Article

An algorithm to transform natural language into SQL queries for relational databases

Garima Singh, **Arun Solanki**

Department of Computer Science and Engineering, Gautam Buddha University, Greater Noida, India

E-mail: garima_singh911@yahoo.com

Received 12 April 2016; *Accepted 18 May 2016*; *Published online 1 September 2016*

Abstract

Intelligent interface, to enhance efficient interactions between user and databases, is the need of the database applications. Databases must be intelligent enough to make the accessibility faster. However, not every user familiar with the Structured Query Language (SQL) queries as they may not aware of structure of the database and they thus require to learn SQL. So, non-expert users need a system to interact with relational databases in their natural language such as English. For this, Database Management System (DBMS) must have an ability to understand Natural Language (NL). In this research, an intelligent interface is developed using semantic matching technique which translates natural language query to SQL using set of production rules and data dictionary. The data dictionary consists of semantics sets for relations and attributes. A series of steps like lower case conversion, tokenization, speech tagging, database element and SQL element extraction is used to convert Natural Language Query (NLQ) to SQL Query. The transformed query is executed and the results are obtained by the user. Intelligent Interface is the need of database applications to enhance efficient interaction between user and DBMS.

Keywords natural language query interface; natural language processing; ambiguity; SQL.

Selforganizology
ISSN 2410-0080
URL: http://www.iaees.org/publications/journals/selforganizology/online-version.asp
RSS: http://www.iaees.org/publications/journals/selforganizology/rss.xml
E-mail: selforganizology@iaees.org
Editor-in-Chief: WenJun Zhang
Publisher: International Academy of Ecology and Environmental Sciences

1 Introduction

In the present fast computing scenario, computer based information retrieval technologies are being highly used to help academic and education institutions organizations, companies to manage their information systems and processes. These are used to manage data that is capable of managing different kinds of data which are stored in the databases also known as DBMS (Rukshan et al., 2013). Despite Information retrieval of a large amount of data being efficient in relational databases, the user still needs to master the DB language/schema to completely formulate the queries. Artificial Intelligence (AI) and Linguistics can be combined to develop programs that can help to understand and produce information in a natural language

(Johnson, 1985; Mckay and Finin; 1990; Wan; 2000; Mohite and Bhojane, 2014; Javubar and Jay, 2015). Database based NLP is thus an important success in processing Natural Language. It is a convenient way of data access by asking questions in natural language to get answers since a layman might not understand the database query language. A NLQ Interface to Database system is an application that accepts a natural language query, creates a SQL query from it and executes it to retrieve the data from relational database. The result retrieved from the database is a stream of elements. The query is generated by identifying the lexical relations of the elements of the NLQ (Gupta and Sangal, 2012). The building of robust and applicable NLIDBS has become acute in recent years. The amount of information on the Internet has grown steadily and also wider population may now access data stored in a variety of repositories via web browsers (Nihalani et al., 2011). Thus NLIDBS are built for to optimize the search results and produce information with more accuracy. The present research extends the existing work further by processing more complex queries along with ambiguity removal.

From the past many years, unending attempts have been made to build efficient natural language query interface. There have been a lot of research works introducing some new theories and implementations of NLIDBs. But the product produced didn't map the desired expectation. LUNAR, launched in 1973 was a system that handled queries related to samples of rock which was brought back from the moon. It used an Augmented Transition Network (ATN) parser and Woods' Procedural Semantics. LIFER/LADDER was one of the best database language processing systems. It was designed to retrieve information regarding US Navy ships. It used semantic grammar to parse user queries in natural language. It could only handle queries related to one table or multiple table queries with easy join conditions. Another general architecture for an intelligent database interface was proposed (Nahalani et al., 2011) whose main characteristic was domain-independence, which means this interface could be used with any database. The interface employs semantic matching technique to convert natural language query to SQL using dictionary and set of production rules. Sontakke and Pimpalkar (2014) introduced a system for people who are snug with Hindi language. The application accepted Hindi sentence as a query, processed it and after execution provided the result to the user in Hindi language itself. Authors developed the rule based system which satisfied the user need by accepting Hindi language as query and the output is displayed in Hindi language only. Savvy, which is a typical application of pattern matching framework (Poole and Mackworth, 2010), uses various patterns written in some different kind of queries which are then executed after the complete queries are entered. The systems are very easy to implement as no elaborate parsing and modules of interpretation are required.

Despite attainment of so many achievements, the present day NLIDBs do not guarantee translation of queries in natural language to database languages. This research design and implements a system called as Natural Language to SQL Convertor (NLTSQLC). It will work to convert a NLP Query to SQL Query.

2 Proposed System Architecture

The proposed system is designed to minimize the communication gap between a human and computer. It is developed to facilitate improved interaction between the two. As it is known databases can only respond to standard queries written in SQL and it is very less possible for a common person to know SQL. Also they might not be aware of the database schema including table names, formats, their fields and the corresponding types. Thus keeping these things in mind, a system is designed which contains an intelligent layer that accepts common user's sentences in natural language as input, converts these sentences into standard SQL queries and executes them to retrieve data from relational databases. The designed system has the following characteristics:

- It can handle ambiguities in NLQ such that the tables with same attribute name clashing are minimized.

- The interface can be configured easily and automatically. It relies on the Metadata set and Semantic sets for tables and attributes.
- The system is designed to accept any relational database schema which is responsible for the intelligent information processing and performing flexible queries.

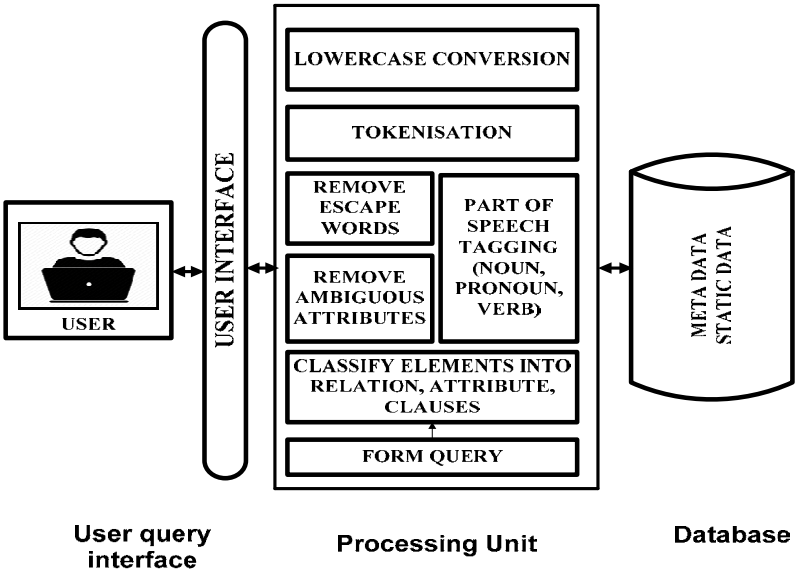

Fig. 1 Three-Tier Architecture of NLTSQLC.

