I.J. Information Technology and Computer Science, **2018, 6, 1-10**
Published Online June 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2018.06.01

SDRED: Smart Data Retrieval Engine for Databases

Mir Shahnawaz Ahmad
SSM College of Engineering & Technology, J&K, India.
E-mail: mirshahnawaz888@gmail.com

Syed Rameem Zahra
Ph.D. Scholar Department of Computer Science and Engineering National Institute of Technology Srinagar Hazratbal, Srinagar, J&K, 190006, India.
E-mail: rameemzahra@gmail.com

Received: 05 January 2018; Accepted: 07 May 2018; Published: 08 June 2018

***Abstract*—**Today computers are continuously betrothed in almost all domains and organizations. Thus, databases act as the heart for storing and retrieving information that contain huge digital data. However, in order to interact with such databases, it is necessary to have knowledge about the Structured Query Language (SQL), which is difficult for non-expert users to understand and manipulate. So, there is an emergent need to develop a smart and a user friendly computational technique to interact with databases. The current work proposed a smart technique that can handle such context. The proposed "Smart Data Retrieval Engine for Databases (SDRED)" provided an environment that allows a non-expert user to write and to execute the database queries easily. Furthermore, it retrieved the data stored in databases without a prior knowledge of the SQL. SDRED, which enables the non-expert user to write database queries in natural language (such as English) and to convert them to their SQL query equivalents. The current work presented a detailed design and evaluation for the proposed system by executing different database queries in English. The results established that SDRED successfully converted the non-expert user's natural language queries into their equivalent SQL queries, thereby providing an easy and user-friendly environment to interact with databases.

***Index Terms*—**Digital data, non-expert user, Structured Query Language (SQL), Natural Language Queries.

I. Introduction

Recently, the increased technology progress leads to a huge amount of data and databases in several domains including small organizations as well as large educational institutes. Such databases require a database management system (DBMS) for their manipulation and management [1]. The DBMS is responsible for performing all types of operations on the database. It represents the data from the databases to the user in an independent way that differs than the actual stored data in the physical medium. Thus, the DBMS has lot of functionality, where many of its commands can perform a particular operation. In order to distinguish between them, these commands are grouped into three main languages, namely data definition language (DDL), data manipulation language (DML) and data control language (DCL) [2]. Furthermore, the relational databases is considered to be the most common type of databases, where data is stored in the form of tables and each table contains many tuples. Each table represents a mathematical relation between the attributes of table. The commands used to retrieve, update or delete any data from relational database, are declarative sentences that are known as SQL (structured query language). The SQL is an ANSI (American National Standards Institute) standard language for accessing and for manipulating the data stored in relational database, which is supported by the DBMS.

The statements in a SQL query are declarative and are Boolean expressions, i.e. either true or false. Different commands are used for manipulating the relational databases using the SQL, where each command has a particular syntax or writing. Memorizing this syntax for each type of query is a bit tedious task. Also, for a non-expert user (one without a prior knowledge of SQL) it is almost impossible. But, in today's world, not only the expert database user, but also non-expert users requires databases for different purposes. Thus, an alternative approach for accessing the databases information easily motivates several researches to create more intelligent databases [3] that can understand queries of the user without requiring the user to have an expertise in SQL. For example, the natural language processing method [4] can be considered as the database or the DBMS is so intelligent that it can understand the queries in natural language in an easy way. Also, with this creation everybody will be comfortable and can easily access the database, as if he/she is talking to a person and not a machine.

However, for non-expert database users, the designers provide an easy interface to extract only limited

Copyright © 2018 MECS

information from the database, depending on the fields provided by the designer. Therefore, if the user wants to extract other fields of database, it is not able to do so because of the limited fields provided by the software designer. In order to overcome this problem, an intelligent engine is proposed in the current work to assist the non-expert users for interacting with the database using a natural language (English). Afterward, the user can extract any data easily without any knowledge about the SQL. Hence, the proposed enhanced system's user can extract limitless information that is present in the database. This proposed scheme is based on the idea of constructing an intelligent layer that works on the top of normal SQL query processing engine [5]. The formed intelligent layer will be responsible for transforming the user's natural language query into SQL query, which can then be easily understood by the SQL query-processing engine. The intelligent layer thus formed is responsible for transforming the user's natural language query into SQL query, which can then be easily understood by the SQL query-processing engine. Moreover, when a common user types a query in natural language, there is a possibility of incurring some mistakes such as incorrect table name, attribute name or any word necessary for the formation of SQL query. The proposed intelligent layer is so smart that it is able to detect those mistakes and will be able to correct them. Finally, the proposed system can construct a correct SQL query out of the user's natural language query. Thus, the intelligent layer will not only be smart, but will be robust, giving an easy environment to database user to interact with database and retrieve any information from the database.

The organization of the remaining sections is as follows. Section II includes the related work followed by the methodology and the mathematical model in Sect. III and IV respectively. The proposed system is demonstrated and discussed in Sect. V. The results and conclusion are given in Sect. VI. Finally, the conclusion is reported in Sect. VII.

II. Related Work

Researchers are interested with designing an easy and user-friendly environment for accessing the data stored in databases due to the widespread of database applications in the different domains [6]. Bertino et al. [7] designed an intelligent database system as a first step to make more intelligent databases. These databases were able to process the user queries in much similar way like that by a human brain, and hence were intelligent. However, for these intelligent databases to be successful, a human friendly way of interaction is required. To achieve that requirement, Lewis et al. [8] proposed the engagement of natural language processing to provide a natural language interface to the database user.

