Automated Key Generation of Incremental Information Extraction using Relational Database System & PTQL

Gayatri Naik, Singhania University, Rajasthan, India.
Dr. Anoop Sharma, Singhania University, Rajasthan, India.

Abstract *Information extraction systems are traditionally implemented as a pipeline of special-purpose processing modules targeting the extraction of a particular kind of information. A major drawback of such an approach is that whenever a new extraction goal emerges or a module is improved, extraction has to be reapplied from scratch to the entire text corpus even though only a small part of the corpus might be affected. In this project, we describe a novel approach for information extraction in which extraction needs are expressed in the form of database queries, which are evaluated and optimized by database systems. Using database queries for information extraction enables generic extraction and minimizes reprocessing of data by performing incremental extraction to identify which part of the data is affected by the change of components or goals. Furthermore, our approach provides automated query generation components so that casual users do not have to learn the query language in order to perform extraction. To demonstrate the feasibility of our incremental extraction approach, we performed experiments to highlight two important aspects of an information extraction system: efficiency and quality of extraction results.*

Keywords — *Text mining, query languages, information storage and retrieval.*

I. INTRODUCTION

Text mining is inter disciplinary field which draws on information retrieval, data mining, machine learning, statistics, and computational linguistics. As most information (common estimates say over 80%) is currently stored as text, text mining is believed to have a high commercial potential value. Increasing interest is being paid to multilingual

data mining: the ability to gain information across languages and cluster similar items from different linguistic sources according to their meaning. Text mining, sometimes alternately referred to as text data mining, roughly equivalent to text analytics, refers to the process of deriving high-quality information from text. High-quality information is typically derived through the divining of patterns and trends through means such as statistical pattern learning. Text mining involves the application of techniques from areas such as information retrieval, natural language processing, information extraction and data mining. These various stages of a text-mining process can be combined together into a single workflow.

1.1 Information Extraction

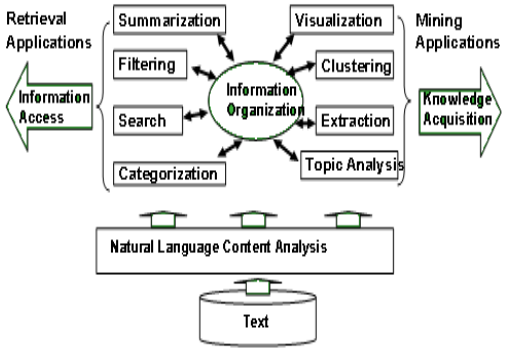

Fig1: Conceptual Framework of Text Information System

Fig 1 shows the Conceptual Framework of Text Information System consist the Retrieval applications & Mining applications. Natural language content analysis access text and send it to information organization, which perform operations like Topic analysis, Extraction, Clustering, Visualization, Summarization, Filtering, Search, Categorization etc. Information organization provide information access & knowledge acquisition.

1.2 Basic Generality Techniques

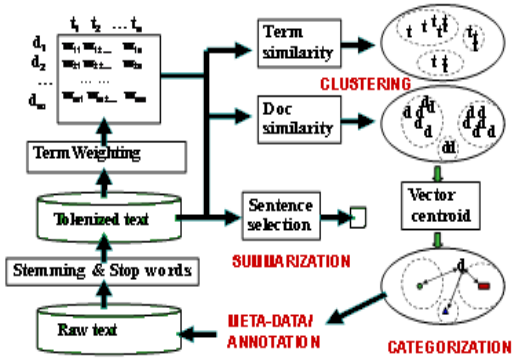

Fig 2: Basic Generality Techniques

IE has been an active research area that seeks techniques to uncover information from a large collection of text. Examples of common IE tasks include the identification of entities, extraction of relationships between entities and extraction of entity attributes from text. Information extraction (IE) is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents. In most of the cases this activity concerns processing human language texts by means of natural language processing (NLP).Information extraction systems are traditionally implemented as a pipeline of special-purpose processing modules targeting the extraction of a particular kind of information. A major drawback of such an approach is that whenever a new extraction goal emerges or a module is improved extraction has to be reapplied from scratch to the entire text corpus even though only a small part of the corpus might be affected.

