Semantic Keyword Search on XML

S.Sundaramoorthy [1] M.Kowsigan [2] J.Rajesh Kumar [3] C.V.Arulkumar [4]
[123] Assistant Professor [4]PG Scholar
Department of Information Technology
SNS College of Technology,
Coimbatore-641035,
Tamilnadu, INDIA.

Abstract

In this paper Semantic Keyword search on xml using CR (Coherency Ranking) is presented. Keyword search is a user friendly mechanism for retrieving XML data in web and scientific applications. An intuitively compelling but vaguely defined goal is to identify matches to query keywords that are relevant to the user. However, it is hard to directly evaluate the relevance of query results due to the inherent ambiguity of search semantics. As an effective search method to retrieve useful information, keyword search has gotten a great success in IR field. However, the inherently hierarchical structure of XML data makes the task of retrieving the desired information from XML data more challenging than that from flat documents.

Key-words: XML, Coherency Ranking, semantics, WSDL, SOAP, XSEarch.

1. Introduction

In recent years, the popularity of the internet and the growing need to share the vast amount of data on the web, has fueled the need for a new web data format that will make data sharing and integration feasible. Information in traditional web pages encoded in HTML (Hyper Text Markup Language) format is usually hard to interoperate and exchange because the tags in HTML documents describe the presentation of data instead of the semantics of data. As such, a new standard was required to encode web data in a simple and usable format so that information providers can interoperate easily over heterogeneous data distributed on the web.

In order to facilitate the sharing of structured data across different information systems over the internet, a new open standard for web data representation and exchange, XML (eXtensible Markup Language), has been recommended and adopted as the new generation web data format. XML started strong and has grown quite rapidly. It has proven itself a very valuable technology, which "turns the web into a database" and allows data integration on the web. In fact, most data exchanged among a variety of web applications are already in XML format, such as web services that use XML-based descriptions in WSDL and exchange XML messages based on the SOAP protocol, e-commerce and e-business, collaborative authoring of large electronic documents and management of large scale network directories.

As a flexible and self describing semi-structured data format, XML holds the promise to yield (1) a more precise search by providing additional information in the elements, (2) a better integrated search of documents from heterogeneous sources, (3) a powerful search paradigm using structural as well as content specifications, and (4) data and information exchange to share resources and to support cooperative search.

Effectiveness in term of result relevance is the most crucial part in keyword search, which can be summarized as the following three issues in XML field. Current XML keyword and natural language query answering approaches rely on heuristics that assume certain properties of the DB schema. Though these heuristics are intuitively reasonable, they are sufficiently ad hoc that they are frequently violated in practice, even in the highest-quality XML schemas. Thus current approaches suffer from low precision, low recall, or both.

Issue 1: It should be able to effectively identify the type of target node(s) that a keyword query intends to search for. Such target node is known as a search for node.

Issue 2: It should be able to effectively infer the types of condition nodes that a keyword query intends to search via. Such condition nodes are known as a search via nodes.

Issue 3: It should be able to rank each query result in consideration of the above two issues.
The first two issues address the search intention problem, while the third one addresses the relevance based ranking problem w.r.t. the search intention.

Regarding to Issue 1 and Issue 2, XML keyword queries usually have ambiguities in interpreting the search for node(s) and search via node(s), due to three reasons below.

Ambiguity1: A keyword can appear both as an XML tag name and as a text value of some other nodes.

Ambiguity2: A keyword can appear as the text values of different types of XML nodes and carry different meanings.

Ambiguity3: A keyword can appear as an XML tag name in different contexts and carry different meanings.

The search intention for a keyword query is not easy to determine and can be ambiguous, because the search via condition is not unique; so how to measure the confidence of each search intention candidate, and rank the individual matches of all these candidates are challenging. Although many research efforts have been conducted in XML keyword search, none of them has addressed and resolved the above three issues yet.

