JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-4, Issue-3 E-ISSN: 2347-2693

Implementation of Fuzzy Text Parser for Querying LAN Configuration Information

Poornima G. Naik[1*] and Kavita S.Oza [2]

[1*] *Department of Computer Studies, CSIBER, Kolhapur, India*
[2] *Department of Computer Science, Shivaji University, Kolhapur, India*

www.ijcseonline.org

Received: Feb/26/2016 Revised: Mar/09/2016 Accepted: Mar/19/2016 Published: Mar/31/ 2016

Abstract— — In this communication era, keeping pace with current advancements in information and communication technologies every organization hosts a private network enabling sharing of resources such as softwares and hardware devices. Such corporate and educational networks facilitate sharing of data in a secure manner but at the same time pose a problem to the lab technician in keeping track of hardware and software configuration information and their working condition. On many occasions the softwares which are rarely used may only be installed on couple of machines. Further, as the size of the network grows locating resources on LAN becomes extremely difficult and time consuming. To address these issues one of the authors has designed a fuzzy text parser for querying hardware and software related information in a local area network in a human-like query language. Human-machine interaction is enabled through a set of query languages nomenclatures as Hardware Query Language (HQL) and Software Query Language (SOQL). In the current paper, authors have provided implementation of the model for fuzzy text parser by considering different application architectures. The fuzziness in the text parser is incorporated by giving due consideration to superfluous and implied words which renders the query language close to natural language.

Keywords—Domain Controller, Human Computer Interface, MySQL, Parse Tree, Synonyms, Workgroup.

I. INTRODUCTION

With the continual advancement in communication and information technologies, there is a paradigm shift from single user machine to a networked environment extending reachability beyond organization's boundaries. While such a hi-tech environment becomes a boon to an end user, it poses a great challenge to a lab technician in keeping track of additional information pertaining to network devices and softwares installed on different machines along with their working condition. These tasks can be brought under control through automation of frequently carried out tasks such as periodic checking of machine status , status of various softwares installed on different machines of a network, periodic checking of security logs etc. Performing any of these tasks manually is time consuming and error prone. On many occasions monitoring changes in network configuration such as IP conflicts, manipulation of IP addresses of individual machines can bring the entire network down and needs the lab technician to virtually check every computer connected to LAN by manually attending each computer. Further, the network configuration is only static over a small period of time but is overall dynamic in nature as continuously new devices are plugged in and new softwares are installed on the network. Lot of efforts and time can be saved by automating such common network routines.

Currently, one of the authors is working on a project pertaining to design and development of human computer interaction (HCI) interface for LAN which accepts a query related to LAN configuration in natural language (NL) which is parsed using NLP parser developed by the author by mapping human query to the corresponding SQL query. The solution poses a unique blend of relational database management system with knowledge bases. Java interface to prolog is employed for mapping human queries to prolog queries. Different types of parsers are developed for the purpose and compared. DFA parser outscores over other parsers where extensibility is a prime issue to be addresses by the parser. However, these parsers are conventional which do not take into consideration fuzziness inherited in natural languages. To render the queries closer to the natural languages and in order to incorporate more human-like behavior in the queries, the authors in the current paper have implemented couple of fuzzy functionalities in the conventional papers. Currently, the fuzzy rule set takes care of only few rules but can easily be extended further as new rules are discovered.

In the current paper, the authors have focused on the following aspects.

- To dynamically discover LAN architecture and to list various computers in a workgroup and a domain controller.
- To dynamically discover the various hardware connected to LAN and softwares installed on different machines of a network, storing the same persistently in a centralized database and knowledge base for future usage.
- To design and implement a fuzzy NLP parser to facilitate an end user and a network administrator. The queries in the natural language submitted by an end user are

parsed and evaluated by mapping them to the prolog queries.

II. LITERATURE REVIEW

In literature there exit numerous papers on natural language processing applied to various areas to reduce the gap between human and machine languages [1-8]. Yue et. al. [9] in their paper have discusses the current situation and process of natural language processing (NLP) and their effect of natural language processing in search engine. Alam [10] has proposed a model for subcategory-based parser 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 which 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. In their paper, the authors have proposed a 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 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. This work is then used as a basis for building a freely available tagging system with good performance and minimal computational requirements. The paper [11] contrasts two strategies for parsing unrestricted natural language text, the grammar-driven and the data-driven approach, and compares the ways in which they deal with the problems of robustness, disambiguation, accuracy and efficiency. The authors have argued that the two strategies are complementary rather than contradictory and that they can be seen as motivated by different goals and optimization strategies in relation to the complex problem of parsing unrestricted text. Fuzzy parsing [12-14] is targeted for reliably extracting information from partially correct or incomplete input texts. For fuzzy parsing the underlying grammar is partial with respect to the original one. Fuzzy approach defines a minimum degree m of certainty (in the interval [0,1]), with which each recognized sentence belongs to the original language. The approach is based on deterministic finite automata where states define precise contexts within the sentences and edges represent potential matches of constructs of interest inside each context. By using this notion of contexts, the search space becomes smaller, reducing both the time to recognize the input and the ambiguity conflicts.

III. CONCEPTUAL FRAMEWORK

For querying hardware and software information, four distinct models have been proposed.

Model 1 :
It is based on manual execution and consists of the following phases [15].

