Using Deep Learning Technique to Query Relational Data using Multi-lingual Query Generator and Translator with NLP support

Sunilkumar N. Beghele, Pallavi V. Kulkarni

Abstract— A smart and intelligent interface utilized & enhance effective interaction between its' users with the underlying databases. Such a system application needs for complex query problem as faced by the user who has an understanding of databases. The database should be efficient and should allow quick access. However, all users are unfamiliar and accustomed to queries and structural implementation in structured_query_language (SQL) because of lack of knowledge, of structure info database. Therefore, naiveusers need an intermediate system interact RDB natural_language that is English. For the same, (Database_Management_System) with the ability to inter-compile natural_language (NL). In the research proposal, we intend to create-develop an interface using Meaningful matching techniques that will translate natural search terms as SQL using a predefined set of written production rules and predefined data dictionaries, the data dictionary will consist of a set of definitions for relationships and properties. Pair of steps, such that lowercase conversion, tagging, tokens, database elements, and SQL separation elements are used for conversion the natural language query (NLQ) in SQL query.

Index Terms— natural language processing, SQL, natural language query interface, ambiguity.

1 INTRODUCTION

Natural(Spoken)_language_processing has many applications that require an in-depth understanding of human spoken-language in creating natural_language or both, such that automatic translation that arbitrates on converting textual contents from human-based language. Different instance to be considered for Natural (N) Language (L) Processing (P) is the bifurcation of broadly classified data related to "Real-Actual data free- text [1]". The topic deals with the vital and important applications which are (N) Natural (S) Spoken (L) Language (I) Interface to the database (NLIDB) [1] from [2]. The database of the concept of NLIDB comes from the method of questioning databases that use natural search terms such as English instead of database language to find information. The article deals with one important application of NLP [2][3], which is a Natural-Language-Interface with DB. The prior lookup for NLIDB [1] application is with the concern that users with little programming expertise can manage the database with ease. Users can utilize verbal-language to transact with DB, i.e. is very easy and convenient. Additional to this, users do not need to own any special training to operate or use such systems. Users do not need to assimilate any DB language as it is difficult to learn any language with conventional queries such as SQL [2][20] for beginners. Although it's easy to use spoken linguistic queries related to various DB tables. The primary purpose of this article is to facilitate the use of DB, where users can extract data using natural statements using intelligent agents that can understand user commands and can create responses.

- *Sunilkumar Nandkishor Beghele is currently pursuing masters degree program in Computer Science and Engineering in Government College of Engineering Aurangabad, 431005, India. E-mail: baghele.sun@gmail.com*
- *Pallavi V. Kulkarni is Completed masters degree program in Computer Science and Engineering. Current designation is Assistant Professor of Computer Science and Engineering in Government College of Engineering Aurangabad, 431005, India. E-mail: Pallavi.k11@gmail.com*

In general, methods that can interact with databases that use NLP[2][3] categorized in three variations: (1) format, format or pattern that uses templates (2) sentence patterns and (3) sentence patterns and meanings. For the first form, the entered query will be managed by equating the set of rules or predefined patterns. The next step will be translated into a logical format according to the format. Using these rules, the DB query will be executed directly. In the second form, the tree is a syntax element using the syntax parser. Use [3] that can be left in the query database mapping process according to the syntax of the preset syntax. The third generation, the purposeful semantics will increase the structuralism of intelligence and the query will be processed according to the meaning. The prominence of the NLIDB system will vary according to the method used to convert search terms into the base language. Data due to SQL generally has four steps to convert: vocabulary analysis, vocabulary analysis/meaning, query creation and retrieval of answers.

2 RELATED WORK

In today's fast-computing environment, using the computer as an information search tool that will help educational organizations to operate and manage its various information-systems. These are utilized for managing information that will help resolve different types of data-values stored in a database called DBMS (Rukshan et al., 2013).

Despite the large voluminous data being retrieved in relational_database but users still want to use the DB schema language to define the query completely. (A)Artificial (I) intelligence and linguistics be integrated to create a process which will assist to interpret and initiate information in NL (Johnson, 1985; Mckay [3] and Finin; 1990; Wan; 2000; [4] Mohite and Bhojane, 2014; Javubar and Jay, 2015).

