ORIENTAL JOURNAL OF
COMPUTER SCIENCE & TECHNOLOGY
An International Open Free Access, Peer Reviewed Research Journal
Published By: **Oriental Scientific Publishing Co., India.**
www.computerscijournal.org

ISSN: 0974-6471
June 2013,
Vol. 6, No. (2):
Pgs.133-140

Hindi Language Interface to Database using Semantic Matching

ASHISH KUMAR[1] and KUNWAR SINGH VAISLA[2]

[1]Department of Computer Science and Engineering, BTKIT, Dwarahat, Almora, Uttarakhand,(India).
[2]Associate Professor, Department of Computer Science and Engineering, BTKIT,
Dwarahat, Almora, Uttarakhand, (India).
Corresponding author E-mail :[1]kumarash1986@gmail.com, [2]vaislaks@rediffmail.com

(Received: June 01, 2013; Accepted: June 10, 2013)

ABSTRACT

In the world of computing, information plays an important role in our lives. One of the major sources of information is database. Database and Database technology are having major impact on the growing use of computers. Almost all IT applications are storing and retrieving the information or data from the database. Database Management Systems (DBMS) have been widely used for storing and retrieving data. However, databases are often hard to use since their interface is quite rigid in co-operating with users. For storing and retrieving the information from database requires the knowledge of database language like SQL. Structured Query Language (SQL) is an ANSI standard for accessing and manipulating the information stored in database. However, everyone may not be able to write the SQL query as they may not be aware of the syntax and structure of SQL and database respectively. In India, the natural language of people is mainly Hindi. Also large number of e-governance applications use database. So, to use such database applications with ease, people who are more comfortable with Hindi language, requires these applications to accepts a simple sentence in Hindi, and process it to generate a SQL query. The SQL query is further executed on the database to produce the results. Therefore, any interface in Hindi language will be an asset to these people. This paper discusses the architecture of mapping the Hindi language query entered by the user into SQL query.

Key words : HLIDB (Hindi language interface to database), NLIDB (Natural language interface to database), NLP (Natural Language Processing), Semantic matching, Tokenizer.

INTRODUCTION

Database Management System is a collection of interrelated data and set of programs to access those data. Database systems are designed to manage large bodies of information. To access this information, we should have the knowledge of Structured Query Language (SQL). Internet is the largest data provider in today's date and it caters to users of all kinds. The vastness of data makes it mandatory that data is saved in an organized manner so that it is easy to search, retrieve and maintain. For this purpose the most logical and commonly used storage method is by the use of databases. But to efficiently use or maintain any database the knowledge of languages such as SQL becomes essential. This would limit the use of data to only those users who

have the knowledge of these languages. Hence, an easy to use user interface comes into picture which would facilitate diverse users to access data. An end user normally doesn't know SQL. There is a need to design and develop an interface in the local language so that user can easily use that system without the knowledge of English language. So, in order to address these issues we have developed Natural language Interface to Database. With the help of this interface, the end user can query the system in natural language like English, Hindi, Telgu, etc., and can see the result in same language. NLIDB system is proposed as a solution to the problem for accessing information in a simple way, allowing ideally any type of users, mainly inexperienced ones; to retrieve information from a database (DB) using natural language (NL).

Natural Language Processing (NLP) is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things. NLP researchers aim to gather knowledge on how human beings understand and use language so that appropriate tools and techniques can be developed to make computer systems understand and manipulate natural languages to perform the desired tasks. The foundations of NLP lie in a number of disciplines, computer and information sciences, linguistics, mathematics, electrical and electronic engineering, artificial intelligence and robotics, psychology, etc. Applications of NLP include a number of fields of studies, such as machine translation, natural language text processing and summarization, natural language interfaces to database, multilingual and Cross Language Information Retrieval (CLIR), speech recognition, artificial intelligence and expert systems, and so on[1]. Next section discusses the literature survey of some already existing systems. Then it deals with system details and architecture of the system.

Literature Review

The first attempts at NLP database had been done many years back and are as old as any other NLP research. Accessing any query and information from the database in natural language is very user friendly, convenient and "free from worry" method to access the data, especially to another and casual users who cannot understand database query language such as SQL[2].