Phase 1 : Retrieving requisite hardware and software information by execution of batch file.
The batch file GetHardwareSoftwareInfo.bat containing the following commands is executed on every client of a local area network.

```
psinfo -s > software.txt

wmic csproduct get vendor, version> vendor.txt

wmic logicaldisk get size, freespace, caption >harddisk.txt

ipconFig.> ip.txt

wmic desktopmonitor> monitor.txt

wmic desktopmonitor get screenheight, screenwidth>
monitorsize.txt

net view > workgroups.txt
netview /DOMAIN:siber1.com
```

Phase 2 :Installation of MySQL server.

MySQL server is installed on a centralized database server and a database with the name softwares is created with consists of the tables shown in Fig. 1.

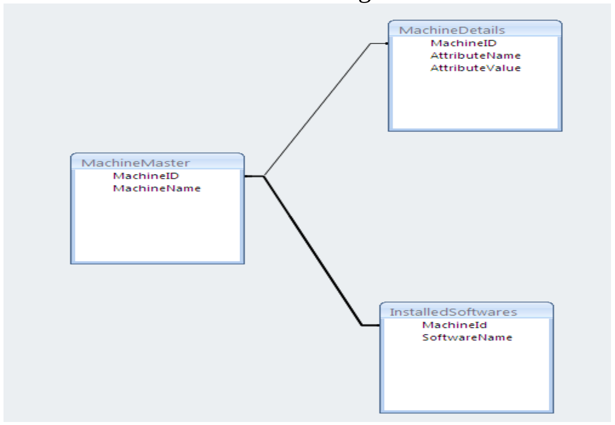


Fig. 1. Structure of Database

Phase 3 : Storing hardware and software information in MySQL database.

The batch file HardwareSoftwareInfo.bat containing the following commands is executed on each client of local area network which stores the hardware information retrieved by the batch file GetHardwareSoftwareInformation.bat in a centralized MySQL database.

```
set path=C:\Program Files\Java\jdk1.5.0\bin

set classpath=MySQL-connector-java-5.1.15-bin.jar;.

javac SearchSoftwares.java

java SearchSoftwares

Pause
```

Phase 4 : Querying Hardware/Software information using HQL/SOQL.

The hardware/software conFig.uration details of any machine can be queried by an end user by connecting to the MySQL database server by issuing a human query language which is parsed using both crisp and fuzzy text parsers. Fig. 2. depicts the Model 1 Application Architecture.

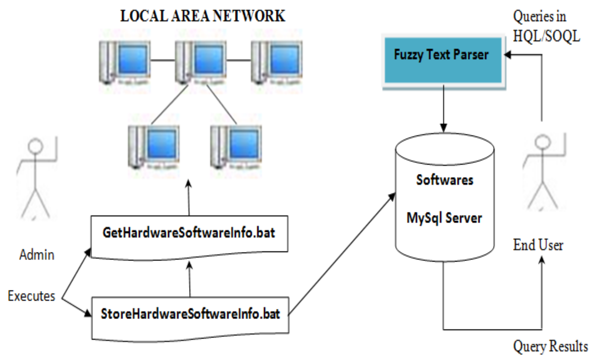


Fig. 2. Model1 Application Architecture

Model 2 :
In this model, a grammar is designed for hardware and software query language which consists of symbol set and rules for parsing phrases in human query language. Natural Language Processor (NLP) parser is used to parse HQL/SOQL queries.

Model 3 :
It is based on execution through a single point of control and has the following pre-requisites on the remote machines.

i) Telnet service should be started on each machine of the network.

ii) File Transfer Service (FTP) should be enabled on each machine of the network.

It consists of the following phases.

Phase 1: Transferring necessary files to the remote machine using FTP.
The batch file and the other requisite files are transferred from the controlling machine to all the clients using FTP.

Phase 2 : Remote execution using Telnet.

Both the batch files are remotely executed using Telnet service.

Model 4 :
It is based on automatic execution using a high level language such as Java which uploads the file using URL and URLConnection classes and remotely executes an application using TelnetClient class present in the package org.apache.commons.net.telnet.

A. Application Architecture

The layered architecture for the execution of HQL is shown below:

End User
HQL/SOQL in Natural Language
SQL Query
Physical Database

The end user issues HQL which is parsed by the text parser and evaluated by mapping to the corresponding SQL query which is then executed against the data stored in the physical database. End user does not have to deal with the intricacies of SQL queries.

B. Control Flow Diagram
Fig. 3(a)-3(b) show control flow diagrams for

- Reading tokens are assigning unique identifiers to each of them.
- Reading a sentence and generating a pattern.
- Parsing a sentence using crisp text parser.

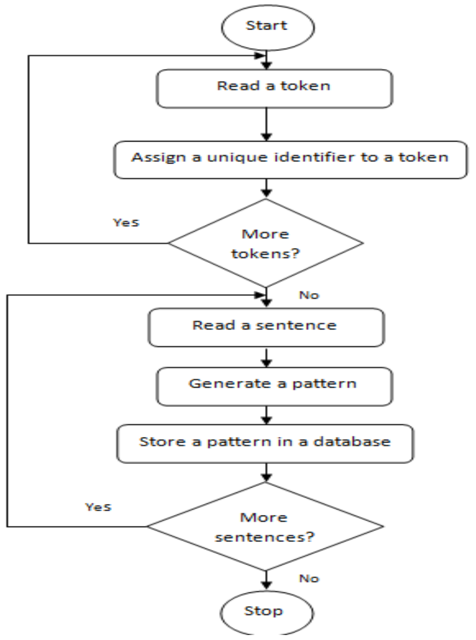


Fig. 3(a) Control Flow Diagram for Parsing and Assigning Unique Identifier to a token.