NLP that uses a database is, therefore, a significant achievement in the NL process. It is an easy way to access

information by asking questions NL for answers because ordinary people may become unsuccessful to understand the language of the database. The NL Q database transacts accepts natural (spoken)-language-queries, generates SQL queries and then executes to select information via relational DB. The results drawn from the DB are a stream of elements. The questionnaire was created by identifying the relationship of words in the elements of NL Q.

The creation of a strong and effective NLIDBS has been vital in later years. The proportion of data present on the internet is increasing fastly and a huge percentage of people have access to information stored in various repositories via the web browser [5]. Therefore, NLIDBS was created to increase efficiency. Search results and data creation with greater accuracy.

From the past several years, there are endless attempts to create an effective natural language questionnaire interface. There is a lot of research that introduces the theory and implementation of new NLIDBs, but products that do not meet the expectations that are needed. LUNAR was launched in 1973 as a system that handles orders related to samples of rocks that have been taken from the moon.

It utilized the Built-Up Transition Network Analyzer (BTN) as one of the best DB language manipulating systems. Designed to retrieve-extract information about US-Navy-ships. It uses semantic syntax to separate user queries in NL. It can handle relative queries one relational structure and multiple queries with simple join conditions.

There is another general architecture presentation for intelligent database interface [5] whose main-attributes are domain independence, which can be interpreted as the interface can-be-used with any DB. The transact uses meaningful blending techniques for conversion of natural(spoken) language queries into Structured-Query Language using dictionaries and sets of production rules. [8]Pimpalkar, Sontakke et .al. (2014) launched the system for those who are satisfied with Hindi.

The author develops the system according to the rules that meet the needs of users by accepting Hindi as a search term with results shown in Hindi only. Savvy, which is a general application pattern matching frameworks (Poole and Mackworth, 2010) uses various forms written in different types of queries that will be processed on-entering the complete questionnaire. This system is very easy to use without having to have a thorough parsing and requires an interpretation module.

Despite receiving many achievements, NLIDBs currently do not guarantee the translation of search terms in natural language into database languages. The research has designed and implemented a system called Natural-Language to SQL Converter (NLTSQLC). It will work and convert the NLP Query to SQL Query. The current research extends existing work by processing more complex queries while removing ambiguity.

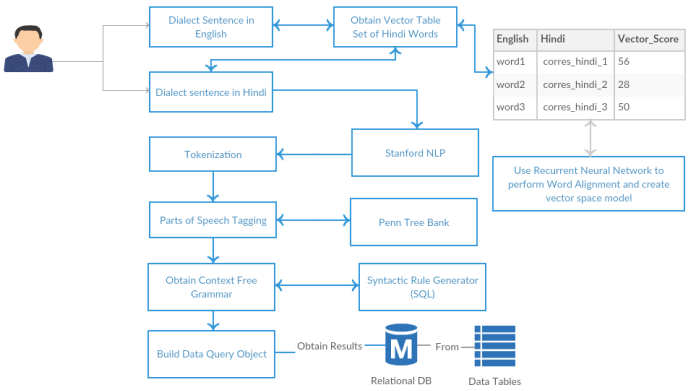
3 SYSTEM ARCHITECTURE

We will design an analyzer that will allow us to extract important characteristics that affect the entire conversion process of natural (spoken)-language queries in equivalent SQL statements. In addition, we will design algorithms to manage the complexity of higher meanings for natural language queries.

We will implement Stanford NLP toolkit with the following architecture:

Figure 1: Proposed Architecture

Typical sentence patterns present linguistic data using tokens,



morphology analyzers, Parts of Speech (POS).In English grammar, there are 8 parts of speech: pronouns, adjectives, nouns, verbs, adverbs, conjunctions, Prepositions and exclamations. For reading items (text) from the DB we use [6] Stanford's Stanford linear tag recording tool [7], which determines the sound portion (Vocabulary category) for every word.