Typically, it is evident that using natural language for interacting with databases requires realizing different patterns in human speech [9], and consequently formulating different rules to transform natural language query into computer understood language. Valverde et al. [10] designed an ontology-based system for querying DBpedia. The authors used a rule-based approach [11], where a particular rule triggers a unique set of operations to be performed. However, some mistakes may occur while writing the database query in natural language. In order to detect such mistakes for further replacement with correct words, researchers have proposed a technique known as Levenshtein distance measurement technique [12]. Also, Ayan et al. [13] proposed a technique "Clarifying natural language input using targeted questions" that used some predefined questions for understanding the natural language input from the user and hence provided suitable results. Llopis et al. [14] provided an example for creating a natural language interface to query a database. The authors explain different constraints and data structures used for creating this interface. Afterward, Li et al. [15] proposed an interactive interface for querying relational databases that used an interactive approach for understanding the user's natural language queries. This interactive approach was inappropriate tool for understanding the natural language queries, as it was time consuming and complex. This system was continuously interacting with the database user in order to understand a single natural language query of database user through asking multiple questions to clearly recognize a single natural language query. Hence, the system can process multiple queries in order to successfully understand and execute a single natural language query, thereby making the system more complex.

From the preceding literatures, a novel smart technique (SDRED) is proposed for querying the databases using natural language. The proposed technique is superior to the previously mentioned system in [15] as it has several features including:

- Instead of asking continuous questions or continuously interacting with the database user, the proposed system only takes a single query from the database user in natural language. Afterward, it processes the single query and finally provides desired results to the user, thus making the interface much simpler with less complexity compared to the work in [15].
- The proposed technique is robust as it corrects the mistakes of the database user automatically that may occur while writing queries in natural language.
- There are different methods embedded in the proposed technique that make it a new system for querying the databases in natural language.

III. Methodology

Generally, the SQL is necessary query language to interact with a relational database. However, the limitations in using SQL include the tedious task of remembering the syntax of different SQL queries. Thus, the current work proposed a SDRED to overcome this limitation and to allow a common user-friendly and easy

Copyright © 2018 MECS

access and interaction database. It allowed the database users interaction with the database in a natural language like English language. In the proposed system, the databases intelligent system SDRED design will be working on the top of the standard SQL query engine. The SDRED is responsible for taking the input from user who typed in natural language, namely the English language to extract useful information from the input to transform it into an SQL query. The generated SQL query is than passed to the standard SQL query engine to execute the SQL query and to produce the desired results for the user. Several training structures were implemented in the SDRED to train the system for the desired processing and for extracting useful information from the input, which finally transformed into SQL query. These training structures train the system for the desired processing and also guide the system when a totally new type of information is encountered. Such training structures are the key which make the system intelligent. The proposed intelligent system consists of different phases/modules through which the user's input passes and formulates an SQL query at the end. Moreover, each module contains different training structures that guide them in performing specific tasks and support them to discover new information during processing and preserving the obtained information for future uses. During the user's natural language query processing, the main motto is to obtain the names of relational database tables or the attributes of tables or any values in a particular table, and finally use it to construct the SQL query. Thus, the Meta data of database is used for searching the above mentioned information that contains all the information regarding a database. It will be also used it to extract useful information from the user's natural language query. Once the intelligent system is able to obtain the names of database's table or the attribute name of any table or both, then the SQL query construction will take place. The whole intelligent SDRED system and its different modules are shown in Figure 1.

The different modules and training structures used in each module are discussed in following subsections.

A. Lexical Analysis

The first step in the processing of the user's query (Q) is the Lexical analysis [16], which is typed in English. Each word in the natural language query is separated. The unnecessary words/information is removed using stop-word (Ws) training structure that consists of all the unwanted words called stop-words. Subsequently, whenever the user's natural language query has certain stop-words in it, they are checked using this data-structure and are thus removed. Initially, this training structure consists of basic Ws, such as if, to, so, at, in, on, etc., but with time when more natural language queries are supplied to the SDRED, this training structure is expanded. When some new Ws are discovered during the processing of user's natural language queries, then they are also automatically added to Ws, thus making it an intelligent training structure.

Once all the stop-words contained in the user's natural language query are removed, then the useful words (known as tokens) are passed to the next phase for further processing. Accordingly, during lexical analysis phase the useful words (Tokens) are generated. These generated tokens may contain the names of database tables or table attribute names or any value contained in database table.

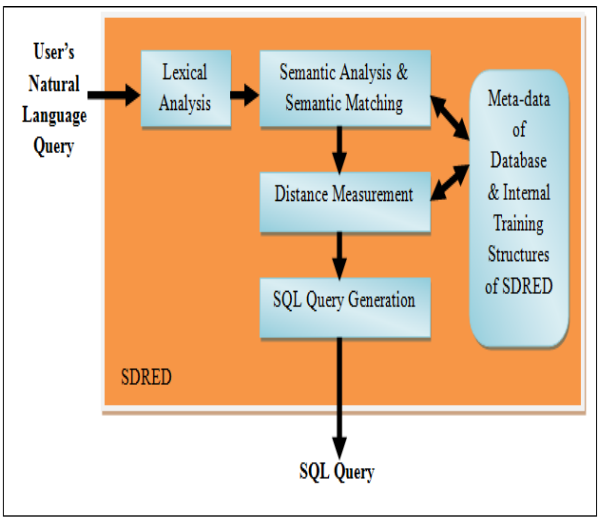

Fig.1. Detailed view of SDRED system

The token generation algorithm is as follows:

Algorithm 1: Token Generation Algorithm

Input: The natural language query entered by database user
Separate each word of the input query
Sort the words in any contiguous memory (a)
for i = 1 to size_of (a)
    If a[i] == Ws, then
        Remove the word from 'a'
    else
        No change to 'a'
End for
Pass the stored words in 'a' to next phase as tokens
Output: Multiple tokens

The procedure for the token generation algorithm is illustrated in Figure 2.

