Semantic Search through Pattern Recognition

Shanti Gunna,
Dept of Computer Science & Engineering
DRK Institute of Science & Technology, Hyderabad
shantireddy22@yahoo.com

N. Raghava Rao
HOD, Dept of CSE
DRK Institute of Science & Technology, Hyderabad
raghavarao.n@gmail.com

Abstract— Semantic search on databases often return a large number of results, only a small subset of which is relevant to the user. We present a semantic search technique considering the type of desired Web resources and the semantic relationships between the resources and the query keywords Ranking and categorization, which can also be combined, to alleviate this information overload problem. Results refinement for databases is the focus of this work. A novel search interface that enables the user to navigate large number of query results using the Pattern recognition. First, the query results are matched with the key word using full Pattern matching. In contrast, previous works expand the hierarchy in a predefined static manner, without navigation refinement modeling. We show that the problem of selecting the best concepts to reveal at each refinement and propose an efficient Pattern matching algorithm. We show experimentally that how results are refined using full Pattern matching at first level and half pattern matching at second level

Keywords: Indexing, Location monitoring, Optimal String matching, Pattern matching, Search process, Semantic Search

I. INTRODUCTION

Search has many parameters and it is, of course, necessary to set up a few of them with independent data in order to reduce the number of degrees of freedom in the model. The main interest of the model is in the interaction between pattern recognition and search. Semantic search seeks to improve search accuracy by understanding searcher intent and the contextual meaning of terms as they appear in the searchable data space, whether on the Web or within a closed system to generate more relevant results. To address the scalability issue in a large, distributed and dynamic setting such as the Semantic (Web), it is often desirable to identify which sources might be potentially relevant to a query before these sources are accessed. We consider an approach for identifying the minimal set of potentially relevant Semantic Web data sources for a given query. In our framework, a Potentially Relevant data source can make assertions about its content's relevance by means of REL statements. A data source provider can use REL statements to summarize the contents of a data source in terms of classes whose instances the data source has information about and the properties used to relate them. REL statements allow us to develop algorithms to choose data sources that may be relevant to a query and ignore sources that are definitely irrelevant. Semantic search systems consider various points including context of search, location, intent, and variation of words, synonyms, generalized and specialized queries, concept matching and natural language queries to provide relevant search results. Two major forms of search: navigational and research. In navigational search, the user is using the search engine as a navigation tool to navigate to a particular intended document. Rather than using ranking algorithms such as Google's Page Rank to predict relevancy, semantic search uses semantics, or the science of meaning in language, to produce highly relevant search results. In most cases the information queried by a user rather than have a user sort through a list of loosely related keyword results.
The size of the query result makes it difficult for the user to find the citations that the user is most interested in, and a large amount of effort is expended searching for these results. In recent years, many researchers have worked on how to apply the methodology of classification to semantic search for acquiring the optimization of search. Theoretically at least, such a search engine could offer advanced querying and browsing of structured data with search results automatically aggregated from multiple documents and rendered directly in a clean and consistent user-interface, thus reducing the manual effort required of its users. Indeed, there has been much research devoted to this topic, with various incarnations of (mostly academic) RDF-centric Web search engines emerging.
Many solutions have been proposed to address this problem—commonly referred to as information overload [1], [2], These approaches can be broadly classified into two classes: ranking and categorization—which can also be combined. Ranking presents the user with a list of results ordered by some metric of relevance [9] or by content similarity to a result or a set of results. In categorization [1], [2], [3], query results are grouped based on hierarchies, keywords, tags, or attribute values. User studies have demonstrated the usefulness of categorization in finding relevant results of exploratory queries [12]. While ranked results are useful when the ranking function is aligned with user preferences or the result list is small in size, categorization is generally employed by users when ranking fails or the query is too "broad" [12].Which rely on the

detection of certain patterns in web content that could potentially be harmful. There are many existing string matching algorithms, such as SBOM (Set Backward Oracle Matching) [3], Aho- Corasick [4], Set Horspool [5], Wu-Manber [6], SOG [7], etc. These matching algorithms are classified into two categories: prefix matching and suffix matching. In general, suffix matching is faster and more effective in handling long patterns than prefix matching; thus, it is used more widely.