In the year 2003, it was presented by a member of the Department of Computer Science (CS) at Stanford University. In Oct. 2014 released last time, if we sent this sentence: "What is the fee for COMP?" To POS (Parts-Of-Speech) tagger we will receive. Reply in Penn format. Binarized in the tree format "What_WP is_VBZ the_DT" with WP tags (Wh-pronoun), VBZ (verb, 3rd person, current singular), NN (noun, plural or singular), DT (Determiner), IN (grouping or preposition) And NNP (Verb, no-3rd person singular present) determined to the query term.

Syntax Analyzer, Token Tag, Sentence Return from Token Analyzer

In our query, there are some keywords and to identify those keywords we use audio labels. After identifying parts of those keywords it sends to the higher level of processing which related to the meaning of the word. We notice that the keyword (node) of the query is: name, adjective, and number. Other words are canceled because it does not affect the conversion process.

Our next stage is to analyze semantic processing:

To understand the queries we required to add additional data or information in the database because Parts-Of- Speech (POS) tag alone is not sufficient to convert the queries of natural language into SQL one. In doing this, we use the Stanford Identifier Named-Entity-Recognizer (NER) to determine the keywords that we have separated from the search to the predefined categories that belong to it. Then, when used in conjunction with POS tags, we use Stanford NER [9] to add meaning to categories or entities that belong to keywords. Therefore, the name provided in the queries recognized or capture as a PERSON and as per our database, the person is

considered as an Employee. We have to practice our own Entity Recognition(ER) model for our own dept_name to appoint various departments as organizations. Suppose let us consider the example if we send the phrase "What is Ahmad's salary working in the programming department?" Accepted as a person, based on our database schema, our system understands that this person is an employee. This understanding allows us to create SQL statements.

A. Query Definer:

The QueryDefiner class performs the first step to create an SQL statement. By defining a list of synonyms we define the type of SQL statement, we expect that users to use for each and every type: SELECT, DELETE, INSERT. I like: "Show me", "Give me" and "what" for SELECT. After the creation of the SQL statement, the second and third step is to separate keywords.

The SQL statements do not disturb the conversion process after the cancellation of any tokens, because the token is added to the appropriate sentence of the SQL statement according to the set of rules defined to process the query structure. There are situations in which POS [13][14] and NER of Stanford do not understand the user's query, so they cannot be converted to SQL. Therefore, we need to use the role tag, meaning that represents the semantic relationship between the display (verb) and Arguments with arc labels. We perform dependency analysis in semantic analysis, to find the syntax dependency structure that each token (word), exemption of the root, has a link (dependency) to the main token. Then we use labeling, meaning roles that each region has meaning information.

B. Psuedo-Code:

• Complete the required process of the input query (INSERT, DELETE or SELECT).

• To extract the keywords delete the filling (filler) words.

• Check Column \ Table name or a synonym of one of them for each keyword: If not: check if it is: Person, Operator, Organization or a WHERE clause's condition.

• After processing it in the appropriate clause of the SQL statement add the keyword if it was A or B.

• Reorganize the SQL string if essential.

For performing analysis start with the Stanford POS marker. Then, to extract the keywords used by the Named-Entity-Recognizer (NER) the keyword extractor uses the info from the POS marker. The Named-Entity-Recognizer (NER) describes the associated domain concepts like a person or department. In the difficult questionnaire, we go through a semantic analysis of dependency. Then, we go to the node assignment that assigns each node in the keywords to the component of the equivalent SQL statement. The SQL statement is performed against our relational DB. The response generation class handles the recovered response to form the response that will be presented to the user.

We will use the labeling of part of the Stanford natural language [7][10] as a syntax parser, the Stanford Entity-Recognizer (ER) [11] as a semantic analyzer and we will compare with OpenNLP [12] to analyze complex queries.

