in Figure 9. The natural language sentence entered by an end user is broken into parts of speech tags and is numerically encoded by a specific numeric encoding mechanism suggested in Table 2. The pattern is saved for future reference.


Form5

Pattern Generation

Enter Sentence: Which Machine has Maximum RAM?

Generated Pattern : 63153

Figure 9. Pattern Generations for NLP Query

If the token is not already present in the database, the new token can be created and saved into the database by clicking on "Add Token…" button. The Token Id is auto generated whereas token name along with the POS tag must be supplied by an end user.

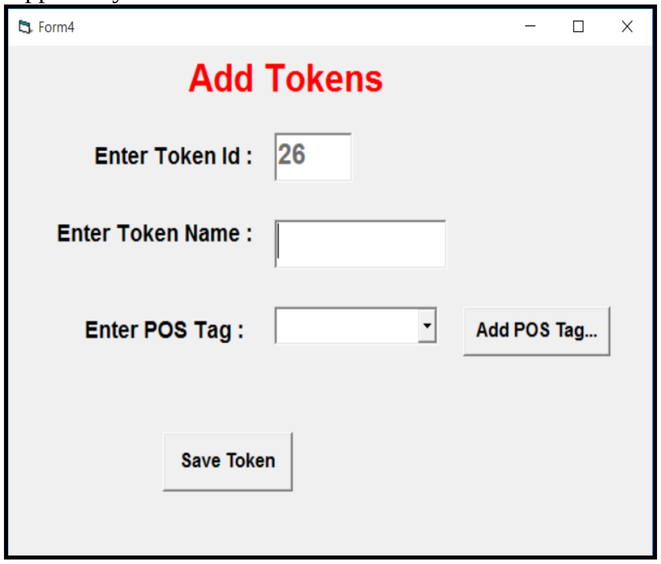


Figure 10. Creating a New Token and Associating with a POS Tag

Figure 10 exihibits GUI for creating and storing new tokens where the Token Id is auto generated and the POS tag is attached to a token. A provision is available for creating and storing new POS tags in the case a particular POS tag is not available in the data store. Figure 11 shows a GUI for adding new POS tag and coding them numerically.

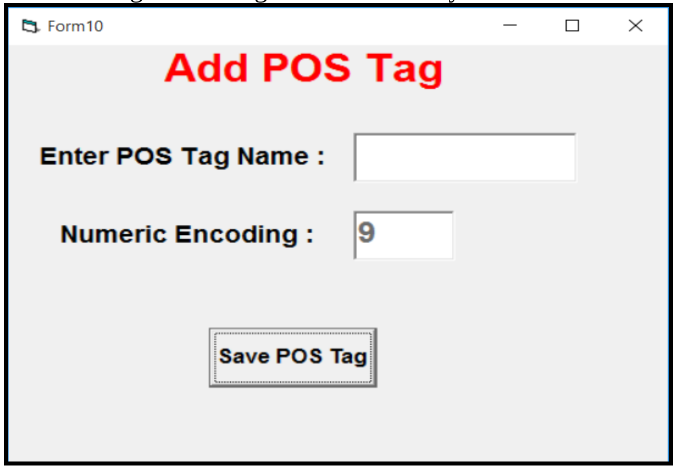


Figure 11 Creating a New POS Tag

Figure 12-13 show parsing of HQL commands for querying hardware and software information in LAN. The natural language query entered by an end user is broken into POS tags, encoded numerically and then the pattern is generated. The generated pattern is looked up in the set of valid patterns. If the pattern exists in the database, the query is parsed successfully, otherwize contains a syntax error.

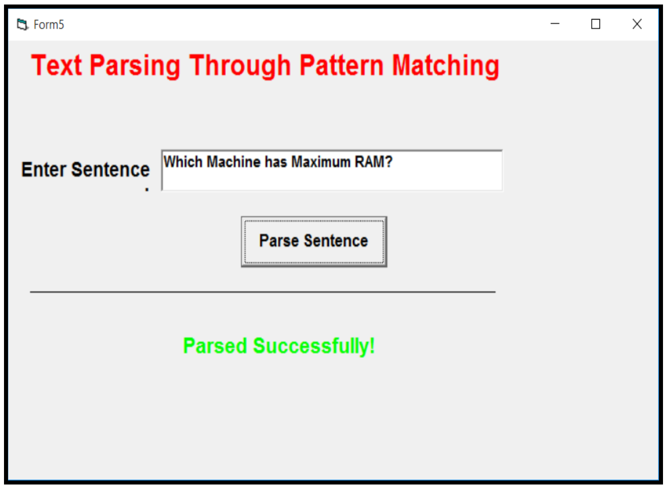

Form5
Text Parsing Through Pattern Matching
Enter Sentence
Which Machine has Maximum RAM?
Parse Sentence
Parsed Successfully!

