Natural language Interface for Database: A Brief review

Mrs. Neelu Nihalani [1], Dr. Sanjay Silakari [2], Dr. Mahesh Motwani [3]

**[1] Reader, Department of Computer Applications, UIT RGPV
Bhopal, MP India**

**[2] Prof. & Head, Dept. of CSE UIT, RGPV
Bhopal, MP, India**

**[3] Reader, Dept. of CSE, JEC, Jabalpur
Jabalpur, MP, India**

ABSTRACT

Information is playing an important role in our lives. One of the major sources of information is databases. Databases and database technology are having major impact on the growing use of computers. Almost all IT applications are storing and retrieving information from databases. Retrieving information database requires knowledge of database languages like SQL. The Structured Query Language (SQL) norms are been pursued in almost all languages for relational database systems. However, not everybody is able to write SQL queries as they may not be aware of the structure of the database. So this has led to the development of Intelligent Database System (IDBS) . There is an overwhelming need for non-expert users to query relational databases in their natural language instead of working with the values of the attributes. As a result many intelligent natural language interfaces to databases have been developed, which provides flexible options for manipulating queries. The idea of using Natural Language instead of SQL has prompted the development of new type of processing called Natural language Interface to Database. NLIDB is a step towards the development of intelligent database systems (IDBS) to enhance the users in performing flexible querying in databases. This paper is an introduction to Intelligent Database System and Natural Language Interface to Databases. Then a brief overview of NLIDB subcomponents is given and then discussion then moves on to NLIDB architectures and various approaches for the development of NLIDB systems.

Keywords: Databases, Database Management System (DBMS), Structured Query Language (SQL), Natural Language Interface for Databases (NLIDB), Intelligent Database System (IDBS), Flexible Querying.

1. INTRODUCTION

Databases are gaining prime importance in a huge variety of application areas employing private and public information systems . Databases are built with the objective of facilitating the activities of data storage, processing, and retrieval associated with data management in information systems. Due to the progress and in-deep applications of computer technologies, the widespread applications of web technology in several areas to be accurate, databases have become the repositories of huge volumes of data In relational databases, to retrieve information from a database, one needs to formulate a query in such way that the computer will understand and produce the desired output. The Structured Query Language (SQL) norms are been pursued in almost all languages for relational database systems. The SQL norms 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. Many novel-generation database applications stipulate intelligent information management necessitating efficient interactions between the users and database. In recent times, there is a rising demands for non-expert users to query relational databases in a more natural language encompassing linguistic variables and terms, instead of operating on the values of the attributes.

Therefore the idea of using natural language instead of SQL has prompted the development of new type of processing method called Natural Language Interface to Database systems (NLIDB). NLIDB is a step towards the development of intelligent database systems (IDBS) to enhance the users in performing flexible querying in databases.

2. Intelligent database System (IDBS)

An IDBS is endowed with a data management system able to manage large quantities of persistent data to which various forms of reasoning can be applied to infer additional data and information. This includes knowledge representation techniques, inference techniques, and intelligent user interfaces – interfaces which extend beyond the traditional query language approach by making use of natural language facilities[1,2]. These techniques play important role in enhancing databases systems : knowledge representation techniques allow one to represent better in the DB the semantics of the application domains, inference techniques

allow one to reason about data to extract additional data and information, Intelligent user interfaces help users to make requests and receive the replies.

Intelligent databases systems are the systems that manage information in a natural way, making that information easy to store, access and use. One of the main reasons for using intelligent database system is that we live in a state of information glut. To simply survive in today's society, we need to access and use this information. By using intelligent databases system we can have better access to, and use of, more kinds of information that they could otherwise. This means intelligent databases systems should[2]

- Provide high-level intelligent tools that provide new insights into the contents of the database by extracting knowledge from data.
- Make information available to larger numbers of people because more people can now utilize the system due to its ease of use.
- Improve the decision making process involved in using information after it has been retrieved by using higher level information models
- Interrelate information from different sources using different media so that the information is more easily absorbed and utilized by the user.
- Use of knowledge and inference, making it easier to retrieve, view and make decisions with information.

In recent times, there is a rising demands for non-expert users to query relational databases in a more natural language encompassing linguistic variables and terms, instead of operating on the values of the attributes. Intelligent interface for database systems, a promising approach, enhance the users in performing flexible querying in databases. The research and advancement of NLIDB, an important step towards the development of intelligent databases system and it has emerged as a new discipline and have fascinated the attention to number of researchers.