3 NLP Rules Used for System Development

In the desired approach, some predefined structures are employed and the system is trained accordingly. The primary advantage of these structures is that they can be expanded whenever some new knowledge is discovered. It uses

- The escape word set (E_w) which contains the list of stop words that occur in NLQ as shown in Table 1.
- The Expression mapping set (E_{map}) which contains the list of conditional clauses which might occur in NL query and their associated mathematical symbols as shown in Table 2.
- A Noun set (N) that contains all elements which are nouns (Table 3) and strictly limited to provided data

  Dictionary of attribute & relation names and further Relation set and Attribute set are extracted from N.
- A Verb set (V) that contains all elements which are verb (Table 4) and act as criteria for forming clauses.
- The semantic set (S) contains the list of all possible semantics related to table names and fields in the database as shown in table-5 and table-6
- A Variable set (Va) that consists of all String and Integer variables used in forming clauses.
- A Relation set (R) consists of relation names that are encountered in user query or are added by analyzing the attribute names present in the NL query.
- An Attribute set (A) that contains all attributes present in the user query.
- An Ambiguity check set (A_c) that contains all attribute fields whose name are used in multiple relation as a field name excluding keys.

- The Conjunction training set (C_T) consists of the list of Conjunctive clauses which occur in NL query and this set is generated at runtime. Whenever new conjunctive clause is encountered, it is appended to the existing Conjunction training set.

Table 1 Escape words list.

A	An	The	Select
find	which	whose	Is
Of	A	With	To
for	Are	And	What

Table 2 Rules for function 'BETWEEN'.

Rule	Rule Symbol	Rule Description
Between	BETWEEN	Rule for function 'BETWEEN'
Range	BETWEEN	Rule for function 'BETWEEN'
Ranges	BETWEEN	Rule for function 'BETWEEN'
Lies	BETWEEN	Rule for function 'BETWEEN'

Table 3 Noun list.

student	Subject	Teacher	Marks
Age	roll_no	subject_code	Name
taught_by	Details	Subjectwise	Id

Table 4 Verb list.

Between	less than
Max	greater than
Min	equal to

Table 5 Rules for attributes in relations.

Rule	Rule Symbol	Rule Description
Marks	Student	Attribute for Relation 'student'
Roll_no	Student	Attribute for Relation 'student'
Emp_id	Teacher	Attribute for Relation 'teacher'
Name	Student Teacher Subject	Attribute for Relation 'student' Attribute for Relation 'teacher' Attribute for Relation 'subject'
Age	Teacher	Attribute for Relation 'teacher'
Subject_code	Student Teacher Subject	Attribute for Relation 'student' Attribute for Relation 'teacher' Attribute for Relation 'subject'
Taught_by	Student Teacher	Attribute for Relation 'student' Attribute for Relation 'teacher'

Table 6 Rules for relation name ‘teacher’.

Rule	Rule Symbol	Rule Description
Teacher	Teacher	Rule for Relation ‘teacher’
Teachers	Teacher	Rule for Relation ‘teacher’
Faculty	Teacher	Rule for Relation ‘teacher’
Professor	Teacher	Rule for Relation ‘teacher’

4 Database Design and Implementation

The present research uses MySQL DBMS as backend. The database schema is an organization of the important information that briefly describes the relation and attributes in the database. Here, Table 7 shows the database schema of the Student database which is generated by the system. The schema holds entries for all the ‘n’ tables (relations) in the database along with all their corresponding fields (attributes) and their unique primary key described in the figures below

Table 7 Rules for database schema.

Relation Name	Foreign Key	Primary Key
Student	subject_code	roll_no
Subject	subject_code	subject_code
Teacher	subject_code	taught_by

Fig. 2 Llist of relations in the database namely Student, Teacher and Subject.

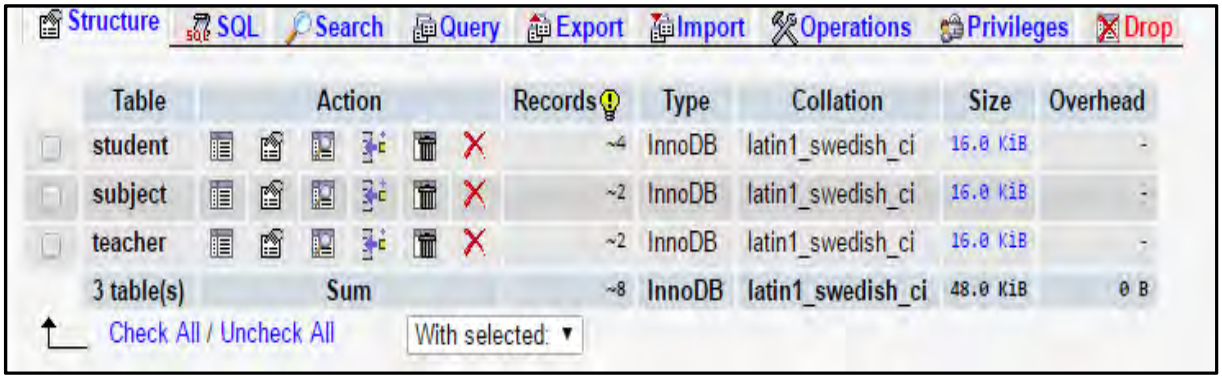

Fig. 2 Relations in database.

The Relation Student has four attributes which are roll_no, name, subject_code and marks as shown in Fig. 3. Roll_no is the primary key of this relation and the foreign key is subject_code.

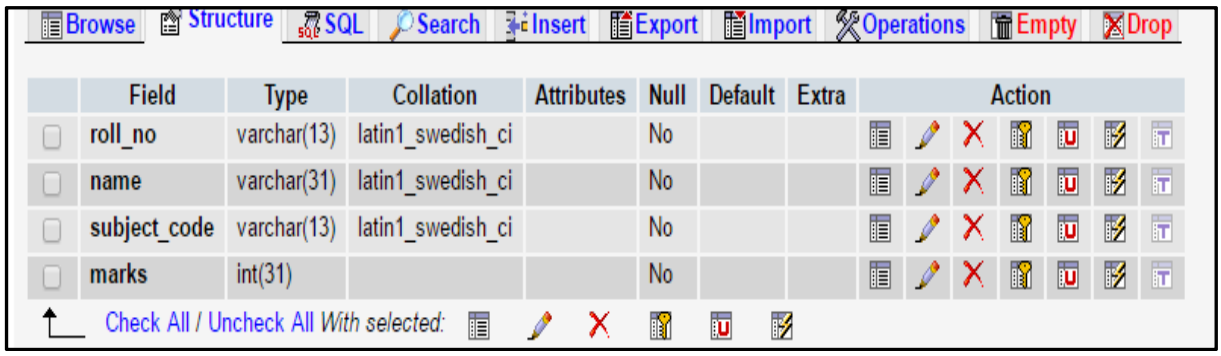

Fig. 3 Attributes in Relation 'Student'.

