International Journal of Engineering Research and General Science Volume 3, Issue 2, Part 2, March-April, 2015
ISSN 2091-2730

NATURAL LANGUAGE DATABASE INTERFACE

Aniket Khapane[1], Mahesh Kapadane[1], Pravin Patil[1], Prof. Saba Siraj
[1]Student, Bachelor of Computer Engineering
SP's Institute of Knowledge College Of Engineering, Pune.

Abstract: This paper presents a natural language interface to relational database. It introduces some classical NLDBI products and their applications and proposes the architecture of a new NLDBI system including its probabilistic context free grammar, the inside and outside probabilities which can be used to construct the parse tree, an algorithm to calculate the probabilities, and the usage of dependency structures and verb sub categorization in analysing the parse tree. Some experiment results are given to conclude the paper.

Keywords: Natural Language database Interface (NLDBI), Structured Query Language (SQL), Syntax analysis, Semantic analysis, Tokens, Tokenizing, Database, Natural Language Processing.

1. INTRODUCTION

N Question Answering (QA) domain, many QA systems, such as START, Ask MSR, NSIR have been developed to support the users searching the correct information about some topics. Among these systems, START may be considered as the best system that can return the good answers for users. However, START is only able to answer questions about concepts, but it could not answer the questions about causes and methods.

Furthermore, we had investigated an open source QA system. That is Open-Ephyra; it is an open framework for question answering (QA). It retrieves answers to natural language questions from the Web and other sources. It was developed on Java framework. This system has: a dictionary, a set of questions, method to find out the correct answers for questions. Once receiving a question, the system classify question into one of defined categories of question to analyse and split keywords base on the dictionary. After that, Open Ephyra uses these keywords to search in data set paragraphs that contain them. The result will be estimated adequate degree and the best result will be display to user.

After considering these QA systems, we assume that current QA systems perform the question processing base on this principle:

- Match the query to existing queries form.
- Generate a set of keywords or a set of queries in knowledgebase.
- Determine the result that fit to the question and generate the answer.
- Without semantic model of query.

In our research project, we specially focus our interests on building a particular QA system model appropriately used in the domain of document retrieval.

2. SOME RELATED EARLIER WORKS

Research in Natural Language Interface for Relational Databases began as far back as the 20th century. Since then the study and interest has continued to grow tremendously such that the area has become the most active in Human-Computer Interaction. The first Natural Language Interface for Relational Databases appeared in the 1970s, the NLIDB system was called LUNAR. After the development of the first NLIDB, many were built which were supposed to be an improvement on the apparent flaws of LUNAR. The presentation and acceptance of LUNAR was huge. The reason for such huge success with NLIDBs includes the fact that there are real-world benefits or payoffs that can be derived from this area of study, other fact is that the earlier experimented domain was a single

domain where execution of non-complex systems are easy and easily adaptable. Same feet were not achieved in the area of using complex databases. We highlight below, the development of some NL interfaces.

A. Lunar (1971)

Man had accomplished the complex task of both having a physical presence on the moon and that of positioning satellites in space that can bring results from observations done on the moon. Information of rock samples brought back from the moon, for example, chemical information were stored in a database, while literature reference on various samples were stored in another database. LUNAR helped provide answers to queries about any of the two information about a rock sample by the use of these databases. LUNAR had linguistic limitations and was able to handle 78% of user-requests.

B. Philiqa [Philips Question Answering Machine] (1977)

This system works by having a clear-cut distinction of the syntactic parsing and semantics of the user-defined query. It has three layers of semantic understanding:

a. English Formal Language
b. World Model Language
c. Database Language

Together, these three layers work to answer user-defined queries. Users did not achieve so much acceptance as the earlier developed LUNAR.

C. Ask (1983)

Ask was a complete information management system with an in-built database and the ability to communicate with multiple external databases using several computer applications which are accessible to users through the user's natural language query. Learning is the ability of a system to experience change based on a certain experience with an input such that it can perform an activity better and more efficiently next time. Since ASK had the ability to be taught new concepts by the user during conversation with the user, it can be said that ASK was a learning system.

D. Team (1987)

TEAM was an NLIDB whose developers concerned themselves with portability issues, as they wanted it to be easily implementable on a wide range of systems without compatibility issues. It was designed to be easily configured by database administrators with no knowledge of NLIDB. These feet affected the functionality of TEAM.

