Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 3, March 2015, pg.485 – 494

RESEARCH ARTICLE

Natural Language to Relational Query by Using Parsing Compiler

Radhika Mali, Pooja Toge, Prerana Gargote, Supriya Mali, Prof. Sanchika Bajpai

Department of Computer Science. University of Pune MH (India)

JSPM's BSIOTR. College of Engineering, Pune, Maharashtra, India

[1] radhikamali1993@gmail.com, [2] poojatoge26@gmail.com, [3] gargote.prerana9293@gmail.com, [4] sups.mali09@gmail.com

Abstract- ***Translating a natural language question into a database query suffers from the translation ambiguity problem, which has not received significant attention in this field. To deal with translation ambiguity, we suggest an ambiguity resolution method based on the proposed database semantics. In natural language database interfaces (NLDBI), manual construction of translation knowledge normally undermines domain portability because of its expensive human intervention. It introduces some classical NLDBI products and their applications and proposes the architecture of a new NLDBI system including its probabilistic context free grammar, the inside and outside probabilities which can be used to construct the parse tree, an algorithm to calculate the probabilities, and the usage of dependency structures and verb sub categorization in analyzing the parse tree. To perform extraction our system provide automated query generation component so that random users do not have to learn the query language. The two important aspects of an information system are efficiency and quality of extraction result, also our system reduces the processing time by 89% as compared to a traditional pipeline approach. Our experiments also describe that our approach archives purity extraction results.***

Index term- Information storage and retrieval, Natural language processing, Query generation, Query language, Text mining

I. INTRODUCTION

The SQL translator acts as a frontend to any information system and provides a natural language interface (NLI) to the end user. The translator is designed to understand everyday English requests and invoke appropriate database reporting tool for a valid query. Each valid query is mapped onto an appropriate SQL command through a neural net based converter/translator[1]

Information Extraction (IE) is a process for the extraction of a particular kind of relationships of interest from a document collection. The purpose of information extraction (IE) is to find wanted pieces of information in natural language texts and store them in a form that is suitable for automatic querying and processing. IE requires a predefined output representation (target structure) and only searches for facts that place this representation.[2] Simple target structures define just a number of slots. Each slot is filled with a string extracted from a text, e.g. a name or a date (slot filler). In order to perform relationship extraction a typical IE setting involves a pipeline of text processing modules. The field of information (IE) search to develop methods for fetching structured information from natural language text. Examples of structured information from natural language are the extraction of entities and relationships between entities. IE is usually deployed as a pipeline of special-purpose programs, which include**:** sentence splitting to identify sentences from a paragraph of text, Tokenization which identifies word tokens from sentences, Named entity recognition used to identifies mentions of entity types of interest**.** To utilize lexical, syntactic and semantic features, Syntactic parsing identifies grammatical structures of sentences; Pattern matching obtains relationships based on a set of extraction patterns. Extraction patterns are typically obtained through manually written patterns compiled by experts or automatically generated patterns based on training data. Different kinds of parsers, which include shallow and deep parsers, can be utilized in the pipeline.[1]

II. RELATED WORK

The main focus has been on improving the accuracy & runtime of information extraction[IE]. NLDBi's can be classified according to the approach employed in deriving an SQL query that retrieves the answer of a given natural language question to a database. Example, system uses this approach, reducing the problem finding a semantic interpretation of ambiguous phrases to a graph matching problem.

III. SYSTEM ARCHITECTURE

In most of the typical NLDBi systems the natural language statement is converted into an internal representation based on the syntactic and semantic knowledge of the natural language. This representation is then converted into queries using a representation converter. Before an natural language query is translated to an equivalent query in technical language like SQL it has to through a lot of steps.