3. Natural language interface to databases (NLIDB)

Natural Language Interfaces is a hot area of research since long. The purpose of Natural language Interface to Database System is to accept requests in English or any other natural language and attempts to 'understand' them or we can say that Natural language interfaces to databases (NLIDB) are systems that translate a natural language sentence into a database query [3]. Although the earliest research has started since the late sixties[3], NLIDB remains as an open research problem.

A complete NLIDB system will benefit us in many ways. Anyone can gather information from the database by using such systems .Additionally, it may change our perception about the information in a database. Traditionally, people are used to working with a form; their expectations depend heavily on the capabilities of the form. NLIDB makes the entire approach more flexible, therefore will maximize the use of a database.

There are many applications that can take advantages of NLIDB. In PDA and cell phone environments, the display screen is not as wide as a computer or a laptop. Filling a form that has many fields can be tedious: one may have to navigate through the screen, to scroll, to look up the scroll box values, etc. Instead, with NLIDB, the only work that needs to be done is to type the question similar to the SMS (Short Messaging System).

3.1 Sub Components of NLIDB

Computing scientists have divided the problem of natural language access to a database into two sub-components:

- Linguistic component
- Database component

Linguistic Component

It is responsible for translating natural language input into a formal query and generating a natural language response based on the results from the database search.

Database Component

It performs traditional Database Management functions. A lexicon is a table that is used to map the words of the natural input onto the formal objects (relation names, attribute names, etc.) of the database. Both parser and semantic interpreter make use of the lexicon. A natural language generator takes the formal response as its input, and inspects the parse tree in order to generate adequate natural language response. Natural language database systems make use of syntactic knowledge and knowledge about the actual database in order to properly relate natural language input to the structure and contents of that database. Syntactic knowledge usually resides in the linguistic component of the system, in particular in the syntax analyzer whereas knowledge about the actual database resides to some extent in the semantic data model used. Questions entered in natural language translated into a statement in a formal query language. Once the statement unambiguously formed, the query is processed by the database management system in order to produce the required data. These data then passed back to the natural language component where generation routines produce a surface language version of the response.

4. Advantages and Disadvantages of NLIDB

This section discusses advantages and disadvantages of NLIDB, most of them cited from[3]**Advantages of NLIDB**

i. No Artificial Language

One advantage of NLIDBs is supposed to be that the user is not required to learn an artificial communication language. Formal query languages like SQL are difficult to learn and master, at least by non-computer-specialists.

ii. Simple, easy to use

Consider a database with a query language or a certain form designed to display the query. While an NLIDB system only requires a single input, a form-based may contain multiple inputs (fields, scroll boxes, combo boxes, radio buttons, etc) depending on the capability of the form. In the case of a query language, a question may need to be expressed using multiple statements which contain one or more sub- queries with some joint operations as the connector.

iii. Better for Some Questions

It has been argued that there are some kind of questions (e.g. questions involving negation, or quantification) that can be easily expressed in natural language, but that seem difficult (or at least tedious) to express using graphical or form-based interfaces. For example, "Which department has no programmers?" (Negation), or "Which company supplies every department?" (Universal quantification), can be easily expressed in natural language, but they would be difficult to express in most graphical or form-based interfaces. Questions like the above can, of course, be expressed in database query languages like SQL, but complex database query language expressions may have to be written.

iv. Fault tolerance

Most of NLIDB systems provide some tolerances to minor grammatical errors, while in a computer system; most of the time, the lexicon should be exactly the same as defined, the syntax should correctly follow certain rules, and any errors will cause the input automatically be rejected by the system. In the case of incomplete sentences, most of computer systems do not provide any support.

v. Easy to Use for Multiple Database Tables

Queries that involve multiple database tables like "list the address of the farmers who got bonus greater than 10000 rupees for the crop of wheat", are difficult to form in graphical user interface as compared to natural language interface.

b. Disadvantages of NLIDB

i. Linguistic coverage is not obvious

Currently all NLIDB systems can only handle some subsets of a natural language and it is not easy to define these subsets. Even some NLIDB systems cannot answer certain questions belong to their own subsets. This is not the case in a formal language. The formal language coverage is obvious and any statements that follow the given rules are guaranteed to give the corresponding answer.

ii. Linguistic vs. conceptual failures

In the case of NLIDB system failures, it is often the case that the system does not provide any explanation of what causes the system to fail. Some users may try to rephrase the question or just leave the question unanswered. Most of the time, it is up to the users to determine of the causes the errors.

iii. False expectations

People can be misled by an NLIDB system's ability to process a natural language: they may assume that the system is intelligent .Therefore rather than asking precise questions from a database, they may be tempted to ask questions that involve complex ideas, certain judgments, reasoning capabilities, etc, which an NLIDB system cannot be relied upon.