B. Semantic Analysis and Semantic Matching

Kaufmann & Bernstein [17] presented an evaluation of natural language interfaces by semantic analysis. Similar technique based on the use of semantic analysis for evaluating the natural language input was presented by Habernal and Konopik [18]. In the current work, similar technique for both the ones in [17, 18, 19] is employed. The input to this phase is the useful words (tokens) extracted by the lexical analyzer. This phase is responsible for several task including i) matching the tokens with the actual information in the database, ii) understanding their meaning, and iii) replacing the correct words in place of wrong tokens (if any). Thus,

Copyright © 2018 MECS

even if the user has typed wrong information in the input, this phase is responsible for correcting the mistakes, thus again making the system intelligent.

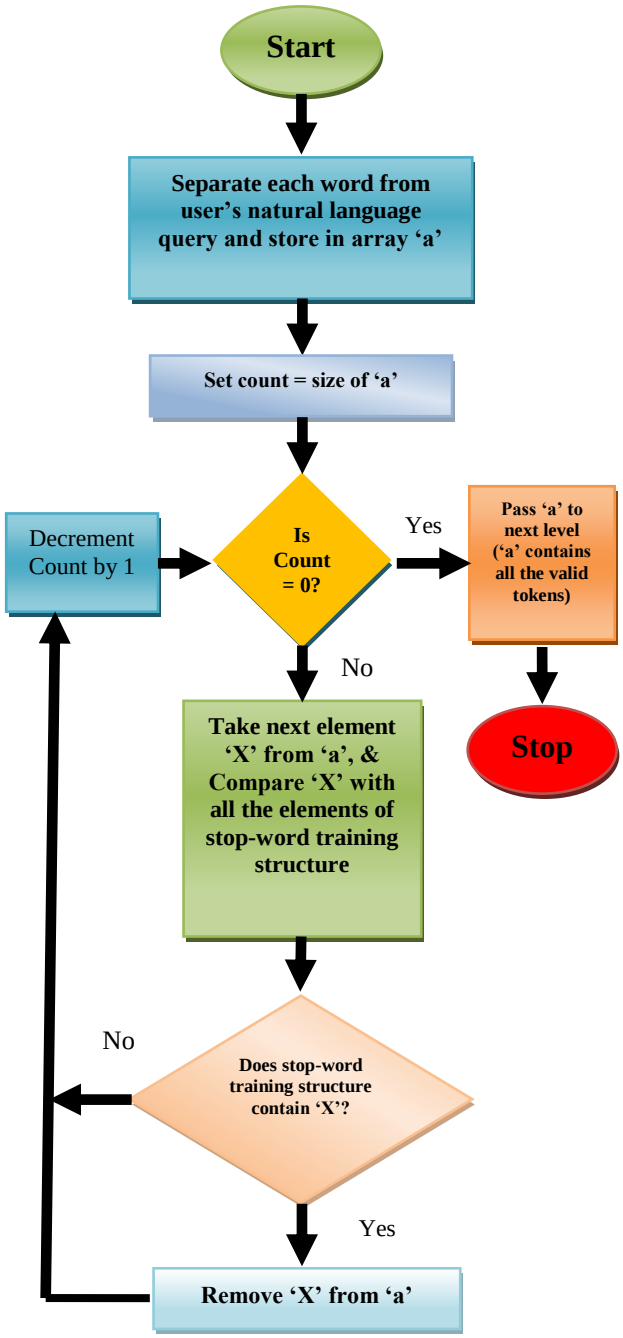

Fig.2. Procedure for Token Generation

This phase processes each token by searching it in the Metadata of database. If the token is not present in the database, then it is replaced with the help of semantic measurement technique [20] and context modeling technique [21], where the token (whose information is not present in the database) is replaced with that token which is present in the meta data of database having the least semantics distance, i.e. which is similar to the meaning of the given token. Thus, through this phase, even if the user enters those table names or attribute names in natural language query which are not in the database (but their synonyms are present in meta data of database), this module is able to replace the wrong information with the correct one, which is present in the metadata of database and which is similar to the meaning of user's input word. Thus, for the efficient working of this module, it required to continuously interact with the metadata of database and its training structures.

The following training structures are proposed for this phase to perform the desired processing of tokens:

- Expression mapping set (Emap): This training structure contains all the relational expression words (e.g. greater than, less than) and their corresponding mathematical expressions (e.g. >, <) that are used in the SQL query. The semantic analyzer uses Emap to search a relational expression words from the tokens list and replaces them with the actual relational expression. Thus, they can be used to construct the final SQL query. In this way, the training structure maps the relational expression words written in natural language to the actual relational expressions used in standard SQL queries.
- Conjunction set (Cs): All the conjunction words used in natural language (English) are stored in this data structure, and are checked for a conditional statement by the semantic analyzer. If a successful match between the extracted tokens and words in the Cs takes place, then there is a conditional statement in the user's query and it needs to be processed accordingly, for which the system is intimated by setting the conjunction flag. When the conjunction flag is set, it means that the user's natural language query has a conditional statement in it. Consequently when transformed, the resulting SQL query too will have a conditional part.
- Semantic set (Sc): This training structure contains all the synonyms of the corresponding database table names and the attribute names of tables. This training structure is used when the user types information which is not present in the metadata of database, but is similar to the meaning of the information stored in the metadata of database, e.g. if the user types "enrollment number" in the natural language query, but it is neither the name of any table nor the name of any attribute (i.e. not present in metadata of database), but it matches with the synonym of one of the attribute name – "roll number" (which is present in metadata of database), then the semantic analyzer will replace "enrollment number" with "roll number" for constructing a correct SQL query.