Come to the preprocessing phase, primary training is given to the system. In this phase, all dictionaries are loaded. The synchronization dictionary consists of all the verbs, nouns, adjectives and adverbs that are acquired from the DB. Then the labeling of part of the speech (POS) [13][14] of the entry is made. The Stanford NLP framework is used to make a part of the speech labeling of the query in English; the ILT Indo Wordnet framework is used for the Marathi and Hindi consultation. A speech part label is an s/w package that reads the text and assigns speech part labels to each word, such as noun, verb, adjective, etc. In this structure, we focus on 10 Parts-Of-Speech (POS) tags: NNP, NP, NN, NNS, JJ,NNPS, JJS, JJR, RB, RBR and RBS for plural names, proper names,, common names, singular names, name in plural, superlative adjective, comparative adjective, adjectives, adverb, superlative adverb and comparative adverb correspondingly. To select a single form of a word as a substitute of different form used Steaming.

C. Word to Vector Representation:

Word2vec translates the textual form into a mathematical form that the deep neural network can easily understand [12][15]. After completion of labeling part of the speech, for each word in the input sentence a Word2vec representation is created. The Word-to-Vector (word2vec) representation module shows the words in the input query with a particular numeric value that is provided as an input to the deep neural network to complete the deep learning process.

CBOW is a continuous word bag model used to guess a word in a given context. On another hand, for prediction of the context when an input word is given to use skip-gram. Skip-gram takes more time and is more accurate than CBOW. But it is not efficient compared to CBOW. Because the CBOW model donates more weight, the CBOW model will be used to train the neural networks. In the proposed system the author uses the CBOW model. The below Figure 2 display the simple CBOW model [14].

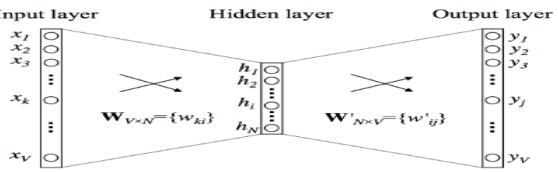


Figure 2: CBOW model

D. Deep Learning Process:

Deep learning is the branch of machine-learning (ML) that uses deep neural networks to train the system to make decisions as a human being. There are several types of neural networks, such as the recurrent neural network, the direct-feed neural network, the convolutional neuronal network, the recursive neural network [15][18]. Each of these neural

networks has its advantages and disadvantages. For the deep learning process, the proposed system uses RRNN. RRNN is the recursive recursive neural network, which combines the functionalities of recursive and recurrent-neural-networks (RNNs). The architecture shown in figure 1 depicts that after the representation of word2vec, the words associated with the user's query are obtained using deep-learning RRNN. This is completed through theme modeling. Modeling of themes comes under in text mining. If the words in the input query do not absolutely match the words in the documents, then the system recovers the result based on the words acquired through the modeling of the topic. There are two ways to done topic modeling: first is the modeling of LDA themes and the second is the modeling of keywords. To train this system using a deep learning model with the help of keyword modeling. The preprocessing phase ends with the completion of the learning process.

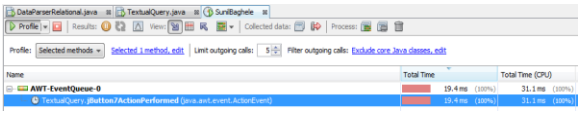
The 3rd phase of bilingual mapping is a non-compulsory phase. If the user wishes to search for the same language, it is not necessary to carry out any bilingual mapping and this phase is excluded. On another hand, if the user requests multilingual results, it is compulsory to carry out bilingual mapping. In the proposed system, the bilingual corpus (developed by the Indian Language Technology Center (CFILT) in IIT, Bombay)[16] and wordnet is used to make bilingual maps. The work of the wordnet is to find the synonyms of the words in the input questionnaire, thus the system makes a comparison based on these synonyms to regain better results.
In the 4th phase, the cosine similarity score is evaluated to obtain the closest match result. In the case of infrequent words, the cosine similarity is a very valuable concept. The below depicts the formula for evaluating the cosine similarity score is:

$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

Finally, the exit phase is last phase of our system. In this phase, the end-user obtains the list of files classified according to the highest score matching the lowest score matching as a result.
The aim of this work is to ease the process of querying relational DB for non-technical users.