5. Various Approaches Used for Development of NLIDB Systems

Natural language is the topic of interest from computational viewpoint due to the implicit ambiguity that language possesses. Several researchers applied different techniques to deal with language. Next few sub-sections describe diverse strategies that are used to process language for various purposes.

a. Symbolic Approach (Rule Based Approach)

Natural Language Processing appears to be a strongly symbolic activity. Words are symbols that stand for objects and concepts in real worlds, and they are put together into sentences that obey well specified grammar rules. Hence for several decades Natural Language Processing research has been dominated by the symbolic approach [6].

Knowledge about language is explicitly encoded in rules or other forms of representation. Language is analysed at various levels to obtain information. On this obtained information certain rules are applied to achieve linguistic functionality. As Human Language capabilities include rule-base reasoning, it is supported well by symbolic processing. In symbolic processing rules are formed for every level of linguistic analysis. It tries to capture the meaning of the language based on these rules.

b. Empirical Approach (Corpus Based Approach)

Empirical approaches are based on statistical analysis as well as other data driven analysis, of raw data which is in the form of text corpora. A corpus is collections of machine readable text. The approach has been around since NLP began in the early 1950s. Only in the last 10 years or so empirical NLP has emerged as a major alternative to rationalist rule-based Natural Language Processing.

Corpora are primarily used as a source of information about language and a number of techniques have emerged to enable the analysis of corpus data. Syntactic analysis can be achieved on the basis of statistical probabilities estimated from a training corpus. Lexical ambiguities can be resolved by considering the likelihood of one or another interpretation on the basis of context.
Recent research in computational linguistics indicates that empirical or corpus –based methods are currently the most promising approach to developing robust, efficient natural language processing (NLP) systems[4,5]. These methods automate the acquisition of much of the complex knowledge required for NLP by training on suitably annotated natural language corpora, e.g. tree-banks of parsed sentences[7].

Most of the empirical NLP methods employ statistical techniques such as n-gram models, hidden Markov models (HMMs), and probabilistic context free grammars (PCFGs). Given the successes of empirical NLP methods, researchers have recently begun to apply learning methods to the construction of information extraction systems[8,9,10]. Several different symbolic and statistical methods have been employed, but most of them are used to generate one part of a larger information extraction system. Majumder, experimented N-gram based language modeling and claimed to develop language independent approach to IR and Natural Language Processing[11].

c. Connectionist Approach (Using Neural Network)

Since human language capabilities are based on neural network in the brain, Artificial Neural Networks (also called as connectionist network) provides on essential starting point for modeling language processing. In the recent years, the field of connectionist processing has seen a remarkable development. The sub-symbolic neural network approach holds a lot of promise for modeling the cognitive foundations of language processing. Instead of symbols, the approach is based on distributed representations that correspond to statistical regularities in language.
There has also been significant research applying neural-network methods to language processing [12,13] However, there has been relatively little recent language research using sub-symbolic learning, although some recent systems have successfully employed decision trees transformation rules and other symbolic methods . SHRUTI[14] system is a neurally inspired system for event modeling and temporal processing at a connectionist level.

6. Architecture of NLIDB systems

This section describes architectures adopted in existing systems.

a. Pattern Matching systems

The early efforts in the NL interfaces area started back in ties [16]. Prototype systems had appeared in the late sixties and early seventies. Many of these systems relied on pattern matching to directly mapping the user input to the database [15]. Formal LIst Processor (FLIP) is an early language for pattern-matching based on LISP structure [17] works on the bases that if the input matches one of the patterns then the system is able to build a query for the database. In the pattern matching
based systems, the database details were inter-mixed into the code, limited to specific databases and to the number and complexity of the patterns. As the usage of databases has spread during the 1970's, the concept of user interface presented new challenges to the designers. One approach was the use of natural language processing, where the user interactively is allowed to interrogate the stored data.

