Volume 3, Issue 5, May 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Semantic Search over an XML data

Bhavana V. Kumbhare[*]
Computer Science and Engineering Dept.
Autonomous, India

S. J. Karale
Computer Science and Engineering Dept,
Autonomous, India

Abstract— A semantic search engine for querying XML data, is presented. Here we present the role of XML search engine in semantic web based on XML data. Due to the popularity of the XML data format, several languages for querying XML have been devoloped. While these languages allows querying the content of an XML document, we propose a search engine that has a simple query language, enables keyword search at the granularity of XML elements and is also suitable for a naive user. It returns semantically related document fragments that satisfy the user's query. Indexing techniques and Evaluation algorithm is developed that allows the system to deal with large documents

Keywords— Data retrieval, Relationship, Semantics, Search Engine, unstructured data

I. Introduction

No doubt, XML is rapidly becoming one of the most important data formats on the web. It has been already used for scientific data and also for data exchange on the Internet. One of the advantage of XML is that, it can be used to represent *structured* as well as *unstructured data*. For example, XML can be used in a hospital to represent information about patients (e.g., name, address, and birthdates) and also observations from doctors. In our work we mainly focused on unstructured data (XML files). Current search engines, are the tools for finding HTML documents, have two main drawbacks when they search XML documents. First, it is not possible to formulate the query that explicitly refers to semantics (i.e., XML tags). The second is, search engines returns an entire XML document as an answer to the users query. Thus reference to the whole document is not an useful answer, since it may contain thousand of elements. So searching must be improved, instead of returning complete XML documents, XML search engine should return the fragments of XML documents. XML query languages, such as XQuery and XPath, queries the data values using predicates with exact semantics. As a result, deep knowledge of the structure of the data is required in order to formulate meaningful queries. But, the XML document contains large amount of plain text within the web. Thus, the retrieval of values based on predicates with exact semantics becomes extremely difficult. To overcome this problem, we employed keyword search for XML data retrieval. The main advantage of keyword search is, it is simple. Users do not have to know a complicated query language and the knowledge about the structure of the data.

In our work, we developed a syntax for search queries that is suitable for an end user and facilitates a fine-granularity search. The results of [11] were adapted so that search engine always returns document fragments that are semantically related, even when only keyword is specified in the query. We developed precomputed index structures and evaluation algorithms that allow the system to deal efficiently with large documents. A suitable work is going on ranking mechanism. In this paper, we focused on how to support semantic search over XML documents.

II. Query Syntax and Semantics

The majority of works as in ([1],[8],[13],[14]]) employs the *list of keywords* as its retrieval language. An XML (semantic) search system identifies the document fragments that best satisfy the query. While processing the query, the system may choose to consider only the textual content, or both the textual content and the labels of each XML node in the searched collection. According to the survey, the SHOE search engine is one of the first form-based semantic search engine. It provides sophisticated web forms that allow users to specify queries. Such forms however are only suitable for those who are familiar with back end ontologies and knowledge bases. Naive users have difficulties in understanding these forms. So they suffer from the difficulty of finding the information they aim. The Corese search engine is one of the RDF-based semantic search engines. In this they have used sophisticated query language. Hence whenever the user asks queries, he should be familiar with the back-end ontologies and the query language. XRANK system as in [13] is one of the keyword based query language over hierarchical and hyperlinked XML documents. Here the XML document is modelled in the form of graph by taking IDREF, Xlink in to consideration. XRANK returns document fragments as answers. In XRANK, there is no distinction between keywords and labels, and each keyword of a query is matched against every word of the document. XRANK may return the answers that are semantically unrelated. It is useful, where references between elements indicate importance.

