International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 12 (2015) pp. 32297-32310

http://www.ripublication.com

An Efficient XML Keyword Search With Efficient Ranking Technique

Dayananda P and Dr. Rajashree Shettar

[1]*Department of Information Science and Engineering, MSRIT, Bangalore*
[2]*Department of Computer Science and Engineering, RVCE, Bangalore*
[1]*dayanandap@msrit.edu,* [2]*rajashreeshettar@rvce.edu.in*

Abstract

Web users can search keyword in XML database using search engine without any knowledge of complex structured query language and XML schemas. The Keyword search will be achieved by searching and ranking process. An efficient searching and ranking method for keyword search is proposed in this paper. The main idea behind searching the keyword in the method is to select the data nodes that match the keywords and to connect them in a meaningful way. As the users are just interested in the relevant part of the results it is meaningless to provide all the connected sub trees to the users. Therefore, the proposed method will not consider the files which are not containing the keyword searched by the user. The main aim is to create a keyword search approach that utilizes the statistics of underlying XML data to return the most promising or relevant result. All the files containing the searched keyword will be assigned with a rank after applying all the techniques of indexing and searching. The most relevant result will be the file with the highest rank. The comparison of proposed technique with existing technique is done with respect to execution time and relevance. The result shows that, proposed technique is better than the existing techniques.

KEYWORDS: XML Keyword search, indexing, scoring, Ranking

I. INTRODUCTION

Simplicity is one of the major advantages of XML keyword search. There is no need to learn complex query language (i.e., X-Query or X-Path) by user, or know the underlying data structure. One disadvantage of this simple query format is that it may not be precise and can return a vast number of results that can be divided into various types. Among those large numbers of results only few are of interest to the users. In

this work, the result types will be automatically derive from the data to be searched by the system and neglects the complex and time consuming process of testing all the result types by the user. The approach which is used in this paper is to find the most relevant or promising result by first compute the query results and then followed by rank them using the ranking function. The rank of the result type denotes the calculated score of the query result, which is calculated by using the terms such as term frequency, inverse element frequency, weight, distance etc. The returned query results are ranked according to their scores. The most promising or relevant result returned is the result with the highest corresponding rank.

This paper proposes a new approach for effectively and efficiently identifying the result type for a keyword search on XML data. Consideration of all the keyword query answers corresponding to different result types is our main idea, which avoids the limitations of the past approaches. Our proposed approach does not require a schema of the XML data. In this work, the suggestion of new improved ranking function is done to predict the most relevant result type.

The main aim of this work is to develop an effective technique for keyword search over XML data. In this paper, two keyword searching methods are discussed which are used for calculating the most relevant result. Existing method uses the factors such as, TF (term frequency) and IEF (inverse element frequency) for calculating the score of the query results in order to determine their rank. While, the proposed method in addition to the existing factors make use of an additional factor IDF (inverse document frequency) to determine the rank of the query result through score calculation. Above mentioned methods are used for experimentation and evaluation. The corresponding results from both the methods are used to conclude the proposed method is retrieving more relevant result for a keyword search.

The proposed method consists of the following steps:

1) Indexing: An XML files are considered as input for indexing. Indexing will be done by forming a table depicting each matching elements of the file along with its parent tag name, and its distance from the root. It makes the process of searching faster. The total number of elements contained in the input file is also calculated in this step.

2) Searching and Ranking: After the completion of indexing step, the keyword will be provided which should be searched in the indexed XML dataset. Then it calculates the TF (term frequency) of the keyword in each file, after that it calculates IEF (inverse element frequency), IDF (inverse document frequency), and it makes use of the total number of documents and the distance of the keyword from the root. The calculation of weight for each keyword is done based on each corresponding file and then finally calculates the aggregated score of the keyword in each file. The ranking order is done using TF*((IDF+IEF)/2) strategy