E. Precise (2004)

PRECISE introduced the concept of Semantically Tractable Sentences which are sentences whose semantic interpretation is done by the analysis of some dictionaries and semantic constraints. It was developed by Ana-Maria Popescu, Alexander Yates, David Ko, Oren Etzioni, and Alex Armanasu in 2004 at the University of Washington. When a natural language query is given to PRECISE, it takes the keywords in the sentence of the query, and matches the keywords to corresponding database structures. This, in fact is the major strength of PRECISE. PRECISE does this matching in two stages. The first is to narrow down the possible keywords using the Maximum Flow algorithm which finds a feasible, constraint-satisfying flow through a Flow Network having just a single source and a single sink, such that the flow is maximum; where a flow network is a directed graph in which each edge has a capacity and each edge receives a flow. By using the Maximum Flow algorithm, the maximum number of keywords is obtained, thereby increasing the chance of the natural language sentence to be accurately transformed to a formal SQL query as there will be enough keywords to compare with the PRECISE dictionary. The second stage is to analyse the syntactic structure of the sentence. PRECISE also has its own limitations.

Generally, some major flaws have been common to these interfaces and their ability to handle natural language processing. Users' feedback system has not been thoroughly handled in existing systems. Such systems learn when the user prompts command such as save text on the interface. This is worsened by the fact that, though they are considered as a NLI, their knowledgebase has been a concern in recent times such that can only get results that keyword based. The area of natural language that can be handled by NLIDBs is just a small subset, and this subset is even difficult to define due to Natural language complexity and the existence of ambiguity.

3. EXISTING SYSTEM

In relational databases, to retrieve information from a database, one needs to write a query in such way that the computer will understand and produce the desired output. The Structured Query Language (SQL) is normally used for retrieving data. The SQL normally are based on a Boolean interpretation of the queries. But some user requirements may not be answered explicitly by a classic querying system. It is due to the fact that the requirements' characteristics cannot be expressed by regular query languages.

4. PROPOSED SYSTEM

Problem statement/System Architecture Our proposed system NLDBI first transform the natural language question into transitional logical query, expressed in some internal meaning illustration language. The intermediate logical query expresses the meaning of the user's question in terms of high level world concepts. The logical query is then translated into the database's query language, and evaluated against the database.