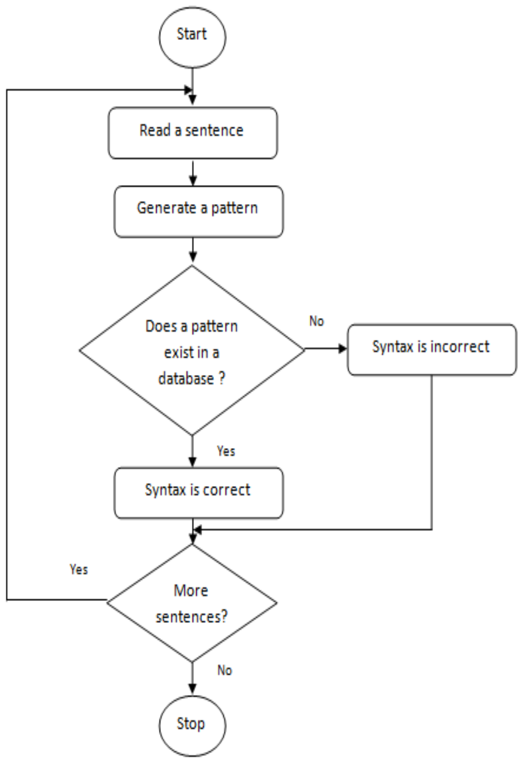


Fig. 3(b) Control flow Diagram for Parsing Tokens

C. Grammar for HQL and SOQL

To implement HQL and SOQL, we have constructed a language by defining the set of rules which specify how to test a string of alphabet letters to verify. A finite set of symbols used in the language is given by

∑ = {a, b, c, d, e, f, h, i, k, m, n, o, p, r, s, t, u, v, w, x, y}

and a set of words over an alphabet is given by

L={ are, brands, capacity, different, disk, hard, has, is, machine, maximum, of, os, processor, ram, speed, version, what, where, which}

D. Syntax and Semantics of a 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/SOQL 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.

The current set of alphabets, words and grammar is designed to address the following HQL/SOQL queries.

Where is matlab?

2. Which machine has maximum ram?

3. Which machine has maximum hard disk capacity?

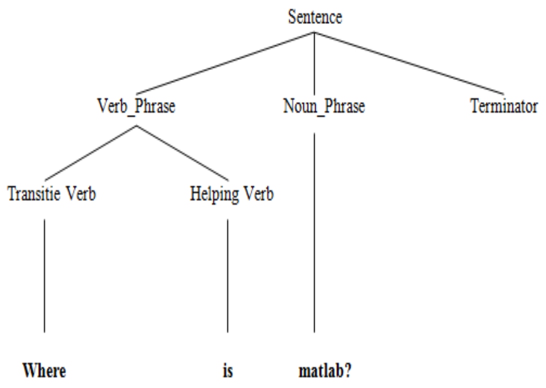
4. Which machine has maximum processor speed?

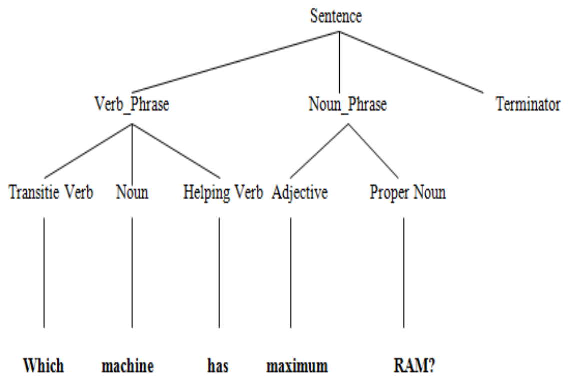
5. What are different OS?

6. What is a version of JDK?

7. What are different machine brands?

The corresponding parse tree is depicted in Fig.s 4(a)-4(e).