II. RELATED WORK

Keyword search is emerged an effective way to search for a record in web.XML keyword search removes the tediousness of learning effort for the user such that in order to search for a record in web the user need not to learn the structure of the document. Ranking the files containing the Keyword is the important component of the searching system. Where the XML dataset is converted into separate files and the files are ranked based on preciseness of the query present in files. XML search provides easier interface to search for a query in the web document. XML searching is the solution for the eradication Of limits and draw backs for searching through html engine, where if user gives query to the XML search, it will return full text document along with the searched results it also gives added information's such as full text titles, references to the searched query for every keyword subsections following the document containing the query. This is effective for a user in various ways and gives extra knowledge about the searched query. XML keyword search is the new method of searching where a non database user can easily feed his query input for search irrespective of data structure and without need of learning special measures to follow in order to search, for example in order to search in sql dataset a user is recommended to learn the structure of the sql data structure and the query processing mechanism. Recently research did by database community reveals the fact that the existing approaches of searching are not user friendly. So the XML dataset search can turn as the future of searching for a record in web. If the keyword search taken into further research and development works then XML keyword search can withstand each and every draw backs of existing searching strategies.

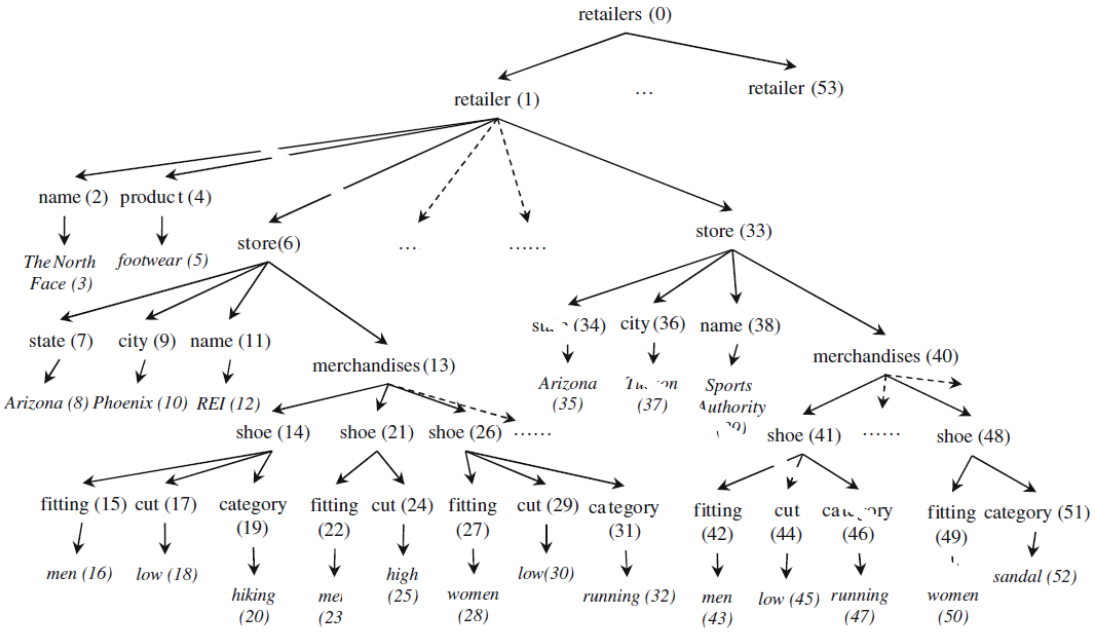

Fig1: XML DATABASE STRUCTURE

Techniques used for keyword search on XML are:

i) Relevant matches

Every keyword can have more than one match in the XML document. Few of them are relevant to the query while others aren't. As a result is necessary to find relevant keyword matches and produce the result appropriately.

Example:-If the user queries "REI, city" in the data given below then the user finds the cities that have REI stores. There are multiple "city" nodes, but only the one labeled node9 is relevant to the query.

ii) Relevant non-matches

Only returning relevant matches is not always meaningful as few non-match nodes may also be of interest to the users depending on queries and data.

Example:-As the query taken in above example which was "REI, city", now we take another query as "REI, pheonix". Now we see that these two queries have similar relevant matches but here the results expected would be different as the users involved have different approaches. In the case of first query the user seems to be interested in the info regarding a store with the name REI and is located in phoenix. While on the other hand in case of second query the user seems to be interested in a part of info – city REI.