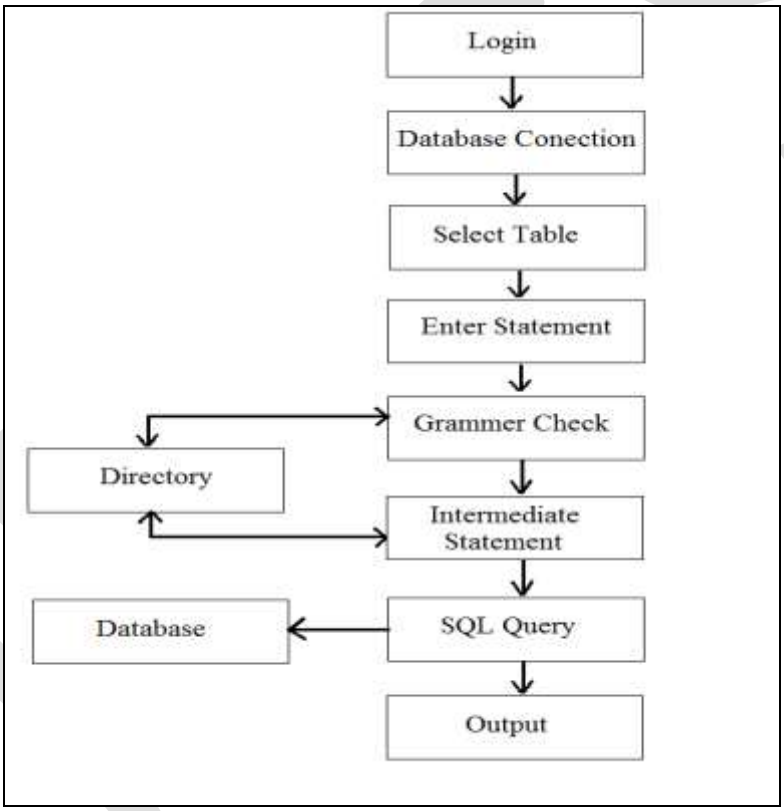

Fig 1: System Architecture

5. MATHEMATICAL MODEL

Table 1.Mathematical model

Sr No.	Description	Observation/Remarks
1	Let S be the system S={S1, S2, S3, S4} Where, S1=module that checks grammar S2=module that tokenize input S3=module that generate parse tree S4=module that generate SQL query	S identifies system set
2	S1={I, O, P, G, In, Cr, S, F} P={I, G, E} G={G1, G2, G3} Where, I= Query in English Language O= Intermediate query in English language In=Enter English statement Cr=Constraint S= Grammar of English query is correct F= Grammar is wrong G=function that check grammar E=function that gives error exception G1=function that scans input statement G2=function that checks semantic of input	The module that checks grammar. Constraint: • Question must be in English statement • Meaningful statements • Question must be in a Wh. form.

	G3=function that divide input into parts	
3	S2={ I, O, P, In, Cr, S, F} P={C, T} Where, I= Intermediate query in English language O= Tokens In= Meaningful statement Cr= Constraint S= Tokens generated F= Unidentified token C=function that scans input T=function that tokenize statement	The module that generate tokens from input statement Constraint: • Tokens should be having alphabets only
4	S3={ I, O, P, In, Cr, S, F} P={ M } Where, I= Tokens O= Intermediate query with mapped keywords P=Function that generate parse tree In= Tokens Cr=Constraints S= Keyword found in directory	The module that generate parse tree and maps keywords from directory Constraint: • Keyword should be present in directory

	F= Keyword on found in directory M=function the keywords from database	
5	S4={ I, O, P, In, Cr, S, F } P={Q, X, R} Where, I= Intermediate query with mapped keywords O= SQL Query P=function that generate SQL query In= Intermediate query with mapped keywords Cr=Constraints S= Valid SQL query generated F=Invalid SQL query generated Q=function that generate SQL query X=function that retrieve data from database R=function that display result	The module that generate SQL query and retrieve result. Constraint: • Complete intermediate query • All of the table names column name should be present in intermediate query

6. SYSTEM REQUIREMENTS

6.1 Functional Requirement:

6.1.1 User Interface:

Swing GUI

6.1.2 Hardware Requirement:

Processor: Dual Core

RAM: Minimum 1 GB

HDD: Minimum 80 GB

Operating System: Minimum Windows XP.

6.1.3 Software Requirements

Table 2: Software requirement of NLDBI system

	Software	Minimum	Recommended
Design Tool	Rational Rose	2000e	2000e
Development Tools	NetBeans	NetBeans 7.0	
Development Kit	JDK		
Development Platform	Windows	Windows XP	Windows XP
Others	Microsoft word	Word 2007	Word 2003

6.2 Non Functional Requirement:

6.2.1 Performance Requirement:

Server programs are supposed to serve multiple requests simultaneously on various TCP/IP connections. Client loads vary and so do requests per client. Taking that into consideration, the performance parameters of servers include the following: number and type of requests per second; latency time, measuring in milliseconds how long it takes to complete each new connection or request; throughput or the amount of data transmitted in response to a request measured in bytes per second. Finally, the server model, whether client-side or server-side, used to execute server programs establishes scalability. Scalability is a system property that refers to a system or network's ability to manage increasing workloads well and the ability to expand gracefully.

6.2.2 Safety Requirement:

The data handled in the Morphing Portal system is very vital. The server should always be confirmed to run properly and the data are saved to the database at consecutive intervals. Power is a significant feature and the power supply should be always taken care of. An Uninterrupted Power Supply is always recommended.

6.2.3 Software Quality Attribute:

A. Reliability:

The system will be designed with reliability as key feature:

- The system is guaranteed of providing the services to user according to his login information.
- This system is guaranteed to be reliable with maximum time.

B. Maintainability:

The system will be developed using the standard software development conventions to help in easy review and redesigning of the system. The system will be backed up by a full fledge documentation of the product which is available online as well as free to download.