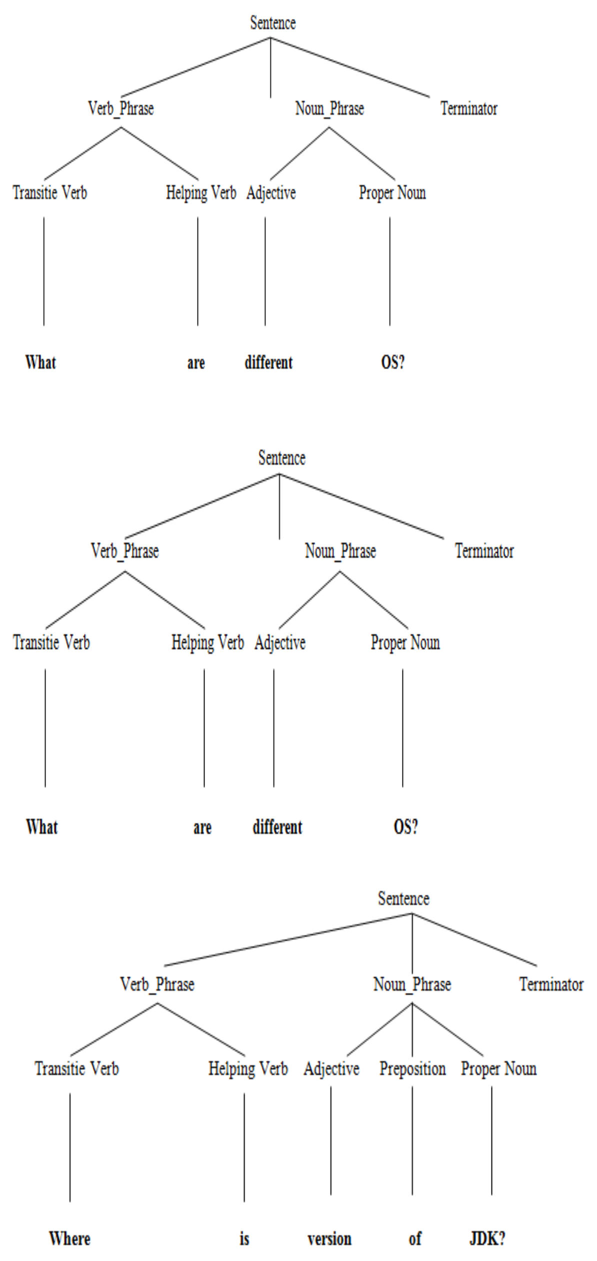


Fig. 4(a)-4(e). Parse Tree for Few commands of HQL.

E. Proposed Algorithm

```
/* Algorithm in C-Style */
struct Token
{
        int id;
        String name;
}
int no_of_tokens;
Token t[50];
String words[10];

function read_tokens_and_assign_identifiers()
{
        read no_of_tokens;
        for(i=1;i<=no_of_tokens;i++)
        {
                read token_name;
                t[i].id=i;
                t[i].name=token_name;
        }
}
function int get_tokenid(String tokenname)

        for(i=1;i<=no_of_tokens;i++)
        {
                if (t[i].name == tokenname)
                        return t[i].id;
        }
        return 0;
}
/*
```

Every high-level language has built-in string manipulation functions present in a string library. The following functions assume the existence of the following string manipulation functions.
instr() – Accepts two string arguments and returns the position of the second string within a first string, if the string is not found returns -1.
Right() – Accepts two arguments of type string and int, respectively and returns a substring of a string passed as the first argument containing rightmost n characters passed as the second argument to a function.

```
*/
function int count_words(String sentence)
{
        int count=0;
        int pos;
        pos=instr(sentence," ");
while (pos != -1)
        {
                count++;
                sentence=right(sentence,pos+1);
                pos=instr(sentence," ");
        }
return count;
}
function split_words(String sentence)
{
words=sentence.split(" ");
}
function String generate_pattern()
{
        String pattern;
read sentence;
```

```
        int cnt=count_words(sentence);
        for(i=1;i<=cnt;i++)
                pattern=pattern+get-tokenid(words[i]);
        return pattern;
}
```

F. Design and Working of Fuzzy Text Parser.

The following HQL query is issued by the user.
"what is the maximum processor speed?"
The tokens in the sentence are
{what, is, maximum, processor, speed}.

These are mapped on one-to-one basis to the corresponding identifiers generated earlier as shown in Fig. 5.

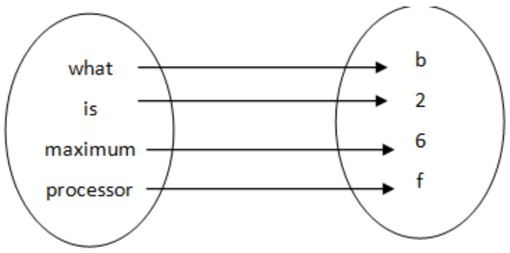


Fig. 5. Mapping of Tokens to Unique Identifiers

The fuzziness in a sentence mainly arises due to the following two main reasons:

i) Use of synonyms
ii) Stripping of superfluous words with the implicit meaning.

In the above human query the only word containing the synonym is 'maximum' for which the possible synonyms are :

i) largest
ii) highest
iii) greatest

Hence the following sentences are also semantically valid and are equivalent to the given query.

i) what is the largest processor speed?
ii) what is the highest processor speed?
iii) what is the greatest processor speed?

Further, the tokens 'what' and 'is' are superfluous. Hence in a layman's language,
"maximum processor speed?"
is also a valid sentence.
Hence incorporating the synonyms and stripping off the superfluous words the query
"what is the maximum processor speed?"

will be converted into its following fuzzy counter parts.

i) what is the largest processor speed?
ii) what is the highest processor speed?
iii) what is the greatest processor speed?
iv) largest processor speed?
v) highest processor speed?
vi) greatest processor speed?

All of which should be syntactically correct when parsed using fuzzy text parser. On the contrary crisp text parser looks for each and every word in a particular sequence and maps it to either "correct" or "incorrect" value. On the other hand fuzzy text parser generates many possible alternative forms of a given sentence taking into account synonyms and superfluous words and maps them all to "correct" value. Mapping of different patterns to correct and incorrect values by both crisp and fuzzy text parsers are depicted in Fig.s 6(a) and 6(b), respectively.