From the above two queries we observe that a query has two parts which are return node and predicate. The predicate function is to restrict the search. e.g. The "where" clause in XQuery. While the return node function is to return the information that the user is looking for.e.g. The "return" clause in XQuery. In the above example, in the second query, REI is a predicate and city is a return node, so as a result the child node phoenix of city node is a relevant non-match node and should be returned.

For searching a keyword in a XML database the following searching technique are used:

i) SLCA (smallest lowest common ancestor):-For a keyword proximity search, tree model is effective and simple semantics. The smallest XML node will be returned for keyword query which contains all keywords in its descendants and has no single proper descendant to cover all query keywords. The SLCA semantics does not catch ID reference data that is generally available and significant in XML databases. It may return a large tree consisting of irrelevant data.

ii) XSEarch :-There are two XSEarch[3] semantics which are star semantics and all pair semantics. In both semantics every result is a pattern match. In order to identify the qualified pattern matches, the interconnection relationship was proposed.

iii) MLCA:-This concept was proposed with schema free XQuery that allows user to combine structured query and keyword search whenever it seems beneficial. Every result under this semantics is a pattern match, where every two nodes are meaningfully related. This relatedness used in MLCA has less dependency on node labels, and it may perform better on heterogeneous data.

iv) CVLCA (compact valuable LCA):-For a query Q on XML data D, a node n is considered as a valuable LCA if there is a pattern match that satisfies all pair semantics. Node n is a compact valuable least common ancestor, if n is the LCA[11] of a pattern match P, and dominates all nodes in P.

v) ELCA (exclusive lowest common ancestor):-It is a superset of SLCA[12] for a given query and data. A node n is an ELCA of a keyword search if it contains at least one match to each keyword in its sub tree after removing the sub tree rooted at each node m, where m is a child of n and the sub tree rooted at m also contain at least one match to each keyword.

iii) Ranking

It is impossible to avoid irrelevant results from being generated by the search engines because it is very difficult for users to exactly describe their intentions as the keyword searches are very ambiguous. The ranking of the query result is done on the basis of its score.

The score depends on following factors

i. Inverse document frequency(IDF) and term frequency(TF)

Term frequency(TF) and inverse term frequency(IDF) are majorly used for calculating the relevance of a Word in a document. Calculation of TF id done for a term by dividing the number of occurrences of the term t in the document d by the Number of occurrences of all terms in document d. Whereas the IDF is calculated by taking the log function of the division of the total number of documents in the file by the number of documents where the term t appears.

ii. Proximity ranking factors

These ranking factors are based on the probability that 2 nodes which are close to each other have a close relationship, therefore a result in which the nodes are close to each other is expected to be relevant. The result proximity can be calculated by the product of Element rank[7] and the decay which measures the proximity of keyword matches.

iii. Page Rank style ranking factors

To analyze the importance of a web page, the search engines use Page Rank[2] as a link analysis algorithm. It is likely that a web page linked by many important web pages is important. The probability that a web surfer, who may randomly click on web

pages and randomly jump to different pages, will arrive at a page determines the Page Rank of this particular page. An initial Page Rank is assigned to each web page, which is computed iteratively until the final result.

iv. Index and Materialized views

They are widely used in answering structured queries and has achieved a great success in pacing up query processing. Various common operations are supported by Indexes on the data efficiently, whereas a query whose results are calculated and stored physically, which can be used to answer new queries is called a materialized view. In order to achieve improvements in efficiency, they both can be used in XML keyword search.

v. IEF based technique

It is the current existing approach for finding the relevant result types. IEF (inverse element frequency)[1] is defined as the total number of elements in the XML data tree over the number of elements that contain the token in the sub tree rooted at the element in question. After the IEF is calculated it is used to find the weight of the keyword in each corresponding file. Along with IEF another factor term frequency (TF) is also used to calculate the weight of the keyword.