For instance, one widely adopted approach so far is to find the smallest lowest common ancestor (SLCA) of all keywords. Each SLCA result of a keyword query contains all query keywords but has no subtree which also contains all the keywords. In particular, regarding to Issue 1 and 2, SLCA may introduce answers that are either irrelevant to user search intention, or answers that may not be meaningful or informative enough.

2. Literature survey

It is becoming increasingly popular to publish data on the Web in the form of XML documents. Current search engines, which are an indispensable tool for finding HTML documents, have two main drawbacks when it comes to searching for XML documents. First, it is not possible to pose queries that explicitly refer to meta-data (i.e., XML tags). Hence, it is difficult, and sometimes even impossible, to formulate a search query that incorporates semantic knowledge in a clear and precise way.
The second drawback is that search engines return references (i.e., links) to documents and not to specific fragments thereof. This is problematic, since large XML documents (e.g., the XML DBLP) may contain thousands of elements storing many pieces of information that are not necessarily related to each other. For example, an author is related to titles of papers she wrote, but not to titles of other papers [7]. Actually, if a search engine simply matches the search terms against the documents, it may return documents that do not answer the user's query.

2.1 Query Language For Xml

A query language for XML, such as XQuery, can be used to extract data from XML documents. However, such a query language is not an alternative to an XML search engine for several reasons. First, the syntax of XQuery is by far more complicated than the syntax of a standard search query. Hence, it is not appropriate for a naive user. Second, rather extensive knowledge of the document structure is requiring in order to correctly formulating a query. Thus, queries must be formulated on a per document basis.

Finally, XQuery lacks any mechanism for ranking answers an essential feature, since there are likely to be many answers when querying large XML documents.A number of XML query languages have been proposed, such as XPath, XML-QL, or the W3C standard XQuery. These languages combine SQL-style logical conditions over element names, content, and attributes with regular-expression pattern matching along entire paths of elements. The result of a query is a set of paths or sub graphs from a given data graph that represents an XML document collection; in information retrieval (IR) terminology this is called Boolean Retrieval. Boolean retrieval does not support relevance ranking [1].

2.2 Ranked Keyword Search

Evaluating keyword search queries over hierarchical XML documents, as opposed to (conceptually) flat HTML documents, introduces

many new challenges. First, XML keyword search queries do not always return entire documents, but can return deeply nested XML elements that contain the desired keywords. Second, the nested structure of XML implies that the notion of ranking is no longer at the granularity of a document, but at the granularity of an XML element.

Finally, the notion of keyword proximity is more complex in the hierarchical XML data model. Keyword search is to rank the query results so that the most relevant results appear first.XQuery is used to query XML documents. While this approach can be very effective in some cases, a downside is that users have to learn a complex query language and understand the schema of underlying XML.The results of keyword search queries over XML documents are of two possible semantics for keyword search queries. Under conjunctive keyword query semantics, elements that contain all of the query keywords are returned. Under disjunctive keyword query semantics, elements that contain at least one of the query keywords are returned.

The notion of proximity among keywords is more complex for XML. In HTML, proximity among keywords translates directly to the distance between keywords in a document. However, for XML, the distance between keywords is just one measure of proximity; the other measure of proximity is the distance between keywords and the result XML element[3].

2.3 Identifying Meaningful Return Information

XRank connects keyword matches by the LCA nodes that contain at least one occurrence of all keywords after excluding the occurrences of keywords in their descendants that already contain all keywords. XSEarch introduces the concept of interconnection. Two matches are interconnected and therefore should be in the same group if there are no two distinct nodes with the same tag on the path between these two nodes (through their LCA), excluding themselves and XKSearch group matches according to their meaningful LCA (MLCA) and smallest LCA (SLCA), respectively. An SLCA is the root of a subtree containing matches to all keywords, and it does not have a descendant whose subtree contains all keywords. Two nodes matching to different keywords are considered to be meaningfully related if their LCA is an SLCA; a set of nodes consisting of one match to each keyword is meaningfully related if every pair is meaningfully related, and a MLCA is defined as the LCA of these nodes.