II. RELATED WORK

The MEDLINE database, on which the Pub Med search engine operates, contains over 18 million citations and is currently growing at the rate of 500,000 new citations each year Other biological sources, such as Entrez Gene and OMIM witness similar growth. As claimed in previous work the ability to rapidly survey this literature constitutes a necessary step toward both the design and the interpretation of any large-scale experiment. Biologists, chemists, medical and health scientists are used to searching their domain literature—such as Pub Med—using a keyword search interface. Here, query language is based on the conjunctive query language for DLs that has been proposed by Horrocks et al.. This query language overcomes the inadequacy of description logic languages in forming extensional queries. Furthermore, it corresponds to the most common SPARQL queries. We refer to the left hand side of :- as the head of the query and the right hand side as the body of the query. The variables that appear in the head must appear also in the body and are universally quantified. Such variables are called distinguished variables and describe the form of a query's answers. All other variables in the query are called non-distinguished variables and are existentially quantified. For a given query Q and substitution θ, we use Q θ as a shorthand for B1θ ^ B2 θ . . .^ Bn θ.

Currently, in an exploratory scenario where the user tries to find citations relevant to her line of research and hence not known a priori, she submits an initially broad keyword-based query that typically returns a large number of results. Subsequently, the user iteratively refines the query, if she has an idea of how to, by adding more keywords, and resubmits it, until a relatively small number of results are returned. Haase and Motik [6] have described a mapping system for OWL and proposed a query answering algorithm. They identify a mapping language that is similar to ours. However, as their language adds rules to OWL, it is undesirable and as such they need to introduce restrictions to achieve decidability. Our language, on the other hand, is a sub language of a decidable language. Furthermore, similar to the DRAGO approach, Haase and Motik do not rely on an explicit reformulation step and process all the maps for a query reformulation. Peer-to-peer systems like Bibster [2] and Some Where have shown promises in providing query answering solutions for the Semantic Web. However, a peer-to-peer system needs special software installed at every server. Our system on the other hand makes use of the existing infrastructure of the Web. A recent work by Liarou et al. [11] uses Distributed Hash Tables (DHT) to index and locate relevant RDF data sources. However, they do not address the schema mapping issue and therefore work on a single ontology environment. Furthermore, DHTs are targeted for a more P2P architecture as opposed to client server web architecture.

2.1Commonly used searching methodologies

The annotation process can be generally divided into two steps. The first is to establish mappings between existing Semantic Web terms and those needs to be annotated in data. The second step is to come up with a local ontological structure constituting the semantic web terms to model the data. Most of previous works in annotating semi structured data focus on the second step. Some skip the first step and bootstrap the ontological terms and structure from the local data itself. For example, a number of systems that map data in RDB to RDF format leverage a set of rules such as "table to class and column to predicate". The Semantic Web may represent a future direction for bringing deep-Web information to the surface, leveraging RDF as a common and exible data model for exporting the content of such databases, leveraging RDFS and OWL as a means of describing the respective schemata, and thus allowing for automatic integration of such data by Web search engines. Efforts such as D2R(Q) [13] seem a natural fit for enabling RDF exports of such online databases. Offering search and querying over a raw RDF dataset collected from the Web would thus entail many duplicate results referring to the same entity, emulating the current situation on the HTML Web where information about different resources is fragmented across source documents. Given a means of identifying equivalent entities in RDF data { entities representing the same real-world individual but identified incongruously} would enable the merging of information contributions on an entity given by heterogeneous sources without the need for consistent URI naming of entities.