1.3 Workflow

A typical IE setting involves a pipeline of text processing modules in order to perform relationship extraction. These include:

1. Sentence splitting: identifies sentences from a paragraph of text.
2. Tokenization: identifies word tokens from sentences.
3. Named entity recognition: identifies mentions of entity types of interest.
4. Syntactic parsing: identifies grammatical structures of sentences.
5. Pattern matching: obtains relationships based on a set of extraction patterns that utilize lexical, syntactic, and semantic features of text as input to perform relationship extraction.

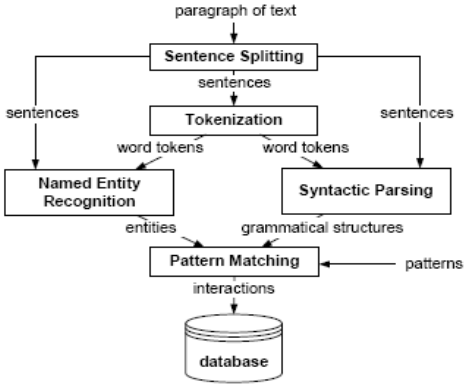

Fig.3. A workflow of text processing modules that takes a paragraph

Fig.3.illustrates a typical text processing workflow in order to perform extraction of relationships. 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. In our work, the Link Grammar parser is utilized as part of our extraction approach. 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. In our work, the Link Grammar parser is utilized as part of our extraction approach.

II. LINK GRAMMAR

2.1 Introduction

Link grammar (LG) is a theory of syntax by Davy Temperley and Daniel Sleator which builds relations between pairs of words, rather than constructing constituents in a tree-like hierarchy. There are two basic parameters: directionality and distance. Link grammar is similar to dependency grammar, but dependency grammar includes a head-dependent relationship, as well as lacking directionality in the relations between words. The Link Grammar parser is a dependency parser based on the Link Grammar theory. Link Grammar consists of a set of words and linking requirements between words. A sentence of the language is defined as a sequence of words such that the links connecting the words satisfy the following properties:

1. The links do not cross
2. The words form a connected graph.
3. Links satisfy the linking requirements of each word in the sentence.

2.2 Linkage

The output of the parser, called a linkage, shows the dependencies between pairs of words in the sentence. For example "RAD53, which activates DNA damage, positively regulates the DBF4 protein."

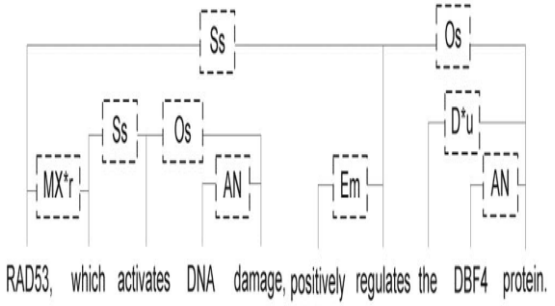

Fig4. Linkage of the sentence "RAD53, which activates DNA damage, positively regulates the DBF4 protein."

III. SYSTEM APPROACH

Our approach composed basic two phases:

- Initial phase: for processing of text and
- Extraction phase: for using database queries to perform extraction.

The Text Processor in the initial phase is responsible for corpus processing and storage of the processed information in the Parse Tree Database (PTDB). 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. Fig illustrates the system architecture of our approach.

Our approach provides two modes of generating PTQL queries for the purpose of information extraction:

1. Training set driven query generation.
2. Pseudo-relevance feedback driven query generation.

3.1 System Architecture

To generate a set of patterns for information extraction using the training set driven approach, the pattern generator first automatically annotates an unlabelled The framework includes the parse tree database for storing intermediate processed information and the query evaluator for the evaluation of PTQL queries through filtering and translation to SQL queries. Document collection with information drawn from a problem-specific database. This step necessitates a method for precise recognition and normalization of protein mentions.

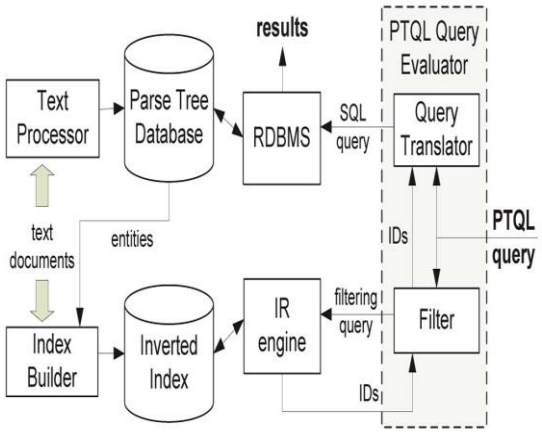

Fig5: System architecture of the PTQL framework

From this labeled data, initial phrases referring to interactions are extracted. These phrases are then refined to compute consensus patterns and the resulting PTQL queries are generated by the query generator. However, training data are not always readily available for certain relationships due to the inherent cost of creating a training corpus. In that regards, our approach provides the pseudo relevance feedback driven approach that takes keyword based queries, and the PTQL query generator then finds common grammatical patterns among the top-k retrieved sentences to generate PTQL queries.