The Relation Subject has three attributes which are name, subject_code and taught_by as shown in figure-4. Subject_code is the primary key of this relation and the foreign key is also subject_code.

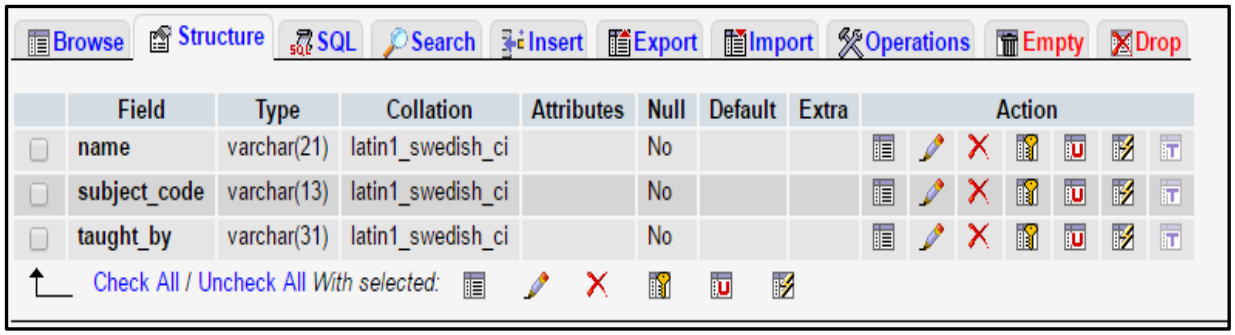

Fig. 4 Attributes in Relation 'Subject'.

The Relation Teacher has four attributes which are taught_by, name, subject_code and age as shown in figure-5. Taught_by is the primary key of this relation and the foreign key is subject_code.

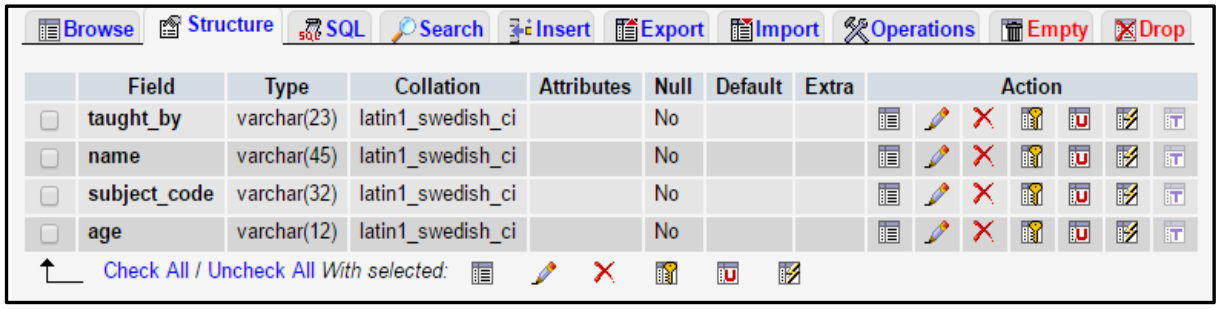

Fig. 5 Attributes in Relation 'Teacher'.

Terms used in NLTSQLC are:

$M = \{T, F, P, F\}$, $0 < y < (n+1)$, where

M=Database for System

T=Set of tables in M (as shown in figure-2)

F=Set of all fields in T (as shown in figure-3,4,5)

P=Primary Key in T

F=Foreign key in T

5 Process Flow of NLTSQLC

The system analyses and executes an NLQ in series of steps and at each stage the data is further processed to finally form a query leading to its execution. At the end the results are fetched from the database and the output is displayed to the user on the GUI as shown in Fig. 6.

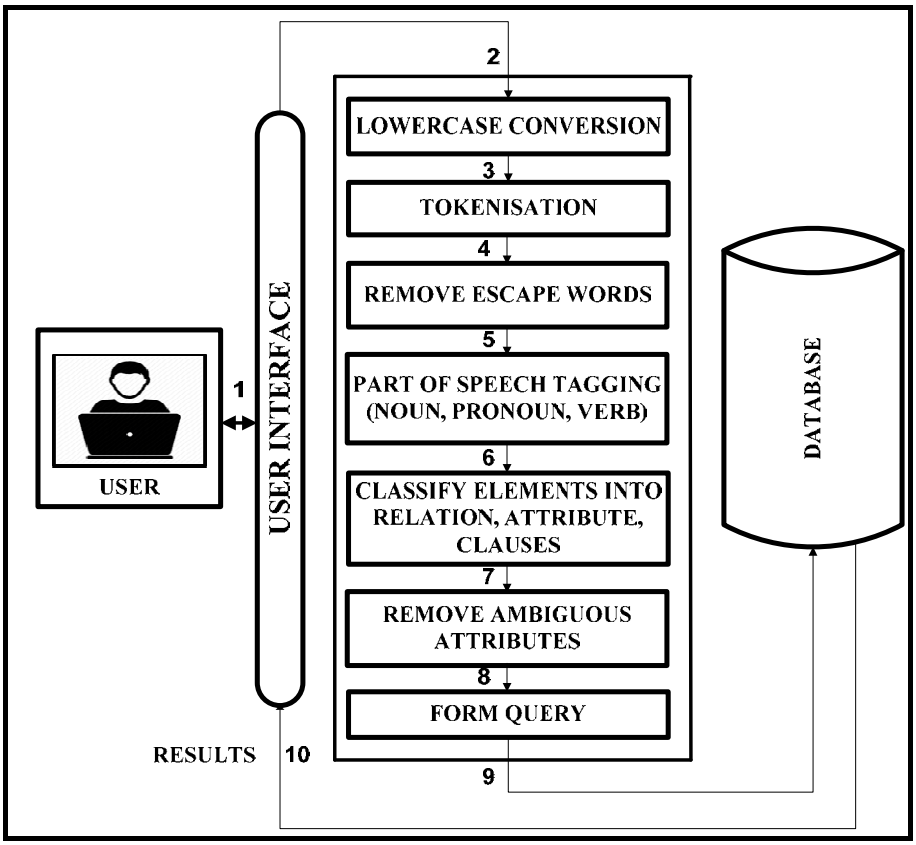

Fig. 6 Process control flow diagram.

The description of the system is as follows:

1. User Interface: The user interacts with the system via GUI and types his/her NLQ.
2. Lowercase Conversion: The NLQ is then translated into lowercase.
3. Tokenization: The query after lowercase conversion is then converted into stream of tokens and a token id is provided to each word of NLQ.
4. Escape word removal: The extra/stop words are removed which are not needed in the analysis of query.
5. Part Of Speech Tagger: The tokens are then classified into nouns, pronouns, verb and string/integer variables.
6. Relations-Attributes-Clauses Identifier: Now the system classifies the tokens into relations, attributes and clauses on the basis of tagged elements and also seperates the Integer and String values to form clauses.