The majority of users are accustomed to expressing their information needs in terms of keywords. It would be interesting to have a semantic search that has a traditional "Google-like" interface (keyword queries), but at the same time performs semantic processing. A semantic search that enables both textual information and RDF annotations querying is presented in [5]. Froogle [7] also presents a very interesting approach for product searches. It is a search engine specialized in querying for products, where the user expresses the products he wants to search

for using keywords that are associated with the product (i.e. its brand, name, model, etc.). Froogle tries to guess the product the user wants to search for by associating the keywords in the query with the metadata that describe the products in their knowledge base. Another interesting semantic searcher is SCORE [15]. It uses automatic classification and information-extraction techniques together with metadata and ontology information to enable contextual multi-domain searches that try to understand the exact user information need expressed in a keyword query.

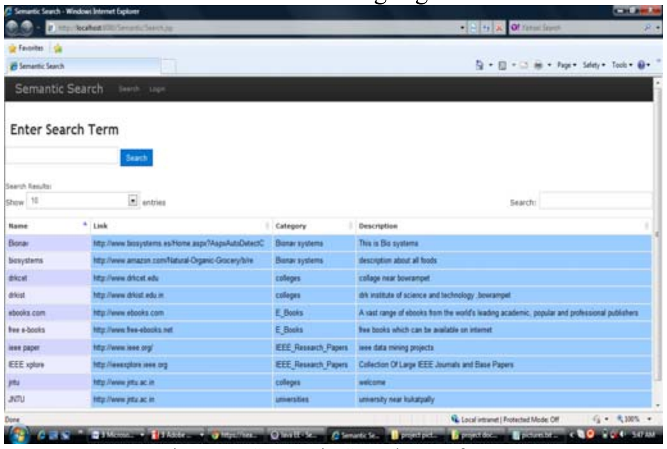
In general the common search algorithms are
RDF Path Traversal - traversing the net formed by the RDF data format.
Keyword to Concept Mapping
Graph Patterns - used to formulate patterns for locating interesting connecting paths between resources. Also commonly used in data visualization.
Logics - by using inference based on OWL
Fuzzy concepts, fuzzy relations, and fuzzy logics
2.2 Single pattern algorithms
This algorithm requires a preprocessing phase, which prepares a table of occurrences of the first and the last characters of the pattern in the given input text. The preprocessing phase time complexity of the Native string search algorithm is less than the Rabin-Karp string search algorithm and the Quick Search algorithms. The preprocessing phase time complexity of the Rabin-Karp string search algorithm is compared with the Native string search algorithm and Quick Search algorithms and is presented in the table *tab 1*,
Let m be the length of the pattern and let n be the length of the searchable text. Asymptotic times are expressed using O, Ω, and θ notation. Before introducing our method, we first investigate the influence on searching time cost when using different strategies of Exact Pattern Matching and Partial Pattern matching algorithm; we extract partial strings from three positions: the leftmost, the rightmost, and the middle of the pattern. Above figure shows the comparison of their corresponding searching speed. We can see that the speed of extracting partial strings from the left most characters is the fastest, while extracting from the rightmost is the slowest. Speed of extracting from the middle position lies.

Algorithm	Preprocessing Time	Matching Time
Native string search algorithm	0(no preprocessing)	θ((n-m+1)m)
Rabin-Karp string search algorithm	θ (m)	average(n+m), worst θ((n-m+1)m)
Finite-state automaton based	θ(m|Σ|)	θ (n)
Knuth-Morris-Pratt algorithm	θ (m)	θ (n)
Boyer-Moore string search algorithm	θ(m+|Σ|)	Ω(n/m), O(n)

In this phase, we find the occurrences of the first and last characters of the pattern in the given input text. Here, we will get two cases: first and last characters of the pattern are similar and the other, dissimilar.