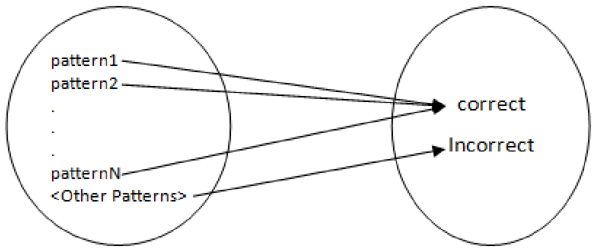


Fig. 6(a). Mapping of Different Patterns by Crisp Text Parser to Correct and InCorrect Values

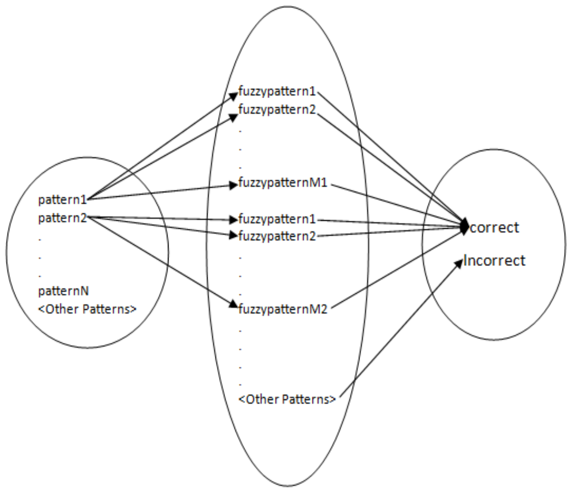


Fig. 6(b). Mapping of Different Patterns by Fuzzy Text Parser to Correct and Incorrect Values

IV. RESULTS AND ANALYSIS

The model1 proposed above is implemented in VB with MS-Access as backend. Fig. 7. depicts the sample database for parsing the sentence using fuzzy text parser.

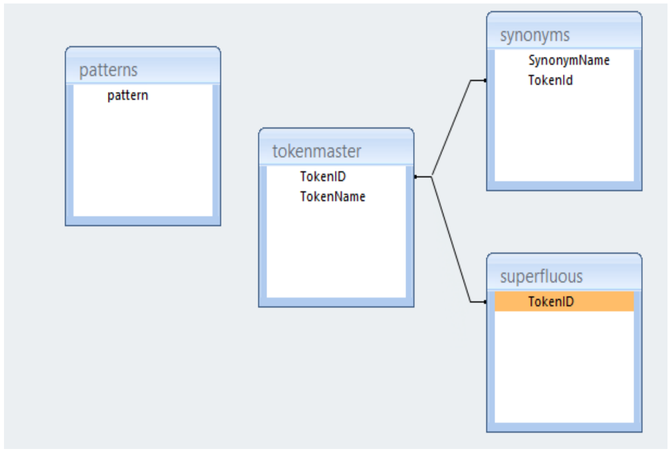
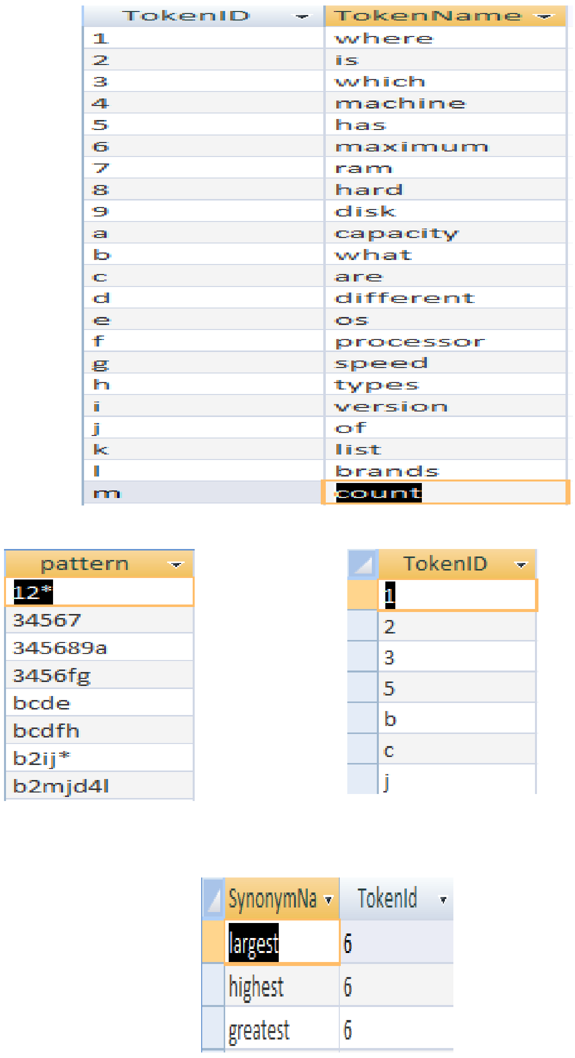


Fig. 7. Sample Database for Implementation of Fuzzy Text Parser.



TokenID	TokenName
1	where
2	is
3	which
4	machine
5	has
6	maximum
7	ram
8	hard
9	disk
a	capacity
b	what
c	are
d	different
e	os
f	processor
g	speed
h	types
i	version
j	of
k	list
l	brands
m	count

pattern
12*
34567
345689a
3456fg
bcde
bcdfh
b2ij*
b2mjd4l

TokenID
1
2
3
5
b
c
j

SynonymNa	TokenId
largest	6
highest	6
greatest	6

Fig. 8. Sample Data Used for Parsing HQL Queries.

Fig. 9 shows dynamic discovery and display of LAN architecture. Fig.s 10(a)-10(c) depict the graphical user interfaces (GUI) for assigning unique identifiers to the various tokens and generating patterns for the various valid statements.