3.2 Parse Tree Database

We propose a new paradigm for information extraction that utilizes database management systems as an essential component of our extraction framework. Database management systems become a logical framework of choice that can serve such dynamic extraction needs over file-based storage systems. As illustrated in Figure 1, text processing components such as named entity recognizers and syntactic parsers are deployed for the entire collection. The intermediate output of the processing modules is stored in a relational database known as the parse tree database. Extraction then becomes a matter of issuing database queries in the form of parse tree query language (PTQL). In the event of a change of extraction goals or a module update, the

responsible module is deployed for the entire text corpus and the processed data is populated into the parse tree database with the previously processed data. Incremental extraction is performed so that database queries are issued to identify sentences with newly recognized mentions. Once the affected sentences are identified, extraction can then be performed only on such sentences rather than the entire corpus. By storing the processed data, our approach avoids the need to reprocess the entire collection of text unlike the file-based pipeline approaches. Avoiding reprocessing of data is particularly important for extraction. We highlight the technical contributions of the architecture proposed Novel Database-Centric Framework for Information Extraction. Unlike traditional approaches for IE, our approach is to store intermediate text processing output in a specialized database called the parse tree database. Extraction is formulated as queries so that it is no longer necessary to write and run special-purpose programs for each specific extraction goal. Our approach minimizes the need of reprocessing the entire collection of text in the presence of new extraction goals and deployment of improved processing components.

3.3 Information Extraction Frame Phases

Our information extraction framework is composed of two phases:

1.1 Initial Phase

We perform a one-time parse, entity recognition and tagging (identifying individual entries as belonging to a class of interest) on the whole corpus based on current knowledge. The generated syntactic parse trees and semantic entity tagging of the processed text is stored in a parse tree database (PTDB).

2. Extraction Phase:

Extracting particular kinds of relations can be done by issuing an appropriate query to the PTDB. To express extraction patterns, we designed and implemented a query language called parse tree query language (PTQL) that is suitable for generic extraction. Our system not only allows a user to issue PTQL queries for extraction, but it can also automatically generate queries from training data or user keyword-based queries.

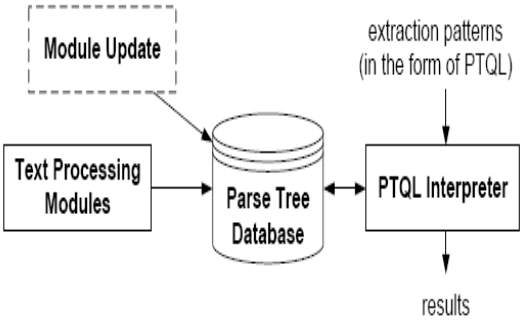

Fig6: Parse tree database use for extraction & update of modules

A fundamental design criterion for the query language is the ability of expressing linguistic patterns based on constituent trees. Standard XML query languages such as XPath and XQuery seem to be the ideal candidates for querying parse trees. An additional design criterion for the query language is the ability to express linguistic patterns based on dependency grammar, such as Link Grammar. Links and link types can be useful in linguistic patterns, such as the type MXsr connects a relative pronoun to its corresponding noun. However, languages such as XQuery and LPath can only express ancestor descendant and sibling relations between nodes. One of the novel features of our proposed query language PTQL is the ability to express links and link types between pairs of nodes, so that PTQL can be used to express linguistic patterns based on constituent trees and links, as well as link types. We propose a high level extraction query language called PTQL. PTQL is an extension of the linguistic query language LPath that allows queries to be performed not only on the constituent trees but also the syntactic links between words on linkages

A PTQL query is made up of four components:

1. Tree patterns,
2. Link conditions,
3. Proximity conditions, and
4. Return expression.

1. Tree pattern: describes the hierarchical structure and the horizontal order between the nodes of the parse tree.
2. Link condition: describes the linking requirements between nodes.
3. Proximity condition: is to find words that are within a specified number of words.
4. Return expression: defines what to return. We start with the basic element of PTQL queries called node expressions.