III.SYSTEM MODEL

When user searches first refinement take place and here on first search button user enters his key word by looking at the description given to it server hit the database location information by raising a query, the server takes it gives information to user. This is the proposed system architecture. In order to make this system to overcome information overload problem at first search exact Pattern Matching used and at the second level of refinement Partial Pattern Matching is used two algorithms are proposed. They are known as Exact Pattern Matching algorithm and Partial Pattern Matching algorithm. Here, we get two cases: first and last characters of the pattern in the given input text may be of similar or dissimilar.
Case 1: If the first and the last characters of the Patterns are similar
If the difference between any two occurrences of the first character of the pattern in the pre-computed table is less than the size of the pattern by one, then, it is taken as one probability for occurrence of an exact pattern match.
Case 2: If the first and the last characters of the Patterns are dissimilar
If the difference between any two occurrences of the last and the first characters of the pattern in the pre computed table is less than the size of the pattern by one, then, it is taken as one probability for occurrence of an exact pattern match.
An interesting functionality is that the system provides, for each node obtained as a result of the propagation, the shortest path from one of the origin nodes. This allows recovering the path followed in the graph to obtain the inference. The node which contributed most for the activation of each node in the result set is also provided. Both these data are very important since they allow the knowledge engineer to better evaluate the results presented, and tune the search engine when necessary. If the results are not satisfactory, this information gives clues that show where the configuration should be changed in order to obtain better results. The result given by the traditional search engine is a set of node instances ordered by their similarity with the query. This set of nodes is supplied to the spread activation algorithm as the initial set of nodes for the propagation. In addition, the ordering information given by the traditional search engine is also used. For each node, the traditional search engine provides a real number that measures the relative importance of that node with respect to the given query. This numeric value is used as the initial activation value for the node. Therefore, the nodes that were ranked well by the traditional search engine will have priority in the propagation since the exploration starts at the nodes with the highest activation values.
The basic Search look like following Figure .1


Semantic Search - Windows Internet Explorer
Favorites
Semantic Search
Page
Safety
Tools
Semantic Search
Search
Login
Search
Done
Local intranet | Protected Mode: Off

Enter Search Term

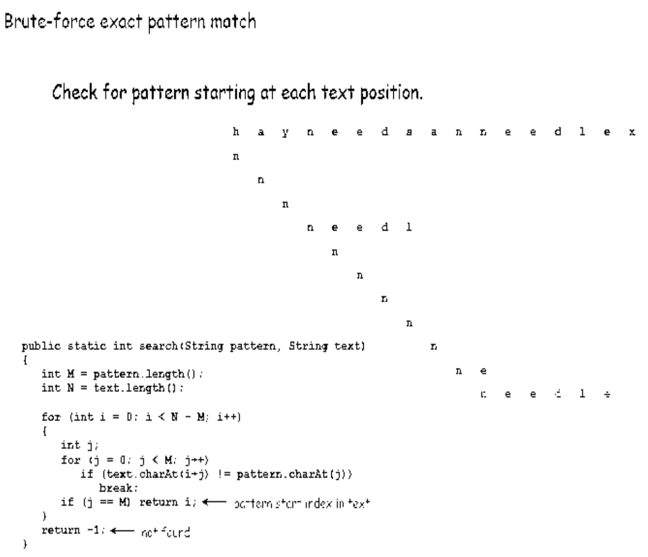
Search Results:
Show 10 entries
Search:

Name	Link	Category	Description
Bionav	http://www.biosystems.es/Home.aspx?AspxAutoDetectC	Bionav systems	This is Bio systems
biosystems	http://www.amazon.com/Natural-Organic-Grocery/b/re	Bionav systems	description about all foods
drkcet	http://www.drkcet.edu	colleges	collage near bowrampet
drkist	http://www.drkist.edu.in	colleges	drk institute of science and technology ,bowrampet
ebooks.com	http://www.ebooks.com	E_Books	A vast range of ebooks from the world's leading academic, popular and professional publishers
free e-books	http://www.free-ebooks.net	E_Books	free books which can be available on internet
ieee paper	http://www.ieee.org/	IEEE_Research_Papers	ieee data mining projects
IEEE xplore	http://ieeexplore.ieee.org	IEEE_Research_Papers	Collection Of Large IEEE Journals and Base Papers
jntu	http://www.jntu.ac.in	colleges	welcome
JNTU	http://www.jntu.ac.in	universities	university near kukatpally

Figure.1: Semantic Search Interface

3.1Exact Pattern Matching Algorithms

The proposed location algorithms are meant for achieving three purposes. The first purpose is that they can enhance the quality of location services. The second purpose is to minimize the computational resources and communication overhead. The third purpose of them is to ensure anonymity of personal location privacy.