The main advantage of the pattern-matching approach is its simplicity. In such systems no elaborate parsing and interpretation modules are needed, and the systems are easy to implement. Also, pattern-matching systems often manage to come up with some reasonable answer, even if the input is out of the range of sentences the patterns were designed to handle. One of the best natural language processing system that role in this style is ELIZA. ELIZA functions by processing users, by these responses to the scripts. It typically says differently and rephrased the statements of the users as questions and replies the answers of those questions to the 'patient. ELIZA was programmed by Mr. Joseph Weizenbaum in nearly from 1964 to 1966.

b. Syntax-Based Systems

In syntax-based systems the users question is parsed (i.e. analyzed syntactically) and the resulting parse tree is directly mapped to an expression in some database query language. Syntax-based systems use a grammar that describes the possible syntactic structures of the users questions. Syntax-based NLIDBs usually interface to application-specific database systems that provide database query languages carefully designed to facilitate the mapping from the parse tree to the database query. It is usually difficult to devise

mapping rules that will transform directly the parse tree into some expression in a real-life database query language (e.g. SQL).

The main advantage of using syntax based approaches is that they provide detailed information about the structure of a sentence. A parse tree contains a lot of information about the sentence structure; starting from a single word and its part of speech, how words can be grouped together to form a phrase, how phrases can be grouped together to form more complex phrases, until a complete sentence is built. Having this information, we can map the semantic meanings to certain production rules (or nodes in a parse tree).

Unfortunately not all nodes should be mapped, some nodes have to be left just as they are without adding any semantic meanings. And it is not always clear which nodes should be mapped and which should not. Moreover the same node in different parse trees is not necessarily going to be translated in all the trees. The second problem is a sentence can have multiple correct parse trees, and if all are translated, they may lead to different query results. The last problem is that it is difficult for a syntax based approach to directly map a parse tree into some general database query language, such as SQL (Structured Query Language). In semantic grammar systems, the question-answering is still done by parsing the input and mapping the parse tree to a database query. The difference, in this case, is that the grammar's categories do not necessarily correspond to syntactic concepts. Semantic information about the knowledge domain is hard-wired into the semantic grammar that"s why systems based on this approach are very difficult to port to other knowledge domains a new semantic grammar has to be written whenever the NLIDB is configured for a new knowledge domain. Semantic grammar categories are usually chosen to enforce semantic constraints [18]. Much of the systems developed till now like LUNAR, LADDER, use this approach of semantic grammar.

c. Semantic Grammar Systems

A semantic grammar system is very similar to the syntax based system, meaning that the query result is obtained by mapping the parse tree of a sentence to a database query. The basic idea of a semantic grammar system is to simplify the parse tree as much as possible, by removing unnecessary nodes or combining some nodes together. Based on this idea, the semantic grammar system can better reflect the semantic representation without having complex parse tree structures. Therefore, a production rule in a semantic grammar system does not necessarily correspond to the general syntactic concepts.
Instead of smaller structures, the semantic grammar approach also provides a special way for assigning a name to a certain node in the tree, thus resulting in less ambiguity compared to the syntax based approach. Both of the ambiguities that can occur in mapping a node to its semantic label and the number of different parse trees which are possible for a particular sentence.

The main drawback of semantic grammar approach is that it requires some prior- knowledge of the elements in the domain, therefore making it difficult to port to other domains. In addition, a parse tree in a semantic grammar system has specific structures and unique node labels, which could hardly be useful for other applications. Regardless, there are on-going attempts to automatically build the grammar rules by obtaining the prior-knowledge based on user interaction or by automatically extracting it from a corpus.

d. Intermediate Representation Languages

Most current NLIDBs first transform the natural language question into an intermediate logical query, expressed in some internal meaning representation language. The intermediate logical query expresses the meaning of the user's question in terms of high level world concepts, which are independent of the database structure. The logical query is then translated to an expression in the database's query language, and evaluated against the database

Due to the difficulties of directly translating a sentence into a general database query languages using a syntax based approach, the intermediate representation systems were proposed. The idea is to map a sentence into a logical query language first, and then further translate this logical query language into a general database query language, such as SQL. In the process there can be more than one intermediate meaning representation language [1]. Figure 6.1 shows a possible architecture of an intermediate representation language system.

