ISSN No. 0976-5697

Volume 2, No. 6, Nov-Dec 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A New Architecture for NLIDB Using Local Appropriator Engine for SQL Generation

Enikuomehin Oluwatoyin*
Computer Science Dept,
Lagos State University
Lagos, Nigeria
toyin@lasunigeria.org

Sadiku J.S
Computer Science Dept,
University of Ilorin
Ilorin, Nigeria
ijssadiku@unilorin.edu.ng

Abstract: In this paper, we present the design and implementation of a special type of Natural Language Interface using the concept of Local Appropriator (LA). The purpose of natural language introduction to database is to allow users post questions coined "natural language" and get discriminated answers which form the result of their query in terms of tables. A Local Appropriator Natural Language Interface, the LANLI, parses semantically, a natural language on a relational database by building the corresponding SQL. The LANLP is a new generation of NL2DB which uses a Syntactic-Semantic (SySe) tree to generate a matching algorithm for its transformation using word net for semantic analysis and a parse tree algorithm for its syntactic analysis. The generated tree is fuzzified to handle ambiguity contained in word. For expert users, this advancement could be likened to the development of an automated structure query language (sql) generator. The essence of this paper is to present a new architecture for querying databases that provide users with the capabilities of extracting information contained in the relational database. Users do not have to learn the process of formulating a query through sql..

Keywords*:* natural language interface, database, local appropriator, query, sql

I. INTRODUCTION

Information Technology and computerization has taken over the way organizations operate. Public and private organizations now depend on the services embedded in IT to perform their services. This is a justification for the continued acceptance of the Database Technology as a way to properly handle data. This has resulted in a simple extraction: *more people will have to access the content of a database*; however, most of these users are non-expert in the field of IT and computer science thus they do not have the knowledge of how to use the ANSI approved universal Database language, the Structure Query Language (SQL). Its therefore becomes necessary to develop a system that can enable non expert users retrieve data from the database by posting request in their natural language. Natural language NL, as language spoken by man such as English, contains high level of ambiguity, thus a NL interface for relational database will only be appropriate for a section of NL at time however this coverage can be extended. We therefore propose the use of a Local Appropriator (LA) as a model that completes the transformation of user natural language request at the front end without having to affect the structure of the database. We examine the earlier transformation processes and propose a new transformation algorithm using appropriate model to measure the accuracy level of the result presented by non sql expert and that presented by an sql expert on the same domain.

Consider the following:

a. By non sql expert; All students in biology department
b. By sql expert: Select * from Student Where dept_name =biology.

The above simple request shall be upheld as a sample request in this paper.

We attempt to compare the result of these two simple queries and use that as a measure of the performance level of the system. Systems that are capable of managing data are classified as Database Management System (DBMS) [1]. Retrieval processes in relational databases have been shown to be very important to the growth of database technology [2]. The growth in database usage is experiencing an all time high as both private and public sectors now depend largely on it to achieve their organizational goals. The database allows for view of data at a logical level. Operations on the database are purely on dedicated languages which are capped as either Data Manipulation language (DML) or Data Definition Language (DDL). Also, in use is the Data Control Language (DCL). All of these are embedded in the structure of a complete structure query language, sql.

SQL is the ANSI approved querying standard for accessing the database, It has been found appropriate for use in most DBMS. A major concern about the performance of sql is its inability to handle ambiguity as pertinent to the way human talk, reason and act. This singular act makes the sql incomplete and inappropriate it for non trained users. Due to the above constraint, it has become necessary and important to develop a system that will be able to extract additional hidden data from the database. Intelligence has been introduced to database itself [4] however, it has become necessary to move the enhancement from the database to the frontend, in this case, the query interface. We therefore suggest the concept of a Local Appropriator. We propose the Local appropriator as an intelligent sql generator that transforms a user request from plain language to sql.