IV. QUERY EVALUATION

Our approach for the evaluation of PTQL queries involves the use of IR engine as well as RDBMS. The role of the IR engine in query is to select sentences based on the lexical features defined in PTQL queries, and only the subset of sentences retrieved by the IR engine are considered for the evaluation of the conditions specified in the PTQL queries by RDBMS. We summarize the process of the evaluation of PTQL queries as follows.

4.1. Translate the PTQL query into a filtering query.
4.2. Use the filtering query to retrieve relevant documents D and corresponding sentences S from the inverted index.
4.3. Translate the PTQL query into an SQL query and instantiate query with document id 2 D and sentence id s 2 S.
4.4. Query PTDB using the SQL query generated in Step 3.
4.5. Return the results of the SQL query as the results of the PTQL query.

V. QUERY GENERATION

An important aspect of an IE system is its ability to extract high-quality results. In this section, we demonstrate the feasibility of our approach by first describing two approaches in generating PTQL queries automatically:

1. Training set driven query generation
2. Pseudo-relevance feedback driven query generation.

The first query generation approach takes advantage of manually annotated data to generate PTQL queries for the extraction of protein-protein interactions. As training data are not always available, we introduce the latter approach that identifies frequent linguistic structures to generate PTQL queries without the use of training data. We illustrate our approach with an application of protein-protein interaction extraction using a set of syntactic patterns that are expressed in PTQL queries. To generate a set of patterns for information extraction, the annotator component is applied to automatically annotate an unlabeled document collection with information drawn from a problem-specific database. This step necessitates a method for precise recognition and normalization of protein mentions. From this labeled data, the pattern generator identifies relevant phrases referring to interactions in order to generate patterns. These initial patterns are then used to compute consensus patterns through the pattern generalization component for protein-protein interactions (PPIs). PTQL queries are then formed by the query generator to perform extraction from the parse tree database.

VI. RESULTS

We showed that we can retrieve the document file names based on the user query and the query searches based on the phrase based search which retrieves semantic data. We can get progressive corpus without extraction from scratch by storing the intermediate results in the table and retrieve the document by using PTQL query. We first illustrate the performance of our approach in terms of query evaluation and the time savings achieved through incremental extraction. We evaluate the extraction performance for our two approaches

1. Automatic training set driven query generation & data processing. Without PTQL.
2. Automatic Query generation by using user keyword based query generation & data processing with PTQL.

➢ **Input Data Sample1:Medical Data**

Your tonsils and adenoids are part of your lymphatic system. Your tonsils are in the back of your throat. Your adenoids are higher up, behind your nose. Both help protect you from infection by trapping germs coming in through your mouth and nose. Treatments include surgery, radiation therapy, photodynamic therapy (PDT), and biologic therapy.

