I.J. Education and Management Engineering **2012, 8, 1-5**
Published Online August 2012 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2012.08.01

Available online at **http://www.mecs-press.net/ijeme**

TakeXIR:a Type-Ahead Keyword Search Xml Information Retrieval System

[a] Yiqun Chen, [b] Jinyin Cao

[a] Guangdong University of Education Sun Yat-sen University Guangzhou, China
[b] The First Affiliated Hospital of Sun Yat-sen University Guangzhou, China

Abstract

As XML (eXtensible Markup Language) is becoming a standard widely applied in data representation and data exchange in web service. In this paper, we propose a novel approach to type-ahead keyword search in XML data, call TakeXIR. The IR-style approach basically utilizes the statistics of underlying XML data to address the following challenges in XML IR system: (1) Identify the user search intention, i.e. identify the keywords to express user interests and identify nodes user wants to search for and search via. (2) Resolve keyword ambiguity problems: synonyms and polysemy exist in natural language, and a keyword can appear as the text values or tag value of different XML node and carry different meanings.

Index Terms: type-ahead search; keyword search; XML information retrieval

1. Introduction

XML(eXtensible Markup Language) documents are one of the best tools for representing and transferring data because of their flexibility and self description, and the amount of data that is stored and exchanged in XML is increasing. The extreme success of web search engine makes keyword search the most popular search model for ordinary users. XML consists of elements and each element is a pair of matching start tags and end tags and all the text that appears between them. Since users can define tags freely, tags represent not only data itself but also data meaning. In traditional keyword search, a user composes a complete query to find relevant answers. If the user has limited knowledge about the targeted entities, often the user has to use a try-and-see method to modify the query and find information. One of the commonly used methods to solve this problem is auto-complete, which predicts a word or phrase that the user may type based on the partial query the user has entered. In this paper, we design an IR-style approach which basically utilizes the statistics of underlying XML data, to bring this type of search to the XML information retrieval system. Our system allows users to explore data as they type, thus brings instant gratification to the users in the search process.

Corresponding author:
E-mail address: [a] ChenYiqun@gmail.com

Our main contributions are summarized as follows:

- Addresses the keyword ambiguity problem in XML: the system can help the user identifying the keyword and finding the information.
- We design the Adapted Type-ahead index to support the type-ahead keyword search in XML document.
- We present the whole working procedure on how to support type ahead keyword search, how to find the LCA (lowest common ancestor), then compute XML TF*IDF, to calculate the confidence of each candidate node type to rank the search result and achieved smallest LCA.

2. Related works

2.1. XML and XML structure definition

XML data exists in two forms: XML documents and XML schemas. XML document defines data while XML schema defines structure. The structure of the XML documents can be declared by several languages: the older (but still more commonly used) Document Type Definition(DTD)[1], the more recent Schema definition[2], and RELAX NG[3].

No matter using which language to specify, a structure definition of the XML documents declares the possible hierarchical structure of the elements and the possible attributes of the elements with their data types. However, as an important concept of XML, the structure definition of XML is not taken full advantage in current applications.

2.2. XML Keyword Query

Keyword searching over XML introduces many new challenges. First, words ambiguity problems still exist and even become worse. Second, the result of the query is a nested XML element, maybe deeply. Third, XML and HTML keyword search queries differ in how query results are ranked.

In tree data model, LCA semantics is first proposed and studied in [4] to find XML nodes,each of which contains all query keywords within its subtree. XSeek [5] infers the semantics of the search, and generates the return nodes inferred by keyword match pattern and the concept of entities in XML data. XKeyword [6] provides keyword proximity search that conforms to an XML schema; but it is constrained by schemas. XReal[7] propose an IR-style approach which basically utilizes the statistics of underlying XML data. It designed novel formulae to identify the search for nodes and search via nodes of a query, and present a novel XML TF*IDF ranking strategy to rank the individual matches of all possible search intentions. XRANK[8] extends Web-like keyword search to XML, can be used to query a mix of HTML and XML documents. Results are ranked by a Page-Rank [9] hyperlink metric extended to XML. A recent work XSEarch[10] provides a semantic search engine for XML, focuses on the semantics and the ranking of the results. It extends information-retrieval techniques (tf*idf) to rank the results. [11] presents algorithms to compute the Minimum Connecting Trees (MCTs) of the nodes that contain the keywords, that is, the sub trees rooted at the LCAs of the nodes that contain the keywords. This is an optimized version of the LCA-finding stack algorithm which returns the set of LCAs along with efficiently (for performance and presentation purposes) summarized explanations on why each node is an LCA. [12] proposes a search technique called Schema-Free XQuery to enable users to query an XML document using XQuery without requiring full knowledge of the document schema. It presents a notion of Meaningful Lowest Common Ancestor Structure (MLCAS) for finding related nodes within an XML document