There are several typical approaches for determining the return information. The first approach is to return the entire documents that contain keyword matches. Most of the existing approaches return the whole subtrees rooted at the LCA or its variants (MLCA, SLCA, etc) of keyword matches , named as subtree return. Alternatively, a tree containing the paths from an LCA node to key word matches can be returned, named as path return in the paper. Sometime there exist many path return subtrees, in order to output information concisely, first reduces each path to an edge labeled with the path length, and then groups the isomorphic reduced subtrees into a generalized tree. Finally, the return information can be specified by users or system administrators. Keyword requires a system administrator to split the schema graph into pieces, called Target Schema Segments (TSS). It also uses presentation graphs with expansion links to present data with multi value dependencies in a concise way. Précis is a keyword search system for relational databases. It determines the schema of the output by requiring users or system administrator to specify a weight for each edge in the schema graph. Then each user further needs to specify degree constraint and cardinality constraint in the schema for the cut-off [4].

2.4 Smallest Lca's In Xml Databases

Two efficient algorithms, Indexed Lookup Eager and Scan Eager, for keyword search in XML documents according to the SLCA semantics. Both algorithms produce part of the answers quickly so that users do not have to wait long to see the first few answers. The Indexed Lookup Eager algorithm outperforms known algorithms and Scan Eager by orders of magnitude when the keyword search includes at least one low frequency keyword along with high frequency keywords.

One widely adopted approach so far is to find the smallest lowest common ancestor (SLCA) of all keywords. Each SLCA result of a keyword query

contains all query keywords but has no subtree which also contains all the keywords. Those SLCA-based approaches only take the tree structure of XML data into consideration, without considering the semantics of the query and XML data. SLCA may introduce answers that are either irrelevant to user search intention, or answers that may not be meaningful or informative enough [8].

2.5 Search Methods

The earlier search method combines IR ranking and structural compactness based DB ranking to fulfill keyword search on heterogeneous data. XRANK and XSEarch are systems facilitating keyword search for XML documents, and they return connected subtrees as answers for keyword queries. XRANK presents a ranking method, where for a given tree T containing all the keywords, a score is assigned to T with an adaptation of PageRank for XML documents. XSEarch focuses on semantics and the ranking of results; during execution, it uses an all-pairs interconnection index to check the connectivity between nodes. XKeyword is a system that offers keyword proximity search over XML documents that conform to an XML schema. However, it needs to compute candidate networks and thus is constrained by schemas. TopX is a prototype search engine for the ranked retrieval of XML, but it processes XML queries with support for XPath axes but not the simpler keyword queries, and it cannot adapt to relational databases. SphereSearch engine provides unified ranked retrieval on heterogeneous XML and Web data. However, it is orthogonal to our method in that: i) it does not support relational databases and it transforms HTML documents into XML, and ii) it depends on its own query language to discover structural relationships and thus is not a pure keyword based search method[2].

2.6 Schema-Free Xquery

Extensive research has been done on structured declarative queries as well as on keyword based text search. In recent years, there have been interests in techniques that merge the two.In those studies, a database is viewed as a graph with objects/tuples as nodes and relationship as edges, and sub-graphs of the database are returned as answers to the original keyword query. Similar approach has also been taken to apply keyword search in XML documents (e.g., XKeyword and XRANK).

Ranking mechanisms have been applied to the search results such that results with perceived higher relevance are returned to the user first. All such keyword search approaches suffer from two drawbacks: (1) they do not distinguish tag name from textual content; (2) they cannot express complex query semantics. A number of attempts have also been made to sup port information retrieval style search by expanding XQuery or other structured query languages (e.g., XIRQL).

