An Intelligent Database System using Natural Language Processing

HESSA ALAWWAD
Department of Computer Science
Al Imam Mohammad Ibn Saud Islamic University
P.O. Box No. 5701, Riyadh-11432
SAUDI ARABIA
hessaalawwad@ccis.imamu.edu.sa

EMDAD KHAN
Department of Computer Science
Maharishi University of Management (MUM)
1000 North 4th Street, Fairfield, Iowa 52557
U.S.A
ekhan@mum.edu

Abstract: - – Enormous amount of data are being processed and exchanged in our daily life, and database, which is used to organize data has been an active research topic for a long time. Database plays a major role in many computer systems and there is always a demand from technical and nontechnical people to ease the process of accessing data on database. Using Natural Language to directly interact with a database is a nice and user friendly solution. In order to achieve this type of communication between the computer (In particular, database) and human we have to make the computer understand what the human asks, and then, be able to respond with the right answer that was expected to be extracted from the database.

In this paper we present an intelligent system for converting Natural Language queries into equivalent database Structured Query Language (SQL). Our system also allows processing complex Natural Language queries. We call this Intelligent Agent based Natural Language Interface to Database (INLIDB). The query results from the INLIDB is presented in an attractive succinctly viewable format. We have obtained encouraging results from INLIDB.

Key-Words: - Natural Language Processing / Understanding, Semantic Parsing, Syntactic Parsing, Intelligent Database, Structured Query Language, Artificial Intelligence.

1 Introduction

Natural Language Processing (NLP) has many applications that require either Natural Language understanding or Natural Language generation or even both - for example Machine Translation, which focuses on translating text from one human language to another automatically. Another good example of NLP is Information Extraction which is concerned with "factual information in free text [1]". This paper is concerned with one of the important applications of NLP, which is Natural Language Interface to Database (NLIDB). Figure 1 from [2] shows the components of Natural Language Interface to Database. The idea of NLIDB systems came from the method of questioning the database that uses Natural Language queries like English instead of database language to query the database.

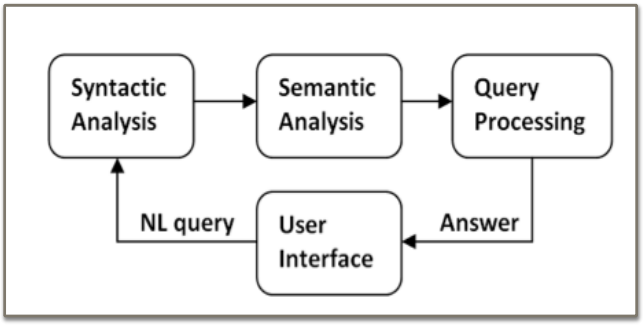
This paper is concerned with one of the important applications of NLP, which is Natural Language Interface to Database. The importance of NLIDB system is that it makes it easy for users with no programming experience to deal with a database. User can just use Natural Language to interact with a database which is very simple and easy. Also, the user would not need a special training on using such systems (maybe some training to know the interface). The user is not forced to learn any database language. It is hard to learn a formal query language like SQL by a naive or inexperienced user. Also, it is easier to use Natural Language in queries that involve multiple database tables.

The main objective of this paper is to make the use of database much simpler where the users can retrieve information using natural sentences by means of implementing an intelligent agent that can understand the user's query and can generate the answer in a nice presentable way.

2 Related Work

NLIDB systems have started in the early 1980s or precisely in the 1970s. Since then, researchers have been very interested in developing these systems, and have always tried to find better solutions for the commercial applications.

The main reason we need such systems is that



they offer a great substitution for people who deal

Figure 1: Components of NLIDB System.

with Database on a daily or periodic basis where they are not required to learn the process of retrieving data
using a database language like SQL. Rather, they only require their language that they use every day to communicate with people around them.

In general, the existing methods to interact with a database using NLP can be divided into three categories: (1) Pattern Matching Models or Template Based Approach. (2) Syntactic Models. and (3) Syntactic and Semantic Models.

In the first model, the entered query is processed by matching it with a predefined set of rules or patterns. The next step then translates it into a logical form according to what pattern it belongs to. From these rules, the database query is directly formulated.
In the second model, a constituent syntactic tree using a syntactic parser is used [3] where the leaves are used in the process of mapping to a database query based on predefined syntactic grammars. In the third model, the use of semantics adds the intelligence concept, and the query is processed according to what it means.

Existing NLIDB systems vary from each other in the method they use to convert the query into a database language such as SQL. Generally, there are four steps to do the conversion: Lexical Analysis, Syntactic / Semantic Analysis, Query Generation and Answer extraction.