- Tokenization
- Grammar Checking
- Query Generation
- Data Collection

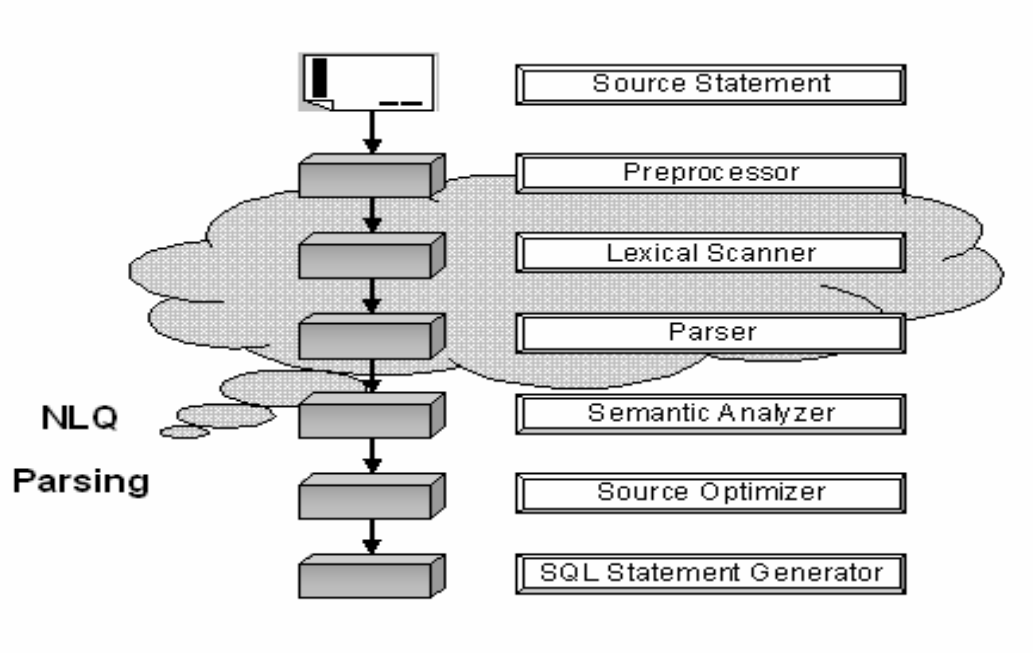

Fig. 1 Architecture of NLDBi System

In this system architecture of natural language database interface developed is given in Fig. 1, which depicts the layout of the processes included in converting NL query into a syntactical SQL query to be fired on the RDBMS.

The extraction patterns over parse trees can be expressed in our proposed parse tree query language. In the extraction phase, the PTQL query evaluator takes a PTQL query and transforms it into keyword-based queries and SQL queries, which are evaluated by the underlying RDBMS and information retrieval (IR) engine. To speed up query evaluation, the index builder creates an inverted index for the indexing of sentences according to words and the corresponding entity types[1].

IV. IMPLEMENTATION

The experimental work is to design an interface for generating queries from natural language statements/questions. It also consists of designing a parser for the natural language statements, which will parse the input statement, generate the query and fire it on the database. The experimental work will understand the exact meaning the end user wants to go for, generate a what- type sentence and then convert it into a query and handover it to the interface. The interface further processes the query and searches for the database. The database gives the result to the system which is displayed to the user[4].

