A Survey on Natural Language Interface to Database

Jiffy Joseph
Kottuppally, Thuravoor P O, Cherthala, Alappuzha – 688532, kjjiffy@gmail.com

Abstract: Natural language processing (NLP) is a field of computational linguistics. NLP deals with analyzing, and understanding the human languages that are used to interface with computers via natural languages instead of using computer languages. In this era databases are used in many fields like banking, human resources, universities etc. That is people around the world need to deal with databases for the extraction of required data. But it is very difficult for a common user to retrieve information from the database using database query languages. Access any kind of data from database using natural language like English is a convenient and easy method instead of using formal query languages such as XQuery, SQL etc. The system which provides such a facility is known as Natural Language Interface to Databases. Some of the works in this area are discussed here.

Keywords: Natural Language Interface to Databases, Natural Language Processing, XQuery.

I. INTRODUCTION

The idea of using natural language instead of formal query languages for retrieving information in a database is a latest application of NLP called Natural Language Interface to DataBase(NLIDB). NLIDB is a step towards the development of Intelligent DataBase Systems (IDBS) to enhance the users in performing flexible querying in databases. The use of Natural Language Interfaces to Databases (NLIDB) is to allow the users to compose questions in natural language and receive responses. Like other systems NLIDB systems are also have some advantages and disadvantages [1]. The advantages and disadvantages of an NLIDB system are listed below.

Advantages

- User is not required to learn any artificial communication language.
- Comparatively good for the queries those involve negation or quantification.
- Easy to use for multiple database tables.

Disadvantages

- It is difficult to understand what kind of questions the NLIDB can or cannot cope with.
- Linguistic coverage is not obvious.
- It is often not clear whether a rejected question is not in the linguistic coverage of the system, or whether it is outside the system's conceptual coverage.
- Users assume intelligence.
- NLIDBs usually require some lengthy configuration phases before they can be used.

A. Components of NLIDB

The operations performed in a NLIDB system can be considered as two processing stages; Linguistic processing and Database processing [2]. In the first stage, the components of natural language query (NLQ) is mapped and translated into the corresponding database query fragments. In the database processing stage, database management and access is performed and the query is executed by the system.

1) Linguistic Component: This component handles the natural language input, convert it to the database query language and generate a natural language output as a result after execution. A lexicon comprise number of tables to create a formal query that store natural language words and their mapping to the objects of database language that is used in the system. These tables can have entries of noun, table name, column names, verbs, adverbs, modifiers etc.

2) Database Component: Database component performs database management functions. After the formation of a formal query from the input natural language sentence, that query is executed and the result in natural language is given to the user.

II. PREVIOUS WORKS

A. NaLIX: A Generic Natural Language Search Environment for XML Data

Yunyao Li, Huahai Yang and H. V. Jagadish [3] proposed a framework for building a natural language interface to XML data. The relationships between words in the NLQ are obtained by analyzing parsed tokens from the MINIPAR dependency parser, which is based on the relationship between words rather than hierarchical constituents. The steps involved in converting natural language queries into XQuery expressions are: identify and classify terms in a parse tree output of the MINIPAR parser. This parse tree is then validated, and the validated parse tree is converted into an XQuery expression. Steps involved in NaLIX are given below:

Parsing: A dependency parser called MINIPAR is used to parse the input NLQ.

Token Classification:- Words in the input query are classified into two categories called markers and tokens.

Tokens: Words or phrases in the original sentence that can be mapped into corresponding components of XQuery.

Markers: Words or phrases in the original sentence that does not match any component of XQuery.

Mapping the tokens in the NLQ to XQuery components:- The tokens in the input NLQ is mapped to corresponding XQuery components.

Determining grouping and nesting for aggregate functions and quantifiers:- If the NLQ contains any quantifier token or function token, corresponding to a quantifier or aggregate function respectively in the XQuery, then the system identifies the nesting and grouping of the XQuery components.

After the determination of grouping and nesting, of quantifiers and aggregate functions, the XQuery fragments are joined together to construct the correct XQuery corresponding to the NLQ.