7. Ambiguity Removal: It removes all the ambiguous attributes that exists in multiple relation with the same attribute name and maps it with the correct relation.
8. Query Formation: After the relations, attributes and clauses are extracted, the final query is constructed.
9. Query Execution and Data Fetching: The query is then executed and data is fetched from the database.
10. Results: The final query result is displayed to the user on the GUI.

6 Implementation and Working of NLTSQLC

The system is developed and implemented in PHP,HTML,CSS and Javascript as front-end and the database is implemented in MySQL as the back-end.

1. **Graphical User interface:** As shown in Fig. 7, the interactive GUI allows user to enter the query. It allows him/her to enter the question in a natural language which will be further processed by the system.

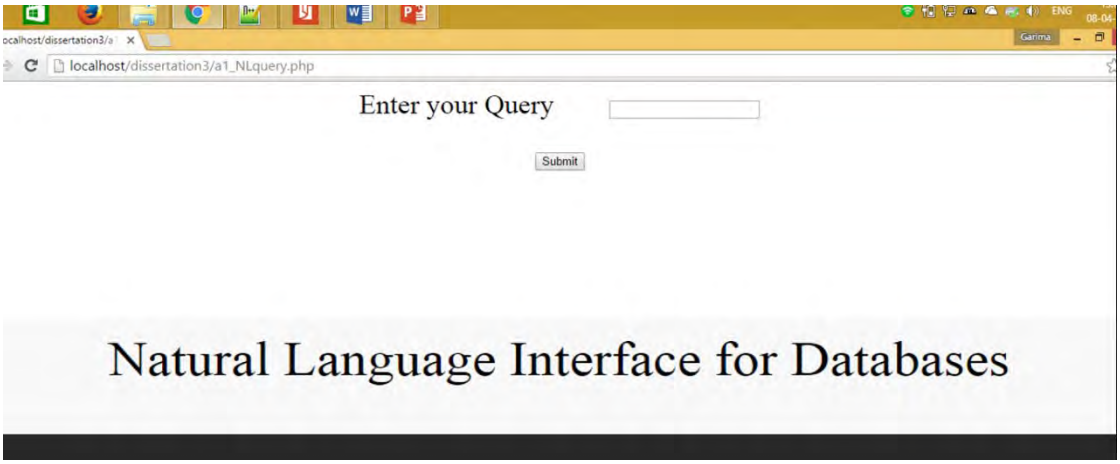

Fig. 7 Graphical User Interface.

2. **Lower case Conversion:** It checks all the words in the user question and converts it into lower case as shown in Fig. 8.

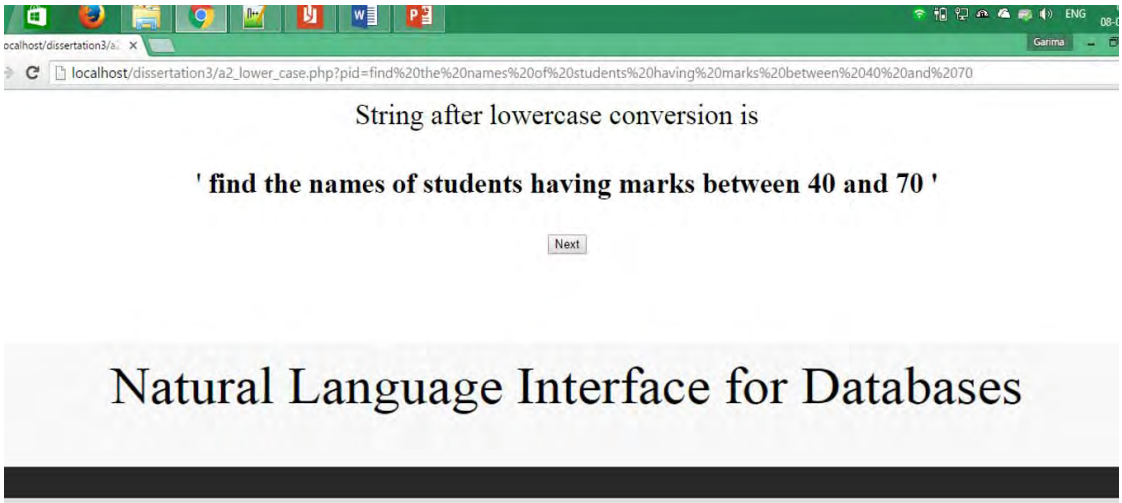

Fig. 8 Lower case conversion.

3. **Tokenisation:** It firstly scans the whole question and then splits the question string into tokens and gives an order number to each token identified, as shown in Fig. 9.

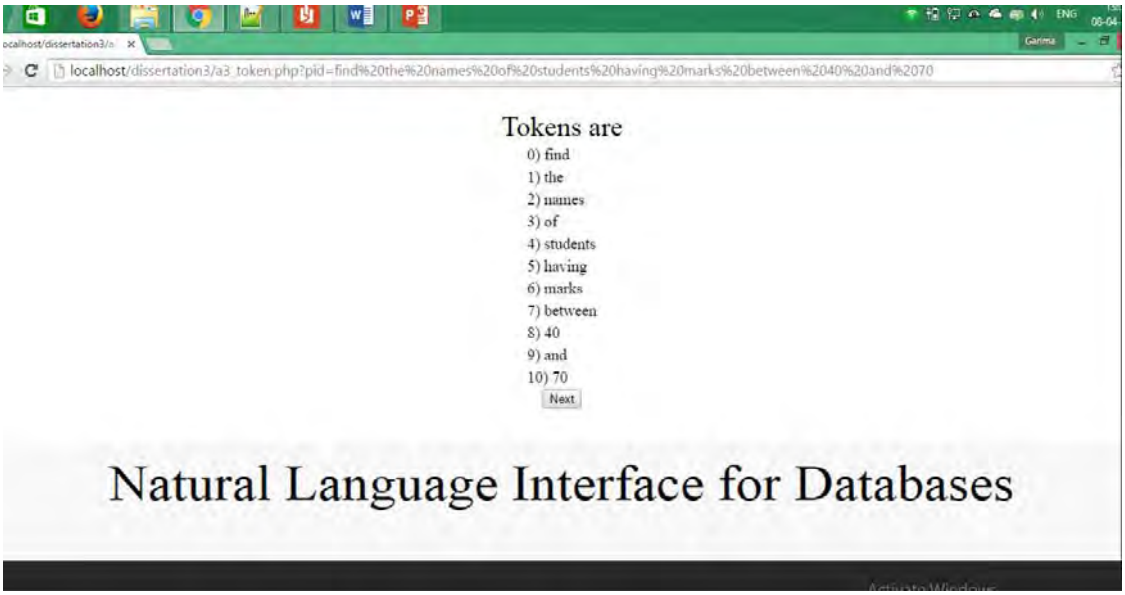

Fig. 9 Tokenisation.

4. **Escape Word Remover:** It removes all words which are not needed in the query and only select those words related to the database content. It store the number of identified SQL elements in arrays shown in Fig. 10.