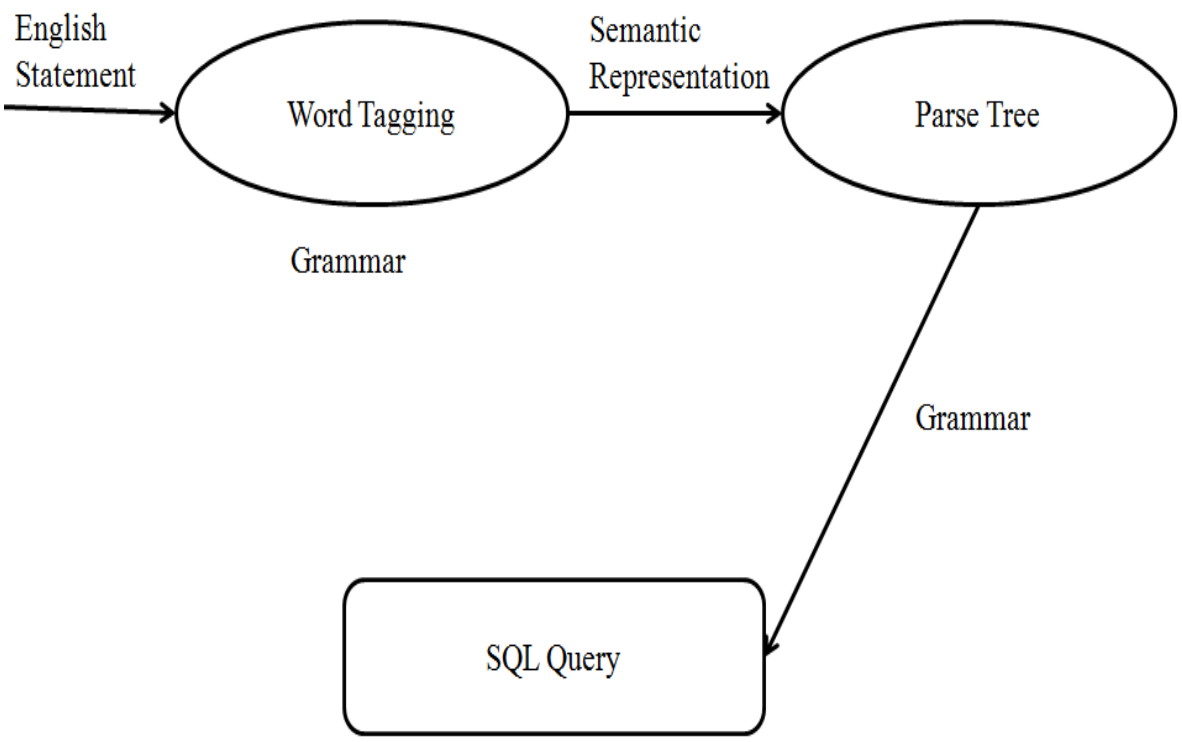

Fig.2. Generation of SQL Query from Natural Language

Fig. 2 depicts the processing of English input statement to generate SQL query. The entire process involves tagging of input statement, apply grammar and semantic representation to generate parse tree, analyze the parse tree using grammar and translating the leaves of the tree to generate corresponding SQL query. The SQL translator generates query in SQL. Using grammar the parse tree is obtained from the input statement. The leaves of the parse tree are translated to corresponding SQL[3].

The following modules were developed.

- An Interface: It allows the user to enter the query in NL, interact with the system during ambiguities and display the query results.
- Parsing: Derives the Semantics of the statement given by the user and parses it into its internal representation, to convert NL input statement into what- type question for selection of data.
- Query Generation: It generates a query against the user statement in SQL and passes on to the database.

The algorithm designed is as given below:

Algorithm 1 Generation of parse tree/s from NL statement using grammar

```
1.  Read input statement S
2.  For each word Wi from S do
3.      If(Wi←Grammar G) then
4.          Add Wi to symbol table ST
5.      End if
6.  End for
7.  For each Wi from ST do
8.      Add Wi to parse terr/s T for What-type question/s
9.  End for
10. Display What-Type question/s Q
11. Read input Q
12. For each Wi from Q do
13.    If(Wi←G) then
14.        Add Wi to parse tree for SQL-query
15.    End if
16. End for
17. Display SQL-query
```


V. EXPERIMENTAL RESULT

The system implemented was tested for variety of NL statement under various categories and the result obtained ware satisfactory under the known constraints.