Natural Language Interface to Database (NLIDB)

One of most wide and interesting area of Natural Language Processing (NLP) is the development of a natural language interface to database systems (NLIDB). In the last few decades many NLIDB systems have been developed. Through these systems, users can interact with database in a more convenient and flexible way. Because of this, this application of NLP is still very and widely used today[2]. Natural Language Interface has been a very interesting area of research since past times. The aim of Natural language Interface to Database is to provide an interface where user can interact with database more easily using their natural language and access or retrieve their information using the same[3]. We can also say that NLIDB is a system that converts the query in native language into SQL and vice-versa.

Advantages of NLIDB

The NLIDB systems allow the people to communicate with database in the same way they communicate with each other. The main advantages of Natural language Interface to Database are given below[4].

1. No requirement of Artificial Language: User is not forced to learn an artificial communication language. Formal query languages like SQL are difficult to learn by non-computer specialists.
2. No need of Training: No special training is required before using the natural language interface. It is highly user friendly and easy to use by the end users.
3. Simple and easy to use: The natural language interface in very simple and easy to use because the end users write the query in their native language.
4. Better for some question: It has been argued that there are some kind of questions (e.g. questions involving negation, or quantification) that can be easily expressed in natural language, but that seem difficult (or at least tedious) to express using

graphical or form-based interfaces.

5. Easy to use for multiple database tables: Queries that involve multiple database tables are difficult to form in graphical user interface as compared to natural language interface.

Disadvantages of NLIDB

Many NLIDB systems have been developed so far for business purpose use but use of NLIDB system is not broad-spread and it is not the primary choice for interfacing to database. This lack of acceptance is mainly due to the large numbers of deficiencies which are given below:

1. Linguistics coverage is not obvious: Currently all NLIDB systems can understand some subsets of a natural language but it is quite difficult to define these subsets. Even some NLIDB systems can't handle certain query belong to their own subsets. This is not the case of formal language like SQL. Because the formal language coverage is obvious and provide the corresponding answers of any statements that follow the given rules[2].
2. Linguistics vs. conceptual failure: When NLIDB system fails, the system does not give any explanation of what causes the system to fail. Some user try to rephrase the question or just leave the question unanswered[2].
3. Inappropriate Medium: It has been argued that natural language is not an appropriate medium for communicating with a computer system. Natural language is claimed to be too verbose or too ambiguous for human-computer interaction[4]. NLIDB users have to type long questions, while in form-based interfaces only fields have to be filled in, and in graphical interfaces most of the work can be done by mouse-clicking. In natural language interface user has to type full sentence with all the connecters (articles, prepositions, etc) but in graphical or form based interfaces it is not required[4].

Some Existing NLIDB Systems

1. LUNAR System[5] deals with the database which has the information about the specimen of rocks brought back from the moon. In this two databases are used one for chemical analysis and second for literature representation.
2. LADDER System[6] is interfacing with the database which stored the data about the US NAVY ships. This system was developed in the late seventies (1978) which uses a worthwhile and meaningful grammar to analyze interrogations and doubts to query a distributed database.
3. CHAT-80 System[4] interfacing the database which stored the data about the geographical facts (such as, oceans, seas, rivers, countries, etc) basically it has the data of around 150 countries. CHAT-80 was introduced in the early eighties. This system was very optimal, result oriented, impressive and sophisticated. It was implemented in PROLOG which transformed an English Query into Prolog expressions and these expressions were evaluated against the Prolog.
4. PLANES System[7] deals with the database which has the information about US NAVY 3-M (maintenance and material management) and aircraft maintenance and flight data. It was developed at the University of Illinois coordinated science laboratory in the late seventies.
5. EUFID System[8] consists of three important modules, without counting the DBMS. First being the analyzer module, followed by mapper module and the last being the translator module.
6. ITS (Intelligent tutoring system)[3] assist the student or user in SQL without interruption of human teacher. It also provides immediate and customized instruction to learners, usually without interruption of human teacher.
7. TEAM System[2] was developed in 1987. A major section of that era was devoted to portability issues. Team was designed by database administrator to be configured easily with least or no knowledge of NLIDBs.
8. DATALOG System[2] based on cascaded ATN grammar, is an English database query system. It achieves a greater degree of portability and extendibility by providing distinct representation schemes for native