A. Document parse tree

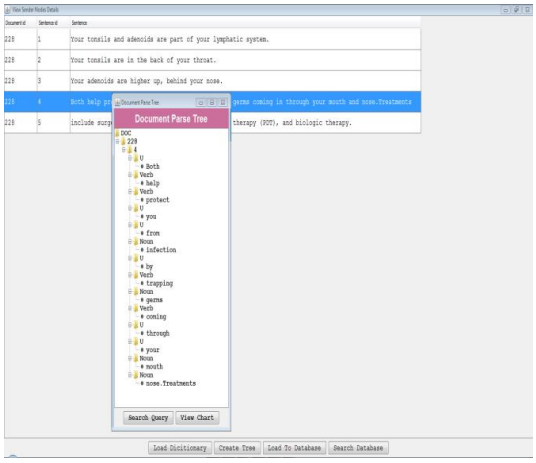

Fig 7: Snapshot of Document parse tree without PTQL

B. Search Query

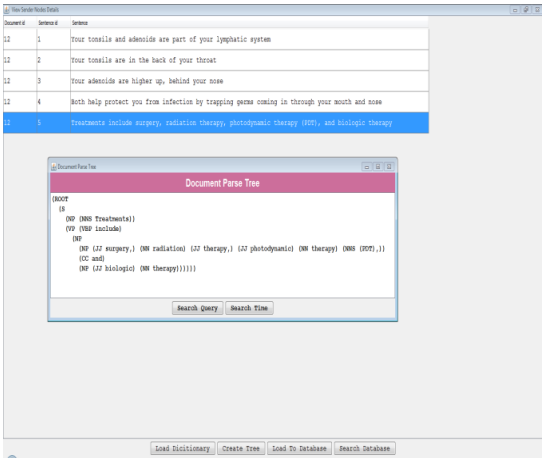

Fig 8: Snapshot of Document parse tree with PTQL

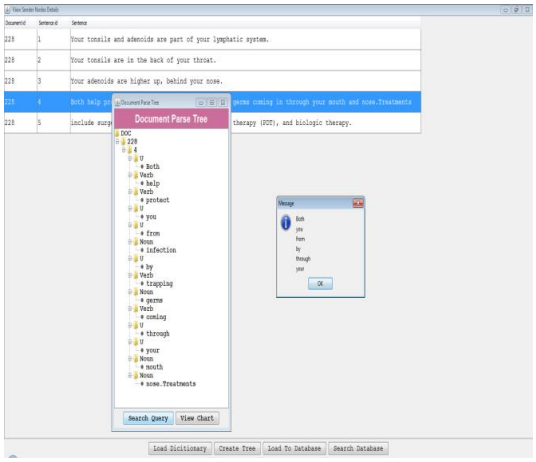

Fig 9: Snapshot of Search Query without PTQL.

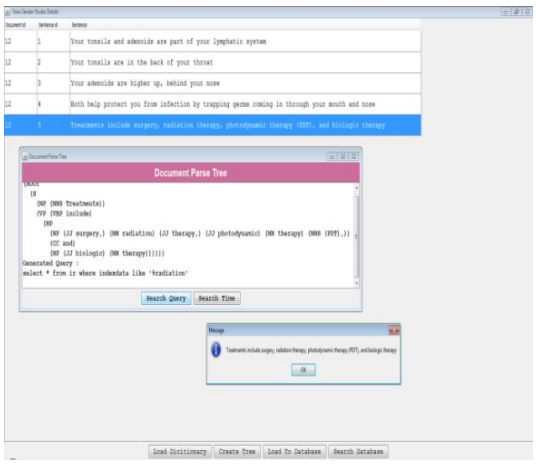

Fig 10: Snapshot of Search Query with PTQL

C. Time Performance Chart

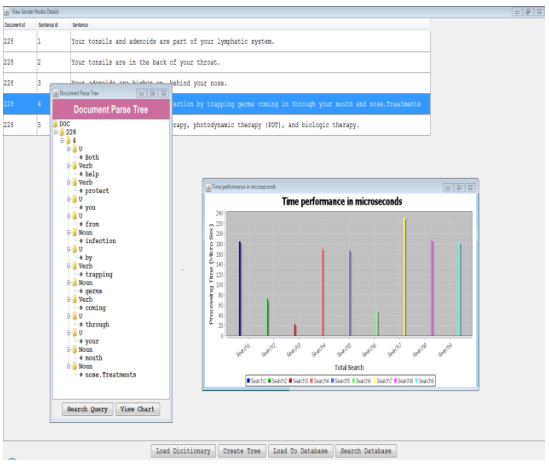

Fig11: Snapshot of Time performance of search query without PTQL.

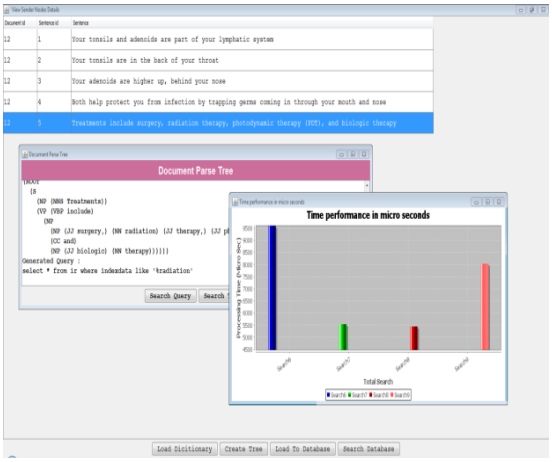

Fig12: Snapshot of Time performance of search query with PTQL

- **Input Data Sample 2: Biomedical Research Data**

Rad53 is a protein kinase required for cell-cycle arrest in response to DNA damage. Rad53 controls the S-phase checkpoint as well as G1 and G2 DNA damage checkpoints. It prevents entry into anaphase and mitotic exit. After DNA damage via regulation of the Polo kinase Cdc5. Rad53 also seems to be involved in the phosphorylation of Rph1.A Zink finger protein that acts as a damage-responsive repressor of the photolyase gene Phr1 in saccharomyces cerevisiae. The 2C type phosphatase, Ptc2, which is required for DNA check point inactivation after double-strand breaks, appears to dephosphorylate Rad53.