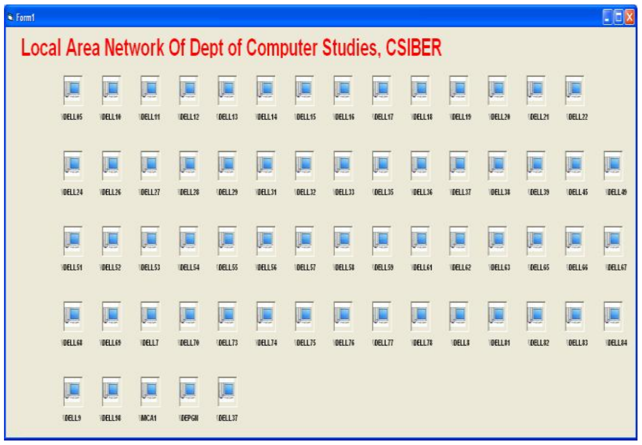
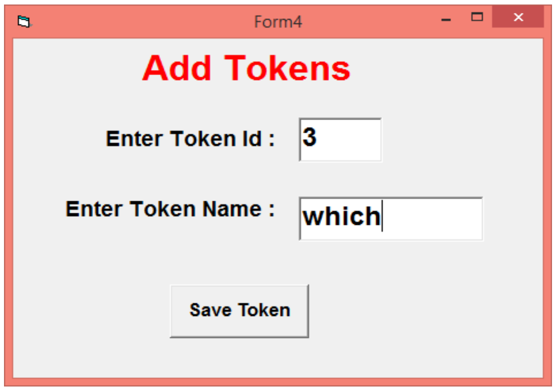
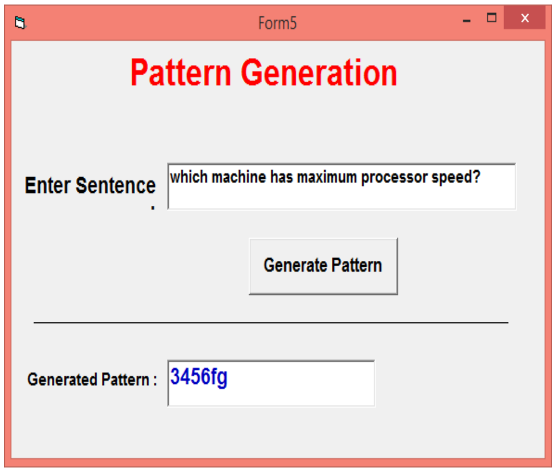


Fig. 9. LAN Architecture at Department of Computer Studies, CSIBER.



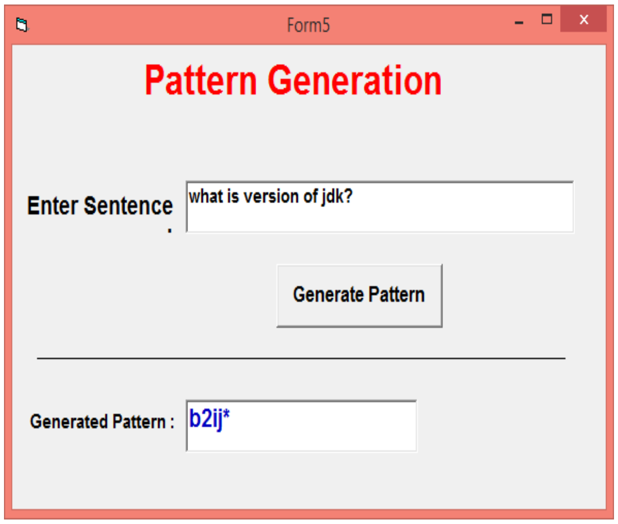
Enter Token Id : 3

Enter Token Name : which



Enter Sentence : which machine has maximum processor speed?

Generated Pattern : 3456fg


Form5

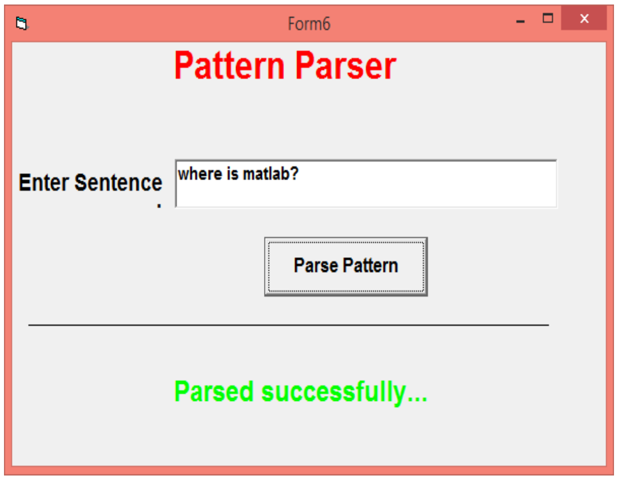
Pattern Generation

Enter Sentence: what is version of jdk?

Generate Pattern

Generated Pattern : b2ij*

Fig. 10(a)-10(c) GUI for Storing Tokens and Pattern Generation

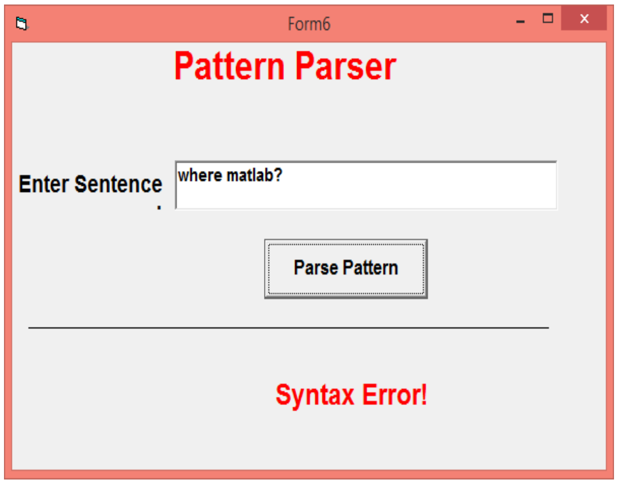

Form6

Pattern Parser

Enter Sentence: where is matlab?