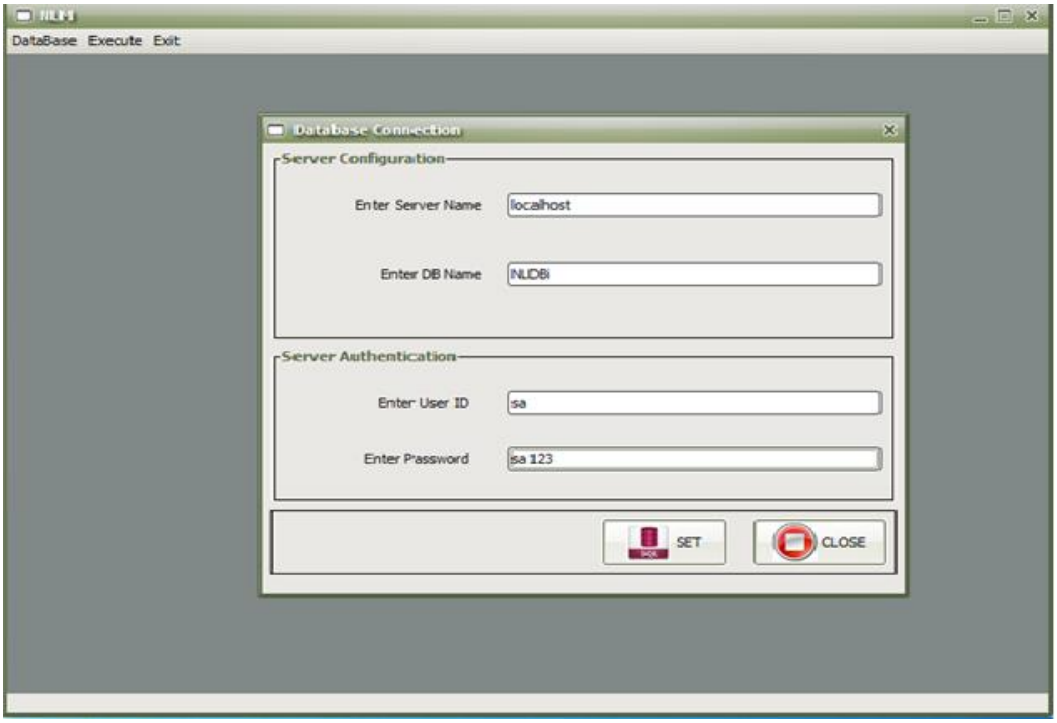

Fig: set database connection

The fig. shows the set connection to database. In the connection two categories are included: server configuration and server authentication .sever configuration ask the server name and DB name.

And server authentication is user ID and the password to be used to create the connection.

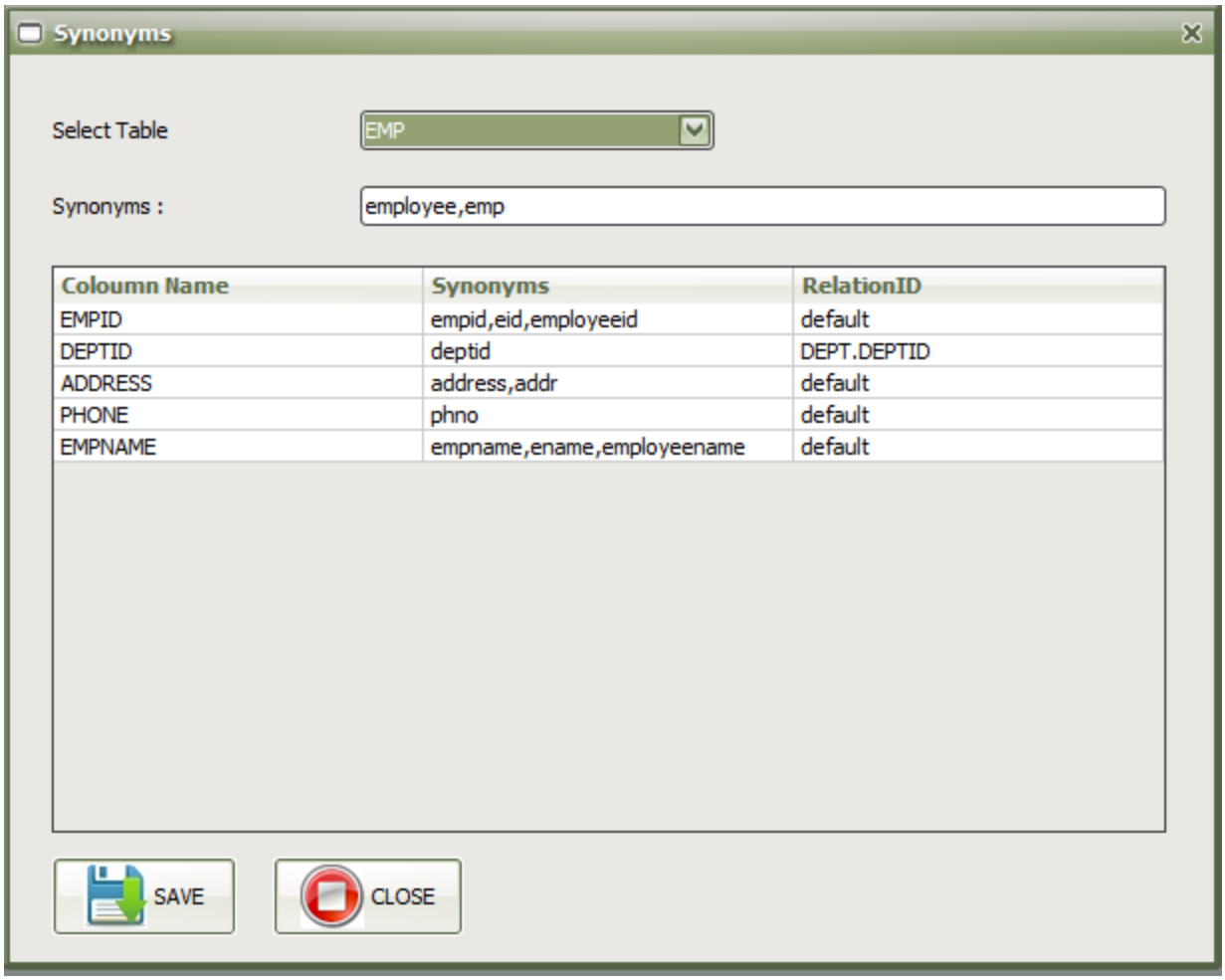

Fig 2: set dictionary using synonyms

Fig. Database create the dictionary, in dictionary different tables are created, using different synonyms.