The semantic analysis algorithm is illustrated as follows:

Copyright © 2018 MECS

Algorithm 2: Semantic analysis algorithm

Input: Token set 'a'
for (i = 1 to size_of(a))
Step 1: Search the token in the Meta data (M) of the database
If a[i] ∉ M
Then use Sc to find its semantic match & replace it with actual token.
Otherwise
No change to 'a'
Step 2: Search the token in Emap
if (a[i] ∈ Emap)
Then replace it with its mathematical expression in Emap.
Else
No change to 'a'
Step 3: Search the token in Cs
if (a[i] ∈ Cs)
Then set the conjunction flag.
End for
'a' contains all the finalized tokens, which are used to generate the SQL query out of user's natural language query.
Pass this token set to SQL query generation module.
Output: Semantically matched and corrected token set

The procedure for semantic analysis is illustrated in Figure 3.

C. Distance measurement

When the semantic analyzer corrects the tokens of user's natural language query, then all the tokens are passed to this module for further processing. After semantic analysis, there are certain tokens whose semantic analysis is not possible due to spelling mistake or by any other reason, thus a distance measurement technique is used for those tokens. In the current work, the Levenshtein distance/ edit distance [12] is used as distance measurement technique for calculating the distance between two tokens/words.

The Levenshtein distance measures the number of edits, which are needed to perform in order to transform one string into another. The allowed edit operations include: insertion, substitution or deletion. There are other edit distance measures such as Hamming distance, Longest Common Subsequence (LCS) etc. but the reason to choose Levenshtein distance as the edit distance are the following: i) Levenshtein distance allows all the important edit operations to be performed on the strings i.e. insertion, deletion and substitution. On the contrary, while LCS allows insertion, deletion, Hamming distance allows substitution only. Hence, while Hamming distance can only be used for strings/words of equal length, LCS cannot be used in cases where we require substitution as well. ii) Levenshtein is one of the most famous edit distances used in software that help in translation of natural languages. iii) Levenshtein is very helpful when the objective is to find matches for short strings in long texts. In the proposed intelligent system, a threshold as

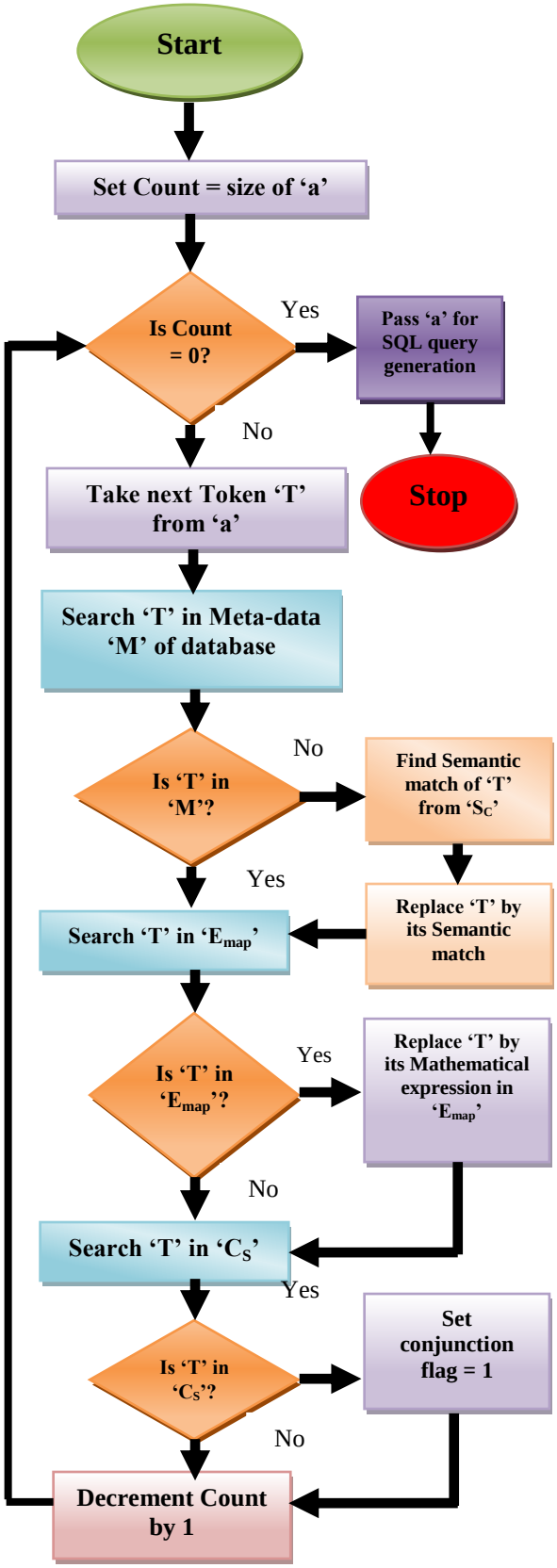

Fig.3. Procedure for Semantic Analysis

been set, and if the Levenshtein distance between two words (one being the un-matched token and the other being the actual table name or attribute name) is less than or equal to threshold then the natural language word is

Copyright © 2018 MECS

replaced by the actual word in the database, i.e. if the calculated distance between a token and a word in the metadata of database is less than or equal to the threshold level, then the token is replaced by the word in metadata. Thus, by using this technique, the query-processing engine will become more robust as it is able to correct the mistakes of user automatically.

