International Journal of Engineering Research and Technology.
ISSN 0974-3154 Volume 7, Number 1 (2014), pp. 69-77

http://www.irphouse.com

# Maximum Entropy based Natural Language Interface for Relational Database

**Deepthi S, Rejimoan R and Vinod Chandra S S**

*Dept. of Computer Science, SCT College of Engineering Thiruvananthapuram, deethis@gmail.com*
*Dept. of Computer Science, SCT College of Engineering Thiruvananthapuram, rejimoan@gmail.com*
*Computer Centre, University of Kerala Thiruvananthapuram, vinodchandrass@gmail.com*

**Abstract**

A Natural Language Interface for DataBase (NLIDB) is a system which accepts the user request in natural language and converts to an SQL query. The system consist of: Language Processor (LP) and Query Translator. LP is used to extract information from user query. Main LP techniques used in our system are Part-of-Speech (POS) Tagging and Chunking that are implemented by Maximum Entropy Models. Query Translator (QT) is used to formulate SQL queries. It has predefined query templates which are selected based on constrains, connectors etc. specified in a user query. Finally the SQL Query is obtained by completing the selected query template with the already identified details like attributes, conditions etc.

***Keywords-*** NLIDB, POS Tagging, Maximum Entropy, QT.

## INTRODUCTION

Access of data stored in databases has always been a problem for regular users who are commonly unaware of query languages. Researches have been going on in this area from the late 1960's. These researches were aimed at making a Natural Language Interface for Database (NLIDB) so that users can query the database directly without query language knowledge. NLIDB let users to query the database in formal English language and it translates query into proper SQL queries.

Early NLIDB systems had many roles in interface based query processing. LUNAR was developed as an interface to the database that held information about rocks collected during American moon expeditions [1] and LADDER is a semantic

grammar based database interface of the US Navy ships database [2]. CHAT-80 interfaced the world's geography facts database [2]. All these systems are either domain oriented or developed to serve a single database. Some new NLIDB systems are Semantic Grammar based System [3], Synchronous Context Free Grammar (SCFG) based System [4], PCFG based System [5], WordNet based System [6], Conversation-based System [7] etc. All these systems accept natural language queries, but required some pre-requisites or they might demand user support. For example, semantic grammar based system requires a set of grammars to be defined, WordNet based system requires an ontology and Conversation based System demands user to communicate with the system till enough information for query formulation is collected. All these systems fail to formulate query accurately and hence provide incorrect results [8].

In order to overcome this flaw, an NLIDB system is implemented using of a semi-parser and Maximum Entropy machine learning model which makes predictions only based on known facts. This method increases accuracy of queries formed and eliminates formation of multiple trees or grammars for the same request.

The NLIDB system described in this paper has two parts: Language Processor (LP) and Query Translator (QT). Figure 1 shows the NLIDB architecture. Language Processor is used to identify constraints, predicates etc. The main components of LP are Tokenizer, POS Tagger, Name Identifier and Chunker. Query Translator holds several query templates and is responsible for formulating the correct SQL query that match the user request.

