Keyword Search on XML Repository with Relevance Ranking

Swati Tonge
Lecturer, PCCOE
Pune

Rashmi Phalnikar
Asst. Professor, MITCOE
Pune

ABSTRACT
Extensible Markup Language (XML) is a simple text format which was designed to describe data using custom tags. The use of custom tags makes XML extremely flexible and enables it to not only describe structured data like information from a table of relational database but also semi-structured data. An XML document is self-describing which has made it a standard means of data exchange between applications and for use in configuration files of enterprise applications. The increasing preference to store and transmit data in the XML format has led to a need for searching these data stores for information. Query languages like Xpath and XQuery are used to retrieve information from xml document. But these query languages are complex for non expert user to learn. Keyword search allows such user to retrieve information without knowledge of complex query language. In this paper we proposed an algorithm for relevance ranking of nodes which retrieved as result by considering keyword ambiguity and intension of user.

General Terms
Information Retrieval.

Keywords
XML, Natural Processing, Keyword Search, Relevance Ranking.

1. INTRODUCTION
eXtensible Markup Language has become universal standard for representation and exchange over the Web due to its simple and flexible representation of data. This, in turn, has increased the demand for ad hoc techniques for XML query processing. As like in any data management application, XML-based systems can achieve effective and efficient query execution by providing a sufficiently expressive query language, such as XQuery [1] and XPath [2] for XML; but these query languages are very complex to understand/learn to the non expert user. Keyword search 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. Such query generates hundreds of nodes as a result which may have different percentage of relevance to query. It is required to rank these results so that the results can be displayed as per their relevance to the keyword query, just like the popular web engine that accepts the keyword query and the desirable text documents are displayed according to their relevance to the keyword query.

2. EXISTING SYSTEMS
Various methods like SLCA, LCA are used to find the smallest sub-structures in XML data that each contains all query keywords in either the tree data model or the directed graph (i.e. digraph) data model. LCA (Lowest common ancestor) basically returns node v which is a LCA of query keyword set K= {w1, w2,…, wk} if the sub-tree rooted at v contains at least one occurrence of all keywords in K, after excluding the sub-elements that already contain all keywords in K. Basically it outputs a node in the XML document that contains all the input keywords. But this can lead to false positive and false negative problems. A false positive problem occurs if the result set contains irrelevant nodes and the false negative problem occurs if some correct results are missing from the answer set. SLCA (Smallest Lowest common ancestor) returns node v if v is a LCA of K and no proper descendant of v is LCA of K [3, 4, and 5]. The SLCA can avoid these problems of LCA but suffers also from other false negative/positive problems. None of these methods has addressed user's intension and relevance ranking problem. Consider a keyword query "name jim gray" issued on the employee data in Figure 1. Most likely it intends to find the employees who having name "jim gray". If adopting SLCA, 3 results will be retrieved, the employee nodes with IDs from 1 to 3 (as these nodes contain "name", "jim" and "gray" in either the tag names or node values) in Figure 1. However, only 2 is desired which should be put as the top ranked one, then 1 and 3 can be displayed. Lastly 4 can be displayed since it is irrelevant to query. XSeek [6] infers the search intention based on the concept of objects and an analysis of the matching between keyword and data node. However, it does not address the ranking problem. XRANK [7], XKSEarch[8] consider only structural compactness of matching results, keyword proximity and similarity at node level

3. XML DATA MODEL
Xml document can be represented by tree model or graph model [9]. We have adopted tree model for our framework where xml document as an ordered tree where internal nodes represents tag element and leaf nodes represent content of the tag element. Nodetype of a node is the path of the node from root node to itself. Two nodes are of the same node type if they share the same prefix path. In figure 1 "e1"," e2"," id"," name" etc represents tag element and "1", "jim gray" are the content of tag element id, name etc. Nodetype of node "id" is "employee\e1\id".