In table column name, synonyms and related ID are used.

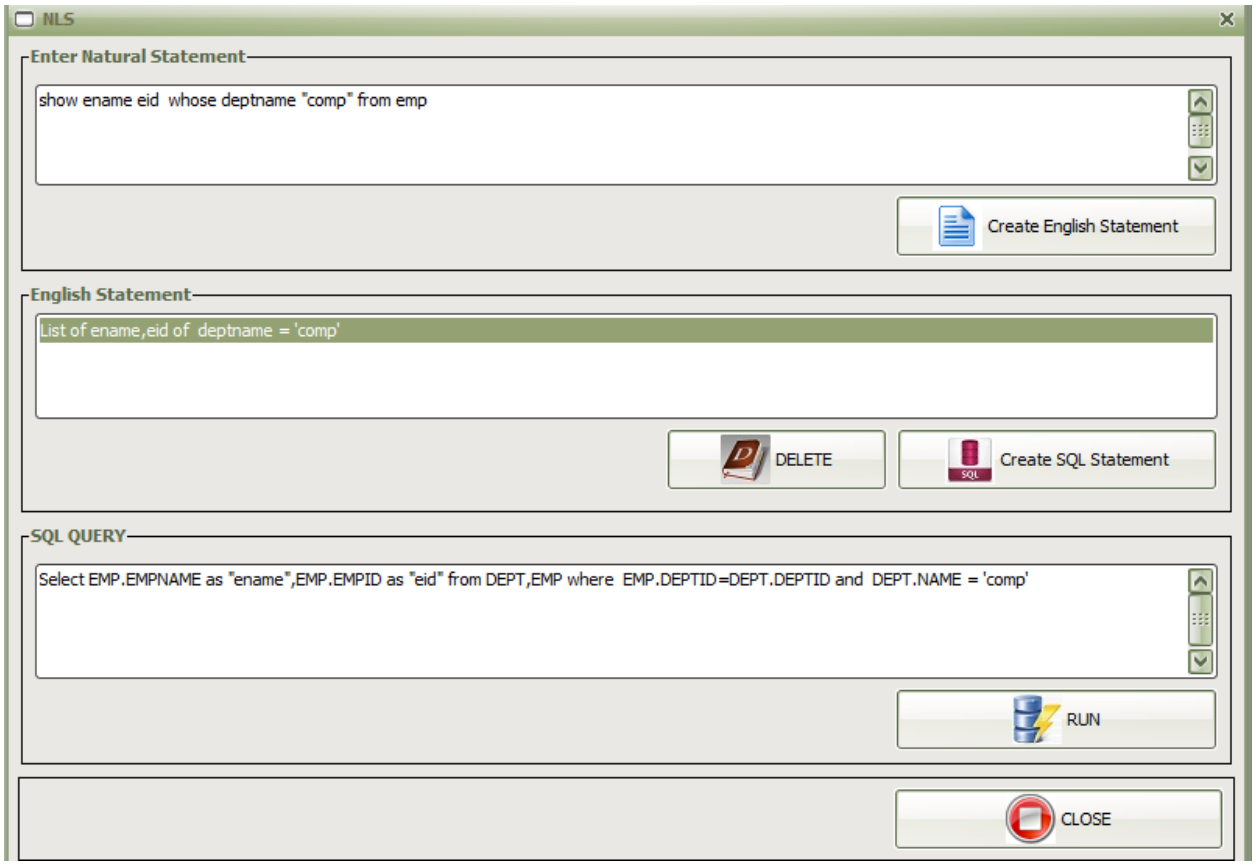

Fig 3: NLDBI system

The fig shows the typical categories of generating ambiguous parse tree.

1. First enter the natural statement in this system, and then these can create the English grammar. Suppose the natural statement are incorrect these system can show the message about wrong statement they cannot be create the English statement.
2. After the English statement creates SQL statement means these SQL query generated.
3. In English statement no of statement are shows these statement are select and u can delete the statement help of the delete button.