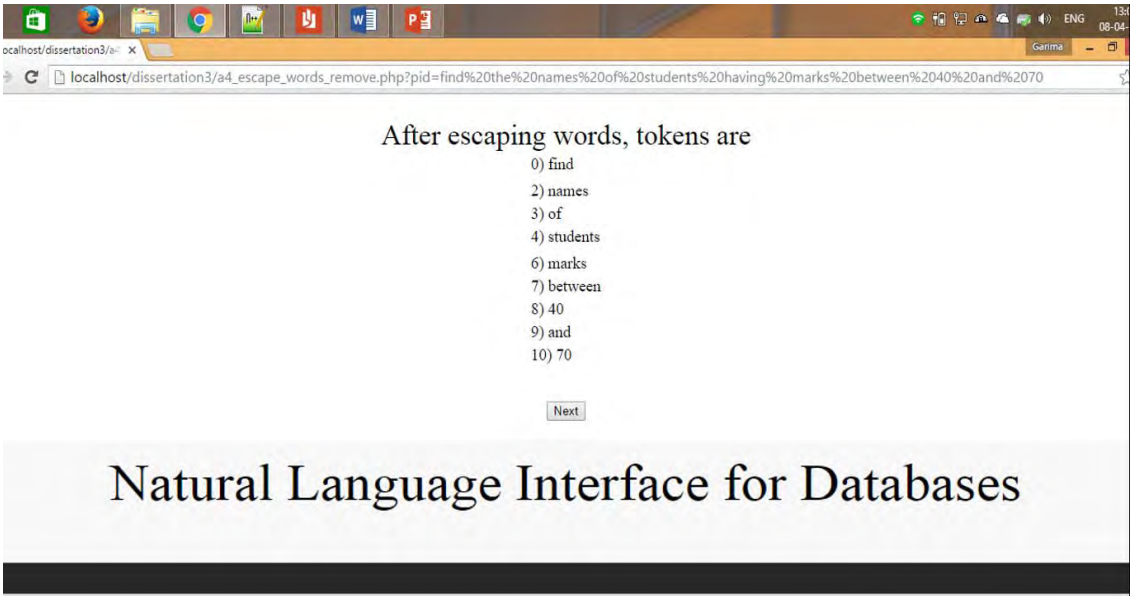

Fig. 10 Escape word Remover.

5. **Noun-Pronoun-Verb Tagger:** It tags the extracted tokens as noun, pronoun or verb and further classyfying them into SQL elements in arrays as shown in Fig. 11.

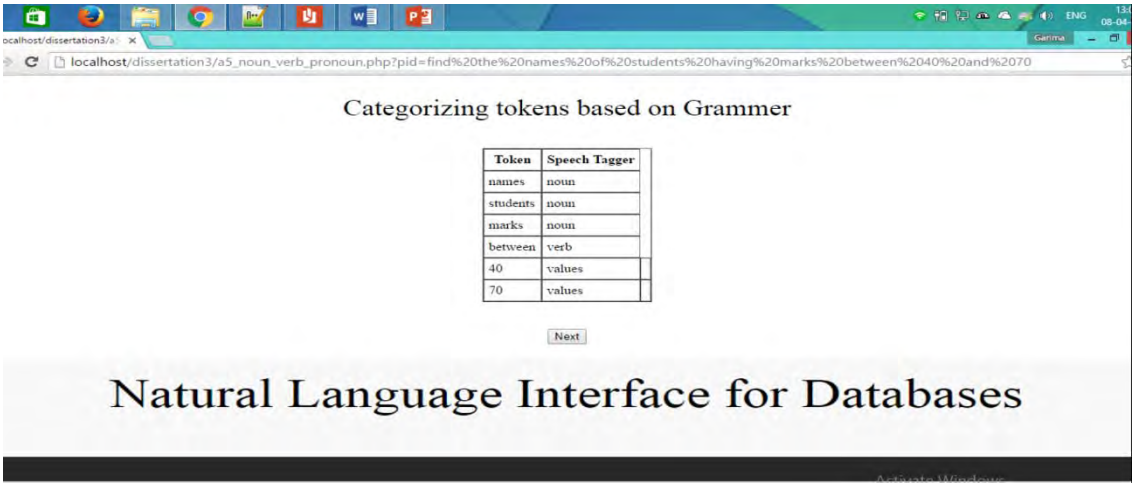

Fig. 11 Noun-Pronoun-Verb Tagger.

6. **Relations-Attributes-Clauses Identifier:** It after tagging the elements into noun-verb-pronoun, then classifies them relations, attributes and clauses on the basis of tagged elements. It also sepearates the Integer and String values for forming the clauses, as shown in Fig. 12.

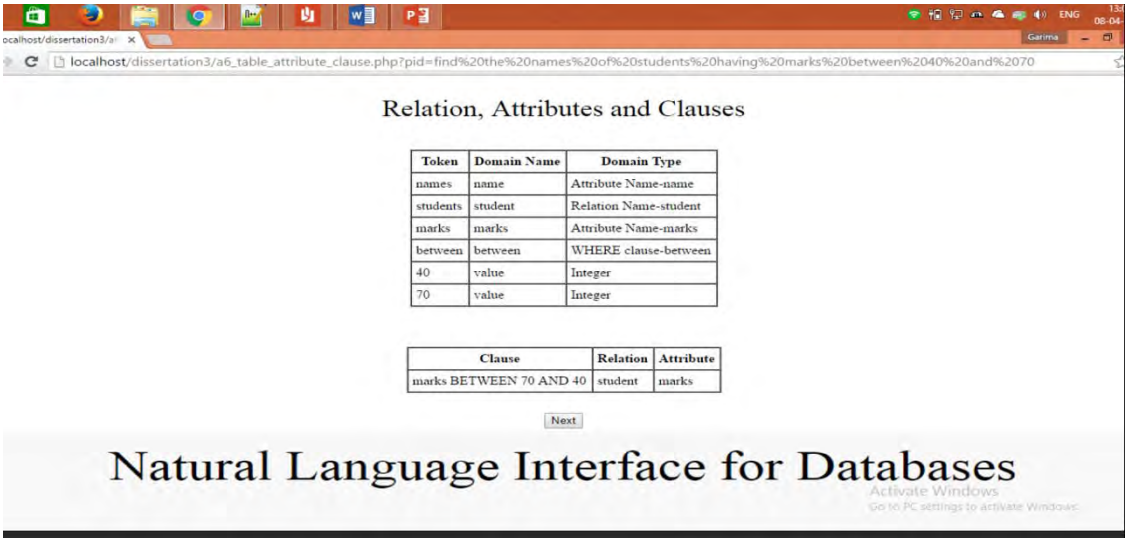

Fig.12 Relations-Attributes-Clauses identifier.

7. **Ambiguity Remover:** It removes all the ambiguous attributes existing multiple times and extracts the most apt attribute and maps it with the relation as shown in Fig. 13.