C. Availability: The System is available on administrator demand.

7. DIAGRAMS

7.1 Data flow Diagram – Level 0

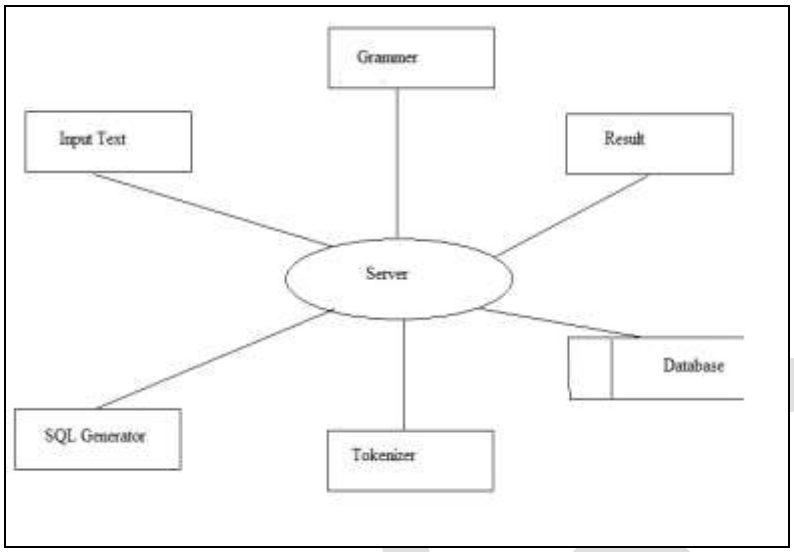

Fig 2: Dataflow diagram level 0

Data Flow Diagram Description

First User has to login then if login is invalid then exit else User proceeds. After successful login user has to input text. Grammar of input text is checked. Input text is tokenized. SQL Query is generated. Query is executed and result is fetched. If query is valid then result is displayed to User else exit. Users can logout from system. (As per shown in figure 2)

7.2. Sequence Diagram

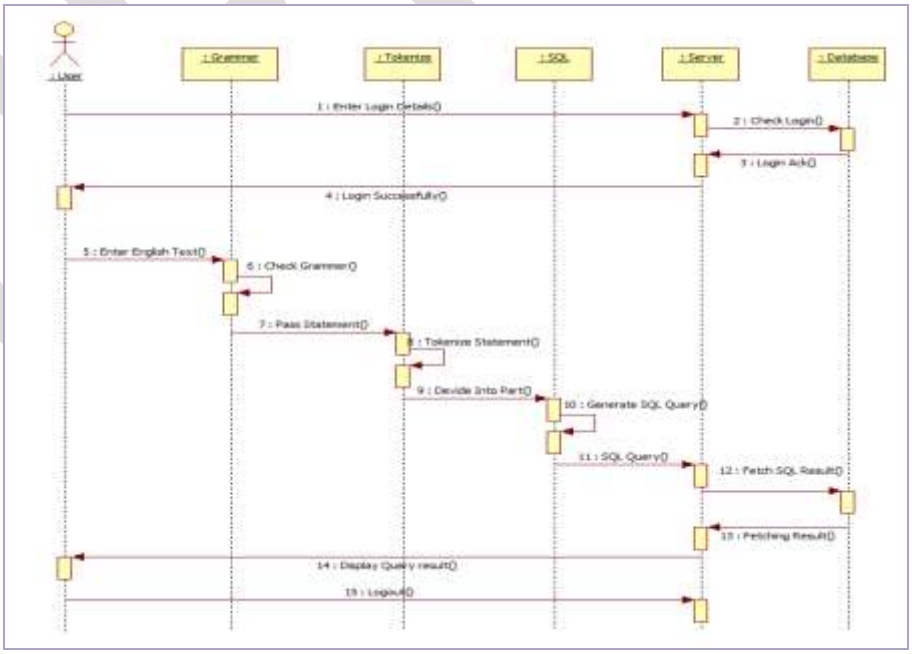

Fig 3: Sequence diagram

8. PERFORMANCE ANALYSIS

Table 3: Performance Table

Total Number of Input NL query	Correct Interpretation	Correct with additional information	Incorrect Interpretation	No Response
100	70	10	25	05