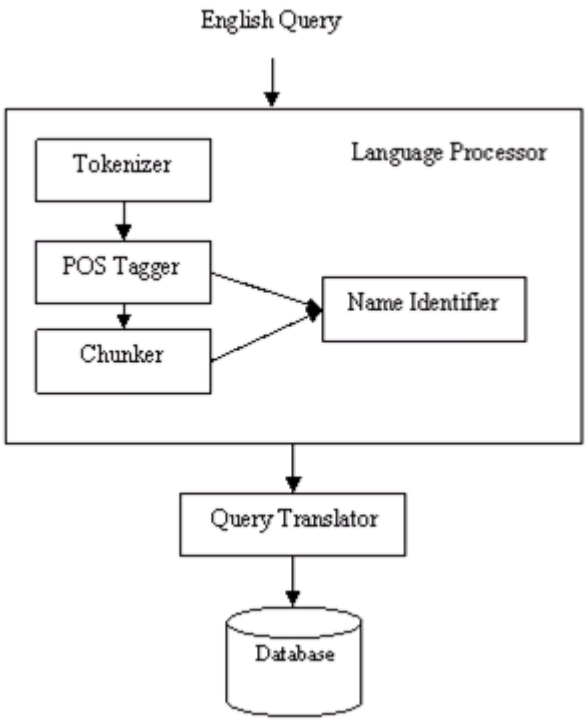


**Figure. 1. NLIDB Architecture**

## LANGUAGE PROCESSOR

Language Processor (LP) is used to analyze user's query, given in an English text. This is first tokenized (sentences are identified and then each sentence is split to words) then passed to POS Tagger. The tagger identifies linguistic category of all words which helps to understand the request. The chunker ensures that no details of given request are miss interpreted. The request and all gathered information are passed to Query Translator for query formulation.

## TOKENIZATION

We split a given text into sentences and end of the sentences is identified by presence of a full stop, question mark or an exclamation mark. If the end of sentence is a fullstop then the abbreviations need to be processed separately. Algorithm1 describes abbreviation check on a text.

***Algorithm 1:***

*Begin*
Let the current End-of-Sentence condition (.) be p
Identify current token, $n_c$ and next token, $n_t$.
*If* $n_c$ is an abbreviation.
    *If* $n_t$, is an end-of-sentence condition, then
    $n_t$ indicates end of current sentence.
    Perform sentence split
    *Else*
    $n_t$ indicates next token of current sentence.
    *Else*
    *if* $n_c$ not an abbreviation
    *if* suffix is whitespace then
    $n_t$ indicates end of current sentence.
    Perform sentence split.
    *Else if* suffix is character, then
    p is part of the token
*End;*

The sentences are tokenized one at a time. Generally tokens are identified by white spaces between words. Special symbols like full stop, question mark, exclamation mark, back slash, double quotes, single quotes, comma, opening and closing braces are also considered as tokens. These tokens are grouped to fix a part-of-speech (POS) tag for a word. For example,

Consider the user query
"List all details of students who joined on 12/3/2013"
Equivalent tokens are
List | all | details | of | students | who | joined | on | 12/3/2013

**POS TAGGING**

Part-of-Speech Tagging marks up a word into corresponding lexical category (verb, noun, adjective, adverb etc.). The tagger makes use of PENN Treebank POS tag set developed by the University of Pennsylvania for NLP related research work [9]. It is a standard tag set accepted around the world. The tagger uses a maximum entropy model trained over the PENN Tag set. Tagger uses WordNet as the underlying dictionary [10].

Maximum entropy based learning is used for predicting tags of words [11]. The feature selected is in the format (a, b) where 'a' is the possible tag, 'b' is the current word and previous two tags. Tag prediction for each word is made by considering the history 'h' (sequence of tags assigned to all previous words of the sentence). Each pair (a, b) has a probability p(a, b) and a tag is selected for 'a' such that it maximizes the entropy H(p) [11]. H(p) is computed using the Shannon's Entropy equation [12].

$$H(p) = -\sum_{x \in A \times B} p(x) \log p(x)$$

$$x = (a,b), \quad where \quad a \in A \wedge b \in B$$

---------- (1)

A denote the set of all tags and B denote set of words and their previous words tags. The current system makes use of twelve features to tag a word: the word, all suffixes and prefixes of the word, special character if any, capitals, regular expressions, previous two words and their tags and two words next to current word. Using these features probability of occurrence of all possible tags are computed, and the tag with maximum probability is fixed as the word's tag.

Tagging process is performed by Algorithm 2. We assume, for each word the history is known to user.