LUNAR [4] is one of the most known syntax-based systems. It came in early seventies (1973). It is an English question answering system that answers questions about the chemical analyzes of the Apollo 11 moon rocks. It has three main parts: the first one is a general purpose grammar and a parser that covers a large subset of the language. The second part is a semantic analyzer to get the meaning from the question. The third part is database retrieval and inference component to store the data to be manipulated. The domain of the system is restricted to lunar geology and chemistry only. The main challenge they faced that there were only a few general-purpose NLP resources available at the time.

The research continued for another decade, where they focused on the syntactic parsing, incorporating domain knowledge, dialog systems and semantic parsing like in LADDER system [5].

3 Contribution

The existing NLIDB systems cover different areas of linguistic and semantic parsing. Authors of existing NLIDB systems proposed different algorithms to handle a certain level of complexity, but the existing systems do not cover complex natural language queries with complex semantics.

We have designed and built an Intelligent NLIDB (INLIDB) that converts the queries from the Natural Language form to its equivalent Structured Query Language form. It starts with the Syntactic analysis performed by Stanford POS tagger. Then, The keyword extractor use the information from the POS tagger to extract the keywords that are used by the Named Entity Recognition tool. The Named entity Recognizer defines the related domain concepts like person or department. The identified keywords are handled by a SQL Generator class. For the complex queries, we propose an algorithm that extracts the main keyword along with its characteristics to be used in further processing.

Section 4 discusses the System Architecture and design decisions we made during design and implementation of our intelligent agent. Section 5 describes how our NLIDB has been tested. It also shows and discusses the results we have achieved.

4 System Architecture

Our architecture has two major parts: a syntactic parser and a semantic parser (Fig. 2).

These parsers help extract the key features that affect the whole process of transforming the natural queries into its equivalent SQL statements. We also show an algorithm to handle a higher level of semantic complexity for Natural Language queries.

4.1 Syntactic Analysis

The Syntactic model in general presents linguistic information based on tokenizers, morph analyzers, part-of-speech tagging (POS). There are eight parts of speech in the English grammar: verbs, nouns,

pronouns, adjectives, adverbs, prepositions, conjunctions and interjections [6].

Stanford Log-linear Part-of-Speech Tagger [7], which was used in this work, is a software that reads an input (text) and assigns part of speech (lexical

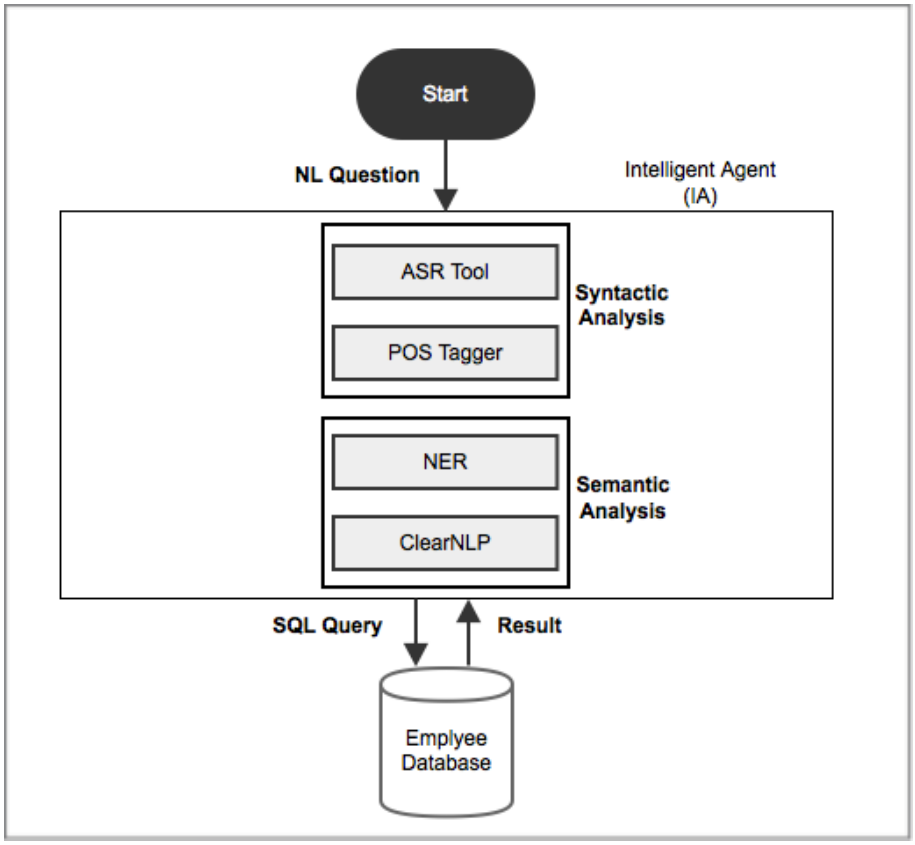


Fig. 2: System Architecture

category) to each word. It was presented in 2003 by members of the computer science department in Stanford University. The last release was in October 2014.