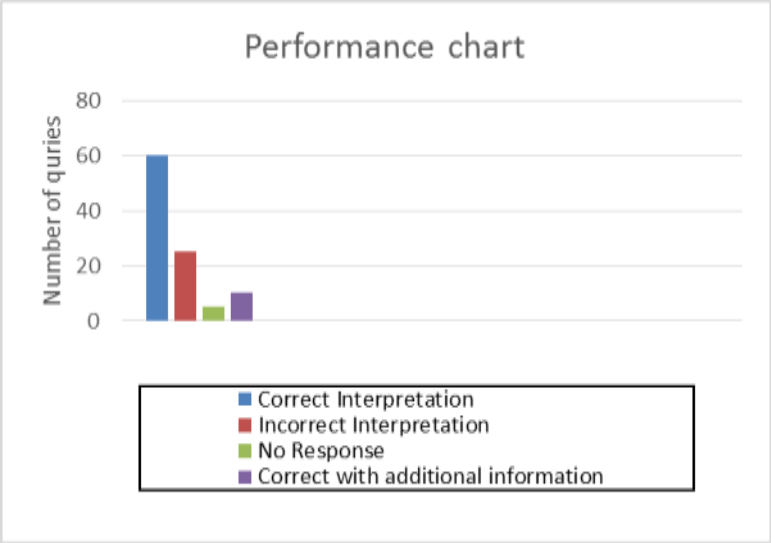

Fig 4: Performance chart

9. CONCLUSION

Natural Language Processing can bring powerful enhancement to virtually any computer program interface. This system is currently capable of handling simple queries with standard join conditions. Because not all forms of SQL queries are supported, further development would be required before system can be used within NLDBI. Alternative for integrating a database NLP component into the NLDBI were considered and assessed.

10. FUTURE SCOPE

- To design user interface for administrator where application can be tuned for new database.
- To execute queries that include aggregate function like sum ( ), max ( ), min ( ).
- To add the features to execute insert type of queries.
- To add features execute DDL statement like create, alter, drop type of queries.

To implement application for regional language like Marathi.

REFERENCES:

[1] "NATURAL LANGUAGE INT ERFACE USING SHALLOW PARSING" by RAJENDRA AKERKAR and MANISH JOSHI International Journal of Computer Science and Applications, Vol. 5, No. 3, pp 70 – 90.

[2] "A noval hindi language interface for databases" by Mr Mahesh Singh and Ms Nikita Bhati, International Journal of Computer Science and Mobile Computing Vol. 3, Issue. 4, April 2014, pg.1179 – 1189.

[3] "Recent Developments in Natural Language Interface to Database Systems" by Preeti Verma and Kulwant Kaur, International Journal of Innovation and Research in Computer Science, ISSN: 2349-2783, pg 20-26.

[4] "Efficient Transformation of a Natural Language Query to SQL" for Urdu by Rashid Ahmad, Mohammad Abid Khan and Rahman Ali, Proceedings of the Conference on Language & Technology 2009.

[5] "An Algorithm for Solving Natural Language Query Execution Problems on Relational Databases" by Enikuomehin A.O., Okwufulueze D.O., (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3, No. 10, 2012, pg 169-175.

[6] "Natural language query processing using probabilistic context free grammar" by Arati K. Deshpande and Prakash. R. Devale, International Journal of Advances in Engineering & Technology, May 2012, IJAET ISSN: 2231-1963, Vol. 3, Issue 2, pp. 568-573.

[7] "NATURAL LANGUAGE QUERY PROCESSING" by GaurRao and Dr.S.H.Patil, International journal of computer applications in applications in engineering, technology and sciences (IJ-CA-ETS), ISSN: 0974-3596, October '09 - March '10, Volume 2: Issue 2, Pg: 1-6.

[8] "Incremental Information Extraction Using Relational Databases" by Luis Tari, ieee transactions on knowledge and data engineering, vol. 24, no. 1, January 2012, pg 86-99.

[9] "Natural language Interface for Database: A Brief review" by Mrs. Neelu Nihalani, Dr. Sanjay Silakari and Dr. Mahesh Motwani, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011, ISSN (Online): 1694-0814, pg 600-608.

[10] "Natural language query processing using semantic grammar" by Gauri Rao, 2 Chanchal Agarwal, Snehal Chaudhry, Nikita Kulkarni and Dr. S.H. Patil, (IJCSE) International Journal on Computer Science and Engineering, Vol. 02, No. 02, 2010, 219-223