ISSN No. 0976-5697

Volume 8, No. 5, May-June 2017

International Journal of Advanced Research in Computer Science

SURVEY REPORT

Available Online at www.ijarcs.info

Survey on Natural Language Database Interfaces

Abhilasha Gautam
Department of Computer Science,
Ambedkar Institute of Advanced Communication
Technologies & Research, Delhi, India

Dr. Shailender Kumar
Department of Computer Science,
Ambedkar Institute of Advanced Communication
Technologies & Research, Delhi, India

Satvic Mittal
Department of Computer Science,
Ambedkar Institute of Advanced Communication
Technologies & Research, Delhi, India

Abstract: This is the era of information technology and most of our data or information is stored in the form of databases on our computers. To extract information from our database we need to use a structured language such as SQL (Structured Query Language). Thus only the person who has a good knowledge of the SQL language can interact with the database. Natural language can be used to retrieve the information from the database in an easier way. But computers cannot understand the natural language without any external help. Thus Natural language database interfaces(NLDBI) were developed, that converts the query given by the user in natural language into a language that is understood by the computer's database management system i.e. Database Query Language(DBQL). In this paper, we present a literature review of various Natural Language Database interfaces which have been built over the time and the new techniques that were added every time to make the interface faster than before. Then we present some future work that can be done in the field of Natural language Database interfaces to make them faster, reliable, robust and easier.

Keywords: Natural Language Database Interface (NLDBI); Database Query Language (DBQL); Structure Query language (SQL); Natural Language Processing (NLP); Artificial Intelligence(AI)

I. INTRODUCTION

Natural Language processing (NLP) [2] have become one of those techniques which are these days frequently used to interact with the computer using the native language used by the humans. Natural language processing comes under the study of Artificial Intelligence [2] which can be used for the retrieval of information, machine translation, and the linguistic analysis.

These days most of the work is done on computers. Thus the Natural language Processing[2] came into the picture to increase the human-computer interaction so that we do not need to learn the computer languages, commands, and the complex procedures. If we can interact with our computer in our natural language then it would make all the work way easier for even those who do not have a good knowledge of computers which was the main objective of Natural language processing.

Now coming to the storage of information on the computers. All the information is stored in the form of databases thus there was a great need to develop something through which we can interact with our databases using our natural language. This problem was resolved by using natural language processing techniques and developing an interface which can be used to retrieve information from the database using natural language and those interfaces are known as Natural Language Database Interfaces (NLDBI) [1].

The main objective of the NLDBIs was to make the retrieval of information easier using NL. While using an NLDBI the user need not have a good knowledge of programming languages or DBQL to have the access to data from the database. Earlier people used to learn the DBQL to interact with their database management systems but after the development of NLDBIs this work became easier but a lot of research is still to be done in the field of Natural Language Database Interfaces to make them faster and generic.

When you give a query in an NLDBI then it first automatically understands the natural language not just semantically but syntactically too and then convert the parsed natural language into a query that is accepted by the Database Management System i.e. a query in SQL. Thus a grammar needs to be developed for every NLDBI for a particular domain. This problem is now being dealt by using a Generic Natural Language Interface [3] which can take any type of natural language query and then convert it into a SQL query. We give details about this interface in further sections.

Our paper further is organized as follows .Section 2 presents the literature review of different NLDBIs that have been developed over the time. In section 3 we present the basic architecture of Natural Language Database Interface. Section 4 represents the methodologies on which mostly NLDBIs are based. Comparative analysis is done between various NLDBIs that have been developed over time in section 5. In section 6 we give an overview of the advantages and disadvantages of using NLDBIs. Section 7 presents the conclusion and possible extensions that can be made in the field of NLDBIs.

II. LITERATURE REVIEW

Various NLDBIs have been developed over time and a lot of research has been done and still going in this field. Some of the work that has been in this field and some popular NLDBIs that have been developed over time are as follows:-

A. LADDER

The LADDER[4] system was a natural language interface to a database developed by the joint efforts of Rich Fikes, Koichi Furukawa ,Gary Hendrix, Paul , Nils Nilsson, Bill Paxton, Jane Robinson, Daniel Sagalowicz, Jonathan Slocum, and Mike Wilber. It was designed specifically for information about US Navy ships. The LADDER[4] system parse questions to query a distributed database using semantic grammar. The system interleaves syntactic processing and semantic processing using semantic grammars techniques. For translating a question, first the input is parsed and the parse tree is then mapped to a database query.