If we send the following sentence: "What is the salary of Ahmad?" to the POS tagger, we get the response in Figure 3: "What_WP is_VBZ the_DT salary_NN of_IN Ahmad_NNP ?_. " with the tags WP (Wh-pronoun), VBZ (Verb, 3rd person singular present), DT (Determiner), NN (Noun, singular or

WP	VBZ	DT	NN	IN	NNP
What	is	the	salary	of	Ahmad ?

plural), IN (Preposition or subordinating conjunction) and NNP (Verb, non-3rd person singular present) assigned to the query words.

The syntactic analyzer tags the tokens of the sentence that are returned from the Token Analyzer.

Figure 3: Part of speech tagging for the sentence: "What is the salary of Ahmad?"

The main reason we used the part-of-speech tagger is to identify the parts that can be identified as our keyword from the query. These keywords are going to be passed to the next level of processing which is concerned with its semantic side.

We notice that the main words (nodes) of the queries are: Nouns, Adjectives and Numbers. Other words were discarded as they do not affect the conversion process.

4.2 Semantic Analysis

The POS is not enough by itself to convert the Natural Language query into SQL, so we need to add
more information that we can use to understand the query. For this, we have used Stanford Named Entity Recognizer (NER) in order to assign the

keywords we already extracted from the query to the pre-defined category it belongs to.

So, along with the POS tagger we used Stanford NER to add some meaning to the category or entity the keyword belongs to. Hence, the names that would appear in the query would be recognized as PERSON which according to our database means that the person is an employee. For our department names we had to train our own named entity recognition model to recognize the departments as Organizations.

For example, if we send the sentence "What is the salary of Ahmad who works in Programming Department?" to be analyzed by the named entity recognizer, we get the output as shown in figure 4. The employee name "Ahmad" has been recognized as Person. Then our module understands that this Person is an employee according to our database schema. This understanding gives us the ability to construct the SQL statement.

Figure 5 illustrates the process of understanding the syntax of the query by using Stanford POS & NER.

The class QueryDefiner does the first step in constructing the SQL statement. It determines the SQL statement type by defining a list of synonym we expect the user to use of each type: SELECT, DELETE, INSERT. Like: 'give me', 'show me' and 'what' for SELECT.

The second and third steps are to extract the keywords. After discarding any tokens that do not affect the transformation process to SQL statement the token is added to the proper clause of the SQL statement according to a set of rules that have been defined to process the structure of the query.

```
what/O
is/O
the/O
salary/O
of/O
ahmad/PERSON
who/O
works/O
in/O
programming/ORGANIZATION
department/O
?/O
```

Figure 4: NER output for the sentence "What is the salary of Ahmad who works in Programming Department?"

4.2.1 Semantic Role Labeling

There were some situations where the Stanford POS and NER failed to understand the user's query and thus was not able to convert it to SQL.

So we needed a Semantic role labeler which presents the semantic relation between the predicates (verbs) and arguments with labeled arcs.

In semantic parsing we perform a dependency parsing to derive a syntactic dependency structure where every token (word) except the root have a link (dependency) to a head token. Then we apply a Semantic role labeling where each predicate has a semantic information.

1. Determine the desired action from the input query (INSERT, DELETE or SELECT).
2. Remove any filler words to extract the keywords.
3. For each keyword:

Check whether it is: Column\Table Name or a synonym of one of them. If not:

Check whether it is: Operator, Person, Organization or a WHERE Clause condition.

4. Whether it was A or B. Add the keyword after processing it to the proper clause of the SQL statement.
5. Re-arrange the SQL string if necessary.

Figure 5: The algorithm used to understand the syntax and semantic of the query by using Stanford POS & NER.

We used ClearNLP [8]. which is an open source project developed at Emory University and has been used in research like in [9].

In general, argument goes through two tasks, argument identification which is the task of finding the argument of each predicate and argument classification where each argument is assigned with a semantic role with respect to the predicate [8].

Figure 6 shows the flow of our framework. It starts with the Syntactic analysis performed by Stanford POS tagger. Then, the keyword extractor use the information from the POS tagger to extract the keywords that are used by the NER. The Named entity Recognizer defines the related domain concepts like person or department.