II. NLIDB AND SOME RELATED WORK

Natural Language Interface to Databases (NLIDB) has been an area of interest over some time. It is a branch of a larger domain of research called Natural Language processing which itself is a study area in Artificial Intelligence. The research on this aspect of AI has been on

since early 80s [4], [5]. The dream has always been to develop a system that can interact with human in an English way[6]. NLIDB are generally seen as easy ways to obtain information from databases. These days they are seen as special types of Question Answering systems. Relationship has been proposed for QA systems like HITIQA[7]

Many times, researchers have linked the NLIDB with the QA systems. Scientifically, they have been shown to have fundamental differences. QA addresses answers using the structure of a text while the NLIDB uses a transformation of its meaning-- the semantic generation. Both systems, are working to make the computer understand human by building a higher level of interaction thereby making users more flexible in system usage. A proper system should be able to interprete a request appropriately.

Most existing database queries that have been proposed do simple keyword match. They initially attempt to break down the request but still execute such by matching relevant keyword other than extracting the actual meaning of a request. They locate the keyword in any query and try a match them across defined database. Some use forms also known as template to analyze users input[8]. The interface of the LA allows users to type in their NL questions which are mostly fuzzy-type sentences without using a form or a template.

III. OVERVIEW OF LOCAL APPROPRIATOR SYSTEM

The concept of Local Appropriator is a new attempt aimed at making the operations of the querying processes end at the frontend without any further activity required from the database. This will enable the system to have the ability to work on many distributed databases and of course, improve time of execution. All activities at the interface takes place at the domain of the Local Appropriator. A nested structure of the syntactic natural language is mapped onto the semantic generations using the concept of wordnet. This leads to a Syntatic-Semantic tree (SySe tree) formation, the syntactic and semantic tree, on the formation can be implemented for retrieval processes using a transformation into the conventional SQL for execution in the RDBMS.

The Earley J algorithm was invoked for use on the parse structure. With Earley's syntactic parse algorithm [9], we can decide if a word will be generated by a given grammar. This semantic algorithm is induced to generate a high level influential and portable interface. The link parser analyzes the syntactic structure of the sentence. The Link Grammar Parser is a syntactic parser of English (and other languages as well), based on link grammar, an original theory of English syntax.

IV. A RECAP OF THE EARLEY'S ALGORITHM

As adapted from [10], given a sentence, the system assigns to it a syntactic structure, which consists of a set of labelled links connecting pairs of words. The parser also produces a "constituent" representation of a sentence. The descriptions, α, β, and γ represent any string of terminals/non-terminals (including the empty string), X and Y represent single non-terminals, and *a* represents a terminal symbol. Earley's algorithm is a top-down dynamic programming algorithm. In the following, we use Earley's dot notation: given a production

$$X \rightarrow \alpha\beta \qquad (1)$$

The notation $X \rightarrow \alpha \bullet \beta$ represents a condition in which α has already been parsed and β is expected. For every input position (which represents a position *between* tokens), the parser generates an ordered *state set*. Each state is a tuple

$$(X \rightarrow \alpha \bullet \beta, i), \qquad (2)$$

consisting of the production currently being matched ($X \rightarrow \alpha\ \beta$), the current position in that production (represented by the dot) and the position *i* in the input at which the matching of this production began(the *origin position).*(Earley's original algorithm included a look-ahead in the state; later research showed this to have little practical effect on the parsing efficiency, and it has subsequently been dropped from most implementations.)

The state set at input position *k* is called S(*k*). The parser is seeded with S(0) consisting of only the top-level rule. The parser then iteratively operates in three stages: *prediction*, *scanning*, and*completion*.. **Prediction**: For every state in S(*k*) of the form ($X \rightarrow \alpha \bullet Y\ \beta, j$) (where *j* is the origin position as above), add ($Y \rightarrow \bullet\ \gamma, k$) to S(*k*) for every production in the grammar with Y on the left-hand side ($Y \rightarrow \gamma$).. **Scanning**: If *a* is the next symbol in the input stream, for every state in S(*k*) of the form ($X \rightarrow \alpha \bullet a\ \beta, j$), add ($X \rightarrow \alpha\ a \bullet \beta, j$) to S(*k*+1).. **Completion**: For every state in S(*k*) of the form ($X \rightarrow \gamma \bullet, j$), find states in S(*j*) of the form ($Y \rightarrow \alpha \bullet X\ \beta, i$) and add ($Y \rightarrow \alpha\ X \bullet \beta, i$) to S(*k*).