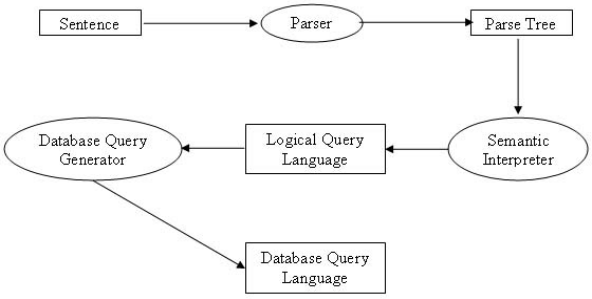

Figure 6.1 Intermediate Representation Language Architecture

The transformation from a logical query language to a database query language does not need to be made in one step. As an example, an NLIDB system developed at the University of Essex uses a multi-stage transformation process [5]. The first logic query is in the from of λ-calculus,

which is then transformed to a first-order predicate logic, universal domain relational calculus, domain relational calculus, tuple relational calculus, and finally SQL.

In the intermediate representation language approach, the system can be divided into two parts. One part starts from a sentence up to the generation of a logical query. The other part starts from a logical query until the generation of a database query. In the part one, The use of logic query languages makes it possible to add reasoning capabilities to the system by embedding the reasoning part inside a logic statement. In addition, because the logic query languages is independent from the database, it can be ported to different database query languages as well as to other domains, such as expert systems and operating systems [1].

7. History of NLIDB

The very first attempts at NLP database interface are just as old as any other NLP research. Asking questions to databases in natural language is very convenient and easy method of data access, specially for casual users who do not understand complex database query language such as SQL.

Here are some examples of the Natural Language Interface to Database systems :

a. LUNAR system

The system LUNAR [20] is a system which answers questions about samples of rocks brought back from the moon. The system was informally introduced in 1971. To accomplish its function the LUNAR system uses two databases; one for the chemical analysis and the other for literature references. The LUNAR system uses an Augmented Transition Network (ATN) parser and Woods' procedural Semantics. According to [19], the LUNAR system performance was quite impressive; it managed to handle 78% of requests without any errors and this ratio rose to 90% when dictionary errors were corrected. But these figures may be misleading because the system was not subject to intensive use due to the limitation of its linguistic capabilities.

b. LADDER

The LADDER system was designed as a natural language interface to a database of information about US Navy ships. According to [22], the LADDER system uses semantic grammar to parse questions to query a distributed database. The system uses semantic grammars technique that inter-leaves syntactic and semantic processing. The question answering is done via parsing the input and mapping the parse tree to a database query. The system LADDER is based on a three layered architecture. The first component of the system is for Informal Natural Language Access to Navy Data (INLAND), which accepts questions in a natural language and produces a query to the database. The queries from the INLAND are directed to the Intelligent Data Access (IDA), which is the second component of LADDER. According to [21], the INLAND component builds a fragment of a query to IDA for each lower level syntactic unit in the English language input query and these fragments are then combined to higher level syntactic units to be recognized. At the sentence level, the combined fragments are sent as a command to IDA. IDA would compose an answer that is relevant to the user's original query in addition to planning the correct sequence of file queries. The third component of the LADDER system is for File Access Manager (FAM).The task of FAM is to find the location of the generic files and manage the access to them in the distributed database. The system LADDER was implemented in LISP. At the time of the creation of the LADDER system was able to process a database that is equivalent to a relational database with 14 tables and 100 attributes.

c. RENDEZVOUS System

This system appeared in late seventies. In this, users could access databases via relatively unrestricted natural language. In this Codd‟s system, special emphasis is placed on query paraphrasing and in engaging users in clarification dialogs when there is difficulty in parsing user input.

d. PLANES

This was developed in late seventies for (Programmed LANguage-based Enquiry System) at the University of Illinois Coordinated Science Laboratory. PLANES include an English language front end with the ability to understand and explicitly answer user requests. It carries out clarifying dialogues with the user as well as answer vague or poorly defined questions. This work is being carried out using database based upon information of the U.S. Navy 3-M (Maintenance and Material Management), it is a database of aircraft maintenance and flight data, although the ideas can be directly applied to other non-hierarchic record-based databases [23].