In complex queries we go through a dependency semantic parsing. Then we move to the nodes mapping which maps each node in the keywords into the corresponding SQL statement component. The SQL statement is executed against our relational database. The retrieved response is

then handled by the answer generation class to form the answer to be presented to the user.

5 Results

We have thoroughly tested our system and we used an input data that consists of hand crafted questions specifically designed to verify that the Intelligent Agent, IA can transform all the covered construction rules (that have been defined) into correct SQL statement. Also, wav files recorded from different users have been used to test speech recognition tool.

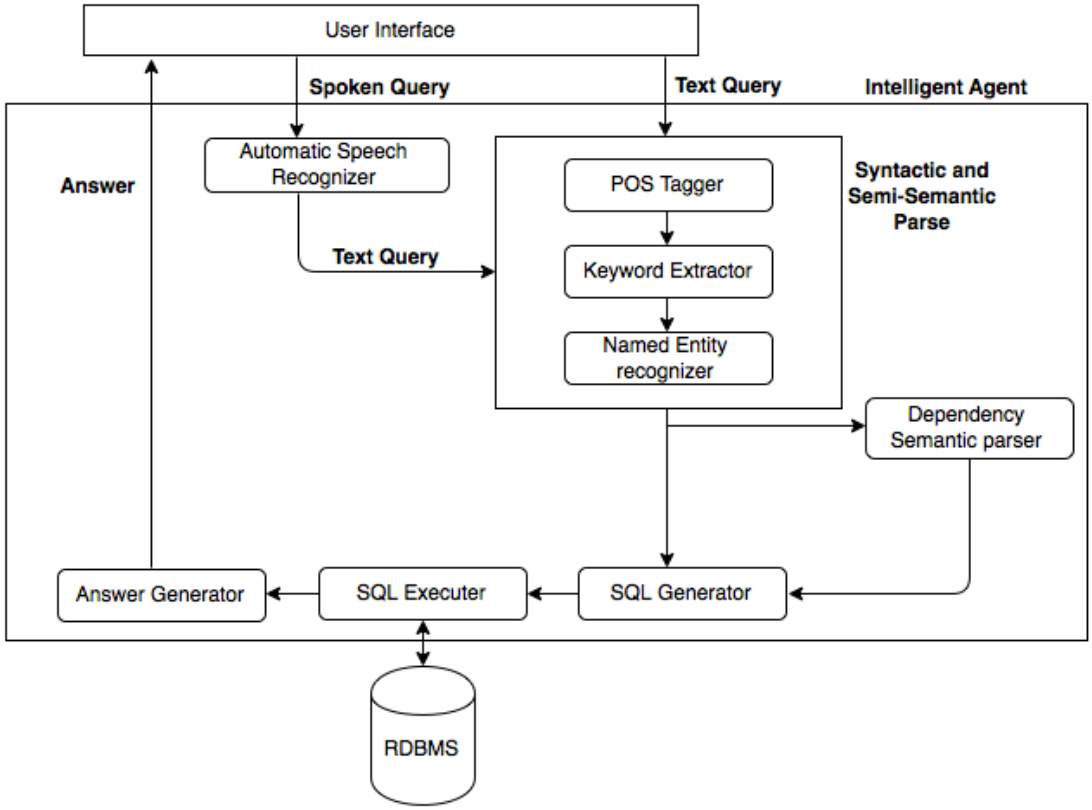


Figure 6: Flow of the framework

We used Sphinx4 [10] as the speech recognition tool, MySQL as the RDBMS, the Stanford Natural Language Part-of-Speech Tagger [7] as the syntax parser, the Stanford Named Entity Recognizer [11] as the semantic parser and ClearNLP [12] for parsing complex queries. We have implemented the IA (to implement our algorithm and integrate all tools mentioned above) in Java.

The goal of this work is to ease the process of querying relational database for non-technical users. As such, we used the measurement in [13] where there are two aspects to be evaluated (with some modification to suite our framework):

- The quality of the returned SQL statement from written Natural Language queries (effectiveness and correctness).
- Whether our module was easy to be used by non-technical users with spoken queries (usability).

5.1 Effectiveness

Number of success	Number of failure	percentage of the success
57	13	81.43%

The effectiveness of our system is evaluated as the percentage of the queries that were successfully translated into SQL statements by our IA.

The test suite contained 70 questions about the database. It has been evaluated by comparing the correct SQL statement that corresponds to the user query with the SQL statement we got from our IA. The percentage of the success of our IA to transform the queries in the test suite was 81.43 % (Figure 7).
Figure 7: The results of the Effectiveness measure of the INLIDB.