The following formula is used to do so:-

$$\text{Weight} = \log_2(1+tf).\log_2(ief) \quad (1)$$

After calculated the weight shown in equation(1) of each keyword, it is used to calculate the score of each keyword. The score[1] of the required keyword in each particular file is the aggregation of the division of weight of that keyword with the distance of that keyword from the root node. Using this formula the score of the required keyword in each containing file is calculated. After the score of all required keyword is calculated, finally all the files containing the required keywords are ranked according to the score shown in equation(2) of those keywords calculated by above mentioned methods.

$$Score(R, Q) = \sum_{i=1}^{K} \frac{weight(v_i)}{dist(N, v_i)} \quad (2)$$

III. PROPOSED TECHNIQUE

The figure 2 shows the overall components of proposed system. It contains the following components:

1. Indexing

Indexing is the basic component of the work carried down which involves to find each keywords in the dataset and indexing is carried down to each and every keywords. Such that distance from each and every keyword to the root of the data set is calculated. The distance calculated is thus used for upcoming modules to find the score.indexing will split up the imported dataset into separate files of a set of tag together which will be helpful to handle with large amount of data efficiently.

2. Searching

Searching for keyword in whole set of keywords is tedious job. The part of searching is made easier from the indexing the dataset into separate files. Hence once the dataset have been splitted into set of files containing keyword and the searching part is easier to search in the files and not in whole dataset at a time. once the keyword which has been given by the user is matched with the keyword in the file the frequency of the occurrence of keyword is saved to the term frequency list.

3. Inverted Element Frequency and Inverted Document Frequency calculation

Inverted Element Frequency is obtained by total number of files divided by number of files containing the keyword. Here come the searched results into use that all the search results which have been saved into he keywords list and the specific file will be directly. Inverted Document Frequency is obtained by total number of documents divided by number of documents containing the keyword. The searched results which have been saved into the keywords list and the specific file will be directly redirected to the IDFList storage array.

4. weight calculation and Score calculation

As depicted in algorithm weight will be calculated as per the formula. where it retrieves the term frequency, inverted element frequency and inverted document frequency from the IEFList, IDFlist and tfList. Weight calculated is hence saved to the weight List will be redirected to the score calculation. For each file in files the keywords the distance of each keyword will be redirected and then weight of each keyword will be divided with the distance of each keyword and then the result will be stored in scoreList.

5. Score aggregation and Ranking

Summation of all the results stored in scoreList will be calculated and then the actual score for a file containing the keyword will be shown and flown to the ranking procedure. sortDescending or sortAscending function are used to rank the obtained aggregated score of each file to either ascending or descending order based on the user Convenience.