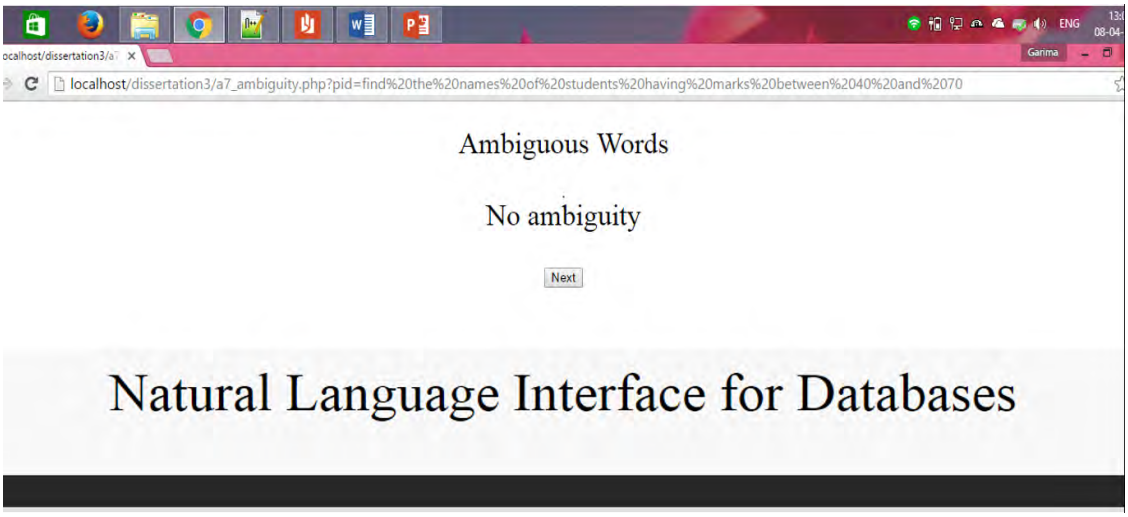

Fig. 13 Ambiguity Remover.

8. **Query Generation:** After classifying the relations, attributes and clauses and further removing ambiguities, the final query is generated on the basis of extracted elements as shown in Fig. 14.

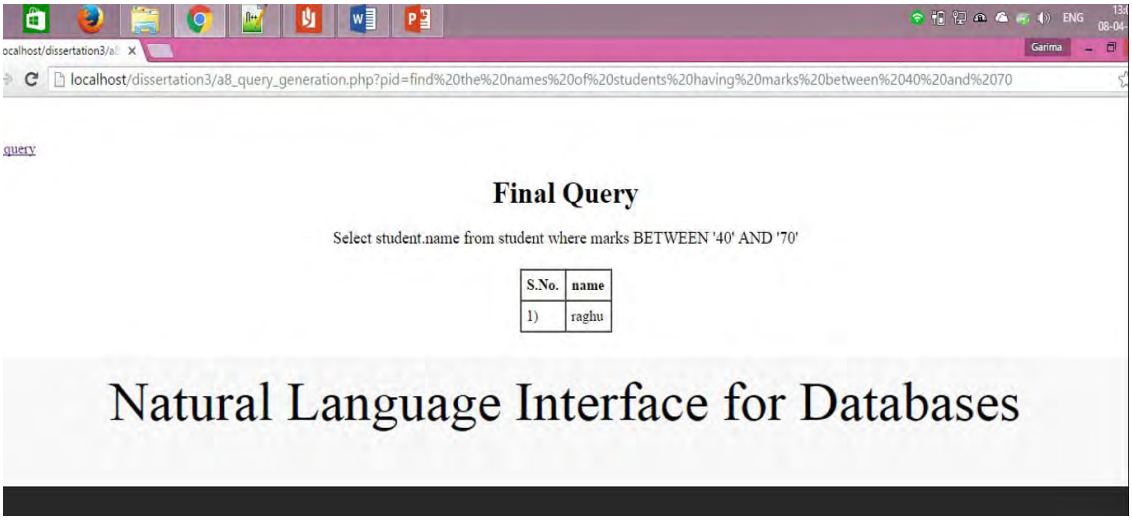

Fig. 14 Query generation.

9. **Results:** The SQL query is executed and results are fetched and displayed to user as shown in Fig. 14.

7 Results

To assess the old and new systems,question set consisting of 28 NLQ questions and 50 NLQ questions were fired. Based on the type of queries, confusion matrix (Tables 8, 9, 10, 11) is created and based on its values various performance factors are evaluated. The Tables 12,13,14 indicates the results that were computed using the values in Confusion Matrix. Same dataset are used to find the result on both system.

Table 8 Confusion Matrix for NLTSQLC (on 28 queries fired).

		Detected: Positive	Detected: Negative
Actual	Positive	True Positive(A)=16	False Negative(B)=4
	Negative	False Negative(C)=1	True Negative(D)=7

Table 9 Confusion Matrix for NLIDBS (on 28 queries fired).

		Detected: Positive	Detected: Negative
Actual	Positive	True Positive(A)=12	False Negative(B)=8
	Negative	False Negative(C)=1	True Negative(D)=7

Table 10 Confusion Matrix for NLTSQLC (on 50 queries fired).

		Detected: Positive	Detected: Negative
Actual	Positive	True Positive(A)=30	False Negative(B)=7
	Negative	False Negative(C)=3	True Negative(D)=10

Table 11 Confusion Matrix for NLIDBS (on 50 queries fired).

		Detected: Positive	Detected: Negative
Actual	Positive	True Positive(A)=23	False Negative(B)=14
	Negative	False Negative(C)=3	True Negative(D)=10

To assess and check the accuracy of systems, evaluation was done based on metrics:

- Recall : The proportion of positive case queries which are correctly identified.

$$Recall(R) = A/(A+B)$$

- Accuracy : The proportion of total number of positive predictions which were correct.

$$Accuracy = (A+D)/(A+B+C+D)$$

- The false positive rate (FPR) : It is the proportion of negatives case queries which are incorrectly classified as positive.

$$FPR= C/(C+D)$$

- The true negative rate (TNR) : It is defined as the proportion of negatives case queries which are classified correctly.

$$TNR= D/(C+D)$$

- The false negative rate (FNR) : It is the proportion of positives case queries which are incorrectly classified as negative.

$$FNR= B/ (A+B)$$

- Precision : It is the proportion of the predicted positive case queries which are correct.

$$Precision (P) = A/ (A+C)$$

- F-measure : It computes some average of the information retrieval precision and recall metrics.

$$F\text{-measure}= 2PR/ (P+R)$$

Table 12 Comparison between NLIDBS and NLTSQLC (for 28 queries fired).