We opted for the query language specifying keywords. Our language, whose syntax has been borrowed from XSearch as in [1], is defined as follow: The query syntax of our search engine extends traditional keyword search by allowing the specification of i) the query label and ii) the combination of Tokens. By the label we mean, the user can specify the expected type of the results and tokens means specifying keywords. We used three operators to specify the query. The first one is l "**::**", which supports the specifying query label. The second one is "**,**", which indicates multiple tokens can be specified. The last one is the word "or ", which indicates the optional tokens. So the complex queries in which multiple tokens are involved can be easily specified by the end user. We divided the query in four terms, has the form label :: Token, label :: ,or :: Token, either Label or Token . A query has the form Q (T) where T = t1,……, tn a sequence terms. In order to satisfy a query, each of the required tokens and the labels must be satisfied, which means that the search results must be semantically related to each of the required tokens. In the case when there are optional tokens, one of the optional token must be satisfied We model an XML documents in the form of trees. Each interior node represent a label and each leaf node represent a Tokens. In Figure 1 there is a tree that represents a small portion of the XML document of the library Record. Let usl refer to this tree as T. Let n be an interior node in a tree T . We say that n satisfies the search term l :: t if n's label equals l and the tokenized text content of n contains the word t. We say that n satisfies the search term l:: if n is labels equals l. Finally, we say that n satisfies the search term:: k if n has a leaf child that contains the token t.

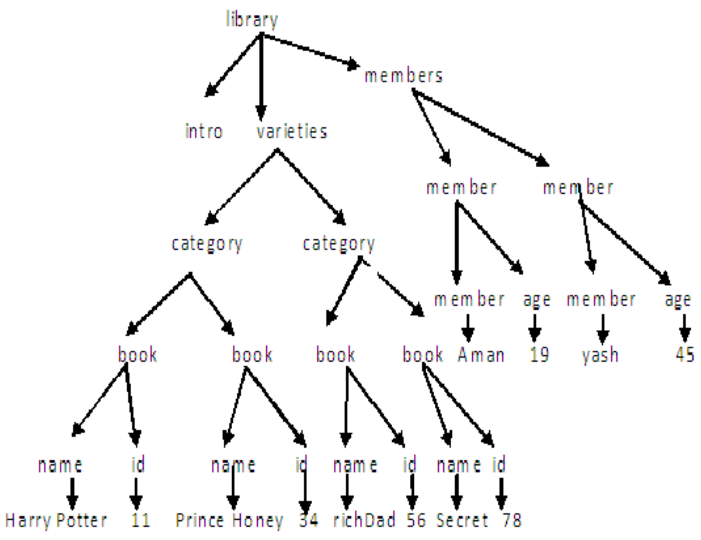

Fig. 1.XML document of Library record

Eg. The node 'member' satisfies 'member :: Aman'. The node 'name' satisfies ' :: HarryPotter'. This query syntax provides a simple, and flexible approach for specifying user queries in semantic search. It overcomes the problem of knowledge overhead as in form-based or view based semantic search engines, since, the end users are not familiar with any ontology or any special query language. In contrast with current semantic-based keyword search engines which only accept one keyword as input, this search engine allows specification of complex queries and the expected type of results.

III. Relationship Between Nodes

The focus of our work is on how to support semantic search over XML documents. Therefore, considering the results as in [1],[11], we decided to investigate whether the relation that were used in [1],[9] can be efficiently employed in this environment. The relationship is defined as follows.

Since the query contains labels, tokens (sequence of terms) forms. Each of the terms in the query should have some relationship between each other. Then only the result we'll get, will be semantically related fragment of XML document. Therefore, we assume that there is a relationship R that determines whether the two nodes are related. Let T be a tree and R relationship between the nodes in T. We assume that R contains pairs of nodes that are meaningfully related. We start by giving the understanding of the relationship in the tree. We say that two nodes are related if and only if they belongs to the same entity. Eg. If n is the ancestor of n1 and n2, means both belongs to the entity n. Hence they are related to each other. Similarly if n1 and n2 have the same labels and they belongs to the same entity which also has the same label then we say that they are meaningfully related .Now suppose if the two nodes n1 and n2 have different ancestors but has same labels say n and n', but n is neither an ancestor of n2 nor n' is an ancestor of n1. Then we may conclude that n1 and n2 are not related. We call this relation as ancestor-decendor relationship.

IV. Query Result

Our search engine always returns document fragments that are semantically related, even when only keyword is specified in the query. Here the query has the form Q(T), where T denotes the terms t1,t2,….,tn. We say that the nodes is an R-answers ,if all nodes are Related in the given query. If there is a optional term in our query, our answer contains the null values. But if there is a required term in the query our answer will contain the values that satisfy the query term. Consider the answer is denoted by ans(Q) and N=n1,n2,n3…….. ans(Q) = n1, n2, …,if n1,n2,n3….are related .