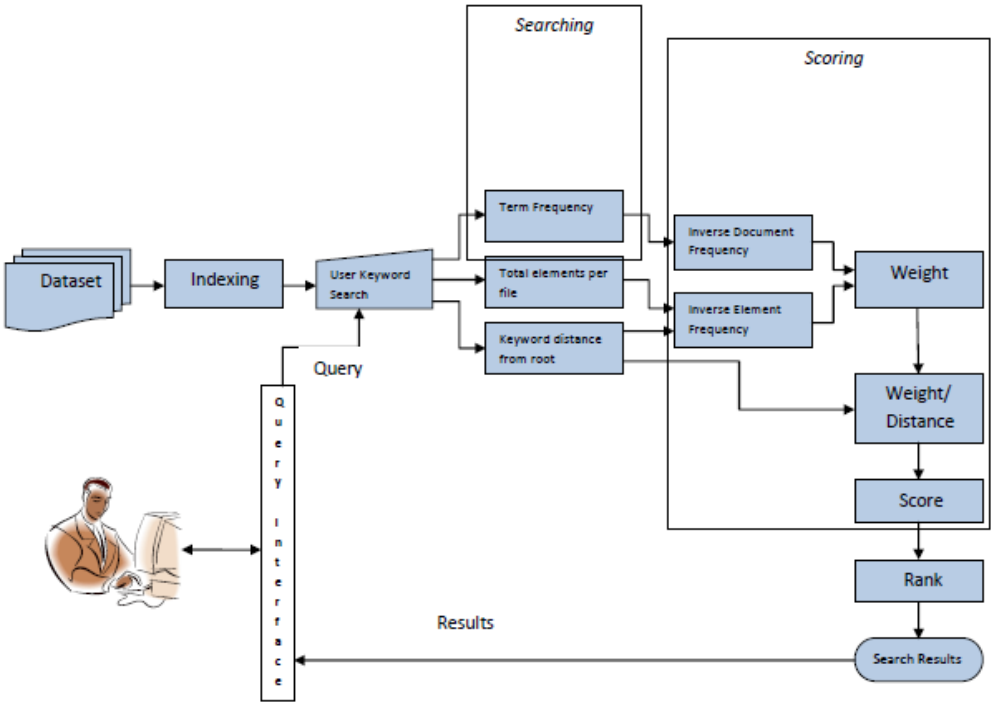

Fig2: Architecture of the system

In algorithm 1, Calculation of Inverse Element Frequency for each keyword is done for each file. IEF is calculated using the formula-No. of elements in file/No. of elements containing the keyword. And also the calculation of Inverse Document Frequency by using the Formula-No. of Files in dataset/No. of files containing the keyword. Calculation of weight for each keyword is done for each file. The following formula used to calculate the score is: (log2 (TF+1)*log2 (ief)+log2 (TF+1)*log2 (idf))/2. Calculating the weight/distance for each keyword in each file is also done. For calculating the weight/distance, the function uses the weight of each keyword in each file and the distance from root of each element containing the keyword in each file. This creates multiple weight/distance entries for each keyword in each file according to the number of elements in each file containing the keyword. The calculation of score of each file is done by adding the weight/distance of each keyword in each file. Finally calculate the rank of each file by comparing the scores of each file.

$$\textbf{Score calculation} = (\log_2(TF+1)*\log_2(ief)+\log_2(TF+1)*\log_2(idf))/2 \quad (3)$$

In comparison with the XBridge score calculation technique and the proposed new technique of calculating score have concluded that score values are better than

the XBridge score values. Ranking is the important component of the XML dataset Keyword search, where files are ranked based on the obtained score values.As the proposed formula of calculating score in equation(3) where the average of Inverted Element Frequency and Inverted Document Frequency is calculated.

The obtained score values in Proposed score calculation had better score values which gives better ranking of the files. The preciseness of the searched keyword is efficiently obtained proposed technique rather than existing XBridge technique[1]. The preciseness of ranked files is depicted using various queries of XML datasets.

IV. RESULTS AND COMPARISON

Our proposed method for keyword search over XML data was experimented by implementing our approach using JAVA software (jdk-1.6 version) on 3.20GHz Intel(R) Pentium(R) D, 1.00GB RAM, and 32-bit operating system with windows 7 professional. The experimental results obtained are tabulated and these results are compared with the existing method of Score. The results generated and compared are tested for the real datasets; viz., DBLP, Yahoo, and Reed. The query notation is used for examining results on three different XML datasets. Table1 shows how query under examination have been made to give particular notation for each keyword and for every dataset.

- DBLP dataset has notations QD1, QD2, QD3, QD4, QD5, QD6.
- YAHOO dataset notations QY1, QY2, QY3, QY4, QY5.
- REED dataset has notations QR1, QR2, QR3, QR4, QR5.

The Xbridge[1] is the existing method used for calculating scores by using Term frequency and Inverted element frequency. Xbridge Score= log2(TF+1)*log2(ief), whereas for proposed method score is calculated based on the Term frequency, Inverted element frequency and Inverted document frequency. Proposed method uses