B. NLKBIDB: Natural Language and Keyword Based Interface to Database

Axita Shah, Dr. Jyoti Pareek, Hemal Patel and Namrata Panchal [4] proposed a combined approach of NLIDB and knowledge based interface to database(KBIDB). NLIDB provides accuracy for generating a SQL by using the SQL generation rule based on Natural language concept. And KBIDB handles syntactically correct and incorrect query both but SQL generation rules of KBIDB is less strong than NLIDB. NLKBIDB applies the generation rules of NLI for syntactically correct query and for incorrect query it applies rules of KBI for SQL conversion.

NLKBIDB tokenizes the NLQ by lexical analyzer and lexicons are parsed by the syntax analyzer. If input query is syntactically valid, lexicons will be analyzed by the semantic analyzer and then SQL Generator will be used to generate SQL. But if Syntax analyzer gets failed to validate the Query then keyword based agent will convert it into SQL by using rule based knowledge.

C. Constructing an Interactive Natural Language Interface for Relational Databases

The architecture proposed by Fei Li and H. V. Jagadish [5] is known as NaLIR. It comprising three functional parts: a first component that transforms a NLQ to a query tree, a second component that verifies the transformation by communicating with the user, and a third component that translates the query tree into a SQL statement. The query interpretation part, which includes parse tree node mapper and structure adjustor, is responsible for interpreting the NLQ and representing the interpretation as a query tree. An interactive communicator is used to communicate with the user to ensure that the interpretation process is correct. The query tree, verified by the user, is then translated into a SQL statement in the query tree translator and then evaluated against an RDBMS.

Parsing: Stanford Parser is used here to generate a linguistic parse tree from the NLQ.

Parse Tree Node Mapper: The parse tree node mapper identifies the nodes in the parse tree that can be mapped to SQL components.

Parse Tree Structure Adjustor: After successful completion of node mapping, structure adjustor adjusts the structure of the parse tree and generate candidate interpretations. Query Tree Translator: In the cases when the query tree does not contain function nodes or quantifier nodes, which means the target SQL query will not have aggregate functions or sub queries, the translation is quite straightforward. In the case when the query tree contains either function node or quantifier node, it implies that the target SQL statements will contain blocks. Given a query tree comprising multiple blocks, NaLIR translate one block at a time, starting from the innermost block, so that any correlated variables and other context is already set when outer blocks are processed.

D. Using Natural Language Processing in Order to Create SQL Queries

In the system proposed by F.Siasar djahantighi, M.Norouzifard, S.H.Davarpanah and M.H.Shenassa [6] they introduced a method which prepares an expert system that can identify synonymous words in any language. It first parses the input sentences, and then the natural language expressions are transformed to SQL language. The functional parts of this system are as:

Tokenizing, and Syntax Parsing: Restoring data in natural language processing, by syntactic viewpoint. It includes shallow parsing, syntactic parsing and extraction of lexical cohesion of the NLQ.

Semantic Knowledge: NLIDB needs a preprocessor to realize the changing of the words in input query. This preprocessor creates a semantic database from different kinds of expository rules of the language and semantic sets for all entities and all their possible attributes. This database helps to present an equal query for more than one sentence with the same meaning.

Expert System: It explores the knowledge and finally decides definitely or indefinitely, it includes two parts: knowledge base, and inference engine.

Semantic Analysis: Mapping of the NLQ query to SQL query is done by matching the input with some particular patterns. Some patterns are given below:

<attribute> of <object>
<attribute> of <object> of <object>
<actionverb> object <attributevalue>

E. DaNaLIX: a Domain-adaptive Natural Language Interface for Querying XML

Yunyao Li, Ishan Chaudhuri, Huahai Yang, Satinder Singh and H. V. Jagadish [7] presented a prototype domain-adaptive natural language interface for querying XML data. The translation from an English query into an XQuery expression involves four main steps.

Parse Tree Tokenization: Tokenize the input query and identify words and phrases that can be directly mapped into XQuery components.

Domain Knowledge Incorporation: The domain adapter checks existing domain knowledge and transforms the parse tree by applying any relevant rules. Domain knowledge is represented as a set of rules representing the mapping between a partial parse tree containing terms with domain meanings with one expressed in terms understandable by a generic system like NaLIX.

Parse Tree Validation: A parse tree may contain terms that cannot be understood by the system, even after transformed using domain knowledge known to the system. The validation phase checks whether the parse tree is one that we know how to map into XQuery. It also checks whether the element/attribute names and/or values contained in the user query can be found in the database. If an invalid parse tree is found, information about the errors will be sent to the message generator to generate appropriate error message.