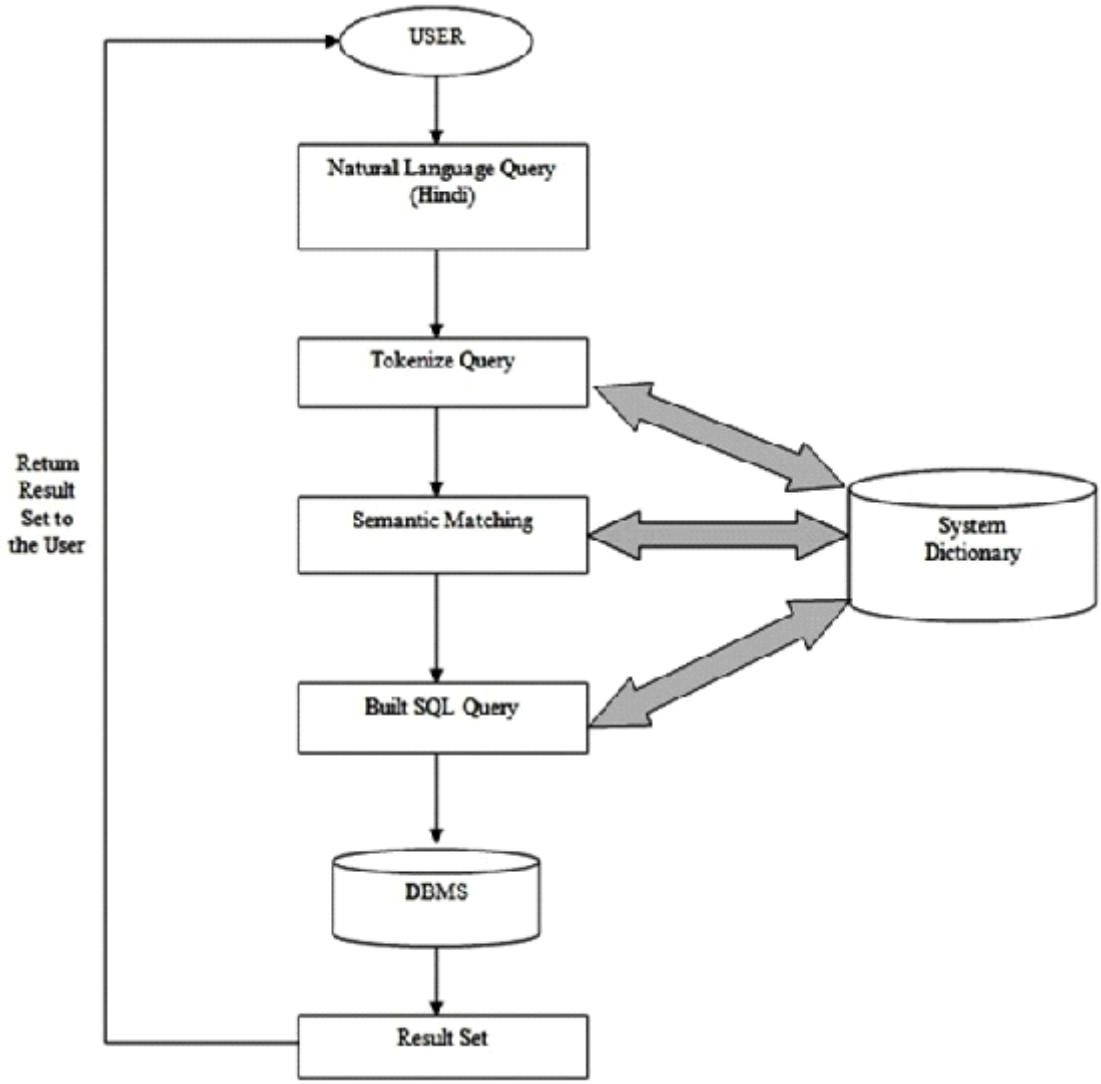

Fig. 3.1 Hindi Language Interface to Database (HLIDB)

language knowledge, general word knowledge and application domain knowledge.

9. START System[2] uses several types of functions which are language dependent such as Parsing, Natural Language annotation to present the suitable information segments to the users.