Brute-force exact pattern match

Check for pattern starting at each text position.

```
public static int search(String pattern, String text)
{
   int M = pattern.length();
   int N = text.length();

   for (int i = 0; i < N - M; i++)
   {
      int j;
      for (j = 0; j < M; j++)
         if (text.charAt(i+j) != pattern.charAt(j))
            break;
      if (j == M) return i; ←— pattern start index in text
   }
   return -1; ←— not found
}
```

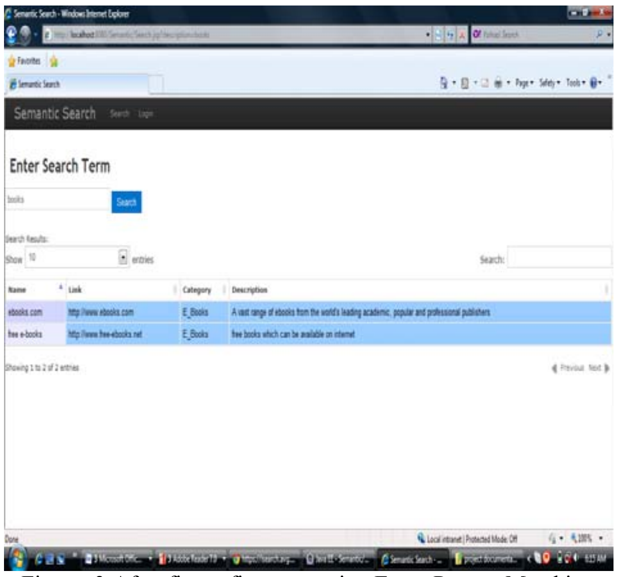
Figure. 2: Exact Pattern Matching algorithm

IV. IMPLEMENTATION

Pattern-matching algorithms scan the text with the help of a window, whose size is equal to the length of the pattern.

The first step is to align the left ends of the window and the text and then compare the corresponding characters of the window and the pattern; this procedure is known as attempt. After a match or a mismatch of the pattern, the text window is shifted to the right. The question is how many characters are required to shift the window on the text. This shift value varies based on the methodology used by various algorithms. This procedure is repeated until the right end of the window is within the right end of the text. The order of comparisons is carried out by comparing the last character of the window and the pattern, and after a match, the algorithm further compares the first character of the window and the pattern. By doing so, an initial resemblance can be established between the pattern and the window, and the remaining characters are compared from right to left until a complete match or a mismatch occurs. After each attempt, the skip of the window is gained by the Quick-Search bad character (qsBc) shift value for the character that is placed next to the window.

By using these algorithm first refinement can be done and no of citations to navigate is also reduced and it will look like follows from fig1 we are selecting books from 12 citations it is reduced to 2 and it look like following figure fig.3



Enter Search Term

books

Search Results:

Show 10 entries Search:

Name	Link	Category	Description
ebooks.com	http://www.ebooks.com	E_Books	A vast range of ebooks from the world's leading academic, popular and professional publishers
free e-books	http://www.free-ebooks.net	E_Books	free books which can be available on internet