Parse Tree Translation: Finally, the translator will utilize the structure of the natural language constructions as reflected in the parse tree to produce an XQuery expression.

F. Semantic Mapping between Natural Language Questions and SQL Queries via Syntactic Pairing

Alessandra Giordani and Alessandro Moschitti [8] presented a method to automatically map NLQ to SQL by carrying out mapping at syntactic level and then applying machine learning algorithms to derive an automatic translator of NLQs into their associated SQL queries. For this purpose, they designed a dataset of relational pairs containing syntactic trees of questions and queries and encoded them in Support Vector Machines by means of kernel functions.

The dataset consists of natural language questions N and SQL queries S related to a specific domain and automatically learn such mapping from the set of pairs P = NXS. The steps included are (a)assume that pairs are annotated as correct when the SQL query answers to the question and incorrect otherwise, (b)train a classifier on the above pairs for selecting the correct queries for a question, (c)rank the latter by means of the question classifier score and by selecting the top one. The problem of assigning a query to a question, is formally described as a ranking problem: (i) given a question n ϵ N and a set of possible useful queries S, generate the set of possible pairs P(n) = {<n, s> : s ϵ S} (ii) classify them with an automatic categorizer; (iii) use the score/probability output by such model to rank P(n); (vi) select the top ranked pairs.

G. The Study on Natural Language Interface of Relational Databases

Xu Yiqiu, Wang Liwei and Yan Shi [9] proposed a design framework based on Ontology. It uses WordNet as basic lexicon, and defines domain lexicon beyond that. The language processing module and database processing module are integrated by intermediate representation language called Discourse Representation Structure (DRS). NLQ sentences are translated into DRS query sentences, and then translated into SQL query. The parser used in this architecture is Stanford parser, universal lexicon used is WordNet. The system expands domain lexicon based on WordNet. Domain lexicon follows WordNet structure and use WordNet's synonymy and antonym relations to connect with WordNet. Semantic interpreter converts syntax tree into DRS. DRS, as an intermediate language of semantic interpreter, can be converted into other format logic.

DRS Structure: DRS represent a logic atom by using tabular label structure. For example, Book A in library management system is an entity, represented by book (A). But here the developers used table(A, book) to represent DRS structure, so that they can use less defined predication structure to express more predication in tabular label structure, and reduce the amount of predication. Semantic

Interpreting Algorithm: In order to get correct result, NLQ sentence inputted by users should include three kinds of information: the table name, query field and query conditions. Steps involved in interpretation algorithm are:

1. Initialize system, consult initialization file nlidb.pl. This file invokes domain lexicon file and Ontology knowledge base file.
2. Input token, conjunction sub-sentence and type of query, which is generated by syntax analysis.
3. Match table form Ontology knowledge base.
4. Match querycolumn from Ontology knowledge base.
5. Matching wherecolumn in token left from Ontology knowledge base.
6. Matching whererelation according to conjuction and or or.
7. Convert the predication generated into DRS

This work uses Velocity to implement mapping from DRS to SQL. DRS expression data and semantic information can be mapped to variables in template files, and then generate SQL sentences.

H. DBIQS - An Intelligent System for Querying and Mining Databases using NLP

Rohit Agrawal, Amogh Chakkarwar, Prateek Choudhary, Usha A. Jogalekar and Deepa H. Kulkarni [10] proposed a Database Intelligent Querying System (DBIQS). The overall system architecture can be divided into three stages; Semantic Building Stage, MR Generation Stage and Query Generation Stage.

Semantic Building: The semantic builder works on the semantic layer and processes and extracts the information from the databases to build up the semantic map. The semantic map is created only once for any database and it consists of lexicon, information regarding the tables and their relationships.

MR Generation: Translation of natural language queries to some intermediate form called Meaningful Representation (MR) is performed in this stage. Meaningful Representation (MR) comprises of one or more tokens and relationships between them. Tokens are divided into three categories; reserved keywords, attributes and values. A tokenizer is used to tokenize the given question. In order for any sentence to be interpreted at least one complete tokenization must map precisely to one set of database elements. A mapping is considered as valid only if there is a satisfactory map of complete sentence tokenization to the set of database elements. If the sentence tokenization contains only distinct tokens and at least one of its value tokens matches a wh-value, then they assumes that the following sentence is semantically tractable. The set of tokens are then passed to the parser which helps in extracting the attachment relationships between tokens from the parse tree.