Hlidb System Details And Architecture

The proposed system maps the Hindi language query to SQL query. So in order to store and access the information in Hindi, a graphical user interface has been used. This graphical user interface requires some basic training for using the system. With the help of this interface, the end user can query the system in Hindi, and can see the result in same language. This gives the idea of Hindi Language Interface to Database (HLIDB). HLIDB system is proposed as a solution to the problem for accessing information in a simple way, allowing ideally any type of users who knows the Hindi language, mainly inexperienced ones; to retrieve information from a database (DB) using natural language (NL)[9]. It is a type of computer human interface. A database is made up of three types of elements: relations, attributes and values. Each element is distinct and unique: an attribute element is a particular column in a particular relation and each value element is the value of a

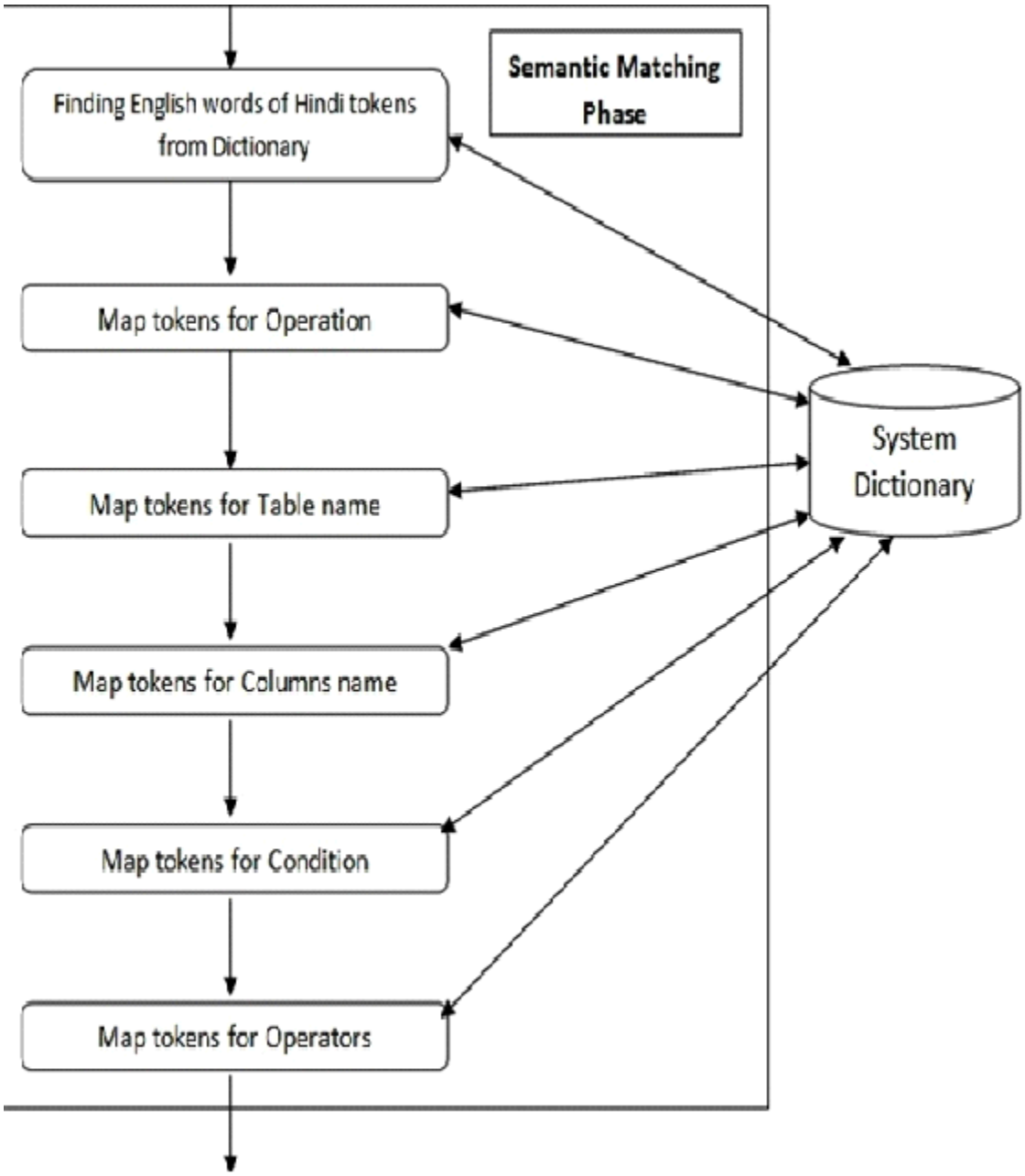

Fig. 3.2: Workflow of the system

particular attribute. A value is compatible with its attribute and also with the relation containing this attribute. An attribute is compatible with its relation. A token is set of word stems that match a database element. Many different tokens might match the same database element, and conversely, a token might match several different elements.