5.2 Usability

The usability in our test is measured by

(a) How easy it is to say (or type) the query naturally and

(b) The simplicity and accuracy of the results obtained from IA.

Users were well satisfied with the results for all the correctly generated SQL statements. They have used 20 different queries using their natural language. When they used spoken sentences (not typed), the accuracy of correctly generated SQL statements suffered to some extent due to the ASR recognition accuracy which was about 80%. This, however, can be improved by using more accurate ASR tool and improved semantics in NLP. Improved semantics can also improve the overall accuracy shown in Figure 7, especially for complex sentences.

6 Conclusion and Future Work

Adding the Natural Language processing capabilities to a database by using an Intelligent Agent (INLIDB), enhances the ease of use by the users with no programming background to query the database with their native language.

One challenge in the NLIDB evolution is not having good ability to overcome the Natural Language problems like the semantics, ambiguity, and universe of discourse which make the transformation process difficult.

The architecture of our system has five parts - first, the syntactic parser where we have used Stanford Part-of-Speech tagger to understand the syntactical structure of the input query; second, the semantic parser where we have used Stanford Named Entity Recognition to recognize the semantics of the entities; third, improving the semantics by using Semantic Role Labeling for which we have used ClearNLP; fourth, the SQL statement generation class, and fifth, generation of a nice presentable result that a nontechnical person can easily understand.

One of the characteristics that distinguishes our research is the focus on extracting the keywords to be processed by the syntactic and semantic parsers. The rules to handle the structure of the query were also designed to use what have been understood by the parsers in the transformation process.

In order to manage complex queries we proposed an algorithm to extract the main keyword and its' characteristics to be used in further processing.

The overall result of our test suite was 81.43% and the result of our proposed algorithm to process complex queries was 80%. These are moderate but encouraging results given the complexity of developing them from scratch and under the hard time constraints. According to the user satisfactory survey, we concluded that the users were pleased with the system performance and some degree of syntactic and semantic processing are needed to improve the results. The intelligent agent still needs improvements in several areas, especially for complex sentences. Key future works are:

(a) Further improving the rules and

(b) Adding more advanced semantics, especially using SEBLA (Semantic Engine using Brain-Like Approach) [15].

References:

[1] J. Piskorski and R. Yangarber, Multi-source, Multilingual Information Extraction and Summarization, 2013.

[2] N. Nihalani, S. Silakari, and M. Motwani, "Natural Language Interface for Database: A Brief review," IJCSI International Journal of Computer Science, 2011.

[3] L. R. Tang, "Using a Machine Learning Approach for Building Natural Language Interfaces for Database: Application of Advanced Techniques in Inductive Logic Programming," 2008.

[4] W.A.WOODS, "Semantics and Quantification in Natural Language Question Answering," Bolt Beranek and Newman Inc., Cambridge, Massachusetts1977.

[5] G. G. HENDRIX, E. D. SACERDOTI, D. SAGALOWICZ, and J. SLOCUM, " Developing a Natural Language Interface to Complex Data," vol. 3, pp. 105- 147, 1978.

[6] A.Rudnicky.SphinxKnowledgeBaseTool--VERSION3.Available:http://www.speech.cs.cmu.edu/tools/lmtool-new.html
[7] L. Niu, J. Lu, and G. Zhang, Cognition in Business Decision Support Systems, 2009.
[8] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields:Probabilistic models for segmenting and labeling sequence data.," 2001.
[9] J. D. Choi, "Optimization of Natural Language Processing Components for Robustness and Scalability," Ph.D, Computer Science and Cognitive Science, University of Colorado Boulder, 2012.
[10] L. Rabiner and B.-H. Juang, Fundamentals of speech recognition, 1993.
[11] (1992).ThePennTreebankProject.Available:http://www.cis.upenn.edu/ ~treebank/
[12] O. Babko-Malaya, "Propbank annotation guidelines," 2005.
[13] F. Li and H. V. Jagadish, "Constructing an Interactive Natural Language Interface for Relational Database," presented at the International Conference on Very Large Data Bases, Hawaii, 2015.
[14] H. Alawwad, "An Intelligent Database System using Natural Language Processing with typed or spoken sentences", Master's Thesis, Al Imam Mohammad Ibn Saud Islamic University (IMSIU), Saudi Arabia, March 2016.
[15] E. Khan, "Intelligent Internet: Natural Language and Question & Answer based Interaction", INTERNATIONAL JOURNAL of COMPUTERS AND COMMUNICATIONS, (NAUN & UNIVERSITY PRESS) Oct. 2013.