These steps are repeated until no more states can be added to the set. The set is generally implemented as a queue of states to process (though a given state must appear in one place only), and performing the corresponding operation depending on what kind of state it is. Adapted from[10]

V. THE PARSING PROCESS

When a sentence or a natural language is placed on the interface, the sentence is firstly broken down into words, at this point, stemming takes place-tokenization. After stemming, they are reformed as a sentence again at the instance of the LA which invokes wordnet for the generation of the semantic tree, knowing that the Earley's algorithm will be used as stated above to generate the syntactic tree. Semantic relation and nodes are created mainly by wordnet. When an input is placed into the interface in LA, the LA invokes word net to place the word in its semantic category, since the grammar is context free and not in the Chomsky normal form, CNF. The CNF is given as follows:

If $L(G)$ does not contain ϵ, then G can have a CNF form with productions only of type

(3)

$$A \rightarrow BC$$
$$A \rightarrow a$$

where $A, B, C \in V, \quad \text{and } a \in T$

$$S \rightarrow aB|bA$$
$$A \rightarrow a|aS|bAA$$
$$B \rightarrow b|bS|aBB$$

Due to complexity, the semantic relations that may also exist are not always all presented in the tree. The LA adheres to the above using a data mapping system contained in word net to provide information about the given word which will be transformed to query usable by the database. The entities, attributes etc of the database are all maintained at the level of the database. Finally, the LA returns the NL query earlier presented by the user based on the generated standard tree. The LA generates a corresponding SQL query from the standard tree. An additional attempt in this study is to further fuzzify the generated sql based on the premise that fuzzy logic closes the gap between man and machine. For the purpose of translation, the tree concept as entrenched in data structure is used. For translation processes, each main node (not root) is travervesed. Consider, again, the request for all student used above, word net generates the following transformation for students.

[student::] [student] pupil, learner

The British National corpus presents a ranking for the first 5000 synsets. Consider a user request query on a table called students as represented above. In our application, students is first stemmed into student(singular), then Wordnet attempts to present words related in meaning, in WordNet 2.0

Student produces student, pupil, learner. Results for 'Synonyms, hypernyms and hyponyms are ordered by estimated frequency' search of noun 'students'. Two (2)senses of students were generated and they are as follows: Student search on wordnet produces pupil(a child or young person in school or in the charge of a tutor or instructor) and learner (To gain knowledge, comprehension, or mastery of through experience or study). Thus the semantic set for student is given as {pupil, learner}. A ranking of synsets derived from word frequencies in theB British national corpus synsets have been used[11]. Problem of execution exist in cases where the same word have different semantic nodes.As example, the word pupil can also mean a layer of the eyes which will lead to another structural formation other than that of the student.

VI. DATABASE AND LA

Database attributes are also assigned a node. The complete query is generated after all nodes including the root have been transverse. This means that it is possible to translate the standard tree into sql using the LA. Note that to handle the preferential nature of human user, a column degree is automatically inserted into the result table to ascertain the level of satisfaction of users, this is the knowledge domain. This is used in measuring the satisfaction level of users and can be expanded to build a suitable feedback system. Some few concepts are germane to the semantic structure of a word. They include the following;

A. Phonology:

Phonetics deals with speech sound of words in the context of a relationship between phonemes: small, distinct sound unit of a language. Most NLS system do not operate at this level.

B. Morphology:

These are the processes of marking, stemming and truncation of words. It deals with the meaningful part the word.

C. Lexicology:

This is the study of words itself. This refers to the area of lexicon. Lexicon are of great importance in the deployment of effective IR system.

D. Semantics:

Involves the study of the meaning of the words. This is a more complex level of importance analysis.

The proposed architecture has the LA as the "CPU". The LA, the knowledge tree and the DB forms a complete set for the proposed system. The LA controls all activities involve in the transformation of Natural Language to sql for execution on Relational Databases.