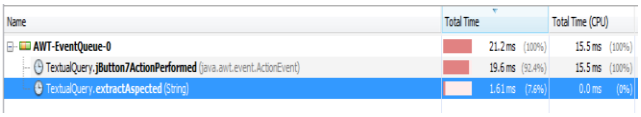
4 EXPERIMENTAL SETUP

A. Profiling with Java Profiler



Our Query generation process requires 19.4 milliseconds to generate SQL query from NLP data processing. With a total CPU time of 31.1 milliseconds.

B. Profiling Stanford NLP Processing



It takes 1.61 milliseconds to process a single sentence from user and process to extract aspects and adjectives to form relation and generate query based on the aspects.

C. WAMP Server for Relational Database

We have used WAMP server with MY-SQL to create our database. WAMP (Windows, Apache, My-SQL, and PHP) is an alternative of LAMP for Windows systems and is frequently installed as a software bunch (Apache, My-SQL, and PHP). WAMP also includes PHP and My-SQL used for creating dynamic websites. MySQL is a high-speed DB.

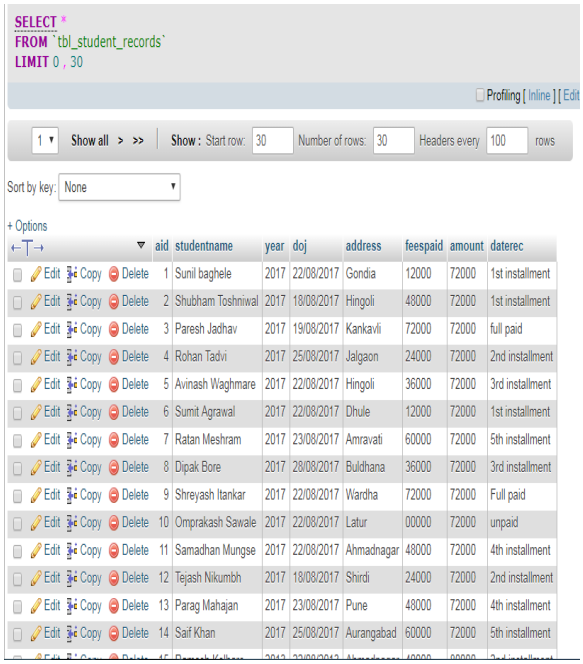


Figure 3: Screen of relational data table

D. Using Synonyms from WordNet Dictionary

After finalizing the relationship between the MRMap and the semantic information query generator outputs the final query. With the help of wordnet, synonyms of the unrelated words can be found. Ex. the set {every, pupil} {entire, seekers} will produce the similar improved set{all, students} in which students is the token which will be mapped to the DB attribute student and all is the token which will be considered as the reserved keyword and can be utilized later.
True Positives (TP) – TP is a result where the model correctly predicts the positive class.
True Negatives (TN) - TN is a result where the model correctly predicts the negative class.
False Positives (FP) – FP is a result where the model incorrectly predicts the positive class.
False Negatives (FN) – FP is a result where the model incorrectly predicts the negative class.
Accuracy – It is an important performance evaluation

technique in which a ratio of correctly predicted observation to the total observations is taken to calculate the result. Mostly, if the accuracy is high then we believe that our model is best. When the values of false-positive (FP) and false-negative (FN) are approximately the same then the only accuracy is an enormous measure will be considered. For that reason, you need to be considering other parameters to estimate the performance of your model. In our model, we have got 97.63% which state that our model is approximately 97.63 % accurate.

$$\text{Accuracy} = (TN+TP) / (TP+FP+FN+TN)$$

Precision - It is basically a ratio of predicted positive observations to the total predicted positive observations. When the precision is high then the false-positive (FP) rate becomes low. Here, we have got 0.788 precision which is best suitable.

$$\text{Precision} = (TP) / (FP+TP)$$

Recall (Sensitivity) – It defines as a ratio of correctly predicted positive observations to the all observations in actual class.