A. Document parse tree

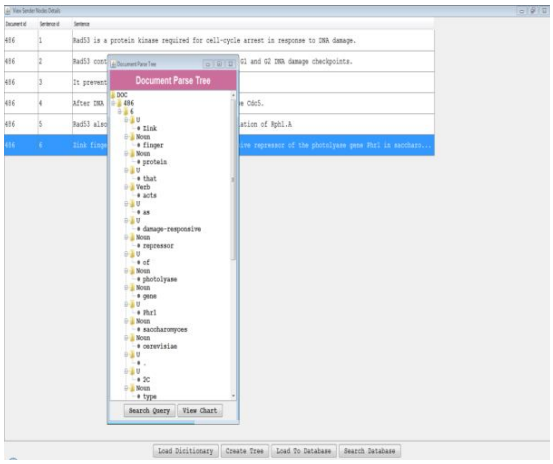

Fig13: Snapshot of Document parse tree without PTQL.

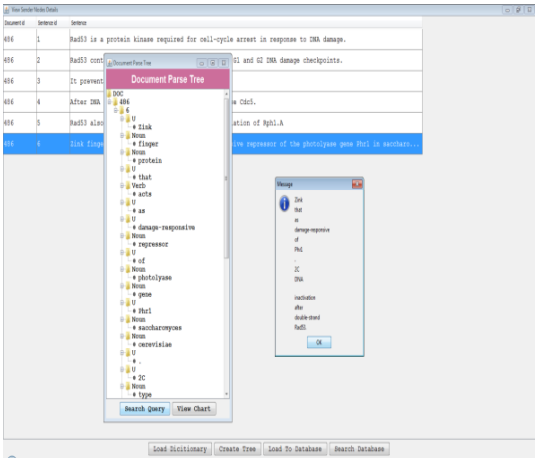

Fig 15: Snapshot of Search Query without PTQL

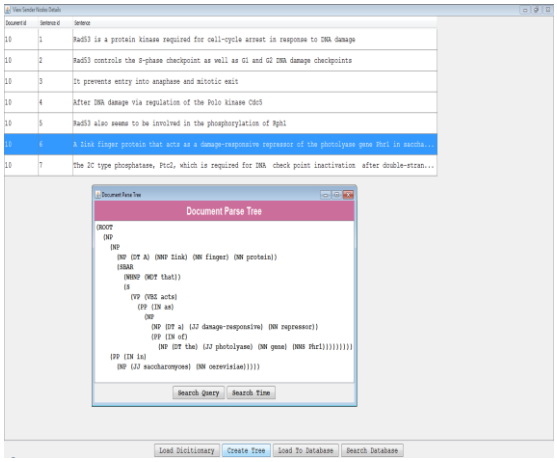

Fig 14: Snapshot of Document parse tree with PTQL.

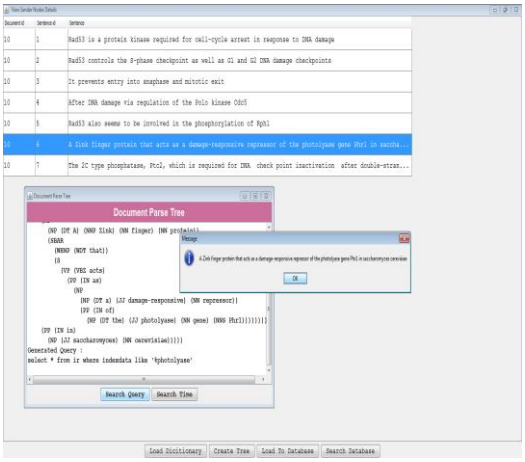

Fig 16: Snapshot of Search Query with PTQL.