After the successful completion of distance measurement operation by levensteins distance [12], the remaining tokens, which are not being used for SQL query generation, are analyzed and are thus added to the already existing training data present in different training structures as feedback [15]. Accordingly, the proposed intelligent system will automatically update the training data whenever a natural language query is passed to it, making the system more robust and efficient.

D. The SQL Query Generation

This phase is the final phase in the natural language query processing, where the valid tokens are transformed into a valid SQL query and is sent to the standard SQL query engine for the execution of SQL query. The process of transforming the valid tokens into the SQL query is systematic process, where different components are appended to the query at different steps and not at once i.e., the retrieved table names or attribute names are appended to the query at different steps, where they fit in the SQL query. If the user directly types the SQL query then it is directly given to the standard SQL query engine for the execution because it is already present in the form that is easily understood by the SQL query engine. At this phase, the intelligent layer has all the necessary information (e.g. table name and attribute name) required to construct the SQL query, which is than formulated in the form of SQL query and is passed to the SQL query engine for execution. Thus, the SQL query is formulated using the following expression:

$$P1 + attribute\text{-}name + P2 + table\text{-}name + P3 + (attribute\text{-}name + P4\ )repeatedly \quad (1)$$

Where, P1, P2, P3 and P4 are the different parts to be appended to the SQL query by the intelligent layer at different stages (e.g. P1 can be "select *" or "select" or "select sum()"), and each part, together with conjunction flag will determine whether the next part be appended to the query or not, and also what will be the structure of next part. The "attribute-name" and "table-name" are the names of attribute and tables of database that the intelligent system has detected during the processing of user's natural language query. The last part of the final query formulated can be repetitive based on the structure of P1, P2 and P3. After appending all the parts to the SQL query by the intelligent layer, the SQL query formulated will be passed to the standard SQL query processing engine, which will execute the SQL query to retrieve the desired results.

IV. Mathematical Model

The mathematical model of the proposed model explains the whole processes of converting the natural language query into SQL query mathematically. Represent the natural language query by 'Nq' and the final SQL query generated by the system is represented by 'Sq'. The different training structures are represented by the stop-word by Ws, the expression mapping set is represented by Emap, Sc denotes the semantic set and Cs represents the conjunction set. Thus, after the Ws removal process, the natural language query input 'Nq' is converted into set of tokens 'T', which can be mathematically represented as:

$$T = N_q - W_s \quad (2)$$

These tokens are then passed for semantic matching process, where the token set is modified, and the resulting token set is calculated as:

$$T_{New} = T - (W_{Emap}) + (E_{map}) \quad (3)$$

Where, "WEmap" is the word in the expression mapping set (Emap) which matches with a particular word of user's natural language query, and "EEmap" is the actual expression against the matched word in Emap.

The table-name of database and the attribute name of a particular table are calculated as:

$$Table\text{-}name\ (T_n) = T_{New} \cap M_T \quad (4)$$

$$Attribute\text{-}name\ (A_n) = T_{New} \cap M_A \quad (5)$$

Where, "MT" is the set of table-names in metadata of database, and "MA" is the set of attribute-names in the metadata of database. If both Tn and An are empty, then semantic matching is performed due to which the new token set is calculated as:

$$T_{New} = T - (W_{Sc}) + (S_{Sc}) \quad (6)$$

Where, "WSc" is the word in the semantic-set (Sc) which matches with a one of the words of user's natural language query, and "SSc" is the synonym of the matched word in Sc. The process of distance measurement also takes place after semantic matching. Afterward, Tn and An are computed for calculating the names of database tables and attributes.

Thus, the final SQL query generated by the intelligent layer is represented mathematically as:

$$S_q = P_1 + A_n + P_2 + T_n + P_3 + (A_n + P_4)_{repeatedly} \quad (7)$$

Where, P1, P2, P3 and P4 are the different parts appended to final SQL query by intelligent layer at different stage. The SQL query "Sq" thus generated by the intelligent layer is given as an input to the standard SQL query processing engine, which executes it and provides the desired results to the common database users for their natural language queries. Based on the preceding

Copyright © 2018 MECS

methodology and mathematical model, the proposed intelligent system is illustrated in the following section.

V. Proposed System

Figure 4 illustrates the proposed intelligent system, where the user can interact with a relational database system using a Natural language query.

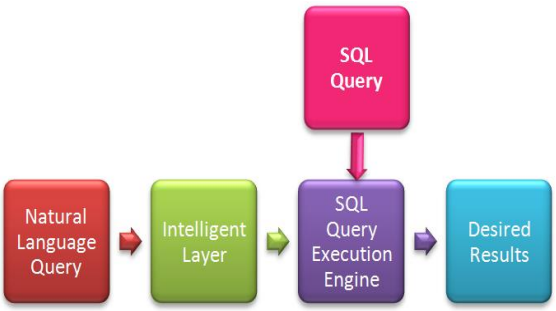

Fig.4. The Proposed scheme overview

In the proposed system represented by the Intelligent Layer in Figure 4, it is unnecessary to remember the syntax of different SQL queries by the user. The database user can type the database queries in natural language, which will be transformed into SQL queries automatically. The basic idea behind the implementation of this system is to design an intelligent layer, which will work on the top of SQL query processing engine. The input to this intelligent layer will be the user's natural language queries as shown in Figure 4. The intelligent layer will then transform the natural language queries into SQL queries that can be easily understood by the standard SQL query-processing engine that will then execute them to provide the desired results to the user. The SDRED steps for executing the natural language query are shown in Figure 5.