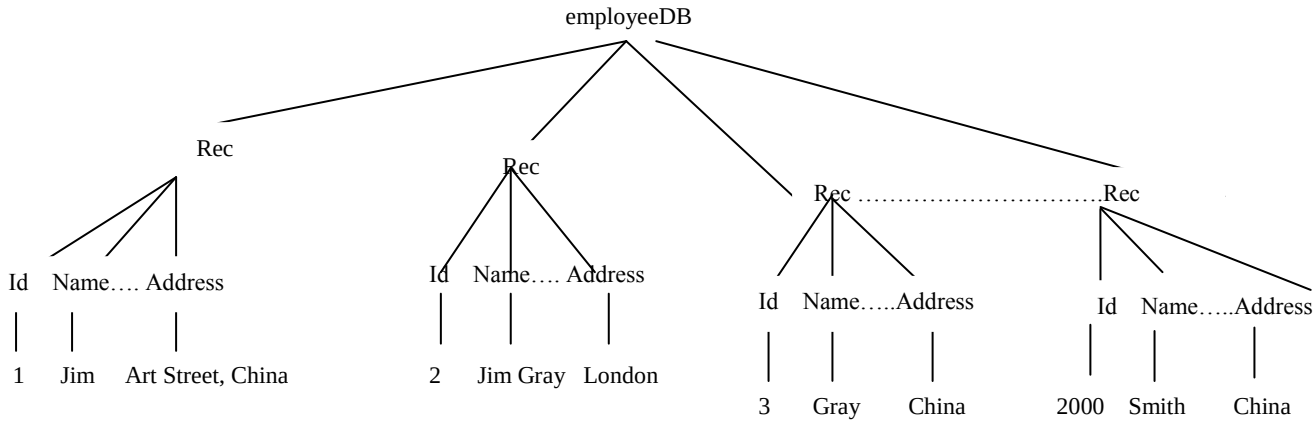

Fig 1: Portion of employee database

4. IDENTIFICATION OF SEARCH INTENSION

To identify the search's intension of user we have used terminology used in [10]. Search intension of keyword query is interpreted based on the statistics of data in xml document and the coocurrence of the keywords in a query. We have used equation 1 to find confidence of nodetype T with respect to a given query. All nodetypes with higher confidence value than predefined threshold value are selected as the search target since search target of the query may not be specified explicitly by user like in structured query language. The desired nodetype to search for is the first issue that needs to address in order to retrieve the relevant nodes. For given keyword query Q, a node type T is considered as the desired node to search for, only if the following conditions are satisfied [10]:

a. T is intuitively related to every query keyword in Q i.e. for each keyword k, there should be some (if not many) T-typed nodes containing k in their sub trees.

b. XML nodes of type T should be informative enough to contain enough relevant information.

c. XML nodes of type T should not be overwhelming to contain too much irrelevant information.

$$Confidence(T,Q) = \log_e(1 + \prod_{k\varepsilon Q} frequency(k,T) * R^{\text{depth }(T)} \quad \text{..... (1)}$$

where R is the reduction factor.

5. RELEVANCE RANKING IN FLAT DOCUMENT

In flat files term frequency * inter document frequency ratio is commonly used for relevance ranking given in equation 2 indicated by Sim(Q,d). Larger value of Sim(Q,d) indicate more relevance of d to query Q.

$$Sim(Q,d) = (\sum_{k\varepsilon Q\wedge d} W_{Q,k} * W_{d,k}\ )/W_Q * W_d \quad \text{..... (2)}$$

Where Q is the query ,d indicate document, $W_{Q,k}$ is the weight of keyword k in query Q ,$W_{d,k}$ is the weight of keyword k in document d ,W_Q is the weight of query and W_d is the weight of document. Following formulas are used to calculate $W_{Q,k}$, $W_{d,k}$, W_Q and W_d .

$$W_{Q,k} = log_e(\frac{N}{f_k+1}) \quad \text{..... (3)}$$

$$W_{d,k} = 1 + log_e(f_{d,k}) \quad \text{..... (4)}$$

$$W_Q = sqrt(\sum_{k\varepsilon Q} W_{Q,k}^2) \quad \text{..... (5)}$$

$$W_d = sqrt(\sum_{k\varepsilon d} W_{d,k}^2) \quad \text{..... (6)}$$

Where N is the total number of documents and f_k is the document frequency which is nothing but the number of documents containing keyword k. The term frequency $f_{d,k}$ in formula 4 is the number of occurrences of k in the document. W_Q and W_d are normalization factor used to balance between short and long documents. Formula 2 makes sure that the keyword appearing in many documents should not be regarded as being more important than the keyword appearing in few. Also a document with more occurrences of keywords in a query should not be regarded as being less important than a document that has less query keywords [10].