$$\textbf{Score= (log2(TF+1)*log2(ief)+log2(TF+1)*log2(idf))/2.}$$

Algorithm 1:

```
Input: XML dataset ,keyword;
Keyword( )
{//indexing//
foreach keyword in keywords  {
for each file in files {
If (file contains keyword) { //searching//
tfList.add(filename,tf);
//tf=no.of times keyword occurs;
If(index contains keyword)
{//IDF,IEF calculation//
IDF=totalNoOfDocuments/NoOfDocumentsContainingKeyword;
foreach file in files(){
IEF=totalNoOfElements/NoOfElementsPerFileContainingKeyword;
distanceList.add(filename,keyword,distance);
 //distance=distance from root to keyword;
 } } }
For each file in files() {
totalNoOfElementsList.add(filename,keyword,elementCount);
foreach keyword in keywords() {
iefList=add(filename,keyword,ief);
idfList=add(filename,keyword,idf);   }}
foreach keyword in keywords {    //score calculation//
 foreach file in files{
 weight=log2(1+tf)*log2(IEF);
 weightList=add(filename,keyword,weight);}}
foreach item in distancelist{
weightByDistance=weight of file containing the element
                          /distance of the element from root;
weightdistanceList.add(filename,keyword,weight/distance);}
 foreach file in files{// score aggregation//
 score=£ weightByDistance
 scoreList.add(filename,keyword,score);  }
// Ranking//
RankList=scoreList.sortDescennding; }}
```

The efficiency test is addressed by evaluating the query response time adopting our proposed method designing the indices for keyword information discussed above. This is executed by measuring the time taken to search for the file relevant to the given query. The response time of individual queries under test is represented in Table 1. Proposed method is compared with the Xbridge Method. In case of DBLP, Yahoo and reed real dataset it is observed that our approach is faster than Xbridge. Fig. 5 shows the response time in seconds on individual queries DBLP, yahoo and reed database