S.No.	Performance Factors	Old System-NLIDBS	New System-NLTSQLC
1)	Recall or True Positive Rate (TPR) or Sensitivity	0.6	0.8
2)	Accuracy	0.678	0.821
3)	Error Rate	0.321	0.178
4)	False Positive Rate	0.125	0.125
5)	True Negative Rate	0.875	0.875
6)	False Negative Rate	0.4	0.2
7)	Precision	0.923	0.941
8)	F-Measure	0.727	0.864

As it can be seen from the Table 12, NLTSQLC has recall value of 0.8 whereas NLIDBS has a recall value of 06. The F-measure for NLIDBS is 0.727 whereas NLTSQLC has 0.864. The accuracy for the new system is 0.821 whereas for the old system it is 0.678. The values for FNR is also low in NLTSQLC. Thus NLTSQLC better values for recall, accuracy, FNR, F-measure with decrease in error rate as compared to NLIDBS. However there is a minimal increase in Precision which can be firther improved in the furture work. Hence the new system, NLTSQLC, is an improved version of existing system.

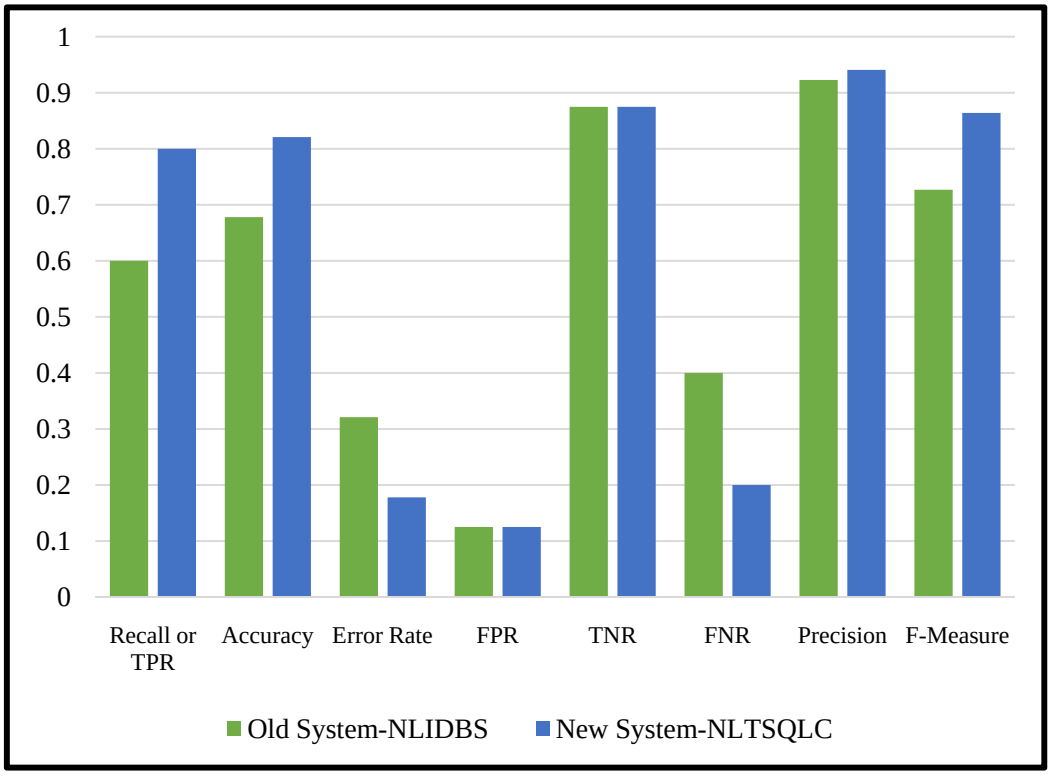

Fig. 15 Comparison between NLIDBS and NLTSQLC (for 28 queries fired).

As it can be seen from the Fig. 15, the peak for accuracy, F-measure, and recall is greater for NLTSQLC as compared to NLIDBS. Also the error rate is lower for the new system. The peak for precision is almost at same height for both the systems which can be improved further in the future work. Thus, we may conclude that the NLTSQLC has better results and is an improved system.

Table 13 Comparison between NLIDBS and NLTSQLC (for 50 queries fired).

S.No.	Performance Factors	Old System-NLIDBS	New System-NLTSQLC
1)	Recall or True Positive Rate (TPR) or Sensitivity	0.621	0.8
2)	Accuracy	0.66	0.8
3)	Error Rate	0.34	0.2
4)	False Positive Rate	0.23	0.231
5)	True Negative Rate	0.769	0.769
6)	False Negative Rate	0.37	0.189
7)	Precision	0.88	0.909
8)	F-Measure	0.729	0.851

From the Table 13 above, the performance of NLTSQLC is again better than NLIDBS when NLQ questions were increased to 50. The Accuracy is 0.8 for NLTSQLC whereas the NLIDBS has an accuracy of 0.621. The Precision for the new system design is 0.9 whereas the old system it is 0.88. The F-measure and Recall values for the new system design are also more than the old system.

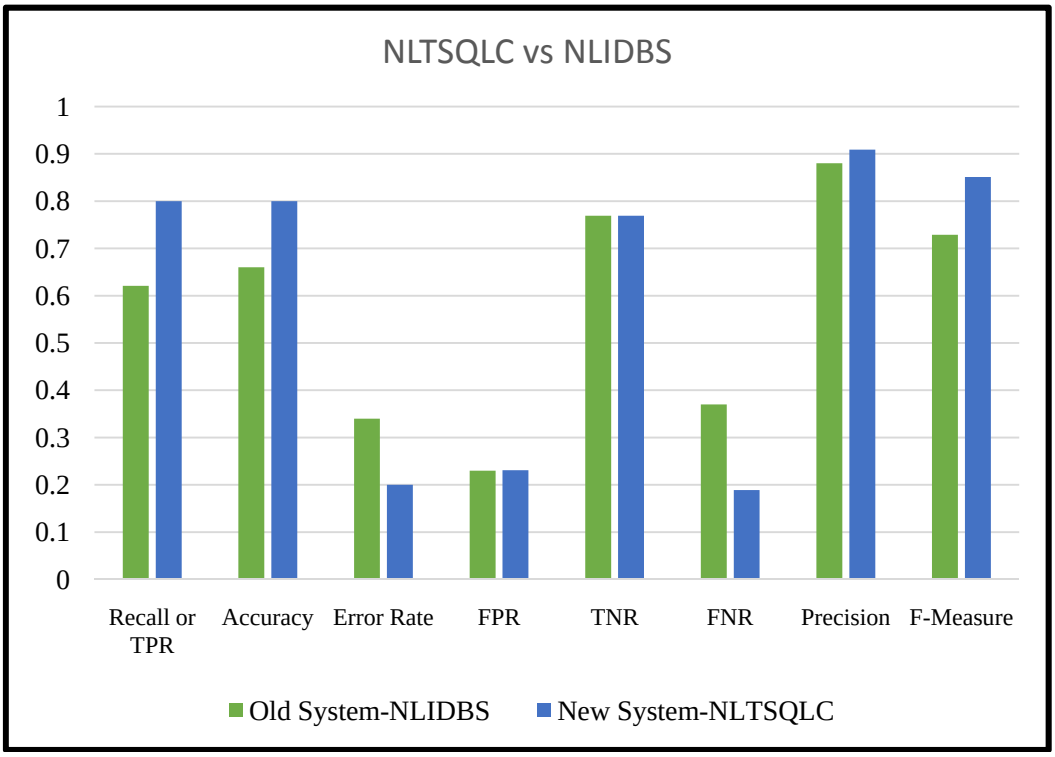

Fig. 16 Comparison between NLIDBS and NLTSQLC (for 50 queries fired).