This NLDBI have a three layered architecture. The first layer is the Informal Natural Language Access to Navy Data (INLAND). INLAND [4] accepts the query in natural language and produces a query to databases. The queries from the INLAND are inputted to the Intelligent Data Access (IDA) [4], which is the second layer of LADDER. The INLAND layer translates a fragment of a query to IDA for each lower level syntactic unit in the English language input query and these fragments are then merged to higher level syntactic units to be recognized. At the sentence level, the merged fragments are used as command for IDA. IDA would build an answer that is in accordance to the user's original queries. IDA also have the responsibility plan the correct sequence of file queries. File Access Manager (FAM) [4] is the third layer of the LADDER system. The responsibility of FAM is to locate generic files and to manage their access mechanism in a distributed database.

The LADDER system was employed in LISP [4]. The LADDER system was able to accurately process a database corresponding to a relational database with 14 tables and 100 attributes.

B. NaLIX

NaLIX (Natural Language Interface for an XML Database) is a Generic interactive natural language query interface to database system developed in 2006 by Ann Arbor, Huahai Yang, Yunyao Liand H. V. Jagadish at the University of Michigan.

NaLIX is system which uses Extensible markup language [5] database with Schema Free XQuery as the database query language. The NaLIX system consists of parse tree classifier, validator and a translator, which is responsible for query translation from natural language queries to XQuery [5]. For translation of natural language queries into corresponding XQuery expressions, three main steps are involved.

First is Parse Tree Classifier which identifies the word or phrase in the original natural language query which can be translated into corresponding components of XQuery [5], and is defined as a token (that is if it matches a XQuery component) or a marker (if it does not). Second is the Parse Tree Validator that validates then classifies parse tree and checks if the parse tree obtained can be mapped into XQuery or not. It also makes sure that the elements, attribute names and values present in the user query are found in the database. Third is the Parse Tree Translator which utilizes the structure of natural language query which is represented by the parse tree to generate the XQuery expression.

Meaningful Lowest Common Ancestor Structure (MLCAS) [5] of a set of nodes, is the concept behind the Schema free XQuery. That is for a given collection of keywords, each keyword has several candidate XML elements to relate. The MLCAS of these elements then automatically finds the relationships among these elements. The main advantage of Schema Free XQuery is that it does not require to map the query to the precise database schema.

C. PRECISE

Precise is a system developed by Ana-Maria Popescu, Oren Etzioni, David Ko, and Henry Kautz at the University of Washington in 2002. Precise reduces the interpretation of semantics in Natural Language Interface to a graph matching problem [6]. Precise returns a correct Structured Query Language (SQL) query for a broad class of natural language questions. Precise maps the word and phrases in the question to the corresponding database by reducing it to a graph matching problem and then computing the maximum flow in the graph. PRECISE was evaluated on GEO-Query [6] domain and hence was domain specific.

The precise Architecture involves many modules. First is the lexicon (1, 6) which is used to get set of ordered pairs in the form (token, database elements). This set of database elements is automatically extracted from the database. Then the natural language question is mapped to the set of all possible complete tokenizations of the question and to each token the types of database element is assigned. Next the matcher is used to generate the graph from the natural language question through the tokens generated by the tokenizer. It involves a constraint that all Token Value and Token Attribute must be matched with a subset of DB Value and DB Attribute. The matcher runs in polynomial time in the length of the natural language question. Then a parser is used to interpret the attachment relationships between the words of the question. Finally, the query generator generates a well formed SQL query using the database elements selected by the matcher.

Precise when compared with Microsoft's English Query product it was found that Precise made fewer errors by a factor of four or more. Although Precise is very accurate, it has its own weaknesses. One weakness is that though it achieves high accuracy in semantic question, this accuracy comes at the cost of recall. Second problem is that since Precise is based on heuristic approach, the system is not able to handle nested structures.