3. Frame work

We will hold a calculation for prefix matching index to support type-ahead search on the underlying XML data “on the fly” as the user types in query keywords. The whole procedure is shown in fig 1. After one keyword is typed, according to its appearance frequency in XML data, the system will suggest the most likely words the user may interest in, after the user selected the keywords, system will find out the related XML data, and rank the result, then present in the client.

A. Data Model

We model XML document as a rooted, labeled tree, such as the one in Fig 1. Our approach exploits the prefix-path of a node rather than its tag name for result retrieval and ranking.

δ-diameter Steiner trees[13]: a δ-diameter Steiner trees contain a complete word for each query keyword.

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

XML TF (fa,k): The number of occurrences of a keyword k in a given value node a in the XML database.

XML DF (f_k^T) : The number of T type nodes that contain keyword k in their sub-trees in the XML database.

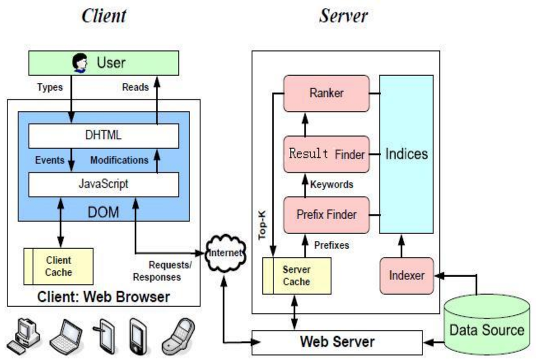

Fig 1. The Architecture of TakeXIR

3.1. Adapted Type Ahead search index

We adapted the trie index in Tastier[13] to index the tokenized words in the XML data. Each word w corresponds to a unique path from the root of the trie to a leaf node. Each node on the path has a label of a character in w. For simplicity, we use a trie node and its corresponding string (composed of characters from the root to the node) interchangeably. For each leaf node, we assign a unique ID to the corresponding word in the alphabetical order. Each leaf node has an inverted list of IDs of vertices in the database graph that contain the word. Each trie node has a keyword range of [L, U,N], where L and U are the minimum and maximal keyword ID in the sub-trie of the node, respectively, N is the appearance frequency of this keyword. It also maintains the size of the union of inverted lists of its descendant leaf nodes, which is exactly the number of vertices in the database graph that have the node string as a prefix.

To answer a single-keyword query, we consider the trie node corresponding to the prefix. (If this node does not exist, the answer is empty.) We access its descendant leaf nodes, and retrieve the graph vertices from their inverted lists as the answers to the query. Whenever the user types in more letters, we traverse the trie downwards to compute the answers and finally find the completed keywords.

Note that the words appear as the tag or as the node value are taken into account following the same way.

3.2. Search for & Search via confidence index

As discussed in XReal[7], given a keyword query q, the confidence of a node type T to be considered as the search for node is defined as

$$C_{for}(T,q) = \log_e(1 + \prod_{k\in q} f_k^T) * r^{depth(T)}$$

And confidence for search via node as:

$$C_{via}(T,q) = \log_e(1 + \sum_{k\in q} f_k^T)$$

As we have work out the Adapted Type Ahead search index and marked every leaf node, for each keyword, we can work out their inverted index according to the XML TF and XML DF.