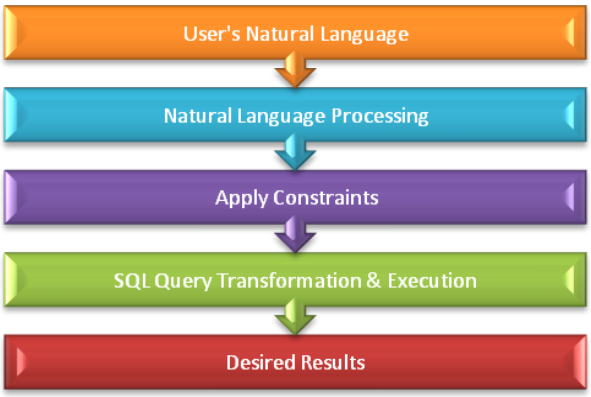

Fig.5. SDRED steps for processing the user's natural language query

Figure 5 illustrated that the SDRED is trained to construct the SQL queries out of user's natural language queries as well as to detect and to correct the errors in the natural language query. Thus, the proposed system will take the natural language input from the user, perform natural language processing on it, apply different constraints and finally formulate a correct SQL query, whose execution yields the desired output to the user. Many training structures were implemented to guide the SDRED in processing the natural language input. The proposed intelligent layer was designed to update the training structures automatically whenever a new query is submitted for execution, thus making the system smart.

VI. Experimental Results and Discussion

The proposed system (SDRED) for database query processing is implemented using ASP.NET with SQL server at backend. The Smart data retrieval engine is designed to allow the common user to type the database queries in natural language (English). Afterward, the intelligent system is responsible for generating the SQL queries out of user's natural language queries automatically. Finally, the output is provided to the user. The different training structures were being implemented using different data structures in ASP.NET. Two University databases are created containing i) records of all the faculty members/workers/students, and ii) details of customers/products of a shopping store. For the proposed SDRED system performance evaluation, different natural language queries (given in Table 1) is used as input to SDRED for execution.

Table 1. Natural Language Queries and their SQL queries generated by "SDRED"

Input:	Natural Language Query – 1:	Display those faculty members who are designated as Assistant professor.
SDRED Output:	SQL Query generated by SDRED:	Select * from faculty as b where b.designation like 'Assistant professor%'
Input:	Natural Language Query – 2:	Show me the details of department with total faculty more than 10 members
SDRED Output:	SQL Query generated by SDRED:	Select * from departments as a where a.Total_Faculty > 10
Input:	Natural Language Query – 3:	Give me the phone number of customer with name mohan
SDRED Output:	SQL Query generated by SDRED:	Select a.name, a,phone_no from customers as a where a.name like '%mohan%'
Input:	Natural Language Query – 4:	Display the total amount in sales
SDRED Output:	SQL Query generated by SDRED:	Select sum(c.amount) as summation from sales as c

The proposed smart system converted these natural language queries into SQL queries and passes them to standard SQL query engine for execution. After the execution of these generated SQL queries, the results are noted as shown in Figure 6 to 9. The natural language queries given to the smart data retrieval engine for query

Copyright © 2018 MECS

execution are written is such a way to consider the common mistakes committed by common-user while interacting with databases. The proposed intelligent system is so smart, where it detects these mistakes and then corrects them to a large extent, and finally formulates the correct SQL queries out of it. After analyzing the SQL queries generated by SDRED for each natural language query shown in Table 1, it is evident that the proposed system is successfully able to convert the natural language queries into their corresponding correct SQL queries. Furthermore, the proposed system is completely different from those mentioned in the literature survey. This proposed technique is completely new and includes some of the simple and efficient techniques for processing the natural language query and converting it into SQL query. Figure 6 depicts SDRED system that is used to investigate the performance of the proposed system.

Figure 7 illustrates query 2 where the user had typed the wrong table name as "department", but the intelligent layer has automatically corrected it and replaced it with the correct table name as "departments".

As illustrates in Figure 7, the SDRED not only constructs the SQL query out of user's natural language query, but also is automatically corrected the mistakes. Similarly, in natural language query 2, the user again types partial attribute name "faculty", which if given to standard SQL query engine will run to error, but the proposed SDRED automatically find the correct attribute name "Total_Faculty" from the metadata of database and formulates the correct SQL query. Figure 8 illustrates the results of query 3, where the user searches the customer with name "Mohan".

Name	ID	Gender	Phone_No	Department	Designation	Specialization-1	Specialization-2	Address	District	State	Country
Mr. Muneer Ahmad	889	M	99865897	Computer sciences	Assistant Professor	Data structures	Algorithms	Muzafarpora	Srinagar	J&K	India
Mr. Azad Ahmad	778	M	77809822	Mechanical	Assistant Professor	Machine drawing	Fluid Mechanics	Owais Colony	Baramulla	J&K	India
Mr. Ajaz Dar	554	M	88254368	Computer Sciences	Assistant Professor	Java Programming	Operating Systems	Bathindi	Jammu	J&K	India
Mr. Asad Kadiri	664	M	77383686	Electronics	Assistant Professor	VLSI	Digital Electronics	M.A. Road	Srinagar	J&K	India
Mrs. Raman Sharma	660	F	55242797	Philosophy	Assistant Professor	Introduction to Logic	India History	Malviya Road	Jammu	J&K	India
Mrs. Supriya datta	880	F	55282598	Mathematics	Assistant Professor	Graph Theory	Coding Theory	Sardari Bazaar	Jammu	J&K	India
Mr. Farooq Shah	663	M	66354390	Mathematics	Assistant Professor	Differential Equations	Differential Calculus	Lalbazar	Srinagar	J&K	India
Mr. Ayaaz Ahmad	552	M	77292772	Philosophy	Assistant Professor	Political Sciences	Introduction to Logic	Ranawari	Srinagar	J&K	India
Mr. Kamal Daas	334	M	33437628	Mathematics	Assistant Professor	Probability	Statistics	Dilshanpora	Ramban	J&K	India
Mr. Umar Kadari	223	M	77383838	Computer sciences	Assistant Professor	Digital Logic	Mobile Computing	Fazabad	Gandarbal	J&K	India
Mr. Raja Hassan	559	M	73636899	English	Assistant Professor	Phonetics	_	Ranawari	Srinagar	J&K	India