B. Search Query

C. Time series Chart

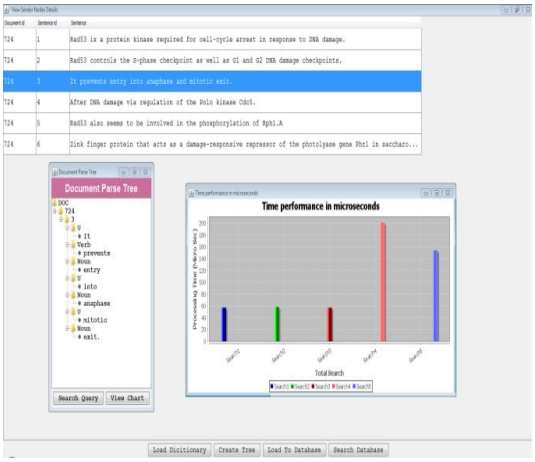

Fig 17: Snapshot of Time performance of search query without PTQL

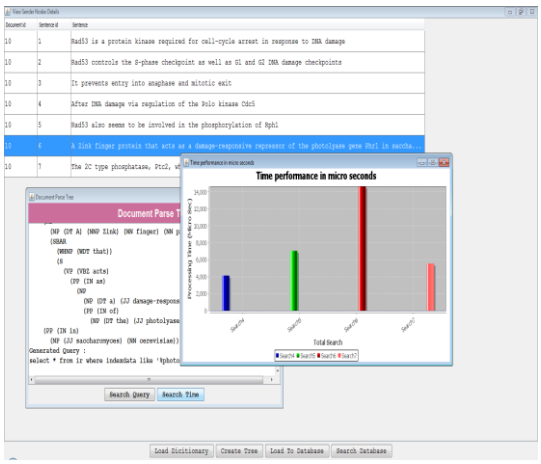

Fig 18: Snapshot of Time performance of search query with PTQL.

Time performance in Micro seconds for search with PTQL evaluation & in Milliseconds for search without PTQL for a biomedical research sample data as given above. The time performance indicates that our proposed framework is acceptable for real-time IE.

VII. CONCLUSION

Information Extraction has been an active research area over the years. The main focus has been on improving the accuracy of the extraction systems, and IE has been seen as an one-time execution process. Such paradigm is inadequate for real-world applications when IE is seen as long running processes. We describe how our proposed extraction framework differs from traditional IE systems, rule-based IE systems and IE systems based on machine learning. While new documents can be added to our text collection, the content of the existing documents are assumed not to be changed, which is the case for Medline abstracts. Our focus is on managing the processed data so that in the event of the deployment of an improved component or a new extraction goal, the affected subset of the text corpus can be easily identified.

The filtering process utilizes the efficiency of IR engines so that a complete scan of the parse tree database is not needed without sacrificing any sentences that should have been used for extraction. Furthermore, our approach provides automated query generation components so that casual users do not have to learn the query language in order to perform extraction. To demonstrate the feasibility of our incremental extraction approach, we performed experiments to highlight two important aspects of an information extraction system: efficiency and quality of extraction result

REFERENCES

[1] Ramakrishnan, and S. Vaithyanathan, Introduction to Special Issue on Managing Information Extraction," ACM SIGMOD Record, vol. 37, no. 4, p. 5, 2008.

[2] E. Agichtein and L. Gravano, "Snowball: Extracting Relations from Large Plain-Text collections," Proc. Fifth ACM Conf. Digital Libraries, pp. 85-94, 2000.

[3] A. Doan, J Naughton, R. Ramakrishnan, A. Baid, X. Chai, Chen, P. DE Rose, B. Gao, C Gokhale, J.Huang,W. Shen, and B.-Q. Vuong, "Information Extraction Challenges in Managing Unstructured Data," ACM SIGMOD Record, vol. 37, no.4, pp. 14-20, 2008.

[4] R. Krishnamurthy, Li, S. Raghavan, S. Vaithyanathan, and H. Zhu, "SystemT: A System for Declarative Information Extraction," ACM SIGMOD Record, vol. 37, no. 4, pp. 7-13, 2009.

[5] P.G. Ipeirotis, E. Agichtein, P. Jain, and L. Gravano, "Towards a Query Optimizer for Text-Centric Tasks," ACM Trans. Database Systems, vol. 32, no. 4, p. 21, 2007.

[6] P.G. Ipeirotis, E. Agichtein, P. Jain, and L. Gravano, "Towards a Query Optimizer for Text-Centric Tasks," ACM Trans. Database Systems, vol. 32, no. 4, p. 21, 2007.