***Algorithm 2:***

Let $h_{ij}$ = the $j^{th}$ highest probability tag sequence up to word $w_i$.
*Begin*
Generate all possible tags for $w_i$
Find the top N tags
For $i$ = 2 to $n$
*Begin*
For $j$ = 1 to $N$
*Begin*
Generate tags for $w_i$ given $h_{(i-1)j}$
Append each tag to $h_{(i-1)j}$ to create new sequence
*End*
Find the highest probability sequences generated
*End*
Repeat for every sentence.
*End;*

After performing POS tagging by Algorithm 2, the user query in our example becomes,

List/NNP all/DT details/NN of/IN students/NNS who/WP joined/VBD on/IN 12/3/2013/CD

where NNP, DT, NN, IN,NNS,WP,VBD and CD denote the POS tags for Singular Proper Noun, Determiner, Singular Noun, Preposition, Plural Noun, Wh-pronoun, Verb past tense and Cardinal Number respectively.

**CHUNKING**

In SQL entities like name, place, date etc. always become part of a query. Since these entities, name, date etc., are often distributed over a piece of text, analysing tokens one by one will not always correctly identify them. This can be solved by performing chunking and hence identifying the beginning and end of such sequences.

Chunking or partial parsing assigns a practical syntactic structure to a sentence and generates a flatter structure than full parsing. Chunks are non overlapping regions of text and every chunk have a head followed by words and their respective tags. Chunker looks through the sequence of POS tags of the tagged text and identifies Noun Phrases (NP) and Verb Phrases (VP). In the context of query formulation NP's will have all possible constrains. For example, details of a particular employ needs to be displayed, the name of employ specified in the request will be present in a NP. So scanning of NP's will ensure that all constraints are identified without fail.

Our chunker is implemented using IOB tags [13]. Each token of a sentence is classified as I (inside), O (outside) and B (begin) chunk tags. The token is tagged as 'B' if it is beginning of a chunk and subsequent tokens within the chunks are tagged as 'I'. Remaining tokens are tagged as 'O'. Tokens are tagged as IOB using the POS tagging algorithm. Here the tagger uses a maximum entropy based model, which is trained over tag patterns. Tag patterns indicate rules based on which the NP, VP etc. are defined. The IOB tags are assigned based on these tag patterns. The features used by this model for performing chunking are: the word, the previous two words, the next two words and their corresponding tags. The chunker works by considering a window of five words at a time and chunk tags are assigned such that the probability of tagged sequence is maximized. For example, the chunking result of our previous sentence becomes,

[NP List/NNP ] [NP all/DT details/NN ] [PP of/IN ] [NP students/NNS ] [NP who/WP ] [VP joined/VBD ] [PP on/IN ] [NP 12/3/2013/CD ]

**NAME IDENTIFIER**

Name Identifier (NI) is used to identify names, locations and dates specified in user query by the use of regular expressions. NI extracts information from POS tagger and Chunker. It use only NP's identified by chunker as names, locations and dates will always be present in NP's alone. In NP's, POS tags of tokens are analysed and nouns that match the defined regular expressions will be either names or locations. In order

to identify dates, words with tag CD (cardinal number) are selected and matched with defined regular expressions. In our example, the NI results in,

List all details of students who joined on <date>12-03-2013</date>

## QUERY TRANSLATOR

Query Translator (QT) converts a given text into SQL query. QT analyzes only the NP's identified by Chunker because details about column names, constraint values etc. are often found in NP. Two pre-requisites used in our work are query templates and column name synonyms. That is, a set of commonly used query formats are pre-defined and they are selected depending on the number of constraints, connectors used etc.

Column name synonyms are natural language alternatives for column names belong to the database under use. A Hidden Markov Model (HMM) is used to identify the column name synonyms in a given text [14]. A large corpus of mappings of all possible column name synonyms and their corresponding database column names are generated. Word probabilities of all possible sequences of every column name synonym is computed. State transition probabilities are calculated by taking the single word probability, double word probability, triple word probability etc. of each column name synonym. For example, consider a column name synonym 'name of employ' whose column name in database would be *e_name*. Now word probabilities are computed for sequences 'name, of, employ, name of, of employ and name of employ'. Their values are 1/3, 1/3, 1/3, 1/2, 1/2 and 1 respectively. Figure 2. shows a sample architecture. These sequences and database column names form the states of the HMM. This model is trained using the corpus and its corresponding computed probabilities.