Parse Pattern

Parsed successfully...


Form6

Pattern Parser

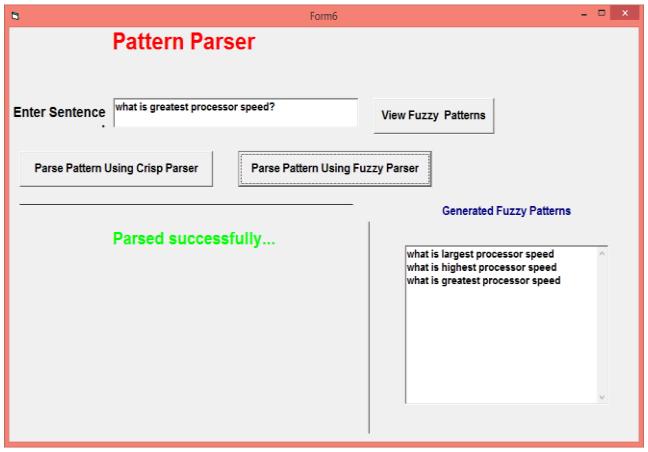
Enter Sentence: where matlab?

Parse Pattern

Syntax Error!

Fig. 11(a)-11(b) Parsing Sentence using Crisp Text Parser

Fig. 12(a) Fig. 12(c) show parsing the sentence using fuzzy text parser whereas Fig. 12(b) and Fig. 12(d) show parsing the same sentences using crisp text parser. It is observed that the fuzzy parser accepts the senetence where as crisp parser rejects the same.


Form6

Pattern Parser

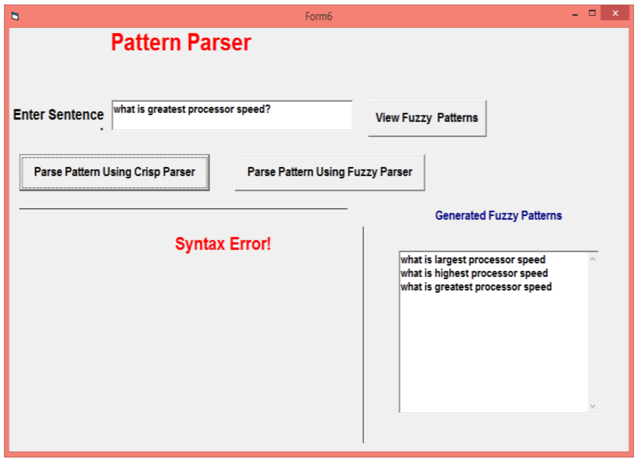
Enter Sentence: what is greatest processor speed?

View Fuzzy Patterns | Parse Pattern Using Crisp Parser | Parse Pattern Using Fuzzy Parser

Parsed successfully...

Generated Fuzzy Patterns

what is largest processor speed
what is highest processor speed
what is greatest processor speed


Form6

Pattern Parser

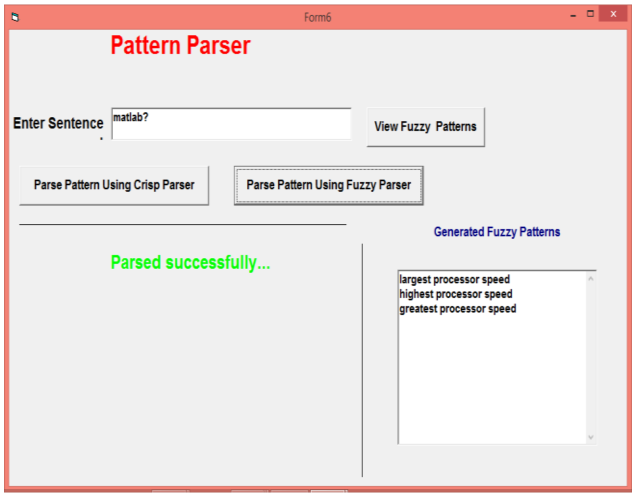
Enter Sentence: what is greatest processor speed?

View Fuzzy Patterns | Parse Pattern Using Crisp Parser | Parse Pattern Using Fuzzy Parser

Syntax Error!

Generated Fuzzy Patterns

what is largest processor speed
what is highest processor speed
what is greatest processor speed


Form6

Pattern Parser

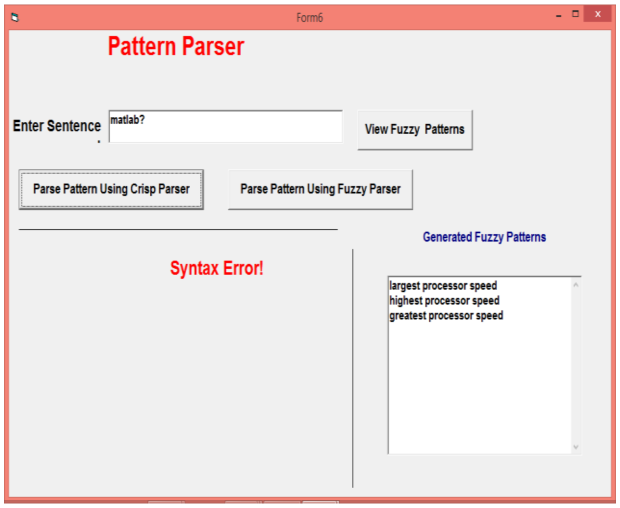
Enter Sentence: matlab?