D. WASP

WASP stands for Word Alignment-based Semantic Parsing [7] developed by Yuk Wah Wong at the University of Texas, Austin. The system was developed to address a comprehensive goal for building a formal, complete, symbolic and meaningful representation of a natural language sentence, and hence could be applied to NLDBI domain. Prolog [7] is a predicate logic which is used as the formal query language. Given a set of natural language sentences marked up with their correct formal query language, WASP builds a semantic parser using the corpus. Statistical machine translation technique is used for the building the knowledge base and therefore WASP does not require any prior knowledge of the syntax. WASP was specific to GEO-QUERY (6, 7) domain. GEO- QUERY corpus contains 250 questions in the test set and of 880 questions in the training set, which are combined together into one larger data set. Each data set was divided in to 10 equal-sized subsets, and the system performance was estimated by a standard 10-fold cross validation. WASP achieved 86.14% precision and 75.00% recall in the GEOQUERY (6, 7) domain. The system was also tested on a variety of other natural languages such as English, Spanish, Japanese and Turkish. The ability of WASP to build a semantic parser from annotated corpora [7] gives it strength. WASP uses statistical machine translation with minimal supervision. Due to this, the system is not required to manually develop a grammar in different domains.

E. NLI-RDB

A natural language interface is created using Conversational Agent (CA)[8], Information Extraction (IE), and Object Relational Mapping (ORM) framework. CA is used to remove ambiguity from the user's queries and improving the user interaction. IE is used in extraction of named entities for mapping natural language queries into database queries. Hibernate framework is used as the ORM [8] framework. Concepts Object Oriented Paradigms (OOPs) differ from Relational Database (RDBs), thus hibernate reduces the complexity in generating SQL statements.

The System contains five components. The first component is the User Interface which provide the interaction between the user and the application. Next component is the Text Preparation which prepares the text for analyzing by first cleaning the text of unnecessary characters and words, and then tokenizing them. Third component is the Engine Algorithm [8]. Engine algorithm is the main component which generates the agent response and Hibernate Query from the natural language query. It uses Information Extraction to extract entities like object names and their attributes. Conversational Agent is used to match the tokenized text with predefined patterns to either guide the user through the responses or to recognize object attributes with their possible conditions. Then the HQL generator [8] generates the HQL query using the extracted information and the matched patterns. The fourth component is the ORM framework, that is, Hibernate. Hibernate maps an object oriented model to a relational database. The HQL query is used by Hibernate to generate the SQL query, which is then executed on a relational database. The results are then presented to the user through the user interface. Fifth and last component is the Relational Database itself which holds the domain information. That is, all the entities with its attributes and values and stored in the form of tables as a database.

III. ARCHITECTURE

Figure shows the various components that can be found in every NLDBI and how a basic NLDBI works and the stages of the natural query processing in an NLDBI.

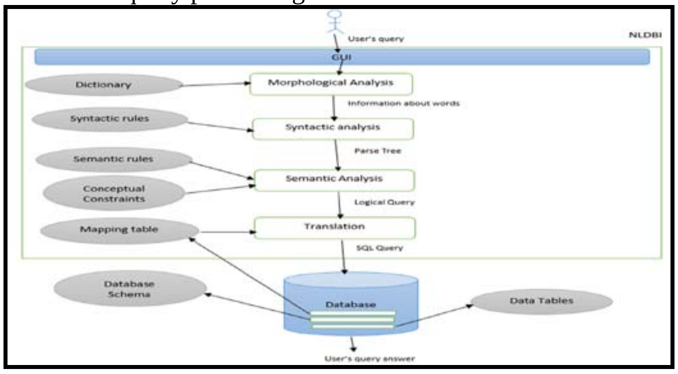

Figure 1: Architecture of NLDBI

A. Selecting a Template (Heading 2)

IV. METHODOLOGY

Various methodologies are used for developing different types of natural language database interfaces. Different techniques are still being developed to make the use of NLDBIs much easier and faster. Some of the common techniques that act as ground for various other techniques are as follows:-

A. Pattern-Matching

While developing an NLDBI the prototype systems were mainly dependent on pattern matching since every user input has to be directly matched with the database. The system can build a query only when a match occurs between a pattern and given user input. The limitations of the pattern matching process are that it has to be made specifically for each database since some details of the database are intermixed within the code of the system. Also, these type of systems works only for less complex and small number of patterns. Some of the advantages of pattern-matching systems are that they are simple as compared to others since there is no need for parsing and interpretation modules like the other systems. When the user input is not in range with the given sentences for which the system is designed then sometimes pattern-matching systems [1] even come up with some kind of justification. The best example till date for the pattern-matching approach is ELIZA [13]. ELIZA rephrased the user input into questions and the gives an answer to those questions.