Fig. 16 shows higher peaks for Recall, Accuracy, Precision and F-measure in the NLTSQLC system. The error rate in the new system is also decreased and hence the performance of NLTSQLC is better than NLIDBS.

Table 14 Comparison between performance of NLTSQLC for 28 NLQ and 50 NLQ.

S.No.	Performance Factors	For 28 queries	For 50 queries
1)	Recall or True Positive Rate (TPR) or Sensitivity	0.8	0.8
2)	Accuracy	0.8	0.8
3)	Error Rate	0.178	0.2
4)	False Positive Rate	0.125	0.231
5)	True Negative Rate	0.875	0.769
6)	False Negative Rate	0.2	0.189
7)	Precision	0.941	0.909
8)	F-Measure	0.864	0.851

Table 14 shows the comparison between the performance of NLTSQLC when the NLQ were increased from 28 to 50. The Recall rate and Accuracy is remained unchanged and has a value of 0.8 but there is slight decrease in F-measure and Precision of the system on increasing the number of queries.

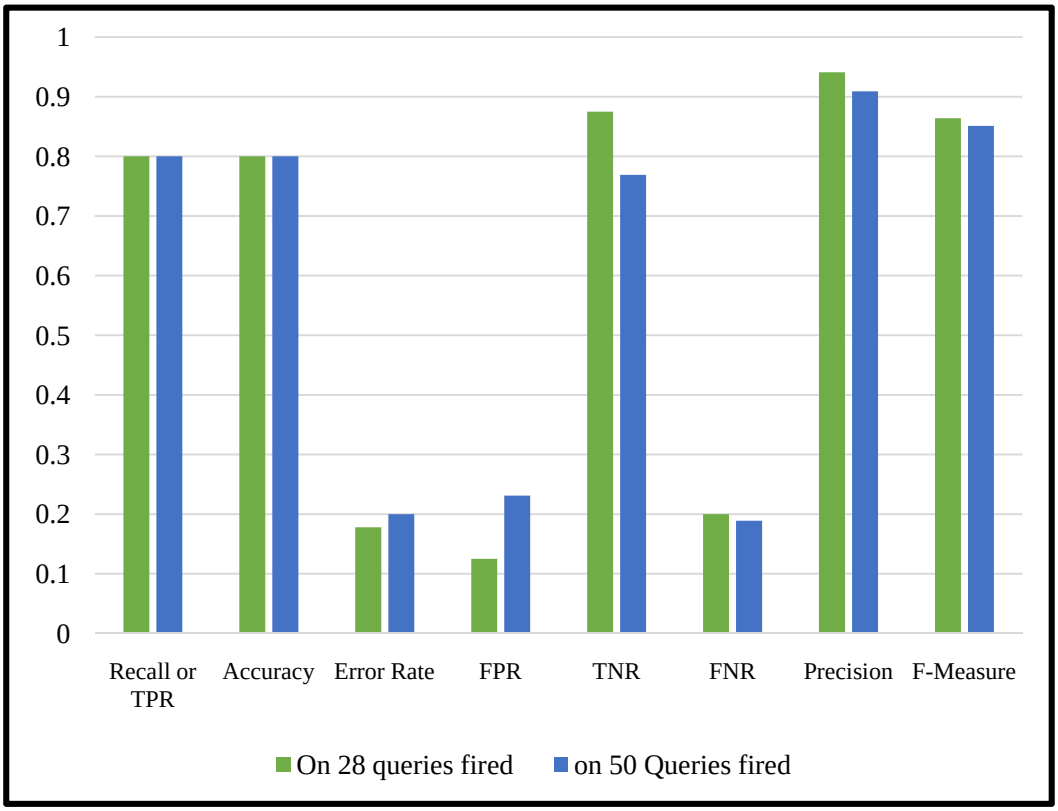

Fig. 17 Comparison between performance of NLTSQLC for 28 NLQ and 50 NLQ.

Fig. 17 shows the comparison on increasing the NLQ from 28 to 50. The NLTSQLC is tested and the peaks for Recall and Accuracy are at the same height. The Precision and F-measure has a slight decrease.

8 Conclusion and Future Work

In this research, the proposed NLTSQLC system is designed to handle challenges in NLQ processing. The aim is to evaluate correct sql translations for NLQ. The intelligent interface developed uses semantic matching technique which translates natural language query to SQL. It also uses set of production rules and data dictionary which consists of semantics sets for relations and attributes. A series of steps like lower case conversion, tokenization, speech tagging, database element extraction, SQL element extraction and ambiguity removal is used to convert Natural Language Query (NLQ) to SQL Query. The validation of the prototype has shown improved performance. The results show that the system had improved Recall, FNR, error rate and Accuracy. However, there was minimal change in precision, TNR and FPR which can be resolved in the future work.

References

Gupta A, Sangal R. 2012. A Novel Approach to Aggregation Processing in Natural Language Interfaces to Databases. Language Technologies Research Centre International Institute of Information Technology,

Hyderabad, India

Javubar SK, Jay A. 2015. Natural language to SQL generation for semantic knowledge extraction in social web sources. Indian Journal of Science and Technology, 8(1): 1-10

Johnson T. 1985. Natural Language Computing: The Commercial Applications. Ovum Limited, London, UK

Mckay DP, Finin TW. 1990. The intelligent database interface: Integrating AI and database systems. Proceedings of the 1990 National Conference on Artificial Intelligence. 677-684

Mohite A, Bhojane V. 2014. Challenges and implementation steps of natural language interface for information extraction from database. International Journal of Recent Technology and Engineering, 3(1): 108

Nihalani N, Motwani M, Silakari S. 2011. An intelligent interface for relational databases. International Journal of Simulation: Systems, Science and Technology, 11(1): 29

Poole D, Mackworth A. 2010. Artificial Intelligence-Foundations of Computational Agents. http://artint.info/index.html

Rao G, Agarwal C, Chaudhry S, et al. 2010. NATURAL LANGUAGE QUERY PROCESSING USING SEMANTIC GRAMMAR. International Journal on Computer Science and Engineering, 2(2): 219-223

Rukshan A, Rukshan P, Mahesan S. 2013. Natural Language Web Interface for Database (NLWIDB). Proceedings of the Third International Symposium. SEUSL, Oluvil, Sri Lanka

Sontakke AR, Pimpalkar A. 2014. A rule based graphical user interface to relational database using NLP. International Journal of Scientific Engineering and Research, 3(4): 81-84

Sreenivasulu M. 2014. Information retrieval using natural language interfaces. International Journal of Computer Applications, 92(12): 34-37

Wan FJ. 2000. A fuzzy grammar and possibility theory – based natural language user interface for spatial queries. Fuzzy Sets and Systems, 113: 147-159