PHILIQA

This was developed in 1977 and was known as Philips Question Answering System [24], uses a syntactic parser which runs as a separate pass from the semantic understanding passes. This system is mainly involved with problems of semantics and has three separate layers of semantic understanding. The layers are called "English Formal Language", "World Model Language", and "Data Base Language" and appear to correspond roughly to the "external", "conceptual", and "internal" views of data.

e. CHAT-80

The system CHAT-80 [26] is one of the most referenced NLP systems in the eighties. The system was implemented in Prolog. According to [25], the CHAT-80 was an impressive, efficient and sophisticated system. The database of CHAT-80 consists of facts (i. e. oceans, major seas, major rivers and major cities) about 150 of the countries world and a small set of English language vocabulary that are enough for querying the database. The CHAT-80 system processes an English language question in three stages as depicted.

f. TEAM

It was developed in 1987. A large part of the research of that time was devoted to portability issues. TEAM was designed to be easily configurable by database administrators with no knowledge of NLIDBs [27, 28].

g. ASK

This system developed in 1983, allowed end-users to teach the system new words and concepts at any point during the interaction. ASK was actually a complete information management system, providing its own built-in database and the ability to interact with multiple external databases, electronic mail programs and other computer applications. All the applications connected to ASK were accessible to the end-user through natural language requests. The user stated his/her requests in English and Ask transparently generated suitable requests to the appropriate underlying systems.

h. JANUS

It had similar abilities to interface to multiple underlying systems (databases, expert systems, graphics devices, etc). All the underlying systems could participate in the evaluation of a natural language request, without the user ever becoming aware of the heterogeneity of the overall system. JANUS is also one of the few systems to support temporal questions [29].

i. EUFID

The EUFID system consists of three major modules, not counting the DBMS. First is analyzer module, second is mapper module and third is translator module. [30]

j. DATALOG

It is an English database query system based on Cascaded ATN grammar. By providing separate representation schemes for linguistic knowledge, general world knowledge, and application domain knowledge, DATALOG achieves a high degree of portability and extendibility [31]. Systems that also appeared in mid-eighties were LDC [32], TQA [33], TELI [34] and many others.

8. Recent Developments in NLIDB

This section provides a brief overview of three specific NLIDB systems developed recently in different universities.

a. NALIX

NALIX (Natural Language Interface for an XML Database) is an NLIDB system de- veloped at the University of Michigan, Ann Arbor by Yunyao Li, Huahai Yang, and H. V. Jagadish (2006). The database used for this system is extensible markup language (XML) database with Schema-Free XQuery as the database query language.

Schema-Free XQuery is a query language designed mainly for retrieving information in XML. The idea is to use keyword search for databases. However, pure keyword search certainly cannot be applied. Therefore, some richer query mechanisms are added [36]. Given a collection of keywords, each keyword has several candidate XML elements to relate. All of these candidates are added to MQF (Meaningful Query Focus), which will automatically find all the relations between these elements. The main advantage of Schema-Free Xquery is that it is not necessary to map a query into the exact database schema, since it will automatically find all the relations given certain keywords.

NALIX can be classified as a syntax based system, since the transformation processes are done in three steps: generating a parse tree, validating the parse tree, and translating the parse tree to an XQuery expression. However, as implied in the paper [35][36], NALIX is different from the general syntax based approaches; in the way the system was built: NALIX implements a reversed-engineering technique by building the system from a query language toward the sentences.

b. PRECISE

PRECISE is a system developed at the University of Washington by Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates (2004). The target database is in the form of a relational database using SQL as the query language. It introduces the idea of semantically tractable sentences which are sentences that can be translated to a unique semantic interpretation by analyzing some lexicons and semantic constraints[37].

PRECISE was evaluated on two database domains. The first one is the ATIS domain, which consists of spoken questions about air travel, their written forms, and their correct translations in SQL query language. In ATIS domain, 95.8% of the questions were semanti- cally tractable. Using these questions gives PRECISE 94% precision. The second domain is the GEOQUERY domain. This domain contains information about U.S. Geography. 77.5% of the questions in GEOQUERY are semantically tractable. Using these questions gives PRECISE 100% accuracy.

The strength of PRECISE is based on the ability to match keywords in a sentence to the corresponding database structures. This process is done in two stages, first by narrowing the possibilities using Maxflow algorithm and second by analyzing the syntactic structure of a sentence. Therefore PRECISE is able to perform impressively in semantically tractable questions.

As other NLIDB systems, PRECISE has its own weaknesses. While it is able to achieve high accuracy in semantically tractable questions, the system compensates for the gain in accuracy at the cost of recall.Another problem is as PRECISE adopts a heuristic based approach, the system suffers from the problem of handling nested structures.

c. WASP

Word Alignment-based Semantic Parsing (WASP) is a system developed at the University of Texas, Austin by Yuk Wah Wong[38]. While the system is designed to address the broader goal of constructing ”a complete, formal, symbolic, meaningful representation of a natural language sentence” , it can also be applied to the NLIDB domain. A predicate logic (Prolog) was used as the formal query language.