3.3. XML TF & DF to search keyword and rank

Based on the Adapted Type Ahead search index and the related inverted lists built after pre-processing the XML document, we presents a flowchart of keyword search and result ranking as shown in the following:

- **Step 1.** When the user is typing the words, system will find ids of candidate words in the index and rank candidate keywords according to their frequency.
- **Step 2.** Check out the keywords' confidence in Search for & Search via index, infer the types of condition nodes that a keyword query intends to search for and search via.
- **Step 3.** Check out the node includes all keywords from the index, following the meet [4] operation, calculate the similarity and rank according to formula (3).

$$\rho_s(q,a) = \begin{cases} \dfrac{\sum_{k\in q\cap a} W_{q,k}^{T_a} * W_{a,k}}{W_q^{T_a} * W_a} & (a) \\ \dfrac{\sum_{c\in chd(a)} \rho_s(q,c) * C_{via}(T_c,q)}{W_a^q} & (b) \end{cases}$$

Where q represents a keyword query; a represents an XML node; and the result ρ_s(q, a) represents the similarity value between q and a.

A XML Document can be converted into a set of relationships following the Monet transform [4]. We use the meet [4] operation to find the SLCA of keyword query. Prior work has shown that Dewey numbers are a good id choice to represents the XML tree [14].

For more than two keywords, we input a set of relations (i.e., sets of nodes of different types) that contain the keywords and output all pair wise LCAs and not global LCAs. Notice that the nodes are grouped by type and not by keywords, so there could be pairwise LCAs that only contain the same keyword twice. We do the intersection operation to merge the answer to achieved SLCAs, and use formula (3) to rank the SLCAs.

4. Conclusions and Future Work

Keyword search is a proven, user-friendly way to query HTML documents in the World Wide Web. Due to the ubiquity and popularity of XML, users often are in the following situation: They want to query XML documents which contain potentially interesting information but they are unaware of the mark-up structure that is used; they can't tell exactly the query keywords. For example, it is easy to guess the contents of an XML bibliography file whereas the mark-up and the keywords depends on the methodological, cultural and personal background of the author(s).

This paper studies a new information-access paradigm that supports type-ahead search in XML information retrieval system. We proposed efficient index structures and algorithms for incrementally computing complete keywords to users' typing queries. In order to achieve an interactive speed on large data sets, we modeled XML data as a graph, analyses the identification of user search intention and result ranking in the presence of keyword ambiguities and use the related definition and formula to build a query prediction technique to improve search efficiency.

References

[1] XML DTD. ://www.w3.org/XML/
[2] XML schema. http://www.w3.org/XML/Schema
[3] RelaxNG. http://relaxng.org.
[4] A. Schmidt, M. L. Kersten, and M. Windhouwer, "Querying xml documents made easy: Nearest concept queries." in ICDE, 2001, pp. 321–329
[5] Z. Liu and Y. Chen, "Identifying meaningful return information for xml keyword search," in SIGMOD Conference, 2007.
[6] V. Hristidis, Y. Papakonstantinou, and A. Balmin, "Keyword proximity search on XML graphs," in ICDE, 2003, pp. 367–378.
[7] Zhifeng Bao, Tok Wang Ling , Jiaheng Lu.Effective XML Keyword Search with Relevance Oriented Ranking.ICDE 2009
[8] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked keyword search over XML documents. In SIGMOD, 2003.
[9] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, 30(1-7):107–117, 1998.
[10] S. Cohen, J. Namou, Y. Kanza, and Y. Sagiv. XSEarch: A semantic search engine for XML. In VLDB, 2003.
[11] V. Hristidis, N. Koudas, Y. Papakonstantinou, and D. Srivastava. Keyword Proximity Search in XML Trees. In IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING. Volume 18, Issue 4 pages525-539
[12] Y. Li, C. Yu, and H. V. Jagadish. Schema-free xquery. In VLDB, 2004.
[13] Guoliang Li, Shengyue Ji, Chen Li, Jianhua Feng. Efficient Type-Ahead Search on Relational Data: a TASTIER Approach. SIGMOD 2009.
[14] I. Tatarinov, S. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang. Storing and querying ordered XML using a relational database system. In SIGMOD, 2002