6. RELEVANCE RANKING IN XML DOCUMENT

Displaying whole xml document as the query result may not be useful for the user. Instead relevant part (sub trees) of the xml document can be displayed as the result. Since xml have hierarchical structure, formulas used for relevance calculation of text document cannot be apply for the xml document. We have used equation 7 to 9 to calculate similarity between a XML node N of the desired node type T to search for and a keyword query Q. These equations provides effective relevance ranking by omitting unnecessary calculation.

$$Sim(Q,N) = \sum_{for\ all\ leaf\ nodes\ n\ of\ N\ having\ k} Sim(Q,n) \quad (7)$$

$$Sim(Q,n) = (\sum_{k\varepsilon Q\wedge n} W_{Q,k}^{T_n} * W(k,n)/W(Q,T_n) * W_n) * Confidence(T,Q) \quad \text{..... (8)}$$

$$W_{Q,k}^{T_n} = COR(Q,a,k) * log_e\ (1 + N_{T_n}\ /(1 + f_k^{T_n}) \quad \text{..... (9)}$$

$$W(k,n) = 1 + log_e(f_{k,n}) \quad \text{..... (10)}$$

$W_n = sqrt(\sum_{k \varepsilon Q} W^2(k,n))$……..…… (11)

$W(Q,T_n) = sqrt(\sum_{k \varepsilon Q} W_{Q,k}^{T_n\ 2})$……..…… (12)

In equation 8 $W^{Tn}_{Q,k}$ is the Weight of keyword k in query Q with respect to nodetype T_n . T_n is the nodetype of parent of n. W(k,n) is the weight of keyword k in leaf node n. W(Q,T_n) is Weight of query Q in T_n nodetype. W_n is weight of node n. In order to capture proximity of keyword kt matching the node setvalue in both query and XML data node respectively, for each query-matching leaf node n in XML data, equation 9 is used. In equation 9 COR(Q,a,k) finds the co occurrence of keyword k in leaf node n using dist. dist is maximum of query distance and structural distance. Query distance is the position distance between k and kt. Structural distance is depth distance between node n and its nearest ancestor having kt. $f_{k,n}$ is the frequency of keyword k in leaf node n. f_k^{Tn} is the frequency of keyword k in nodetype T_n .

7. DATA P ROCESSING AND INDEX CONSTRUCTION

While parsing the each xml document, we have collected following information of every node n.

a. Assign dewey id[12,13]
b. Assign nodetype using global hashtable
c. If it is leaf node then calculate normalization factor W_n and frequency of keyword in a node $f_{k,n}$

We have created two indexes, Frequency table and an inverted list of keywords which retrieves a list of leaf nodes having input query keywords, in document order. Frequency table is a matrix of all keywords in the document and nodetypes. It stores frequency of keyword appearing in each nodetype.

8. ALGORITHM

In step 1-3 , confidence of each nodetype are calculated and whose having confidence value greater than threshold value are selected as desired nodetype to be search for. Then the lists of all leaf nodes corresponding to the keywords in query are extracted. In step 5 relevance of each node of nodetype selected in 2 is calculated.

Key_Search(Q[m],Frequency_matrix, Key__IList[m])

1. For each document d in repository do
Calculate Confidence (T,Q) of each node type for each keyword $\in$ q $\cap$ d
2. Choose all the nodetype greater than threshold value
3. Select all nodes N for chosen nodetype in step 3.
4. Get Inverted Key__IList(IL) of all the leaf nodes in document order for each keyword in query
5. For each node N_i in set N do
While (!end(Key__IIL[1]) ….(!end(Key__IIL[m]))) do
Node *n* = getMinimum(Key__IIL[1],Key__IIL[2],...,
Key__IIL[m])
if(isAncestor(N_i , n))
Calculate Similarity Sim(Q,N_i)+=Getsim(Q,n)
else
Set Sim(Q,N_i)=0
6. Sort N in descending order of S
7 Return N

Getsim(q,n)
For each k $\in$ q $\cap$ n
{
COR(q,n,k)=getQueryKWCo-occur(q,n,k)
$W^{Tn}_{Q,k}$+=COR(Q,a,k) * $\log_e(1+N_{Tn}/(1+ f_k^{Tn})$
}
W (n,k) = 1+log ($f_{n,k}$);
Sum += $W^{Ta}_{q,k}$* $W_{n,k}$
Return Sum/ W^{Tn}_{q} *Wn

9. EXPERIMENTAL SETUP

This framework is implemented in Java and run on a 1.87GHz Pentium 4 machine with 1GB RAM running Windows XP. The indexing and search performance of the search tool has been tested with mainly the DBLP (small), Orders, Wsu, ebay and SigmodRecord datasets [16].