WASP learns to build a semantic parser given a corpus a set of natural language sentences annotated with their correct formal query languages [38]. It requires no prior-knowledge of

the syntax, because the whole learning process is done using statistical machine translation techniques.

WASP was evaluated on the GEOQUERY domain, the same domain as PRECISE. GEO- QUERY corpus consists of 880 questions in the training set and 250 questions in the test set, which are merged together into one larger data set. Each data set was divided to 10 equal-sized subsets, and standard 10-fold cross validation was used to estimate the system performance. WASP achieved 86.14% precision and 75.00% recall in the GEOQUERY do- main. The system was also evaluated on a variety of other natural languages: English, Spanish, Japanese and Turkish. There were no significant differences observed between Eng- lish and Spanish, but the Japanese corpus has the lowest precision and the Turkish corpus has the lowest recall.

The strength of WASP comes from the ability to build a semantic parser from annotated corpora. This approach is beneficial because it uses statistical machine translation with minimal supervision. Therefore, the system does not have to manually develop a grammar in different domains. Moreover, while most of NLIDB systems use English as their natural language, WASP has been tested on several languages.

In spite of the strength, WASP also has two weaknesses. The first is: the system is based solely on the analysis of a sentence and its possible query translation, and the database part is therefore left untouched. There is a lot of information that can be extracted from a database, such as the lexical notation, the structure, and the relations within. Not using this knowledge prevents WASP to achieve better performances. The second problem is that the system requires a large amount of annotated corpora before it can be used, and building such corpora requires a large amount of work.

9. CONCLUSION

Research is done from the last few decades on Natural Language Interfaces. With the advancement in hardware processing power, many NLIDBs mentioned in historical background got promising results. Though several NLIDB systems have also been developed so far for commercial use but the use of NLIDB systems is not wide-spread and it is not a standard option for interfacing to a database. This lack of acceptance is mainly due to the large number of deficiencies in the NLIDB system in order to understand a natural language.

10. REFERENCES

[1]. Bertino, B. Catania, G.P. Zarri, "Intelligent database systems", Reading, Addsion Wesley Professional, 2001.

[2]. Kamran Parsaye, Mark Chignell, Setrag Khoshafian and Harry Wong, "Intelligent databases-object-oriented, deductive hypermedia technologies", New York, John Wiley& Sons, 1989.

[3]. Androutsopoulos, G.D. Ritchie, and P. Thanisch, Natural Language Interfaces to Databases - An Introduction, Journal of Natural Language Engineering 1 Part 1 (1995), 29–81

[4]. Charniak E. 1993, "Statistical Language Learning", *MIT Press*.

[5]. Church K., Mercer R. 1993, "Introduction to the special issue on computational linguistics using large corpora", *Computational Linguistics*,19 (1), pp. 1-24.

[6]. Miikkulainen R. 1993, "Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory", *MIT Press*, Cambridge, MA.

[7]. Marcus M., Santorini B., Marcinkiewicz M. 1993," Building a large annotated corpus of English: The Penn Treebank", Computational Linguistics, 19 (2), pp. 313-330.

[8]. McCarthy J, Lehnert W ,1995, "Using decision trees for coreference resolution", *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1050-1055.

[9]. Riloff E. 1993, "Automatically constructing a dictionary for information extraction tasks", *Proceedings of the Eleventh National Conference on Artificial Intelligence*,pp. 811-816.

[10]. Riloff E.1996, "Automatically generating extraction patterns from untagged text", *Proc eedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 1044-1049.

[11]. Majumder P., Mitra M., Chaudhari B.,2002, "N-gram: A Language Independent Approach to IR and Natural Language Processing", Lecture Notes.

[12]. Miikkulainen R., 1997,"Natural language processing with subsymbolic neural networks", *Neural Network Perspectives on Cognition and Adaptive Robotics.*

[13]. Reilly R., Sharkey N. (Eds.),1992 "Connectionist Approaches to Natural Language Processing", *Lawrence Erlbaum and Associates*, Hilldale, NJ.