Figure 12 Parse sentence screen

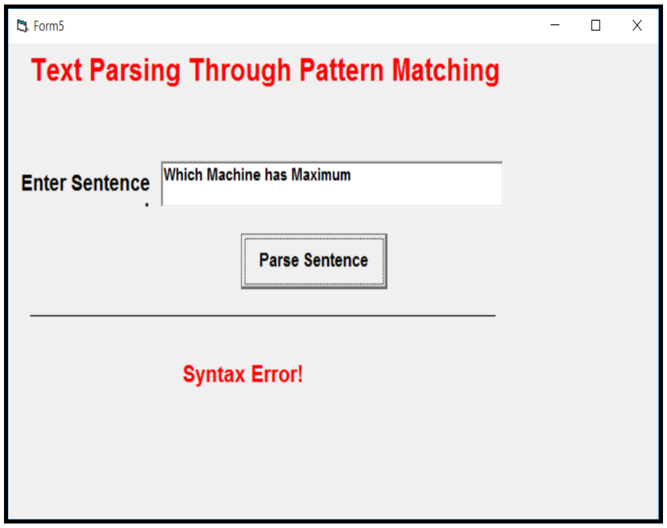

Form5
Text Parsing Through Pattern Matching
Enter Sentence
Which Machine has Maximum
Parse Sentence
Syntax Error!

Figure 13 Parsing HQL Commands Using NLP Parser

V. Conclusion and Future Scope

In the current research, a query language is designed by the author for querying the hardware/software information of LAN. The end user instead of querying the database directly will use the natural language, termed as Hardware Query Language (HQL) and Software Query Language (SoQL) designed by the author, which is interfaced with RDBMS using prolog. In the current work, few HQL/SoQL queries are used for reference. However, the set can be extended in future.

An expert system is designed and implemented for querying the software information stored in knowledgebase and implement a NLP parser for parsing sentences entered by the user in a natural language as it is very flexible and quickly adapts to the changes in a query language.

To render the model generic the dynamic data is separated out and stored in an XML format which is then parsed using MS XML parser in VB.

For designing NLP parser, a context free grammar is devised. The natural language sentences which are parsed successfully are then executed employing three tier application architecture with Visual Basic in presentation tier, Java in middle tier and MS-Access in data tier. Visual basic presents a GUI for end user interation which collects the user input, generates a Prolog query on fly and invokes a Java application residing in middle tier which interfaces with SWI prolog for execution of Prolog query. The output is routed back to the presentation tier before being presented to an end user. The dynamically generated Prolog query is also displayed for reference to a technical person.

A. Scope for Future Work and Suggestions-

The future work focuses on designing a supervised neural network for parsing the HQL/SOQL queries into two different classes “correct” and “incorrect” based on the input pattern.

i) The tool can be further modified

a) For incompatible bootstrap technology to make it available on small Hand-held devices.

b) To instantly send SMS to a lab technician if any anomaly or any conflict is observed in network.

In the current work the design configuration information is stored in XML format which was also employed for data exchange. However, recently Java Script Object Notation (JSON) is emerging as the most promising technology for storing and exchanging data in the form of name/value pair in a platform independent and file system neutral manner which is slowly replacing XML and is an output for choice for most of the web services and RESTful services. The tool implemented in current research can be rendered widely accessible by converting them into web services which can be deployed on organization’s private cloud interfacing with each other employing JSON as a mechanism for data interchange.

REFERENCES

[1] Dr.P.G.Naik, Mr. M. B. Patil “ Design and Development of Network Monitoring and Controlling tool for Domain Controller @Department of Computer Studies, CISBER using RMI Technology- A case study”. International Journal of Engineering Trends and Technology (IJETT) – Volume 26 Number 2- August 2015

[2] Wei Wang , Huankai Shi “Analysis of the Combination of Natural Language Processing and Search Engine Technology “. International Workshop on Information and Electronics Engineering (IWIEE), 2012.

[3] JE Cooling “Survivable LANs for distributed control systems”. Published in Computer communication, Publisher- Elsevier, Volume17, Issue 5, May 1994, Pages 317-331.

[4] Yukoko Sasaki Alam “A Subcategory-based Parser Directed to Generating Representations for Text Understanding”. Published by Elsevier Ltd, Pacific Association for Computational Linguistics (PACLING)., Procedia - Social and Behavioral Sciences 27 (2011) 194 – 201.

[5] Marta Andersson, Adnan Ozturely, Silvia Pareti “Annotating Topic Development in Information Seeking Queries”. Published by Research at Google, 2016.

[6] Ankur P. Parikh, Dipanjan Das “A Decomposable Attention Model for Natural Language Inference.” Proceedings of EMNLP 2016, subject-co mputation and language, cite as arXiv:1606.01933[cs.CL] Submitted on 6 Jun2016 (v1), last revised 25 Sep2016(this version,v2)

[7] Dan Gillick, Cliff Brunk , Oriol Vinyals, Amarnag Subramanya “Multilingual Language Processing From Bytes. Proceedings of EMNLP 2016, subject-computation and language, cite as arXiv:1606.01933[cs.CL] Submitted on 1 Dec 2015 (v1), last revised 2 Apr 2016 (this version, v2))

[8] Ronan Collobert, Jason Weston , Leon Bottou, Michael Karlen “Natural Language Processing (Almost) from Scratch.” Journal of Machine Learning Research 12 (2011) 2493-2537 Submitted 1/10; Revised 11/10; Published 8/11.

[9] Abu Taha Zamani, Javed Ahmad “ Wireless LAN Security : IEEE 802.11g & VPN”. International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization) Vol. 4, Issue 6, June 2016. ISSN(Online): 2320-9801 ISSN (Print) : 2320-9798.

[10] Vijay Chandramouli “A Detailed Study on Wireless LAN Technologies”. Publisher RearchGate, scientific-contributions/2024466239 in 2017.