Showing 1 to 2 of 2 entries

Done Local intranet | Protected Mode: Off

Figure. 3:After first refinement using Exact Pattern Matching

Now at second level refinement we are using for example Fig illustrates the searching process with a certain pattern set. In Figure two pointers have moved d1 off the last position in this window, which means there are only d1 characters that were successfully matched. The window is then shifted after the failed search by the oracle. The two pointers are set to the last position of the new window again. These two pointers are decreased until the pointed characters fail to match. These steps are iterated until the end of the text, with the total number of times in which the window is slid being k. Within the i-th window, the comparisons between the corresponding two characters have been completed for di times. Location algorithm is presented as follows:

```
Input: P(Original Pattern Set), R(Train Data)
Output: Q(the set of extracted partial strings of length m in P)
1.  for each pattern pt in P
2.      $S_{sub} \longleftarrow \{s^{p,x}_{y,z}\}$
3.      $Ac \longleftarrow Build\_AC(S_{sub})$
4.      $C_{sub}=\{c^{p,x}_{y,z}\} \longleftarrow Search\_InAC(R)$
5.      for each partial string ppt in pt
6.          minx $\longleftarrow$ 0;  mincost $\longleftarrow +\infty$ ; ps $\longleftarrow \xi$
7.          for x=0 to x<|ppt|-m
8.              costx $\longleftarrow \sum_{y=0}^{m}\sum_{z=1}^{m-y} c_{y,x}^{ppt,z}$
9.              if costx<mincost do
10.                 mincost=costx;  Minx=x;  ps=ppt
11.             end of if
12.         end of for
13.     end of for
14.     Add ps to Q
15. end of for
```

Our algorithm allows the proper partial strings extraction from a pattern set, with random pattern lengths meeting most suffix matching algorithms' requirements. Each of the experiments discussed here shows that choosing the partial strings extracted by our method can achieve better searching speed than those by other methods. Moreover, the advantage of our method is more obvious on the Snort pattern set. When the scale of pattern set increases, our method can achieve more graceful degradation.

V. RESULTS ANALYSIS

To assess the performance of our algorithm, we considered all the well-known algorithms for comparison with the proposed algorithm. We have analyzed two types of data, consisting of small (ό) 4) and big (ό) 20) alphabet sizes.The first one is the pattern set available in the and the second is the searching speed sequences. We have executed and tested all the algorithms under study using a 3.06 GHz processor, 1 GB of RD-RAM with 512 KB of cache memory.

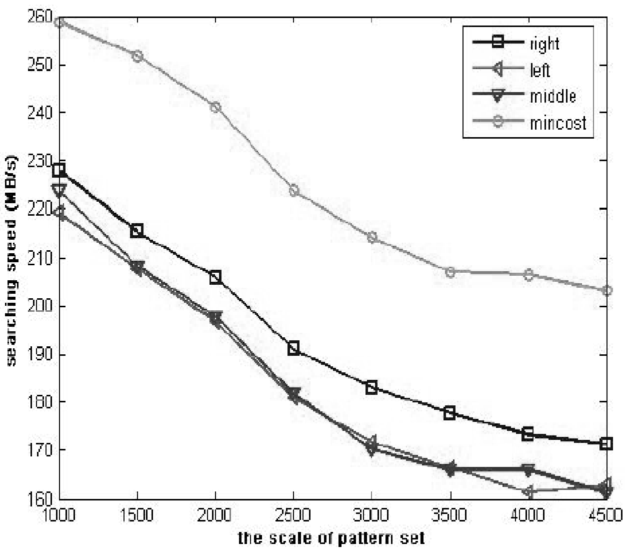


Figure 4. Speed comparisons of partial string extractions at different positions with pattern sets changing. Pattern sets are from Snort. (pattern number range: 1000-3500, pattern length range: 5-18).

IV.CONCLUSION AND FUTURE WORK

Information overload is a common phenomenon encountered by users searching huge databases We address this problem by organizing the query results according to their associations to concepts In this paper, we propose a novel method to Semantic Search using both Exact Pattern Matching and optimal partial strings based on achieving the fastest searching speed. these citations are refined such that the information overload observed by the user is minimized. We propose database indexing for large data. A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of slower writes and increased storage space. Indices can be created using one or more columns of a database table, providing the basis for both rapid random lookups and efficient access of ordered records For example, an index could be created on upper(last name), which would only store the upper case versions of the last name field in the index. Another option sometimes supported is the use of "filtered" indices, where index entries are created only for those records that satisfy some conditional expression. A further aspect of flexibility is to permit indexing on user-defined functions, as well as expressions formed from an assortment of built-in functions.