A recent closely related work is XSEarch, which attempts to return meaningful results based on query as well as document structure using a heuristic called interconnection relationship. In XSEarch, two nodes are considered to be semantically related if and only if there are no two distinct nodes with the same tag name on the path between these two nodes (excluding the two nodes themselves). Queries are allowed to specify tag names and attribute value pairs. However, interconnection does not work when two unrelated entities are present in entities of different types. For example, two author nodes may be considered as interconnected, even though one of them belongs to an article node and the other belongs to a book node.

Moreover, due to the simple query semantics used, XSEarch suffers from drawbacks similar to keyword search methods: difficulty to express complex knowledge semantics. The MLCAS operator, on the other hand, takes full advantage of well-defined XQuery, and enables the user to take more control of the search results without knowing the document structure. Finally, the system allows query answering across schemas by deploying schema mapping a query rewriting techniques. Users are still required to have extensive knowledge of at least one schema to pose queries [5].

3. Data Model

We model XML document as a rooted, labeled tree plus a set of directed IDRef edges between XML nodes, such as the one in Figure 1. Our approach exploits the prefix path of a node rather than its tag name for result retrieval and ranking. The existing works rely on DTD while our approach works without any XML schema information.

Node Type: The type of a node n in an XML document is the prefix path from root to n. Two

nodes are of the same node type if they share the same prefix path.

Data Node: The text values that is contained in the leaf node of XML data and have no tag name is defined as a data node.

Structural Node: An XML node labeled with a tag name is called a structural node. A structural node that contains other structural nodes as its children is called an internal node; otherwise, it is called a leaf node. In this paper, we do not consider the case that an internal node n contains both data nodes and structural nodes, as we can easily avoid it by adding a dummy structural node with a tag name say "value" between n and the data nodes during node indexing without altering the XML data.

With the above two definitions, the value part and structure part of the XML data is separated.

Single-valued Type: A structural node t is of single-valued type if each node of type t has at most one occurrence within its parent node.

Multi-valued Type: A structural node t is of multi-valued type if some node of type t has more than one occurrence within its parent node.

Grouping Type: An internal node t is defined as a grouping type if each node of type t contains child nodes of only one multi-valued type.

XML nodes of single-valued type and multi-valued type can be easily identified when parsing the data. A node of single-valued (or multi-valued, or grouping)type is called a single- valued (or multi-valued, or grouping) node.

4. Keyword Search And Motivation

With increasing volumes of XML data transferred over the Internet, retrieving relevant XML fragments in XML documents and databases is particularly important. Keyword search provides a simple and user-friendly query interface to access XML data in web and scientific applications. To identify relevant results for an XML keyword query, different systems use different underlying principles and heuristics, leading to different query results in general. How to guide the design and evaluate XML keyword search strategies is becoming a critical research problem.

However, due to the inherent ambiguity of search semantics, it is hard, if not impossible, to directly assess the relevance of query results and reason about various strategies. The extreme success of web search engines makes keyword search the most popular search model for ordinary users. As XML is becoming a standard in data representation, it is desirable to support keyword search in XML database. It is a user friendly way to query XML databases since it allows users to pose queries without the knowledge of complex query languages and the database schema.

4.1 Tree Model For Xml Keyword Search

In the tree model, SLCA (Smallest Lowest Common Ancestor) is a simple and effective semantics for XML keyword proximity search. Each SLCA result of a keyword query is an XML sub tree rooted at one XML node that satisfies two conditions. First, the node covers all keywords in its sub tree; second, it has no single proper descendant sub tree to cover all query keywords. However, the SLCA semantics based on the tree model does not capture ID reference information which is usually present and important in XML databases.

As a result, SLCA is insufficient to answer keyword queries that require the information in XML ID references and may return a large tree including irrelevant information for those cases. Moreover, SLCA results may not be a good choice for direct result display without using application semantic information. However, it is not informative to display just the title without other information of the course. In this case, it is better to display the information of the course with the matching title.