The architecture of the Hindi language interface to database system is shown in the Fig 3.1. Architecture of the system is divided into three phases:

- o Tokenize the query.
- o Semantic Matching and Generation of SQL query.
- o Execution of SQL query.

Workflow of the System

1. Natural Language Query: In this phase user can query the system in Hindi language. The working of the system has been explained with the help of an example (sentence) as shown in (3.1).

उस विद्यार्थी का नाम, अंक बताओ जिसका अनुक्रमांक 12 है...3.1

2. Tokenize Query: User is providing input in the form of simple sentence to the system in Hindi Language. System tokenizes that sentence and produce tokens that are to be searched in the system dictionary for mapping. The tokenizer for the input sentence given in (3.1) is shown in (3.2).

उस, विद्यार्थी, का, नाम, अंक, बताओ, जिसका, अनुक्रमांक, 12, है...32

3. Finding English words of Hindi tokens: After tokenizing, the tokens are mapped to their English with the help of a system dictionary. Basically, it works only on logical word in input sentence. So, we neglect the some useless words in the query. The English words generated by the dictionary for input sentence given in (3.1) are shown in (3.3).

उस-(neglect)
विद्यार्थी-student
का-from
नाम-name
अंक-marks
बताओ-select
जिसका-where ...3.3
अनुक्रमांक-roll no
12-12
है-(neglect)

4. Map tokens for Operations: After finding all the tokens, their corresponding English words are found from the system dictionary which contain the Hindi and English words. All the words of input sentences are stored in the dictionary except proper noun. After finding the English words from the dictionary, input sentence is analyzed so that system understands the type of the query entered, i.e., information retrieval, update, insert or delete query. If the input sentence contains the word "करो" or "कीजिए", system understands it is an "update" query. If the input sentence contains the word "हटाओ" or "मिटाओ", system understands it is a "delete" query. If the input sentence contains the word "बढ़ाओ", system understands it is an "insert" query; otherwise it is information retrieval query. The input sentence given in (3.1) contains the word बताओ. So, the system understands it is an information retrieval query and "select" keyword is generated by the system.
5. Map tokens for Table name, Columns name, Condition and Operators: The system contains the tables like TABLE_NAME, COLUMNS_NAME, CONDITION, OPERATORS. TABLE_NAME and COLUMNS_NAME table contain the data required to map tables name and columns name that are found in Hindi sentence into their corresponding English words respectively. TABLE_NAME table contains mapping for names of available tables name and COLUMNS_NAME table contains mapping for names of available columns name. CONDITION table contains mapping for "where" keyword of SQL query. OPERATORS table contains mapping of conditional words or operators like less than (<), greater than (>).
6. Generates SQL Query: After mapping all the tokens, SQL query is generated according to the type of the input Hindi sentence. After that, the query is executed on the Database. The SQL query generated by this function for input sentence given in (3.1) is shown in (3.4).

Select name, marks from student where roll no=12. ..3.4

7. Execute SQL Query: The SQL query is executed on the database. The retrieved data from the database is provided to the user as Hindi output.

Algorithms To Implement The System

The system's dictionary contains the tables SQLWORDS, TABLE_NAME, COLUMN_NAME, CONDITION, OPERATORS, which have two columns Token_Word and Mapped_Word.

The SQLWORDS table contains the

keywords like, select, update, insert, delete and from. TABLE_NAME table contains the name of all tables in the database. COLUMN_NAME table contains the columns name of tables. CONDITION table contains the only "where" keyword and OPERATORS table contains the all operators like <, >, = etc.

Algorithm

1. Enter Hindi input sentence in the system.
2. Tokenize the input Hindi sentence and generates the tokens.
3. Map each token in the system's dictionary for finding what is the type of the query i.e. select, update, delete and insert type:- Call algorithm-3.1
4. Map the token in the system's dictionary for finding the table name in the query:- Call algorithm -4.1
5. Map the token in the system's dictionary for finding the columns name in the query:- Call algorithm -5.1
6. Map the token in the system's dictionary for finding the condition in the query:- Call algorithm -6.1
7. Generates the SQL query.
8. Execute the SQL query on DB.