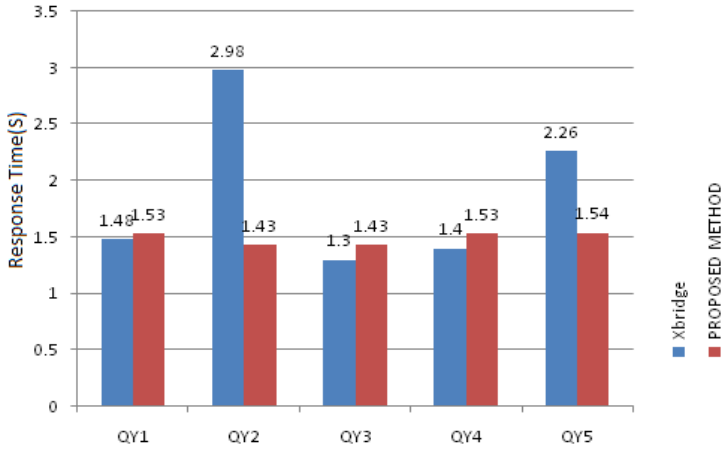

Fig3 : Response time on Yahoo dataset

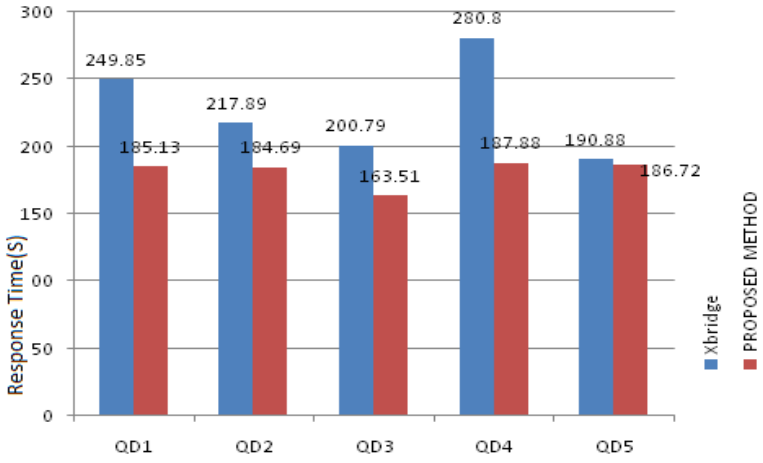

Fig4 : Response time on DBLP dataset

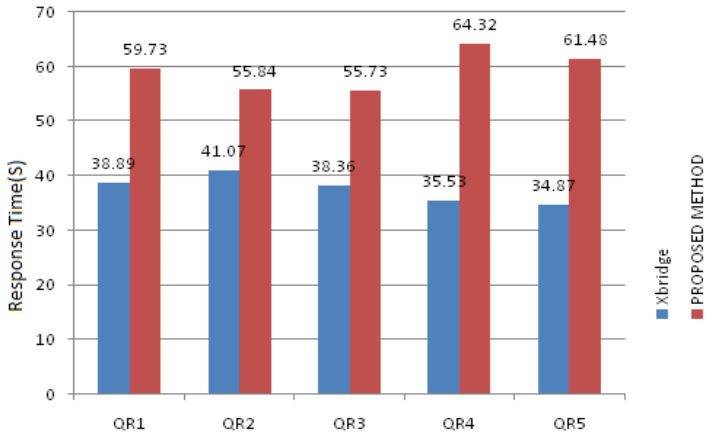

Fig4: Response time on Reed dataset

The Fig 3 shows the repose time for Individual query on yahoo datasets, Fig4 shows the repose time for Individual query on DBLP dataset and Fig5 shows the repose time for Individual query on Reed datasets. The proposed method take more time over existing XBridge method but the proposed method retrieved results are more accurate and relevant result than existing data. The table1, table2 and table3 are used for query notation used for examining results on three different XML datasets.

Table1: DBLP DATASET

Notation **DBLP DATASET**	Query under test
QD1	"sanjay"
QD2	"KURT"
QD3	"SANJAY KURT"
QD4	"1993"
QD5	"sanjay 1993"
QD6	"KURT 1993"

Table2: Yahoo DATASET

Notation **YAHOODATASET**	Query under test
QY1	"seller"
QY2	"mike"
QY3	"seller mike"
QY4	"ultra"
QY5	"seller ultra"
QY6	"mike ultra"

Table3: Reed DATASET

Notation **REED DATASET**	Query under test
QR1	"brightman"
QR2	"parker"
QR3	"brightman parker"
QR4	"f01"
QR5	"brightman f01"
QR6	"parker f01"

The score value is observed by both proposed method and XBridge method and is shown in Fig 6, fig7. The higher the score value is higher the ranking of file for a particular keyword searched. Proposed method gets more relevant result than Xbridge method among dataset considered above. The results are observed from two data sets corresponding to the proposed and existing method. Proposed method

retrieves more relevant result. The change in ranking here occurs because the new weight takes into account with reference to the number of files the keyword exists in. As the number of files containing one keyword is more important than the number of files containing other keyword, the file containing keywords had higher weight, which further influenced the scores of the file containing it. Using proposed technique the experimental results shows the precision is higher than the Xbridge technique