The algorithm used by the QT proceeds by searching for these column name synonyms in every NP. The HMM returns column name of the sequence in NP whose probability evaluate to 1. Algorithm 3 gives the procedure for QT. It is assumed that all column names and table names of current database are known.

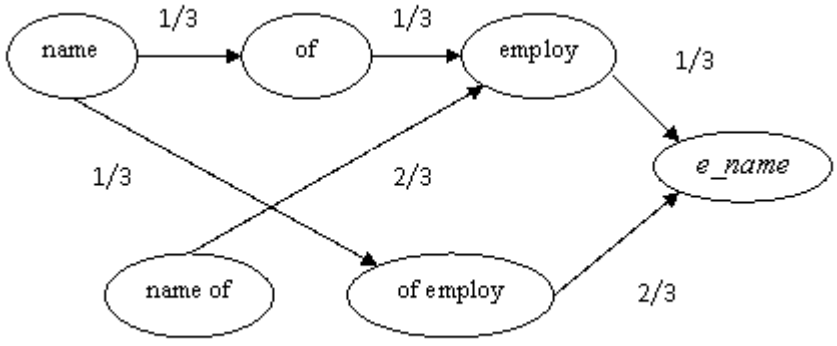


**Figure 2. HMM for the example 'name of employ →*e_name*'**

***Algorithm 3:***
*Begin*
For every NP
*Begin*
*If* NP contain column name synonym
*If* a value succeed it
A constraint is identified.
*Else*
A column to be retrieved is identified.
*Else if* NP contain names
A name column synonym will precede it.
A constraint is identified.
*If* POS tag JJR/JJS exist
'<'/'>' relations exist.
*If* POS tag CC exist
Predicates AND/OR exist.
Identify column names and then table names
*If* no of tables >=2
Add condition joining the tables.
Select appropriate SQL query template
Execute query and display result
*End*
*End;*

**RESULTS AND DISCUSSION**

Our system accepts query in raw English text and do not require conversion to any format for further processing. The main operations of our system are POS Tagging and Chunking. For semantic grammar based system, a set of grammars need to be defined, CFG based system demand input to be represented as in Codd's Tuple calculus expressions, Wordnet based system requires input to be converted to an intermediary format and Conversation based system demand communication till query is derived. The Maximum Entropy based NLIDB do not have any pre-requisites when compared to the systems mentioned in Table1 [8].This system does not require any kind of human intervention to reach the SQL query like the Conversation based system. The SQL query is formulated by bringing together the information collected at the various parts of the system. Other systems perform query formulation by conversion from an intermediate form. For example, the SCFG based system convert the λ-SCFG rules to SQL query, PCFG system function by generating SQL from the grammar tree, the Wordnet based system converts the intermediate representation to SQL and Conversation based system generates SQL query from the leaf node of the tree. It can be observed that the Maximum Entropy based NLIDB is more accurate when compared to the other systems. Once the system is trained it does not need any manual help.

**TABLE I Comparison of NLIDB Systems**

| NLIDB Systems | Requirement | Main parts of system | Important operation | Conversion to query |
|---|---|---|---|---|
| Semantic Grammar based System | A set of grammar should be defined | Parser | Parsing | Query generator |
| Synchronous CFG based System | Expressions in Codd's Tuple Calculus | Noun Phrase analysis | Lambda calculus expression | λ-SCFG rules |
| PCFG based System | Natural language | POS Tagging and PCFG Parser | Finding grammar tree with maximum probability | Tree to SQL |
| WordNet based System | Ontology | Language and database processing module | DRS | DRS to SQL |
| Conversation based System | Conversation | Conversation agent, knowledge tree and conversation manager | Knowledge tree traversal | Leaf node to query |
| Maximum Entropy based NLIDB | Natural Language | Language Processor and Query Translator | POS Tagging and Chunking | Query Translator |