Algorithm

Finding Type Of The Query

Traverse each token and check:-

1. If the input sentence contains the word "करो" or "कीजिए", system understands it is an "update" query.
2. If the input sentence contains the word "हटाओ" or "मिटाओ", system understands it is a "delete" query.
3. If the input sentence contains the word "बढ़ाओ", system understands it is an "insert" query.
4. The input sentence contains the word बताओ. So, the system understands it is an information retrieval query and "select" keyword is generated by the system.
5. Otherwise system gives an error.

Algorithm

Finding Table Name From Input Hindi Sentence

1. The proposed system searches the tokens first in the TABLE_NAME table in the database. If the token matches with the value stored in this table system maps that value with the table name and finds the table name to which that query associates.
2. Is the system doesn't find table name; it searches the tokens which match with the column names stored in a file. This file contains column names and their corresponding table names.
3. If any token matches with the column name value, corresponding table name retrieves from it.
4. If both the cases don't apply, then system reports an error.

Algorithm

Finding Column Name From Input Hindi Sentence

1. System searches tokens generated by input sentence in COLUMNS_NAME table. If any token matches with the value stored in the table, system maps that value with the column name and finds that column name to which that query associates.
2. System again searches in COLUMNS_NAME table for column names that appear in condition.
3. If no token matches with column name, system executes the query "select * from table-name".

Algorithm

Finding Condition From Input Hindi Sentence

1. System searches token generated by input sentence in CONDITION table. If any token matches with the "where" keyword stored in the table, system maps that value with the condition name.
2. If token maps with "where" keyword, then system map the token in the system's dictionary for finding the operators in the query. If the tokens are matches with any of the words present in the table OPERATORS the system map the value of their corresponding symbols.
3. If no condition occurs, system executes the query "select column-name from table-name".

CONCLUSION

The proposed Hindi Language Interface to Database accepts the query in Hindi, tokenize the sentence (split sentence into words called tokens) and maps the Hindi words with their corresponding English words with the help of system dictionary maintained. After mapping the sentence, it is checked whether it is data retrieval, update, insert or delete type of sentence. This is done by analysing the input Hindi sentence. After analysing, the input Hindi sentence, table names, column names, condition and operators are searched in the dictionary. After mapping, SQL query is generated and executed on database to display the result set to the user.

REFERENCES

1. Gobinda G. Chowdhury, "Natural Language processing", Strath cis publication, pp 1 (2001).
2. G. Hendrix, E. Sacerdoti, D. Sagalowicz, and J. Slocum, "Developing a Natural Language Interface to Complex Data", *ACM Transactions on Database System*, pp. 105-147, (1978).
3. B.Sujatha, Dr.S.Viswanadha Raju, and Humera Shaziya, "A Survey of Natural Language Interface to Database Management System", *International Journal of Science and Advanced Technology*, ISSN 2221-8386., **2(6)**, (2012).
4. D.L. Walt "An English Language Question Answering System for large Relational Database", communication of the ACM, pp. 526-539, (1978).
5. Gauri Rao, Chanchal Agarwal, Snehal Chaudhry, Nikita Kulkarni, Dr. S.H. Patil "Natural Language Query processing using Semantic Grammar", *International Journal on Computer Science and Engineering*", **2(2)**, 219-223, ISSN: 0975-3397 (2010).
6. A. Kaur, and P. Bhatiya, "Punjabi Language Interface to Database", M.tech thesis, Department of CSED, Thapar University, (2010).
7. Mrs. Neelu Nihalani, Dr. Sanjay Silakari, Dr. Mahesh Motwani. "Natural language Interface to Database using Semantic Matching", *International Journal of Computer Application,* **31(11)**, ISSN: 0975 - 8887 (2011).
8. M. Joshi, R.A. Akerkar, "Algorithm to improve Performance of Natural Language Interface", *International Journal of Computer Science & Application*, **5(2)**, pp. 52-68, (2008).
9. M. Templeton and J. Burger, "Problems in Natural Language Interface to DBMS with Examples from EUFID", In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, pp 3-16 (1983).