[14]. Shashtri L.,1997, "A model of rapid memory formation in the hippocampal system", *Proceeding of Meeting of cognitive Science Society*, Stanford.

[15]. Abrahams P. W. et al. "The LISP 2 Programming Language and System", in proceedings of FJCC, No. 29, USA, 1966, pp. 661– 676.

[16]. J. McCarthy, "LISP Programmers Manual, Handwritten Draft" MIT AI Lab., Vambridge, USA, 1959.

[17]. T. Warren, "A Step toward Man-Computer Symbiosis", Ph.D. Thesis, Massachusetts Institut of Technologie, Project on Mathematics and Computation (MAC), Technical Report MAC-TR-32, Cambridge, MA, USA, 1966.

[18]. Rohit J. Kate and Raymond J. Mooney, Using String-Kernels for Learning Semantic Parsers, COLING-ACL (2006).

[19]. Woods, W. (1973). An experimental parsing system for transition network grammars. In Natural language Processing, R. Rustin, Ed.,Algorithmic Press, New York.

[20]. Woods, W., Kaplan, R. and Webber, B. (1972). The Lunar Sciences Natural Language Information System. Bolt Beranek and Newman Inc., Cambridge, Massachusetts Final Report. B. B. N. Report No 2378.

[21]. Hendrix, G. (1977). The LIFER manual A guide to building practical natural language interfaces. SRI Artificial Intelligence Center, Menlo Park, Calif. Tech. Note 138.

[22]. Hendrix, G., Sacrdoti, E., Sagalowicz, D. and Slocum, J. (1978). Developing a natural language interface to complex data. ACM Transactions on Database Systems, Volume 3, No. 2, USA, Pages 105 – 147

[23]. D.L. Waltz., "An English Language Question Answering System for a Large Relational Database",

Communications of the ACM, 21(7): (July 1978), pp 526– 539

[24]. R.J.H., Scha., "Philips Question Answering System PHILIQA1", In SIGART Newsletter, no.61. ACM, New York, (February 1977)

[25]. Amble, T. (2000). BusTUC – A Natural Language Bus Route Oracle. 6th Applied Natural Language Processing Conference, Seattle, Washington, USA

[26]. Warren, D., Pereira, F. (1982). An efficient and easily adaptable system for interpreting natural language queries in Computational Linguistics. Volume 8 pages 3 – 4.

[27]. B.J. Grosz, "TEAM: A Transportable Natural-Language Interface System", In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, (1983), pp 39–45

[28]. B.J. Grosz, D.E. Appelt, P.A. Martin, and F.C.N. Pereira, "TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces", Artificial Intelligence, 32: (1987), pp 173–243

[29]. P. Resnik, "Access to Multiple Underlying Systems in JANUS", BBN report 7142, Bolt Beranek and Newman Inc., Cambridge, Massachusetts, (September 1989)

[30]. M. Templeton and J. Burger, "Problems in Natural Language Interface to DBMS with Examples from EUFID", In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, (1983), pp 3–16

[31]. C.D. Hafner, "Interaction of Knowledge Sources in a Portable Natural Language Interface", In Proceedings of the 22nd Annual Meeting of ACL, Stanford, California, (1984) pp 57–60

[32]. B.W. Ballard, J.C. Lusth, and N.L. Tinkham, "LDC-1: A Transportable, Knowledge based Natural Language Processor for Office Environments", ACM Transactions on Office Information Systems, 2(1): (January 1984), pp 1–25

[33]. F. Damerau, "Operating statistics for the transformational question answering system", American Journal of Computational Linguistics, 7: (1981), pp 30–42

[34]. B. Ballard and D. Stumberger, "Semantic Acquisition in TELI", In Proceedings of the 24th Annual Meeting of ACL, New York, (1986), pp 20–29

[35]. Yunyao Li, Huahai Yang, and H.V. Jagadish, Nalix:an Interactive Natural Language Interface for Querying XML, SIGMOD (2005).

[36]. Yunyao Li, Huahai Yang, and H.V. Jagadish, Constructing a Generic Natural Language Interface for an XML Database, EDBT (2006).

[37]. Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates, Modern Natural Language Interfaces to Databases:Composing Statistical Parsing with Semantic Tractability, COLING (2004).

[38]. Yuk Wah Wong, Learning for Semantic Parsing Using Statistical Machine TranslationTechniques, Technical Report UT-AI-05-323, University of Texas at Austin, Artificial Intelligence Lab, October 2005.