B. Syntax-Based Systems

In systems which uses syntax based approach [9], a parse tree is generated. The parse tree is generated by parsing the query given by the user and then it is directly mapped to and expression in DBQL. In this approach the syntactic structure of user's questions is defined by the generated grammar. In this approach the detailed information about the structure of the sentence is provided which is the main advantage of syntax-based systems. This parse tree starts from a single word and its part of speech extraction, then a cluster of words is formed to form a phrase structure, and then these phrases are combined to form complex phrases. A continuous check on the structure is done till the complete sentence is built and accessed. This information can be used to map certain production rules with their semantic meaning. Some systems that were developed using syntax-based approach are LUNAR [12], LADDER [4] etc.

C. Semantic-Based Systems

Semantic-based systems [11] are similar to the syntax-based systems except the fact that the parse tree generated in semantic based approach are generally simpler than the ones in syntax-based approach. The parse trees are made simpler by using several approaches like removing the unnecessary nodes or by combining them. In semantic based systems without the need of complex parse tree structures, we can reflect the semantic representation in a better way using semantic grammar [11]. The only difficulty which we face in this technique is that we need to have a prior knowledge of domain elements. Semantic grammar approach results into less ambiguity by providing a special way of naming the tree node as compared to the syntax-based approach.

D. *Intermediate Representation Languages*

This approach is used widely these days because it is easier to use and less domain specific as compared to the other approaches. In this approach user input is not directly converted into Database query language, rather it is first converted into an intermediate logical query [3] which is then transformed into some DBQL. Directly translating a sentence in natural language into query language using syntax-based approach is very difficult and the reason behind the upcoming of this approach. In this approach, a sentence in the natural language is mapped into logical query first and then this logical query is translated into general DBQl such as SQL. A good example of this approach is the NLDBI system developed at the University of Essex [3] which have a multi-stage transformation process. In this process, we can also use more than one intermediate representation languages.

V. COMPARATIVE ANALYSIS

All the parameters on the basis of which we have compared the systems are explained as follows: -

Table I: Comparative analysis of various NLDBIs

Systems / Properties	LADDER (4)	PRECISE (5)	NALIX (6)	WASP (7)	NLI-RDB (8)
RELIABILITY	LOW	HIGH	LOW	HIGH	HIGH
DOMAIN DEPENDENT	YES	YES	NO	YES	NO
ACCURACY	LOW	HIGH	LOW	HIGH	HIGH
DOMAIN	US NAVY SHIPS	GEO-QUERY	GENERIC	GEO-QUERY	GENERIC
PORTABILITY	LOW	HIGH	LOW	LOW	HIGH

- Reliability: - It measures the degree of correctness of the results given by the system in stated conditions.
- Domain dependent: - It tells us if the system can work only for a particular domain or it can be used for any type of questions.
- Accuracy: - It gives the measure of accuracy of the SQL query returned by the NLDBI.
- Domain: - It gives us for which domain the system can be used.
- Portability: - It tells us if we can use the NLDBI for different database management system or not.

A. *Advantages of NLDBIs*

There are various advantages of using an NLDBI which are as follows:-

- NO ARTIFICIAL LANGUAGE: - While using an NLDBI there is no need to learn any artificial language for gaining access to the database. For data retrieval or data manipulation in a database, we need to learn a formal query language like SQL which are difficult to learn and master especially for a non-technical person.
- SIMPLE, EASY TO USE: - NLDBI makes the user interaction with the database very easy. In a normal database, we need to put a formal query sentence designed in some formal query language while using an NLDBI we need to provide only sentences written in our natural language.
- BETTER FOR SOME QUESTIONS: - There is some kind of questions which cannot be easily expressed in DBQL such a questions involving negation or quantification but they can be expressed easily in natural language. So, for this kind of questions NLDBI serves very helpful.
- FAULT TOLERANCE: - For writing a query in a DBQL we need to follow certain rules and if any errors are made then the system automatically rejects the input. While in the case of NLDBI there is some tolerance of minor grammatical errors. In the case of incomplete sentences, NLDBI provides some support while computer system does not.
- EASY TO USE FOR MULTIPLE DATABASE TABLES: - Queries in which multiple databases are involved are easily expressed in natural language interface as compared to the normal graphical user interface.