Fig.6. Database Result for Query – 1 by SDRED

Name	H.O.D	Total_Faculty	Total_Administrative_Staff	Total_Labs	Total_Students	Cource_offered-1	Cource_Offered-2	Cource_Offered-3
Computer Sciences	Mr. Ramanuj Sharma	15	8	4	600	B.Tech.	M.Tech	M.C.A.
Mechanical	Mr. Mir Furqaan	16	3	3	310	B.Tech.	M.Tech.	P.hd.
Electronics	Mr. Murzamil Sheikh	11	5	4	600	B.Tech.	M.Tech.	_
Mathematics	Mr. Jain Sharma	13	3	_	400	M.Sc.	P.hd.	_
Philosophy	Mr. Majid Husain	12	5	_	200	M.A.	P.hd.	_

Fig.7. Database Result for Query – 2 by SDRED

Name	Phone_No
Mr. Mohan Lal	55625275
Mr. Mohan Kumar	66383622

Fig.8. Database Result for Query – 3 by SDRED

Figure 8 includes the natural language query 3, where the user searches the customer with name "Mohan" (which if directly given to standard SQL query engine will provide no results for this query, since "Mohan is not present in the database"), while the proposed SDRED formulates the SQL query in such a way that all the names are displayed which contain word "Mohan", and then user can easily search particular tuple from the retrieved result. Table V includes the results of query 4, where the natural language query 4 is transformed into the SQL query.

Figure 9 includes the transform of the natural language query 4 into SQL query. The proposed SDRED first checks if there is any column with name "Total amount", if not, then SDRED formulates SQL query in such a way that it finds the summation of values under column name "amount".

Summation
129800

Fig.9. Database Result for Query – 4 by SDRED

Consequently, from the preceding results it is evident that SDRED not only formulates the SQL queries for natural language queries, but also checks for the user mistakes. Accordingly, it corrects them, so that the formulated SQL queries will not produce erroneous results to the user. The proposed SDRED system do not require the database's user to remember the exact names of tables and their attributes in a database, because SDRED automatically replaces the mistaken names of tables and their attributes by exact and correct names.

From the promising results obtained by the proposed system, it is recommended to use other natural languages rather than English in the future. In addition, in future implementation of all types of SQL queries can be considered, by which the database user can not only retrieve, but also can be able to modify the database using natural language queries. Besides, this proposed technique can be extended for image retrieval system, where the Natural language processing technique can parse the user queries and a back propagation based neural network can be used to retrieve the images from database, thereby creating an easy and efficient image retrieval system.

VII. Conclusions & Future

In this paper, a smart data retrieval engine for databases has been reported, which is able to provide an easy interface to the common database users, who can type the queries in natural language and finally retrieve the desired results from the database. All the components

of proposed intelligent system were discussed in details and a complete procedure of transforming a natural language query into SQL query has also been defined.

The main task of the proposed system was to extract useful information from natural language query and then use it to construct the corresponding SQL query. The system has been implemented and multiple queries (in natural language) were supplied as input. The results established that the system efficiently converted the natural language queries into their corresponding SQL queries. It executed them and finally provided the desired results to the user. Furthermore, the system automatically corrected the mistakes (if any) in natural language queries using semantic matching and distance measurement techniques, thereby making the complete system easy, simple and robust.

Also, in future we will increase the applicability of our proposed system to much more complex natural language queries (which will generate more complex nested SQL queries).

APPENDIX A

The notations used in the present work are as follows:

S. No.	Notation	Meaning	Remarks
1	**Q or Nq**	User's query in Natural Language	-
2	**W_S**	Stop Word Training set	Used to remove unwanted words from Nq.
3	**a**	Array of Tokens	-
4	**E_{map}**	Expression mapping training set	Used to map mathematical expressions.
5	**C_S**	Conjunction training set	Contains all the conjunction words.
6	**S_C**	Semantic set	Dictionary of synonyms.
7	**Sq**	SQL query	Generated by SDRED.
8	**T**	Set of Tokens	-
9	**W_{Emap}**	Word corresponding to expression in E_{map}	-
10	**W_{SC}**	Word in SC which matches with particular word in Nq	-
11	**E_{Emap}**	Expression word in E_{map}	-
12	**S_{SC}**	Synonym of matched word	-
13	**T_n**	Database Table name	-
14	**A_n**	Database Attribute name	-
15	**M_T**	Set of Table names in Meta-Data	-
16	**M_A**	Set of Attributes in Meta-Data	–

REFERENCES

[1] ElmasriRamez, and Shamkant B. Navathe, Fundamentals of database systems, *Pearson*, (2014).