V. SYSTEM ARCHITECHURE

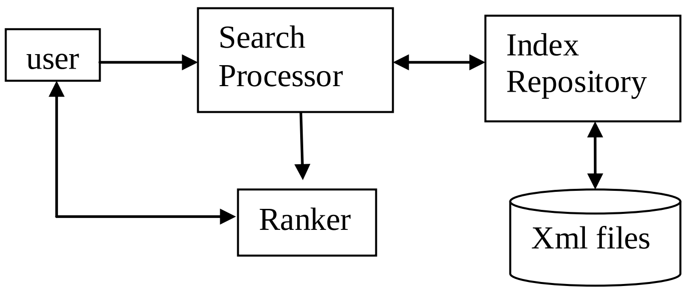

Fig.2. System architechure

The system architecture is shown in Fig. The flow of information is as follows. A user enters a query using browser. A Search Processor then parses the query into a list of terms. The Index Repository is used to find nodes that satisfy that terms and also find whether pairs of nodes are meaningfully related or not. The Index Repository checks the indices and then responds with the relevant information to the search processor. Search Processor, then creates the answers. These answers are then ranked using ranking algorithm and then they are returned according to their relevance to the user. The query processor is based on the algorithms as in [1],[11], uses the index structures such as inverted indices and nodeconnection indices . The search engine is built on precomputed index structures for evaluating the various search conditions in a query. Because of space limitations, we do not discuss in detail all of the index structures used. But here we mostly focus on the nodeconnection index. A nodeconnection index stores the pair of nodes which are related to each other. We say that nodes n1 and n2 are related if they belong to same entity and are also labeled differently

VI. RELATED WORK AND CONCLUSIONS

Several query languages for XML have been developed. XSEarch as in [1] has a simple query language in the form of keywords , suitable for an end user. It returns semantic document fragments that satisfy the user's query. Query answers are ranked using vector space model. Advanced indexing techniques were developed for XSEarch. It can be used as a general framework for searching an XML documents. The XXL search engine as in [14] has an SQL-like syntax, extended with ontological knowledge for similarity metrics. These languages are not suitable for naive user, since the query syntax is always complex. The EquiX as in [15] language for XML documents can only deal with documents that have a DTD. In [21], another search language for XML was proposed. Their query language consists of fragments of XML documents, and they only require matching of the queries to the documents. However, their query answers consist of entire documents. XRANK system as in [13] for keyword searching in XML documents has a ranking mechanism and it returns document fragments as answers. In XRANK, there is no distinction between keywords and labels, and each keyword of an XRANK query is matched against every word of the document . Actually, XRANK may return answers with parts that are semantically unrelated . The TAP search engine as in [11] was one of the first keyword-based semantic search engine. The TAP search engine first compares the keyword term against semantic entities. It then selects one entity from the candidate matches . After finding the semantic entity, the TAP search engine uses graph-walking techniques to collect and cluster search results in order to decide what to show as results and in which order. To choose the right keyword for a successful search, end-users should have the knowledge about the problem domain. Our semantic search engine overcomes the problem of knowledge overhead by supporting a keyword query, since the user do not need to have the knowledge about the structure of data and ontology. Keyword query provides a simple but powerful way of specifying queries. It is able to produce fragments of XML documents as an answers for user queries by providing relationship between the nodes in the query. Finally, naïve user can easily specify the complex query by allowing multiple keywords in the query.

ACKNOWLEDGMENT

` Mrs. Bhavana v. Kumbhare and Mr. S. J. Karale wish to thanks International Journal of Advanced Research in Computer Science and Software Engineering. This work was supported in part by the department of Computer Science and Engineering from Yashwantrao Chavan College of Engineering.

REFERENCES

[1] Cohen, S. Mamou, J. Kanza, Y. Sagiv, Y "XSEarch: A Semantic Search Engine for XML*" proceedings of the international conference on very large databases,* pages 45-56, 2003

[2] H. L. Wang, S. H. Wu, I. C. Wang, C. L. Sung, W. L. Hsu, and W. K. Shih, "Semantic search on Internet tabular information extraction for answering queries," *in Proceedings of CIKM '00 McLean, 2000*, pp.243-249.