B. *Disadvantages of NLDBIs*

There are some disadvantages too of using an NLDBI which are as follows:-

- FAILURES ARE NOT HANDLED PROPERLY: - No explanation for system failures is provided. Most of the time the user has to identify the cause of error itself. Some users try to put the question in other ways and some might just leave it unanswered.
- LINGUISTIC COVERAGE IS NOT OBVIOUS: - Most of the NLDBIs available are domain specific i.e. they can handle only some subsets of a natural language. In some cases, the system cannot answer certain answers belonging to their own subset. Some generic NLDBIs have started to be developed but still, these interfaces also depend upon the reach of grammar that is provided to them which is not in the case of formal languages. The coverage of formal language is obvious and statements following given rules are guaranteed to give proper results.
- FALSE EXPECTATIONS: - NLDBIs are assumed to be intelligent systems as they are processing a natural language. Thus sometimes users tempt to ask questions involving certain judgment, complex ideas etc. for which we cannot rely upon an NLDBI system.

VI. CONCLUSION

This paper attempts to serve two purposes mainly which are: Firstly, to introduce the reader with the basic concept of NLDBIs and common techniques based on which NLDBIs are developed. It presents the user with some of the research work that has been done in the field of NLDBIs. Secondly, we conclude some of the advantages and disadvantages of using a NLDBI. Lots of research has been still on going in this field since last few decades and thus surveying the field cannot be achieved completely at any given moment. Over the period, many natural language database systems have been developed, all with different capabilities and specifications. On the top we have LADDER which was developed in 1978 for information about US Navy ships. LADDER was domain specific and had high error rate. This urged for systems with higher reliability and performance which led to the development of systems like PRESICE and WASP. Though these systems boosted higher performance but were still domain dependent. To deal with this problem generic systems were developed like NALIX and NLI-RDB.

Though development in natural language database system have advanced in last few years, it is still not being commonly used. Algorithms are needed to developed and incorporated for query optimization. We are going to pursue the research further with the intent of incorporating the functions of natural language database interface in PostgreSQL.

VII.REFERENCES

[1] N.Nihalani, S.Silakari, M.Motwani. Natural language Interface for Database: A Brief review, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 2, 2011.

[2] R.Reilly, N.Sharkey, Connectionist Approaches to Natural Language Processing, Lawrence Erlbaum and Associates, Hilldale, NJ, 1992.

[3] H.Bais, M.Machkour, L.Koutti, A Model of a Generic Natural Language Interface for Querying Database, MECS, 2016.

[4] E.Sacerdoti, Language Access to Distributed Data with Error Recovery, Knowledge Repr.- l : Sacerdoti 196.

[5] Y.Li, H.Yang, H.Jagadish, NaLIX: an Interactive Natural Language Interface for Querying XML, SIGMOD, Baltimore, Maryland, USA, 2005.

[6] A.Popescu, O.Etzioni, H.Kautz, High Precision Natural Language Interfaces to Databases: a Graph Theoretic Approach, American Association for Artificial Intelligence 2002.

[7] Y.Wong, R.Mooney, Learning for Semantic Parsing with Statistical Machine Translation, Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL-2006). pp. 439-446, New York City, NY, June 2006

[8] A.Alghamdi, M.Owda, K.Crockett (2017) Natural Language Interface to Relational Database (NLI-RDB) Through Object Relational Mapping (ORM).

[9] A.Sawant, P.Lambate, A.Zore, Natural Language to Database Interface, International Journal of Engineering Research & Technology (IJERT) Feb 2014.

[10] F.Li, H.Jagadish, NaLIR: An Interactive Natural Language Interface for Querying Relational Databases, SIGMOD'14, Snowbird, UT, USA, 2014.

[11] G.Rao, C.Agarwal, S.Chaudhry, N.Kulkarni, et.al, NATURAL LANGUAGE QUERY PROCESSING USING SEMANTIC GRAMMAR, (IJCSE) International Journal on Computer Science and Engineering.

[12] W.Beranek, N.Inc, Progress in natural language understanding: an application to lunar geology, AFIPS '73 Proceedings of the June 4-8, 1973, national computer conference and exposition Pages 441-450

[13] E. Rich, "Natural-Language Interfaces," in Computer, vol. 17, no. 9, pp. 39-47, Sept. 1984.doi: 10.1109/MC.1984.1659244IEEE.