4.2 Graph Model For Xml Keyword Search

On the other hand, XML documents can be modeled as graphs (or digraphs) when ID reference edges are taken into account. With the graph (or digraph) model, a keyword search engine captures a richer semantics than that based on the tree model. The key concept in the existing semantics is called reduced sub graph. Although there exist very efficient algorithms on SLCA with the tree model, unfortunately, to our knowledge, there is no efficient algorithm for reduced sub graphs. The reason is twofold. Firstly, the number of all reduced sub graphs

may be exponential in the size of G. In contrast, the number of LCA sub trees is bounded by the size of the given XML tree. Note that different reduced sub graphs present different connected relationships in the real world; and most of them cannot be easily considered as redundant results.

Secondly, enumerating results are considered by increasing sizes of reduced sub graphs for ranking purposes according to the general assumption of XML keyword proximity search, this problem can be NP-hard; the well-known Group Steiner tree problem for graph can be reduced to it. Although there are a multitude of polynomial time approximation approaches that can produce solutions with bounded errors for minimal Steiner problem, they require an examination of the entire graph. These algorithms are not desirable since the overall graph of XML keyword search is often very large.

5. Principles Of Keyword Search In Xml

Compared with flat documents, keyword search in XML has its own features. We have considered the following three principles that the search engine should adopt.

Principle 1: When searching for XML nodes of desired type D via a single-valued node type V ideally only the values and structures nested in V - typed nodes can affect the relevance of D-typed nodes as answers, whereas the existence of other typed nodes nested in D-typed nodes should not. In other words, the size of the subtree rooted at a D-typed node d (except the subtree rooted at the search via node) shouldn't affect d's relevance to the query.

Example: When searching for customer nodes via street nodes using a keyword query "Art Street", a customer node (e.g. customer C1 in Figure 1) with the matching keyword "street" shouldn't be ranked lower than another customer node (e.g. customer C3 in Figure 1) without the matching keyword "street", regardless of the sizes, values and structures of other nodes nested in C1 and C3. Note this is different from the original TF*IDF similarity that has strong intuition to normalize the relevance score of each document with respect to its size (i.e. to normalize against long documents).

Principle 2: When searching for the desired node type D via a multi-valued node type V1, if there are many V1 typed nodes nested in one node d of type D, then the existence of one query-relevant node of type V1 is usually enough to indicate, d is more relevant to the query than another node d1 also of type D but with no nested V1-typed nodes containing the keyword(s). In other words, the relevance of a D-typed node which contains a query relevant V 1 typed node should not be affected (or normalized) too much by other query-irrelevant V1-typed nodes.

Example: Consider when searching for customers interested in art using the query "art", a customer with "art"-interest along with many other interests (e.g. C4 in Figure 1) should not be regarded as less relevant to the query than another customer who doesn't have "art"-interest but has "art street" in address (e.g. C1 in Figure 1). Compared to the existing works which blindly exploit the compactness of the query results in result ranking a significant difference of the above two principles is: the internal structure of a query result should be exploited as a critical factor to reflect the real relevance of the query results.

Principle 3: The proximity of keywords in a query is usually important to indicate the search intention.

We have considered the following properties monotonicity and consistency, for XML keyword search with respect to data and query.

- Data monotonicity,
- Query monotonicity,
- Data consistency,
- Query consistency.

Capturing the reasonable connection between an original query result and a new query result obtained after an update to the query or to the data. These properties are non-trivial, non- redundant, and satisfiable.

Monotonicity: Monotonicity describes the desirable change to the number of query results with respect to data updates and query updates.

Data Monotonicity: If a new node is added to the data, then the data content becomes richer, therefore the number of query results should be (non-strictly) monotonically increasing.

Query Monotonicity: If a keyword is added to the query, then the query becomes more restrictive, therefore the number of query results should be (non-strictly) monotonically decreasing.

Consistency: Describes how the content of query results should change upon an update to the data or query.

Data Consistency: After a data insertion, each additional subtree that becomes (part of) a query result should contain the newly inserted node.