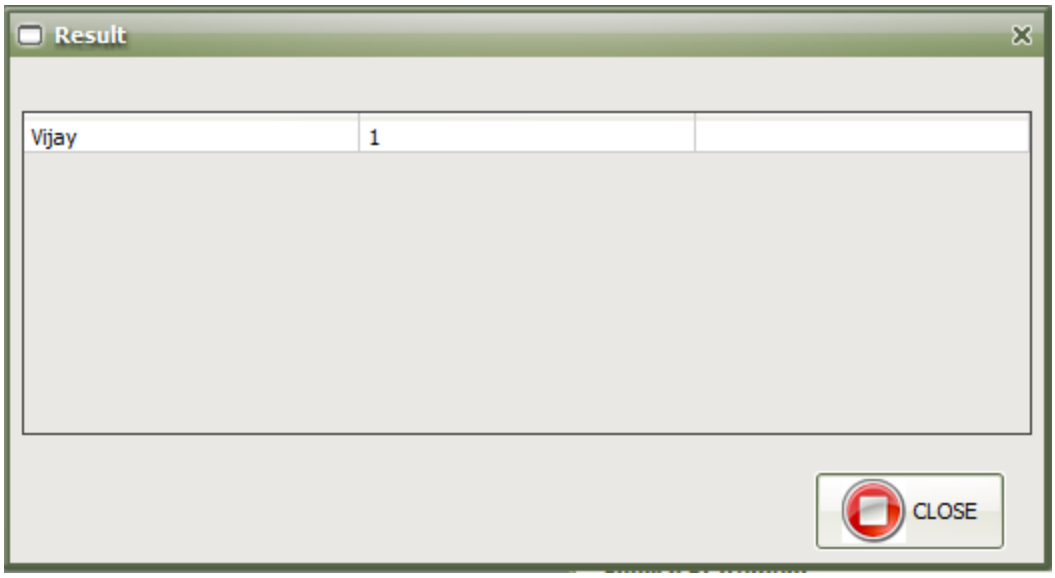

Fig. 4 Final output as a result.

Above fig4 shows the final output of the natural statement. When SQL query is fired on database then we got final result as a information which we want.

VI. CONCLUSION

Information extraction system is to extract query language. Natural query language transfer to PTQL(Parse Tree Query Language) tree. This leads to the unnecessary reprocessing of the entire text when the extraction goal is modified or improved, which can be computationally exclusive and time consuming one. To reduce this unnecessary reprocessing time the intermediate process data is stored in database system. The database is in the form of parse tree. To extract information from parse tree the extraction goal written by the user in natural language text is converted into PTQL and then extraction is performed on text corpus. This increment extraction approach reduces time 89% as compared to performing extraction by first processing each sentence one at a time with linguistic parse and then other components.

REFERENCES

[1] http://piserjournal.org/2014/09/natural-statement-to-relational-query-by-using-parsingcompiler/

[2] D. Ferrucci and A. Lally, "UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment," Natural Language Eng., vol. 10, nos. 3/4, pp. 327- 348, 2004.

[3] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications," Proc. 40th Ann. Meeting of the ACL, 2002.

[4] F.Chen, A. Doan, J. Yang, and R. Ramakrishnan, "Efficient Information Extraction over Evolving Text Data," Proc IEEE 24th Int'l Conf. Data Eng. (ICDE '08), pp. 943-952, 2008.

[5] S.Sarawagi, "Information Extraction," Foundations and Trends in Databases, vol.1, no. 3, pp. 261-377, 2008.

[6] M.J. Cafarella and O. Etzioni, "A Search Engine for Natural Language Applications," Proc. 14th Int'l Conf. World Wide Web(WWW '05), 2005.

[7] "XQuery 1.0: An XML Query Language," http://www.w3.org/ XML/Query, June 2001.

[8] C.Lai, "A Formal Framework for Linguistic Tree Query," Master's thesis, Dept. of Computer Science and Software Eng., Univ. of Melbourne, 2005.

[9] E.Agichtein and L. Gravano, "Querying Text Databases for Efficient Information Extraction," Proc. Int'l Conf. Data Eng. (ICDE), pp. 113-124, 2003.

[10] M.Huang, S. Ding, H. Wang, and X. Zhu, "Mining Physical Protein-Protein Interactions by Exploiting Abundant Features," Proc. Second BioCreative Challenge, pp. 237-245, 2007.