## CONCLUSION

Our system accepts user request in natural language and converts it to SQL query. In order to do this the request is processed by the LP module. POS Tagger will tag each word in English request based on its lexical category. It will lead to a chunker which extracts the noun phrases present in the request. Next, NI identifies constraints specified in the query with the help of regular expressions. It is done by performing a search in NP's identified by the chunker.

The information gathered by LP is given to the query translator. It will identify the column names of specified constraints, column names of requested data etc. Table names of these required columns are extracted and all these data are mapped to an appropriate query template (pre-defined). Query templates are chosen depending on the identified number of tables, number of constraints, connectors etc. An advantage of this system is that it could be made to work with any database.

## REFERENCES

[1] Huangi; Guiang Zangi; Phillip C-y Sheu, "A Natural Language database interface based on probabilistic context free grammar", *IEEE international workshop on Semantic Computing and Systems*, 2008, pp 155-162.

[2] I. Androutsopoulos; G Ritchie; P Thanisch. "Natural language interfaces to databases-an introduction", *Journal of Language Engineering*, Vol. 1, 1995, pp. 29-81.

[3] Gauri Rao; Chanchal Agarwal; Snehal Chaudhry; Nikita Kulkarni; Dr. S H Patil, "Natural Language Query Processing using Semantic Grammar", *International Journal on Computer Science and Engineering*, Vol. 2, 2010, pp. 219-223.

[4] Zhao; Tie-jun Jun; Xu Chong, "A Head-Annotated Synchronous Context-Free Grammar for Hierarchical Phrase-Based Translation", *IEEE International Conference on Natural Language Processing and Knowledge Engineering*, 2011, pp. 51-55.

[5] Bei-Bei Huang; Guigang Zhang; Phillip C-Y Sheu, "A Natural Language Database Interface Based on a Probabilistic Context Free Grammar", *IEEE International Workshop on Semantic Computing and Systems*, 2008, pp. 568-573.

[6] Hu Li; Yong Shi, "A WordNet-Based Natural Language Interface to Relational Databases", *IEEE Conference on Computer and Automation Engineering*, 2010, pp. 514-518.

[7] Majdi Owda; Zuhair Bandar; Keeley Crockett, "Conversation-Based Natural Language Interface to Relational Databases", *IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, 2007, pp. 363-367.

[8] Deepthi.S; Rejimaon R; Vinod Chandra S S, "A Review on Natural Language Interface for Database", *International Journal for Applied Engineering Research*, Vol.8 No.4, 2013, pp. 399-412.

[9] Mitchell P Marcus; Beatrice Santorini; Mary Ann Marcinkiewicz, "Building a Large Annotated Corpus of English: The Penn Treebank", *Association of Computational Linguistics*, Vol. 19, No. 2, 1993, pp. 313-330.

[10] George A Miller, "WordNet: A Lexical Database for English", *Communications of the ACM*, Vol.38, No. 11, 1995, pp. 39-41.

[11] Adam Berger; Stephen A. Della Pietra; Vincet J. Della Pieta, "A Maximum Entropy Approach to Natural Language Processing", *Computational Linguistics*, Vol. 22, 1966, pp. 39-71, 1966.

[12] Edwin C May; S James; P Spottiswoode; Christine L. James, "Shannon Entropy: A Possible Intrinsic Target Property", *The Journal of Parapsychology*, Vol.58, 1994, pp. 384-401.

[13] Hans van Halteren, "Chunking with WPDV Models", *Proceedings of CoNLL-2000 and LLL-2000*, pp. 154-156.

[14] Rabiner L; Murray Hill, "A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Preceedings of the IEEE*, Vol.77, 2002, pp. 257-286.