Query Consistency: If a new keyword is added to the query, then each additional subtree that becomes (part of) a query result should contain at least one match to this keyword.

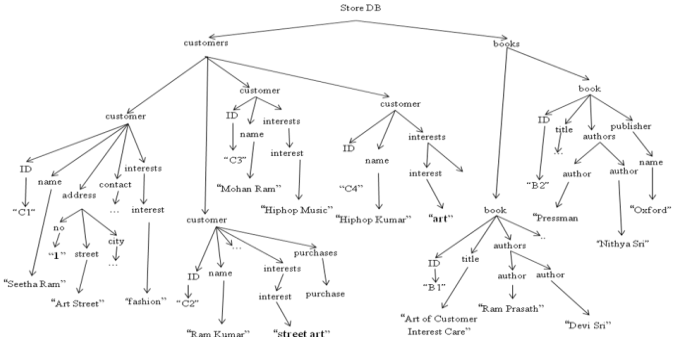

Fig.1, XML database for book store

5. Keyword Ordering

The order of keywords in a query is important to indicate the search intention. We have considered the input parameter as a keyword query which contains many keywords. Keyword query is splitted to single words to identify the node type by using the space separator.

The following are the steps to perform keyword ordering.

1. Get the Query as input
2. Get the xml Document
3. Split the Query using Space Separator
4. Store that in an array
5. Compare each sting in an array with xml document nodes
6. If that string appears as a node ,remove that string from that array
7. Finally the text to search is arrived

From the keyword ordering we can identify what type of the given query is,either it can be a node or a value. When the keywords are identified it will be useful for searching process.

6. Coherency Ranking

Coherency ranking (CR), a domain and database design-independent ranking method for XML keyword queries that is based on an extension of the concept of mutual information. With CR, the results of a keyword query are invariant under schema reorganization. Ideally, the query answer must include all portions of the data that are related to the query (high recall), and nothing unrelated (high precision). A ranking approach is proposed for keyword a query that exploits XML structure while avoiding overreliance on shallow structural details, and has higher precision and recall and better ranking quality than previous approaches. CR finds the most probable intention(s) for queries containing terms with multiple meanings.

7. Conclusion

In this paper, the problem of effective XML keyword search which includes the identification of user search intention will be solved by using keyword ordering followed by the searching mechanism and result ranking will be solved in the presence of keyword ambiguities using Coherency Ranking which gives relevance results according to the query and provide high recall and high precision. In future keyword search on XML database can be done with different file formats and images.

References:

1. Cohen.S, Mamou.J, Kanza.Y, and Sagiv.Y, (2003), 'XSEarch: A semantic search engine for XML', In Proc. of VLDB Conference, pp. 45–56.
2. Guoliang Li, Beng Chin Ooi Jianhua Feng, Jianyong Wang, Lizhu Zhou, (2008),'EASE: An Effective 3-

in-1 Keyword Search Method for Unstructured, Semi-structured and Structured Data', in ACM.

3. Guo.L, Shao.F, Botev.C, and Shanmugasundaram.J, (2003), 'XRANK: Ranked keyword search over XML documents', In SIGMOD.
4. Liu.Z and Chen.Y (2007), 'Identifying meaningful return information for xml keyword search', In SIGMOD Conference.
5. Li.Y, Yu.C, and Jagadish.H.V, (2004), 'Schema-free XQuery', In VLDB.
6. Luk.W, Leong.H.V, Dillon.S, Chan.T.S, Croft.W, and Allan.J, (2002),'A survey in indexing and searching xml documents', Journal of the American Society for Information Science and Technology, vol. 53, pp. 415–437.
7. Weimen He, (2008),'Searching and ranking Xml data in a distributed environment', The University of Texas
8. Xu.Y and Papakonstantinou.Y, (2005). 'Efficient keyword search for smallest LCAs in XML databases'. In SIGMOD, pages 537–538.