[2] Burleson, Donald K., et al., Handbook of Advanced SQL Database Programmer, *Rampant Tech Press*, (2003).

[3] Ma, Zongmin, et al., Intelligent databases: technologies and applications, *IGI Global*, (2007).

[4] Weizenbaum, Joseph, ELIZA—a computer program for the study of natural language communication between man and machine, *Communications of the ACM*, vol. 9, no. 1, pp. 36-45, (1966). DOI:10.1145/365153.365168

[5] Ilyasu Anda, Isah Omeiza Radiu, Enesi Femi Amine, "A safety data model for data analysis and decision making", *International Journal of Information Engineering and electronic business (IJIEEB)*, Vol. 9, No. 4, pp. 21 – 30, 2017. DOI: 10.5815/IJIEEB.2017.04.04.

[6] Harrison John Bhatti, Babak Bashari Rad, "Databases in Cloud Computing: A literature review", *International journal of Information Technology and Computer Science (IJITCS)*, Vol. 9, No. 4, pp.9-17, 2017. DOI: 10.2815/ijitcs.2017.04.02

[7] ElisaBertino, Barbara Catania, and Gian P. Zarri, Intelligent database systems, *ACM Press, Addison-Wesley*, (2001). ISBN: 0-201-87736-8.

[8] Lewis, David D., and Karen Spärck Jones, Natural language processing for information retrieval, *Communications of the ACM*, vol. 39, no. 1, pp 92-101, (1996). DOI:10.1145/234173.234210

[9] Daniel Jurafsky, and James H. Martin, Speech and Language Processing, *Pearson*, (2000).

[10] Paredes-Valverde, Mario Andrés, et al, ONLI: An ontology-based system for querying DBpedia using natural language paradigm, *Expert Systems with Applications*, vol. 42, no. 12, pp. 5163-5176, (2015).

[11] Ngamnij, Somjit et al., Semantic ontology mapping for interoperability of learning resource systems using a rule-based reasoning approach, *Expert Systems with Applications*, vol. 40, no. 18, pp7428-7443, (2013). DOI:10.1016/j.eswa.2013.07.027

[12] Heeringa, Wilbert, Measuring dialect pronunciation differences using Levenshtein distance [Dissertation], *Rijksuniversiteit Groningen*, (2004).

[13] NF Ayan, ArindamMandal, and Jing Zheng, Clarifying natural language input using targeted questions, *U.S. Patent 13/866,509*, (2013).

[14] Li, Fei, and Hosagrahar V. Jagadish, NaLIR: An interactive natural language interface for querying relational databases, Proceedings of the 2014 *ACM SIGMOD international conference on Management of data*. ACM, (2014). DOI:10.1145/2588555.2594519

[15] Damljanović, Danica, et al., Improving habitability of natural language interfaces for querying ontologies with feedback and clarification dialogues, *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 19, pp1-21, (2013). DOI:10.1016/j.websem.2013.02.002

[16] Deebha Mumtaz, Bindiya Ahuja, "A Lexical Approach for opinion Mining in Twitter", *International Journal of*

Education and Management Engineering (IJEME), Vol. 6, No. 4, pp. 20 – 29, 2016. DOI: 10.5815/ijeme.2016.04.03.

[17] Kaufmann, Esther, and Abraham Bernstein, Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases, *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 8, no. 4, pp377-393, (2010). DOI:10.1016/j.websem.2010.06.001

[18] I. Habernal, M. Konopík, SWSNL: Semantic web search using natural language, *Expert Systems with Applications*, vol. 40, no. 9, pp3649–3664, (2013). DOI:10.1016/j.eswa.2012.12.070

[19] Wen-Tau Y., Ming-Wei C., Xiaodong H., Jianfeng G., "Semantic parsing via staged query grape generation", *Microsoft Research, Redmond, WA 98052*, USA.

[20] Llopis, Miguel, and Antonio Ferrández, How to make a natural language interface to query databases accessible to everyone: An example, *Computer Standards & Interfaces*, vol. 35, no. 5, pp 470-481, (2013). DOI:10.1016/j.csi.2012.09.005

[21] Melyara. Mezzi, Nadjia. Benblidia, "Study of context Modelling criteria in information Retrieval", *International journal of Information Technology and Computer Science (IJITCS)*, Vol. 9, No. 3, pp. 28-39, 2017. DOI: 10.5815/ijitcs.2017.03.04

Authors' Profiles

Mir Shahnawaz Ahmad received the B.Tech. degree in Computer Science and Engineering from university of Kashmir, Srinagar, J&K, India, and the M.Tech. degree in Computer Science and Engineering from SMVDU, katra, J&K, India. He is currently working as an Assistant Professor in SSM college of Engineering and Technology, Parihaspora Pattan, J&K, India. His main research focus lies in database systems, Software Defined Networks, MANETs, IOT and Data Sciences.

Syed Rameem Zahra received the B.Tech. degree in Computer Science and Engineering from university of Kashmir, Srinagar, J&K, India, and the M.Tech. degree in Computer Science and Engineering from SMVDU, katra, J&K, India. She is currently pursuing the Ph.D. degree with the department of Computer Science and Engineering, National Institute of Technology Srinagar, J&K, India. Her area of research is database systems, Wireless sensor networks, VANETs and IoT security.

How to cite this paper: Shahnawaz Ahmad, Syed Rameem Zahra, "SDRED: Smart Data Retrieval Engine for Databases", International Journal of Information Technology and Computer Science(IJITCS), Vol.10, No.6, pp.1-10, 2018. DOI: 10.5815/ijitcs.2018.06.01