REFERENCES

[1] J.S. Agrawal, S. Chaudhuri, G. Das, and A. Gionis, “Automated Ranking of Database Query Results,” Proc. First Biennial Conf. Innovative Data Systems Research, 2003.

[2] K. Chakrabarti, S. Chaudhuri, and S.W. Hwang, “Automatic Categorization of Query Results,” Proc. ACM SIGMOD, pp. 755- 766, 2004.

[3] Z. Chen and T. Li, “Addressing Diverse User Preferences in SQLQuery-Result Navigation,” Proc. ACM SIGMOD, pp. 641-652, 2007.

[4] L. Comtet, Advanced Combinatorics: The Art of Finite and Infinite Expansions, pp. 176-177, Reidel, 1974.

[5] R. Delfs, A. Doms, A. Kozlenkov, and M. Schroeder, “GoPubMed Ontology-Based Literature Search Applied to Gene Ontology and PubMed,” Proc. German Conf. Bioinformatics, pp. 169-178, 2004.

[6] D. Demner-Fushman and J. Lin, “Answer Extraction, Semantic Clustering, and Extractive Summarization for Clinical Question Answering,” Proc. Int’l Conf. Computational Linguistics and Ann. Meeting of the Assoc. for Computational Linguistics, pp. 841-848, 2006.

[7] Entrez Programming Utilities, http://www.ncbi.nlm.nih.gov/ entrez/query/static/eutils_help.html, 2008.

[8] U. Feige, D. Peleg, and G. Kortsarz, “The Dense k-Subgraph Problem,” Algorithmica, vol. 29, pp. 410-421, 2001.

[9] V. Hristidis and Y. Papakonstantinou, “DISCOVER: Keyword Search in Relational Databases,” Proc. Int’l Conf. Very Large Data Bases (VLDB), 2002.

[10] R. Hoffman and A. Valencia, “A Gene Network for Navigating the Literature,” Nature Genetics, vol. 36, no. 7, p. 664, 2004.

[11]iHOP—Information Hyperlinked over Protein, http://www. ihop-net.org/UniPub/iHOP/, 2008.

[12] M. Kaki, “Findex: Search Results Categories Help When Document Ranking Fails,” Proc. ACM SIGCHI Conf. Human Factors in Computing Systems, pp. 131-140, 2005.

[13] Snort Rule. http://www.snort.org/snort-rules

[14] ClamAV Rule. http://www.clamav.net/lang/en/download/cvd/

[15] C. Allauzen, M. Crochemore and M. Raffinot, “Efficient Experimental String Matching by Weak Factor Recognition”, in Proc. 12th Annu. Symp. on Combinatorial Pattern Matching, Jerusalem, July 1–4, 2001, pp. 51-72.

[16] A. Aho and M. Corasick, Bell Laboratories. “Efficient String Matching:An Aid to Bibliographic Search”, in Communications of the ACM, vol. 8, 1975, pp. 333-340.

[17] G. Navarro and M. Raffinot, Flexible Pattern Matching in Strings: Practical on-line search algorithms for texts and biological sequence. Cambridge: Cambridge University Press, 2002.

[18] S. Wu and U. Manber, “A fast algorithm for multi-pattern searching”, Dept. of Computer Science, University of Arizona, Tucson, AZ, TR-94-17, 1994.

.

Mrs. Shanti Gunna is pursuing M.Tech from DRK Institute of Science and Technology, Hyderabad.Her main research interest includes Data Mining, Compiler design.

Mr. N. Ragha Rao has received his M.Tech degree in Computer Science Engineering. He has many years of experience in teaching field, presently working as HOD and Associate Professor, Computer Science & Engineering at DRK Institute of Science and Technology, Hyderabad, Andhra Pradesh.