$$\text{Recall} = (TP) / (FN+TP)$$

F1 score – It defines the weighted average of Precision and Recall. For that reason, we need to consider both false-positives (FP) and false-negatives (FN). However, understanding of this procedure is not easy as accuracy. Although F1 is usually more valuable compare to accuracy, especially having an uneven class distribution. The same cost of both false-positives (FP) and false-negatives (FN) is best suited for Accuracy. We need to look at both Precision and Recall, if the cost of false-positives (FP) and false-negatives (FN) are very different. F1 score is 0.701.

$$\text{F1 Score} = 2*(\text{Precision}* \text{Recall}) / (\text{Precision}+ \text{Recall})$$

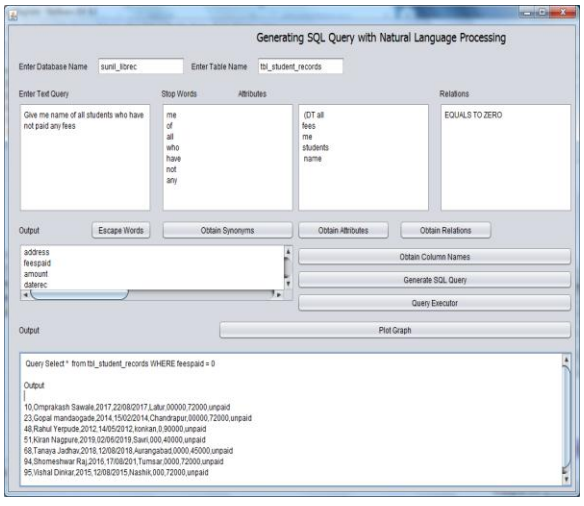


Figure 4: Output of Natural Language Input into Query Translation with final output displayed.

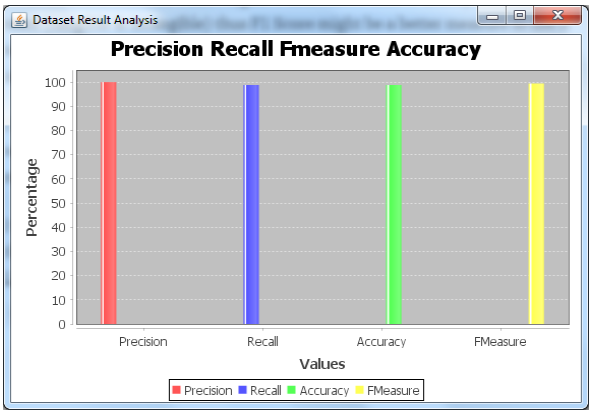


Figure 5: Precision Recall FMeasure and Accuracy

We have performed tokenization and splitting. Using Stanford NLP we could achieve this on text while giving Java only 20MB of memory. java -mx20m -cp "$STANFORD_CORENLP_HOME/*" edu.stanford.nlp.pipeline.StanfordCoreNLP -annotators tokenize, split outputFormat text.
Our annotation speed is about 200 tokens/second.

E. Using Parts of Speech

When the Parts-of-Speech (POS) tagger starts utilizing memory and time linearly with sentence length, the Parts-of-Speech (POS) tagger supported pos.maxlen flag should rarely required. The default english-left3words-distsim.tagger is quicker than the comparison of english-bidirectional-distsim.tagger.

F. Using coref

Neural-English coref is a reasonable option for higher quality co-reference. Some properties of the number co-reference model will speed up their application to long documents:

- dcoref has dcoref.maxdist with value the maximum number of tokens back to look for a coreferent declares.
- Both neural coref and statistical coref have coref.maxMentionDistance and coref.MentionDistanceWithStringMatch provide two variants of the same kind of limiting how far back you look for coreferent declares.

5 CONCLUSION

By adding natural (spoken)-language processing abilities to a DB through an intelligent agent (INLIDB), the ease of use is improved for users with no programming background to consult the database with their own spoken language. A challenge in the evolution of NLIDB is not having a good capacity to overcome the problems of natural languages, such as semantics, ambiguity and the universe of discourse, which hinders the process of transformation. Using Stanford parts of speech label we understood the syntactic and structure of the input query. Using Stanford NLP we have generated an intermediate SQL query that allows user to obtain meaningful

results that are understandable by individuals who are non-technical users.