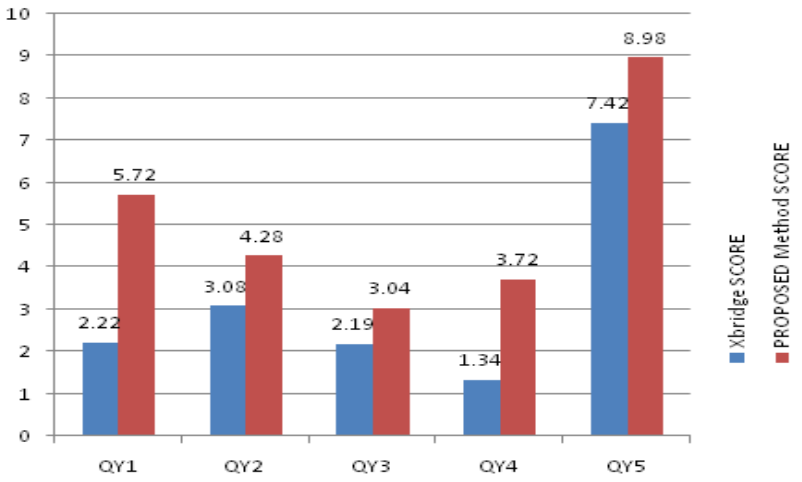

Fig6: Score on Yahoo Dataset

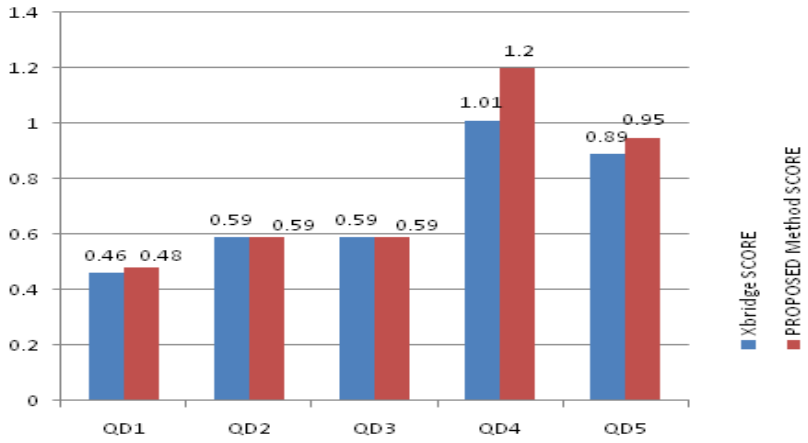

Fig6: Score on DBLP data set

V. CONCLUSION

In this paper, a new XML keyword searching is proposed that can retrieve relevant results type. Based on the statistics of TF, IEF and IDF the calculation of rank takes place in our work. Inverse element frequency will be considered based on total no. of elements over the no. of elements containing the keyword, while Inverse document frequency takes into

account with the log of total no. of documents over the no. of documents containing the keyword. As deducible from the formula that the value of "IDF" for a particular keyword will remain constant. While the value of IEF will always be different for a keyword in different files. In the propose work the Average of IEF and IDF will be consider for calculation of scoring and ranking, The Experimental results shows the ranking of the files containing the required keyword using the proposed technique are more relevant than existing Xbridge method.

REFERENCES

[1] Jianxin Li, Chengfei Liu, Rui Zhou, : XML keyword search with promising result type recommendations, In:www, pp.127-157(2014)

[2] Bao, Z., Ling, T.W., Chen, B., Lu, J.: Effective xml keyword search with relevance oriented ranking. In: ICDE, pp. 517–528 (2009)

[3] Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: XSEarch: a semantic search engine XML. In: VLDB, pp. 45–56 (2003)

[4] Florescu, D., Kosssmann, D., Manolescu, I.: Integrating keyword search into XML query processing. Comput. Networks 33(1–6), 119–135 (2000)

[5] Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.:XRANK: ranked keyword search over XML documents. In: SIGMOD Conference, pp. 16–27 (2003)

[6] Hadjieleftheriou, M., Chandel, A., Koudas, N., Srivastava, D.: Fast indexes and algorithms for set similarity selection queries. In: ICDE, pp. 267–276 (2008)

[7] Hristidis, V., Papakonstantinou, Y., Balmin, A.: Keyword proximity search on xml graphs. In: ICDE, pp. 367–378 (2003)

[8] Kulkarni, S., Caragea, D.: Computation of the semantic relatedness between words using concept clouds. In: KDIR, pp. 183–188 (2009)

[9] Li, Y., Yu, C., Jagadish, H.V.: Schema-free XQuery. In: VLDB, pp. 72–83 (2004)

[10] Li, G., Feng, J., Wang, J., Zhou, L.: Effective keyword search for valuable lcas over xml documents. In: CIKM, pp. 31–40 (2007)

[11] Li, J., Liu, C., Zhou, R., Wang, W.: Suggestion of promising result types for XMLkeyword search. In: EDBT, pp. 561–572 (2010)

[12] Li, J., Liu, C., Zhou, R., Wang, W.: Top-k keyword search over probabilistic XML data. In: ICDE, pp. 673–684 (2011)

[13] Liu, Z., Chen, Y.: Identifying meaningful return information for xml keyword search. In: SIGMOD Conference, pp. 329–340 (2007)

[14] Liu, Z., Chen, Y.: Reasoning and identifying relevant matches for xml keyword search. PVLDB 1(1), 921–932 (2008)

[15] Liu, C., Li, J., Yu, J.X., Zhou, R.: Adaptive relaxation for querying heterogeneous XML data sources. Inf. Syst. 35(6), 688–707 (2010)

[16] Sun, C., Chan, C.Y., Goenka, A.K.: Multiwayslca-based keyword search in xml data. In: WWW, pp. 1043–1052 (2007)

[17] Termehchy, A., Winslett, M.: Effective, design-independent xml keyword search. In: CIKM, pp. 107–116, (2009)