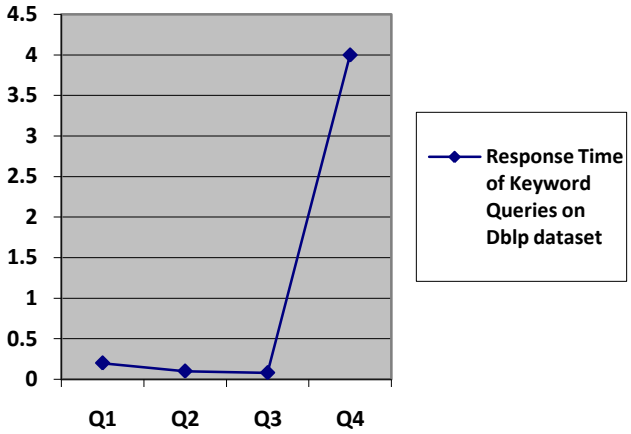

Fig 2. Response time in sec on DBLP dataset

Q1: author,chen,lei
Q2: jim, gray,article
Q3: xml,twig
Q4: title,acm,year, 2000

These datasets represent both document and text centric XML data [16]. Comprehensive experiments are performed to compare the effectiveness, efficiency and scalability of our framework. The datasets along with the total indexing time are reported in Table 1. It has been observed that our framework is able to infer a desired search for node in most queries, especially when the search for node is not given explicitly in the query. Execution time for some of the queries on dblp dataset is listed in fig 2.

10. CONCLUSION

Proposed method allow user to retrieve information without knowledge of complex query processing language from xml document. We have proposed novel method to calculate the relevance of retrieved nodes by considering the hierarchical structure of xml document.

Table 1. Data and Indexing Time

Sr.No	Dataset[16]	Indexing Time(sec)
1	Orders	3.5
2	Wsu	1.17
3	Dblp (small)	5.01

4	Book	0.2
5	eBay	0.047
6	SigmodRecord	0.4

It is also includes the identification of user search intention, keyword ambiguities and the keyword proximity.

11. REFERENCES

[1] S. Boag, D. Chamberlin, and M. F. Fernandez. XQuery 1.0: An XML query language. W3C Working Draft 22 August 2003.

[2] A. Berglund, S. Boag, and D. Chamberlin. XML path language (XPath) 2.0. W3C Working Draft 23 July 2004.

[3] Y.Xu and Y.Papakonstantinou."Efficient Keyword Search for Smallest LCAs in XML Databases", SIGMOD,2005.

[4] Haitao Wu Zhenmin Tang , "An Efficient Algorithm for Meaningful SLCA in XML Keyword Search ",IEEE, 2009.

[5] Chong Sun, Chee-Yong Chan "Multiway SLCA-based Keyword Search in XML Data", WWW 2007.

[6] Ziyang Liu, Jeffrey Walker, Yi Chen "XSeek: A Semantic XML Search Engine Using Keywords" xseek.asu.edu/xseekdemo.pdf

[7] Lin Guo, Feng Shao ,Chavdar Botev, Jayavel Shanmugasundaram "XRANK: Ranked Keyword " Search over XML Documents, SIGMOD 2003.

[8] S.Cohen,J.Mamou,Y.Kanza,andY.Sagiv."XSEarch: A Semantic Search Engine for XML", VLDB, 2003.

[9] Gang GOU , Rada Chirkova "Efficient querying large XML data Repositoties : A Survey" IEEE 2007.

[10] Zhifeng Bao, Jiaheng Lu, Tok Wang Ling and Bo Chen, "Towards an Effective XML Keyword Search" , IEEE 2010.

[11] Albrecht Schmidt Martin Kersten Menzo Windhouwer "Querying XML Documents Made Easy:Nearest Concept Queries" , IEEE 2001

[12] V. Vesper. Let's do dewey. http://www.mtsu.edu/ vvesper/dewey.html.

[13] Li Ying ,Ma Jun, Sun Yuyin "Applying Dewey Encoding to Construct XML Index for Path and Keyword Query " IEEE, 2009.

[14] Arash Termehchy "Keyword and Natural Language Query Processing for Semi-Structured ata Sources" , IDAR 2009.

[15] Yi Chen, Wei Wang, Ziyang Liu, Xuemin Lin "Keyword Search on Structured and Semi- Structured Data" , SIGMOD'09.

[16] XMLData Repository: http://www.cs.washington.edu/ research/xmldatasets.