Query Generator: A query generator generates final SQL queries by mapping MR to semantic information. While mapping, the synonyms are identified using Wordnet. After mapping, query generator makes use of a predefined engine containing reserved keywords or phrases and the actions associated with them to build the final SQL query.

I. D-HIRD: Domain-Independent Hindi Language Interface to Relational Database

Rajender Kumal, Mohit Dul and Shivani Jinda [11] proposed an architecture that accepts a NLQ in Hindi and then it generates corresponding SQL query, and returns the result by extracting the data from corresponding databases. Different phases of D-HIRD are as given below:

Query Analyzer: This phase analyze the input Hindi query and divide it into simpler elements called tokens.

POS Tagger: POS Tagger uses the grammatical category to tag the token of the input query. Annotated corpora are used by the POS tagger for finding the lexical category.

Morphological Analyzer: This phase analyzes grammatical features of the word. In this phase, take words and try to identify the suffix or prefix and check whether this suffix is valid or not. If this suffix is present in the word-list of annotated corpora then it is valid otherwise it is unknown word for morphological analyzer. If the suffix is valid then the system compares the stem to the root word stored in annotated corpora. If it is not found in annotated corpora, then it is declared as unknown word by morphological analyzer. If the suffix and root words are valid then take the line number of the word in word-list and get the feature structure. After that, root word and feature structure are passed to the next phase.

Semantic analyzer: Semantic analyzer uses the semantic grammar, database metadata and lexicon to determine the semantic meaning of the word.

Domain Identifier: Domain identifier uses the domain oriented knowledge base for deciding a particular domain. Domain oriented knowledge base stores the schema of relations with their name. It also stores the domain name with all their relation. Knowledge expert is responsible for generating domain knowledge.

SQL Query Generator: SQL query generator uses the mapping rules which are defined to map the NLQ tokens to the elements of SQL. Three routines are taken for constructing the SQL query. First routine determine the command of the query. Second routine determine the attributes and relation name for the query on the basis of command. Third routine determine the condition part of the query. Final SQL query is generated by concatenating all of these routines.

SQL Executor: SQL executor executes the SQL query on specific domain's database and generates the result.

III. COMPARATIVE STUDY ON DIFFERENT NLIDB SYSTEMS

The previous works can be classified into four categories based on the techniques used. The categories are systems that use pattern matching system, syntax-based systems, semantic grammar systems, and the systems that uses intermediate representation languages.

Pattern Matching Systems: The query construction is done by matching the input sentence with a given pattern. This system is easy to implement. But it is not an efficient technique.

Syntax-Based Systems: The user input is parsed first. The parse tree delivers detailed information about the structure of a sentence. The resulting parse tree is directly depicted to an expression in database query language. This technique can handle more types of NLQs than pattern matching technique. But the problem with such systems are: it is not always clear that which nodes should be mapped and which should not, the NLQ with multiple parse trees may results multiple formal queries, and it is difficult to map a parse tree directly into some formal database language.

Semantic Grammar Systems: The basic design of a semantic grammar system is to make simpler the parse tree as much as possible, by removing unneeded nodes or combining some of the nodes at the same time. Such a system is more accurate than the syntax-based systems. The main issue of semantic grammar approach is that it needs some prior knowledge of the elements in the domain, therefore making it hard to port to other areas.

Intermediate Representation System: The idea is to translating a sentence into a intermediate representation language first and then further converts this intermediate representation into a general database query. In this process there can be more than one intermediate meaning representation language. The advantage of this system is separating the language processing module from the database processing module ensures system's portability. But the time taken for converting NLQ to database query is slightly greater since it contains two translation steps.