[3] D. Tümer, M. A. Shah, and Y. Bitirim, An Empirical Evaluation on Semantic Search Performance of Keyword-Based and Semantic Search Engines: Google, Yahoo, Msn and Hakia, 2009 4th *International Conference on Internet Monitoring and Protection (ICIMP'09)* 2009.

[4] C Mangold (2007). "A survey and classification of semantic search approaches*". International Journal of Metadata, Semantics and Ontologies, pp. 23–34.*

[5]. D. Florescu, D. Kossmann, and I. Manolescu. Integrating keyword search into XML query processing. *The International Journal of Computer and Telecommunications Networking, 33(1):119–135,* June 2000.

[6] D. Ferrucci and A. Lally. UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, June 2004.

[7] Manning, C.D., Raghavan, P., Schtze, *H.: Introduction to Information Retrieval.* Cambridge University Press, New York, USA (2008)

[8] Yuangui Lei, Victoria Uren, and Enrico Motta *"SemSearch: A Search Engine for the Semantic Web",* Knowledge Media Institute (KMi), The Open University, Milton Keynes

[9] Maciej Ceglowski, Aaron Coburn, and John Cuadrado*" Semantic Search of Unstructured Data using Contextual Network Graphs"* National Institute for Technology and Liberal Education College, Middlebury,Vermont,05753USA

[10] R. Guha, R. McCool, and E. Miller. Semantic Search. *In Proceedings of the 12th international conference on World Wide Web,* pages 700–709, 2003.

[11]. S. Cohen, Y. Kanza, and Y. Sagiv. Generating relations from XML documents. *In Proc 9th International Conference on Database Theory,* Siena (Italy), Jan. 2003. Springer-Verlag.

[12] E. Kandogan, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. Zhu, "Avatar semantic search: a database approach to information retrieval," *in Proceedings of SIGMOD '06* Chicago, 2006, pp. 790-792.

[13] L. Guo, F. Shao, C. Botev, and J. Shanmu gasundaram. XRANK: Ranked keyword search over XML documents. *In Proc. 2003 ACM SIGMOD International Conference on Management of Data,* San Diego (California), June 2003.

[14] A. Theobald and G. Weikum. The index-based XXL search engine for querying XML data with relevance ranking. *In Proc. 8th International Conference on Extending Database Technology,* pages 477–495, Prague (Czech Republic), March 2002. Springer-Verlag.

[15] S. Cohen, Y. Kanza, Y. Kogan, W. Nutt, Y. Sagiv, and A. Serebrenik. EquiX: A search an query language for XML. *Journal of the American Society for Information Science and Technology,* 53(6):454–466, 2002.

[16] Thanh Tran, Philipp Cimiano, Sebastian Rudolph and Rudi Studer. Ontology-based Interpretation of Keywords for Semantic Search. *Institute AIFB, Universität Karlsruhe, Germany {dtr,pci,sru,rst}@aifb.uni-karlsruhe.de*

[17] Imen Zemmar*, Abdallah Benouareth*, Labiba Souici-Meslati. A SURVEY OF INDEXING TECHNIQUES IN NATIVES XML DATABASES. *LABGED Laboratory, Badji Mokhtar University, BP 12, 23000, Annaba, Algeria imen_zemmar@hotmail.com,* benouareth@yahoo.com

[18] G.Sudeepthi1, G.Anurada , Prof.M.Surendra Prasad Babu . A Survey on Semantic Web Search Engine. *IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 1, March 2012*

[19] Manoj Manuja , Deepak Garg Semantic Web Mining of Un-structured Data: Challenges and Opportunities. *International Journal of Engineering (IJE), Volume (5) : Issue (3) : 2011 268*

[20] Chulho Ahn, Quing Li, Ramez Elmasri, Shalli Prabhakar, Niroj Manandhar, Do Youn Kim. A Survey of Three Types of XML Indexing Techniques. *ACM Transactions on Computational Logic, Vol. 37, No. 4, 12 2005, Pages 1{24.*

[21] D. Carmel, Y. Maarek, Y. Mass, N. Efraty, and G. Landau. An extension of the vector space model for querying XML documents via XML fragments. *In ACM SIGIR 2002 Workshop on XML and Information Retrieval, Tampere (Finland),* Aug. 2002.