REFERENCES

[1] Gupta, R Sangal. 2012. Novel Approach to Aggregation Processing in Natural Language Interfaces to Databases. Language Technologies Research Centre International Institute of Information Technology, Hyderabad, India

[2] Javubar SK, Jay A. 2015. Natural language to SQL generation for semantic knowledge extraction in social web sources. Indian Journal of Science and Technology, 8(1): 1-10

[3] Johnson T. 1985. Natural Language Computing: The Commercial Applications. Ovum Limited, London, UK Mckay DP, Finin TW. 1990. The intelligent database interface: Integrating AI and database systems. Proceedings of the 1990 National Conference on Artificial Intelligence. 677-684

[4] Mohite A, Bhojane V. 2014. Challenges and implementation steps of natural language interface for information extraction from the database. International Journal of Recent Technology and Engineering, 3(1): 108

[5] Nihalani N, Motwani M, Silakari S. 2011. An intelligent interface for relational databases. International Journal of Simulation: Systems, Science and Technology, 11(1): 29

[6] Poole D, Mackworth A. 2010. Artificial Intelligence-Foundations of Computational Agents. http://artint.info/index.html

[7] Rao G, Agarwal C, Chaudhry S, et al. 2010. NATURAL LANGUAGE QUERY PROCESSING USING SEMANTIC GRAMMAR. International Journal on Computer Science and Engineering, 2(2): 219-223
Rukshan A, Rukshan P, Mahesan S. 2013. Natural Language Web Interface for Database (NLWIDB). Proceedings of the Third International Symposium. SEUSL, Oluvil, Sri Lanka

[8] Sontakke AR, Pimpalkar A. 2014. A rule-based graphical user interfaces to a relational database using NLP. International Journal of Scientific Engineering and Research, 3(4): 81-84

[9] Sreenivasulu M. 2014. Information retrieval using natural language interfaces. International Journal of Computer Applications, 92(12): 34-37

[10] Wan FJ. 2000. A fuzzy grammar and possibility theory-based natural language user interface for spatial queries. Fuzzy Sets and Systems, 113: 147-159

[11] I. Androutsopoulos, "Interfacing a Natural Language Front-End to a Relational Database(MSc thesis)," Technical paper 11, Department of Artificial Intelligence, University of Edinburgh, 1993.

[12] Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates, "Modern Natural Language Interfaces toDatabases Composing Statistical Parsing with Semantic Tractability," COLING 2004.

[13] Y. Li, H. Yang, and H.V. Jagadish, "NALIX: an interactive natural language interface for querying XML," in Proceedings of the International Conference on Management of Data, pp. 900–902, 2005.

[14] I. Androutsopoulos, G.D. Ritchie, and P. Thanisch, "Natural Language Interfaces to Databases – An Introduction," j. Lang, pp. 29-81, Eng.1995.

[15] E.W. Hinrichs. Tense, "Quantifiers, and Contexts. Computational Linguistics," 14(2), pp.3–14, June 1988.

[16] P. Reis, J. Matias, and N. Mamede, "Edit – A Natural Language Interface to Databases: A New Dimension for an Old Approach, in Proceedings of the Fourth International Conference on Information and Communication Technology in Tourism (ENTER' 97), Edinburgh, 1997.

[17] Jurgen Albert, Dora Giammarresi, and Derick Wood, " Normal form algorithms for extended context-free grammars," in Theoretical Computer Science 267, pp. 35–47, 2001.

[18] I. Androutsopoulos, G. Ritchie, and P. Thanisch, "Natural language interfaces to the databases-an introduction," in Journal of Language Engineering, v. 1(1), pp. 29–81, 1995.

[19] Manning C. and Schütze H., " Foundations of Statistical Natural Language Processing," MIT Press, Cambridge, 1999.

[20] Luis Tari, Phan Huy Tu, Jorg Hakenberg, Yi Chen, Tran Cao Son, Graciela Gonzalez, and Chitta Baral, " Parse Tree Database for Information Extraction," in IEEE transactions on knowledge & data engineering, 2010.