View Fuzzy Patterns | Parse Pattern Using Crisp Parser | Parse Pattern Using Fuzzy Parser

Parsed successfully...

Generated Fuzzy Patterns

largest processor speed
highest processor speed
greatest processor speed



matlab?

Syntax Error!

largest processor speed
highest processor speed
greatest processor speed

Fig. 12(a)-12(d). Parsing HQL Queries using Fuzzy and Crisp Text Parsers.

V. CONCLUSION AND SCOPE FOR FUTURE WORK

In the current work authors have designed and developed a model to dynamically discover LAN architecture and query for hardware and software configuration information of each machine connected to LAN. The information is stored in a centralized MySQL database which can be queried using Hardware Query Language and Software Query Language designed by the authors. Both the crisp and fuzzy text parsers are designed and implemented. Both the parsers are compared by parsing few test queries by generating the respective parse tree. Currently, few fuzzy rules are implemented in the fuzzy parser which can be extended in future with minimum efforts. The model proposed is general and can be applied to any local area network.

Our future work focuses on replacing RDBMS with a knowledge base and developing NLP parser interface with the knowledge base using Java interface to Prolog. The human queries can be converted into Prolog queries which can be parsed using NLP parser. The queries which are successfully parsed can be evaluated to generate the desired output.

REFERENCES

[1]. Klein and C. D. Manning. Natural language grammar induction using a constituent-context model. In Advances in Neural Information Processing Systems (NIPS 14), pages 35–42, **2002.**

[2]. Kudo and Y. Matsumoto. Chunking with support vector machines. In Conference of the North American Chapter of the Association for Computational Linguistics & Human Language Technologies (NAACL-HLT), pages 1–8, **2001.**

[3]. Mnih and G. E. Hinton. Three new graphical models for statistical language modeling, International Conference on Machine Learning (ICML), pages 641–648**, 2007.**

[4]. Ronan Collobert, JasonWeston, L´eon Bottou, Michael Karlen , Koray Kavukcuoglu, Pavel Kuksa, Natural Language Processing (Almost) from Scratch, Journal of Machine Learning Research 12 , pages 2493-2537**, 2011**.

[5]. Saif Mohammad, Bonnie Dorr, and Graeme Hirst, Computing word-pair antonymy. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, , Honolulu, HI. ACL, pages 982–991, **2008**.

[6]. Stanley Kok and Pedro Domingos, Extracting semantic networks from text via relational clustering. In Proceedings of the Nineteenth European Conference on Machine Learning, Antwerp, Belgium. Springer, pages 624–639, **2008.**

[7]. Booth, Taylor L. & Richard A. Thompson ,Applying probability measures to abstract languages. IEEE Transactions on Computers C-22.5, pages 442–450**, 1973.**

[8]. Bod, Rens, Beyond Grammar, CSLI Publications, University of Chicago Press, **1998.**

[9]. Xiaoguang Yue, , Guangzhi Di, Yueyun Yu, Wei Wang, Huankai Shi, Analysis of the Combination of Natural Language Processing and Search Engine Technology, Procedia Engineering, Volume 29, International Workshop on Information and Electronics Engineering, pages 1636–1639, **2012**

[10]. A Subcategory-based Parser Directed to Generating Representations for Text Understanding, Yukiko Sasaki Alam, Procedia - Social and Behavioral Sciences, Volume 27, Pages 194–201, **2011.**

[11]. Joakim Nivre, Two Strategies for Text Parsing, CSLI Publications, University of Chicago Press

[12]. Peter R. J. Asveld. Fuzzy context-free languages – Part 2: Recognition and parsing algorithms. Theoretical computer science, 347(1):191–213, **2005**.

[13]. Rainer Koppler. A systematic approach to fuzzy parsing. Software Practice and Experience, 27:649, **1996**

[14]. Kang , Soon Ju, Kwon , Yong Rae, A Tightly Coupled Approach to Fuzzy Syntactic Parsing and Neural Networks for Event-Synchronous Signal Inspection, Journal of Intelligent and Fuzzy Systems, vol. 3, No. 3, pages 215-227**, 1995.**

[15]. Santosh Patil and Poornima G. Naik, Design and Development of Fuzzy Text Parser for Querying Hardware and Software Information in Local Area Network, International Journal of current Research, Vol 8., Issue 03, pp 27434-27437, March 2016.

Author Profile

Dr. Poornima G. Naik, received M.Sc. degree in Physics and Mathematics and Ph.D. degree in physics from Karnataka University, Dharwad. She received MCA degree from IGNOU with first class Distinction. Currently, she is working as Professor in the Department of Computer Studies, SIBER, Kolhapur. Her areas of interest are network security, soft computing and cloud computing. She has participated in several national and international conferences and has published more than 30 papers in International and national journals of repute.

Dr. Kavita S. Oza , received Ph.D. degree in Computer Science from Shivaji University, Kolhapur. Currently, she is working as Assistant Professor in the Department of Computer Science, Shivaji University, Kolhapur. Her areas of interest are Data Mining, ICT, Algorithms, Theory of Languages. She has participated in several national and international conferences and has published more than 15 papers in International and National journals of repute.