Table 1: Comparative Study on Different NLIDB

Paper	*Technique Used*
NaLIX: A Generic Natural Language Search Environment for XML Data	Semantic grammar system
NLKBIDB: Natural Language and Keyword Based Interface to Database	Syntax-based and pattern matching system
Constructing an Interactive Natural Language Interface for Relational Databases	Semantic grammar system
Using Natural Language Processing in Order to Create SQL Queries	Syntax-based and pattern matching system
DaNaLIX: a Domain-adaptive Natural Language Interface for Querying XML	Semantic grammar system
Semantic Mapping Between Natural Language Questions and SQL Queries via Syntactic Pairing	Syntax-based system
The Study on Natural Language Interface of Relational Databases	Intermediate representation system
DBIQS - An Intelligent System for Querying and Mining Databases using NLP	Intermediate representation system
D-HIRD: Domain-Independent Hindi Language Interface to Relational Database	Syntax-based system

IV. SUMMARY AND CONCLUSION

Natural language interface to databases (NLIDBs) are the systems which helps a user to interact with databases without any knowledge about the database querying languages. The different types of NLIDB systems are as explained in the previous section. Some of them provide interfaces to XML database and the others provide interface to SQL databases. The most noticeable advantage of NLIDB system is, it is user friendly. End user accepts this kind of system only if it is easy to use. Though several NLIDB systems have also been developed so far for commercial use but the use of NLIDB systems is not wide-spread because the linguistic coverage of those systems is less.

REFERENCES

[1] Yohan Chandra, "Natural Language Interfaces to Databases", http : //digital:library:unt:edu/ark : /67531/metadc5474/m2/1/high_res_d/thesis:pdf, August 2015

[2] Neelu Nihalani, Sanjay Silakari, Dr. Mahesh Motwani, "Natural language Interface for Database: A Brief review", http : //ijcsi:org/papers/IJCSI - 8 - 2 - 600 – 608.pdf, August 2015

[3] Yunyao Li, Huahai Yang, H. V. Jagadish, "NaLIX: A Generic Natural Language Search Environment for XML Data", ACM Transactions on Database Systems (TODS), Vol. 32, Issue 4, November 2007.

[4] Axita Shah, Dr. Jyoti Pareek, Hemal Patel, Namrata Panchal, "NLKBIDB: Natural Language and Keyword Based Interface to Database", International Conference on Advances in Computing, Communications and Informatics (ICACCI), Pages 1569 - 1576, August 2013.

[5] Fei Li, H. V. Jagadish, "Constructing an Interactive Natural Language Interface for Relational Databases", Proceedings of the VLDB Endowment, Vol. 8, Issue 1, Pages 73-84 , September 2014.

[6] F.Siasar djahantighi, M.Norouzifard, S.H.Davarpanah, M.H.Shenassa, “Using Natural Language Processing in Order to Create SQL Queries”, International Conference on Computer and Communication Engineering, ICCCE, Pages 600 - 604, May 2008.
[7] Yunyao Li, Ishan Chaudhuri, Huahai Yang, Satinder Singh, H. V. Jagadish, “DaNaLIX: a Domain-adaptive Natural Language Interface for Querying XML”, SIGMOD ’07 Proceedings of the 2007 ACM SIGMOD international conference on Management of data, Pages 1165-1168, June 2007.
[8] Alessandra Giordani, Alessandro Moschitti, “Semantic Mapping Between Natural Language Questions and SQL Queries via Syntactic Pairing”, 14th International Conference on Applications of Natural Language to Information Systems, Vol. 5723, Pages 207-221, June 2009.
[9] Xu Yiqiu, Wang Liwei, Yan Shi, “The Study on Natural Language Interface of Relational Databases”, International Conference on Environmental Science and Information Application Technology (ESIAT), Vol. 2, Page 596 - 599, July 2010.
[10] Rohit Agrawal, Amogh Chakkarwar, Prateek Choudhary, Usha A. Jogalekar, and Deepa H. Kulkarni, “DBIQS - An Intelligent System for Querying and Mining Databases using NLP”, International Conference on Information Systems and Computer Networks(ISCON), Page 39 - 44, March 2014.
[11] Rajender Kumal, Mohit Dul, Shivani Jinda,“D-HIRD: Domain- Independent Hindi Language Interface to Relational Database” , International Conference on Computation of Power, Energy, Information, and Communication(ICCPEIC), Page 81 - 86, April 2014.

Jiffy Joseph is a post graduate in Computer Science and Engineering. She obtained her M. Tech. (Specialization: Computer and Information Science) degree from College of Engineering Cherthala, Cochin University of Science and Technology (CUSAT), Kerala, India. She has authored a conference paper in the field of Natural Language Processing. Her research interests are Natural Language Processing and Image Processing.