The LA used in this study is an extension of a query agent called ADAM [12] . A general overview of the system is shown below

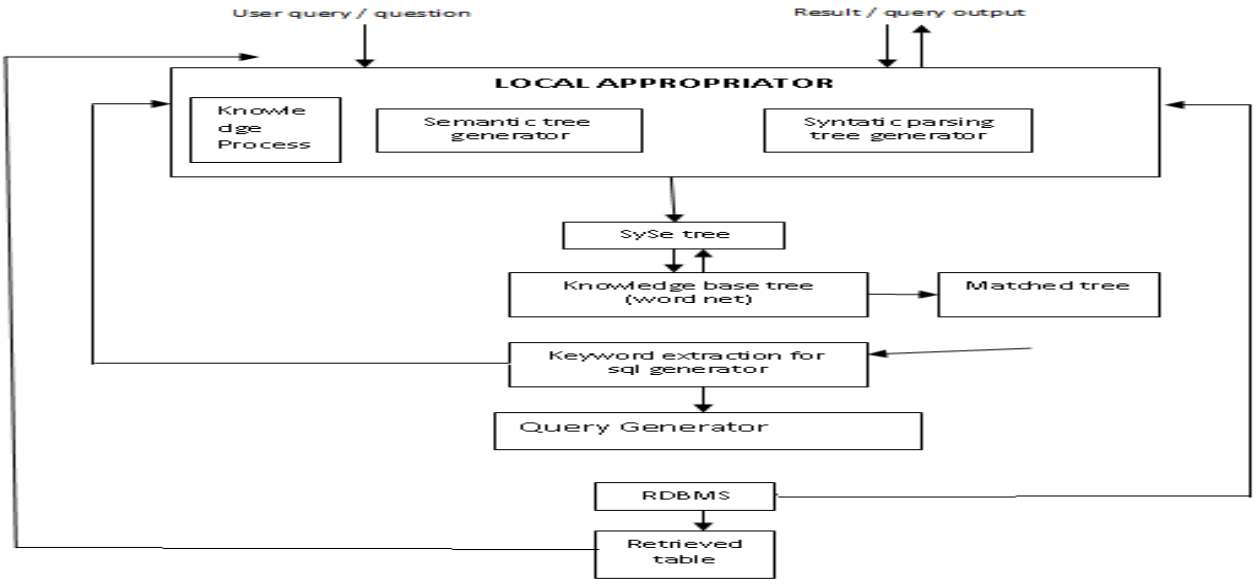

Figure 1: A theoretical view of the parsing system

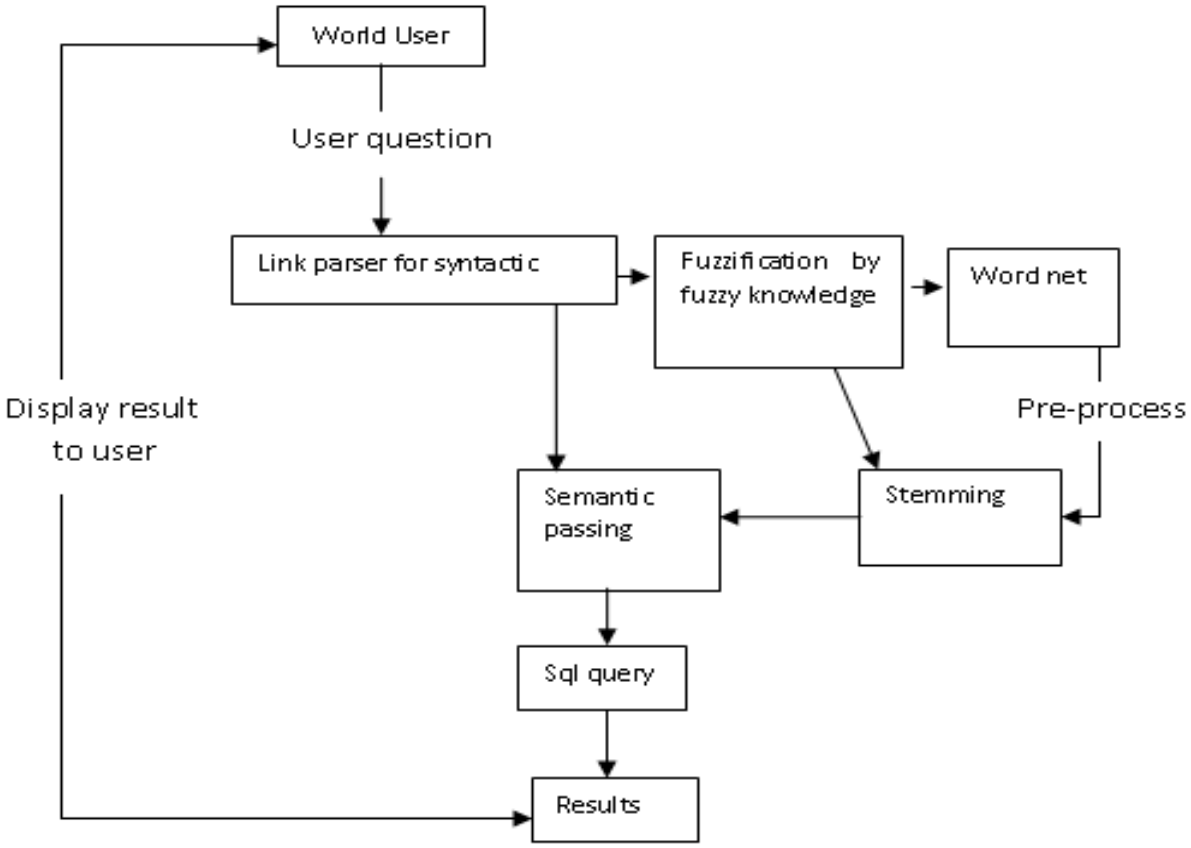

Figure 2. The flow of the parsing structure

VII. SYNTAX BASED SYSTEM

Natural language interfaces has shown that the user is not required to learn any artificial motional language. Since formal queries like sql are difficult to learn and must by a non computer expert. NL has been described as an easy to use process for implementing the desire of the user against a database.

In a syntax based system, the user input/question is parsed syntatically. The question is translated from its language syntatically in this form, from English to SQL, parsing is the process of identifying structure in data [13]. It determines the whether the input data has some pre-determined structure and respond accordingly. Parsing requires the definition of a set of grammars. Gradually, all syntax based system uses the grammar to describe the possibility of a syntactic structure embedded in the users question. These grammars are set of rules which define how the language is structured. Rules are specific platform of data, which appears in the input. Sub rules also exist as those rules with reference to the main rule.

The syntactic system is regularly based on the rules defined in the grammar. It can be summarized as follows:

Grammar: rulers #one or more rule
Rule: productions # can be form in several ways depending on structure of input

The bottom up parser is given as:

Productions: terminal # used to describe complex structure of inputs
Sub rule: terminal

The bottom – up parser are series of syntax encoded in a look-up table. Parsers are transitive as the begin act a pre-defined state and move to other states depending on the next valuable write with which parser works. Tokens can be terminal and non-terminal. Token groupings are determined by the grammar rules.

Transforming a SySe tree to an sql cannot be achieved in a single step. Consider, the universal of Essex NLI which is a multi – stage transformation process[14].

A. Implementation and Result:

The proposed system is implemented using php on an sql server. The processes described above were all implemented in the underline coding. Matching and generation rule are developed within the preprocessor. This rules are created in other to form a dependent relation. That it, match will occur if and only if a relation exist. This is the benchmark that maps the user input into the required SQL. It means that for any given query with term a, b there must be corresponding tables a-like, b-like on the database. In such case, we can have a rule that for any query containing student and department then system list all content of table and department from the database. Our system associate rules to each table content as been implemented in php and ensure that they are well weaken by the pre-processor. Consider the request for list all of students in the department of biology. It is evaluated as figure 5:

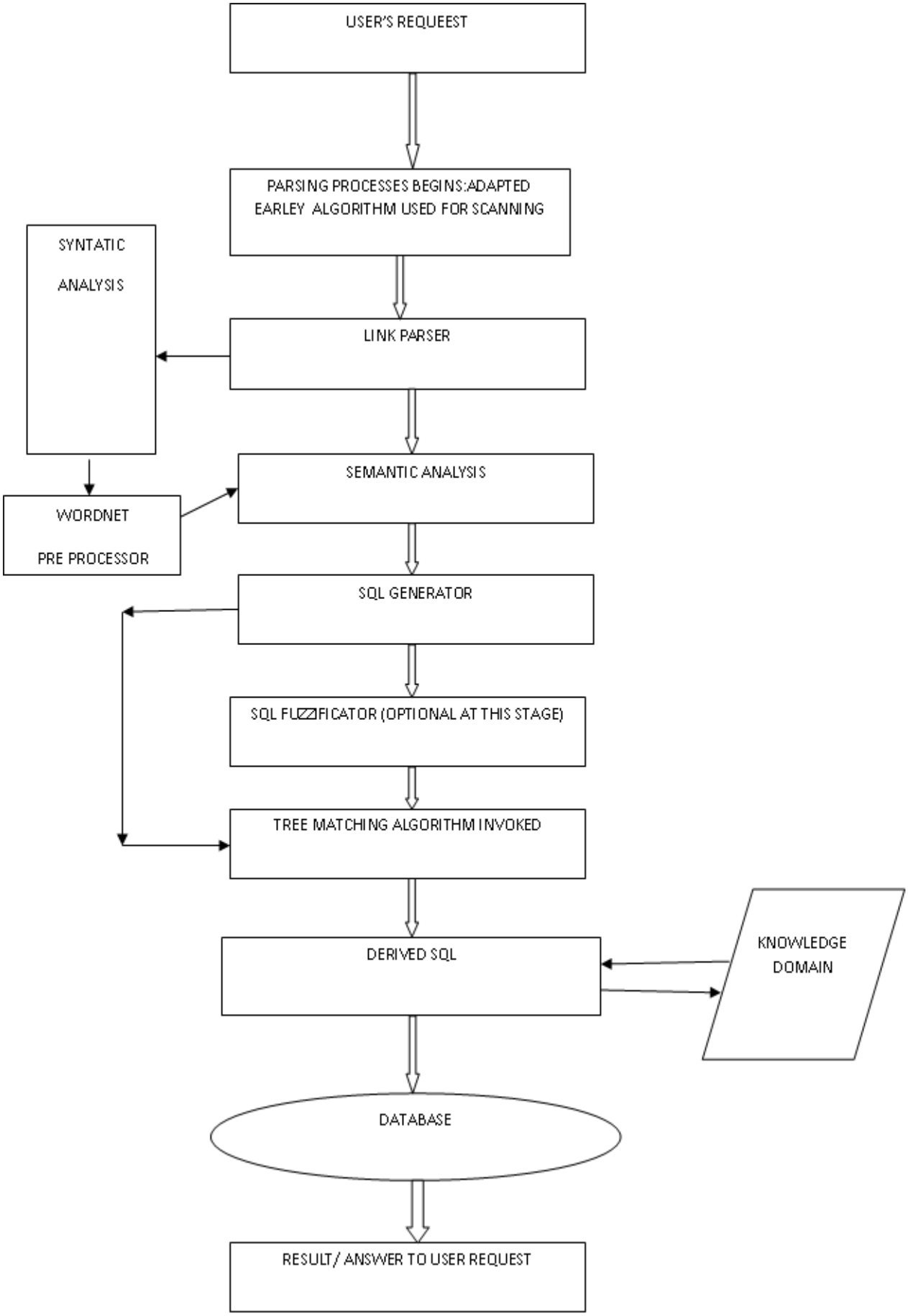

Figure 3. The main Architecture

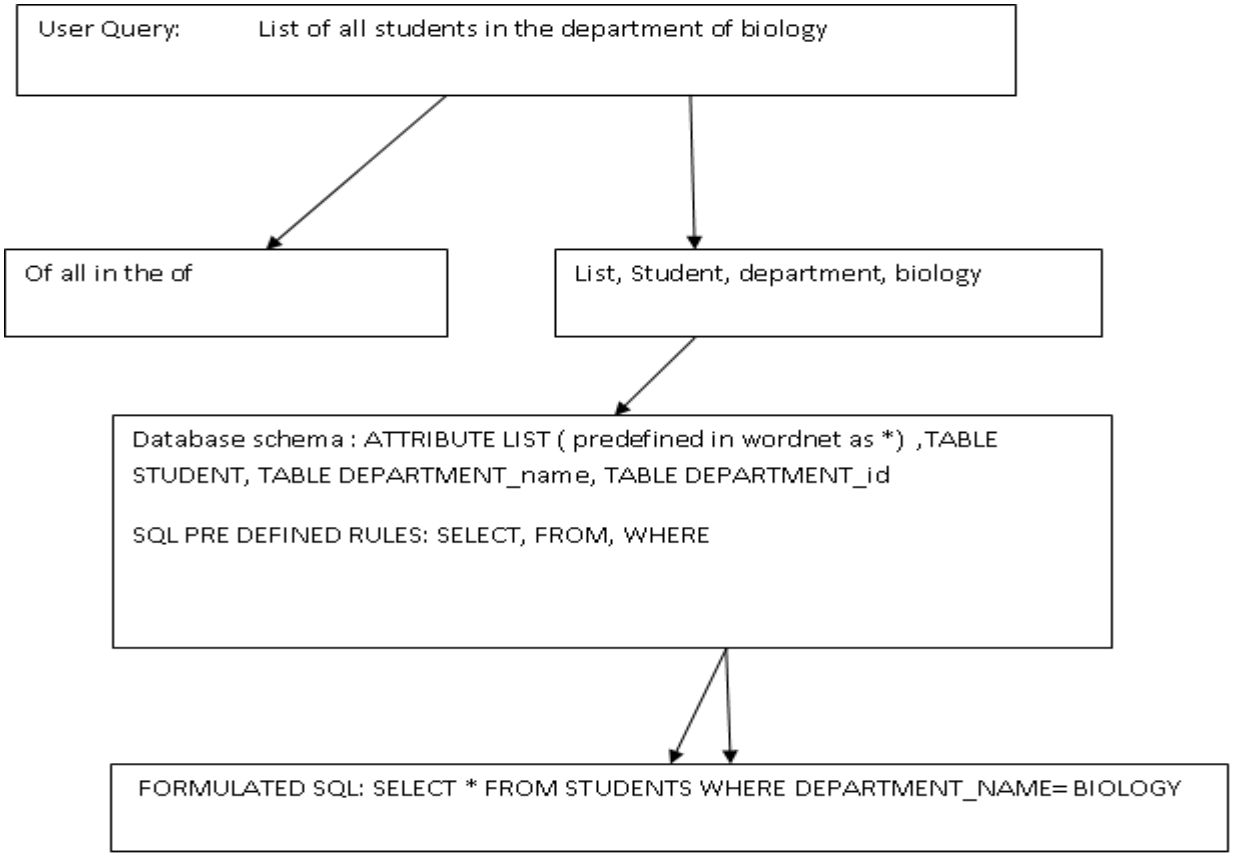

Figure 5. A sample execution flow

VIII. RESULTS AND CONTRIBUTIONS

On implementation, the runtime test of the project over a university database is presented below. It shows that our model is completes its task over a shorter period than others that were tested. We may not be able to give confirm further enhancement at this stage as the model was just tested on a sample database. We are of the opinion that the model will be appropriate for use in large databases.
The success rate in the result received is presented as a chart below:

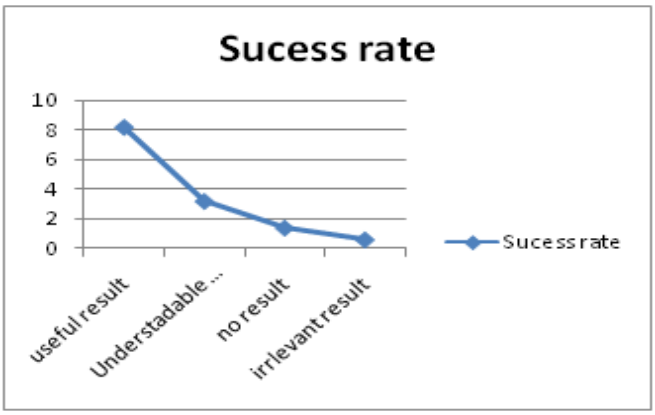

Figure 6. Rate at which result match ordinary sql at several runs

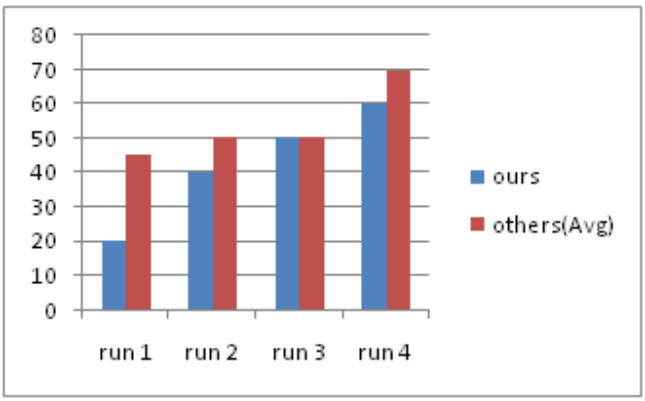

Figure 7. The LA(ours) based query against some commercial NLPs(others).

IX. CONCLUSION AND FUTURE WORK

Initial results showed the prototype implementation to be successful. The system users were able to put their request into the system and the system was able to display expected and required result. The success rate of several training of the system is presented below. Future work will include the enhancement of the fuzzy logic part of the sql translation process. We shall attempt to include some weighted constant into the query. An attempt will be made to understand the possibility of the database itself handle the semantic interpretation. This will make the execution process faster than as it is now.

X. REFERENCES

[1]. M. Zongmain, ‘’Intelligent databases technology and applications’’, IGI publishing, 320 pages, 2007

[2]. W. Dietmar, “Application of sql for infometric data processing”, proceeding of the 33rd conference of the Canadian Association for information science, 2005

[3]. Intelligent databases, available at http;//serachsqlserver.techtarget.com, accessed nov. 2011

[4]. I. Androutsopoulos, G. Ritchie, and P. Thanisch, “Natural language interfaces to databases – an introduction”, Journal of Natural Language Engineering 1 (1) (1995) 29–81.].

[5]. Giordani, Alessandra, .Kapetanios, Epaminondas, Sugumaran, Vijayan, Spiliopoulou and Myra, “Mapping Natural Language into SQL in a NLIDB”, Natural Language andInformationSystems, Lecture Notes in Computer Science,Springer Berlin / Heidelberg, 2008 pp367-371

[6]. M. bordie, Future Intelligent information system: AI and Database Technologies Working Together in Rendering artificial Intelligence and Database Morgan, Kaufmann, san mateo, CA, 1988

[7]. S.Small, T.Strzalkowski, T. Janack, T. Liu, S. Ryan, R. Salkin, N. Shimizu, P. Kantor, D. Kelly, R. Rittman, N. Wacholder, and B. Yamrom. HITIQA: scenario-based question answering. In Proceedings of the Workshop on Pragmatics of Question Answering at HLT-NAACL Boston, MA, 2004. pp. 52-59].

[8]. M.Blazquez, Q. Sheu , “linguistic Hedges on trapezoidal Fuzzy set” a revisal (2001)

[9]. N Nihalani, S. Silakari and M. Motwani , “Design of Intelligent layer for flexible querying in databases”, International Journal of Computer Science and Engineering (IJCSE) Vol 1(2) 2009

[10]. J. Earley "An efficient context-free parsing algorithm", Communications of the ACM **13** (2): 94–102, doi:10.1145/362007.362035. 1970.

[11]. Available at http://www.natcorp.ox.ac.uk/, accessed on Tuesday 1st, November, 2011

[12]. Adam: Student Debt Advisor, Convagent Ltd, Manchester,UK, 2001, Available at:http://www.convagent.com/convagent/adam3.aspx

[13]. D. Klein and D. Manning. “Fast Exact Inference with a Factored Model for Natural Language Parsing”. In Advances in Neural Information Processing Systems 15 (NIPS 2002), Cambridge, MA: MIT Press, 2003 pp. 3-10.

[14]. K.church, R. Murer,“Introduction to the special issue on computational linguistic using large corpus”, computational linguistic, 19 (1), 1993; pp. 1 -24).

